Untitled

April 15, 2023

0.1 Exercise1

Let's start creating a function that given the key word mentioned on the exercise, returns a permutation of the letters in the alphabet, following the steps indicated on the exercise:

[1]:
```
def permutation(keyword):
    alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    non_used_letters1 = ''
    non_used_letters2 = ''
    last_letter = keyword[-1]

    for letter in alphabet:
        if letter not in keyword:
            if alphabet.index(letter) < alphabet.index(last_letter):
                non_used_letters1 += letter
            else:
                non_used_letters2 += letter

    permutation = keyword + non_used_letters2 + non_used_letters1

    return permutation
```

The function takes the keyword as argument. Then, we define the alphabet and we create 2 null strings called non_used_letters1 and non_used_letters2 were we are going to append the letters that are not in keyword. Also, we are going to store the last letter of the keyword in last_letter. In non_used_letters1 are letters that are not in keyword and with index below last_letter index on the alphabet. On the other hand, in non_used_letters2 are letters with higher index than last_letter index. We create the permutation adding the keyword with the non_used_letters2 (that are after last_letter in alphabet) and then non_used_letters1 (that are before last_letter in alphabet).

In the exercise we have the example using LIME and we should get LIMEFGHJKNOPQRSTUVWXYZABCD. Let's see if it works:

[2]:
```
permutation('LIME')
```

[2]:
```
'LIMEFGHJKNOPQRSTUVWXYZABCD'
```

[3]:
```
if permutation('LIME') == 'LIMEFGHJKNOPQRSTUVWXYZABCD':
    print('It works')
else:
    print('It doesnt work')
```

```
It works
```

Now, we have that the keyword is APRICOT and we want to decrypt the following message:

[4]:
```
message = 'VOQDK MABJJ DZAXC ABAEE YCEVC OGDZH RGAJR UQDKZ KHJOV GHJRG CAJCJ␣
 ↪UCKBV LCGHC'
alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
keyword = 'APRICOT'
alphabet_permutation = permutation(keyword)
```

We are going to create a function for decrypting the messages given the alphabet, the corresponding permutation and the message we want to decrypt:

[5]:
```
def decrypt(alphabet, permutation, message):

    mapping = dict(zip(permutation, alphabet))

    decrypted = ''
    for c in message:
        if c in mapping:
            decrypted += mapping[c]
        else:
            decrypted += c

    return decrypted
```

We create a dictionary that maps each character in the permutation to its corresponding character in the alphabet. We do this using the zip() function to pair up each character in permutation with its corresponding character in alphabet, and then passing the resulting pairs to the dict() constructor to create a dictionary.

Then, we loop through each character in message, checking if it appears in the mapping dictionary. If it does, we replace the encrypted character with its corresponding decrypted character. If it doesn't, we simply append the original character to the decrypted string.

[6]:
```
decrypt(alphabet,alphabet_permutation,message)
```

[6]:
```
'IFYOU WANTT OMAKE ANAPP LEPIE FROMS CRATC HYOUM USTFI RSTCR EATET HEUNI VERSE'
```

Now, with the message already decrypted, we place the corresponding spaces and obtain:

[7]:
```
'If you want to make an apple pie from scratch you must first create the␣
 ↪universe'
```

[7]:
```
'If you want to make an apple pie from scratch you must first create the
universe'
```

0.1.1 Auxiliar Fucntions

[8]:
```
def frequency(message,alphabet):
    counter = [0]*len(alphabet)
    for char in message:
        if char != ' ':
            num = alphabet.find(char)
            counter[num] += 1
    freq = {}
    i = 0
    for char in alphabet:
        freq[char] = counter[i]
        i += 1
    return freq
```

[9]:
```
def count_bigrams(string):
    bigrams = {}
    for i in range(len(string)-1):
        bigram = string[i:i+2]
        if bigram in bigrams:
            bigrams[bigram] += 1
        else:
            bigrams[bigram] = 1
    bigram_counts = dict(sorted(bigrams.items(), key=lambda x: x[1],␣
 ↪reverse=True))
    return bigram_counts
```

[10]:
```
def count_trigrams(string):
    trigrams = {}
    for i in range(len(string)-2):
        trigram = string[i:i+3]
        if trigram in trigrams:
            trigrams[trigram] += 1
        else:
            trigrams[trigram] = 1
    trigram_counts = dict(sorted(trigrams.items(), key=lambda x: x[1],␣
 ↪reverse=True))
    return trigram_counts
```

0.2 Exercise 2

We have a ciphertext encrypted using the mechanism from the previous exercise.

[11]:
```
message = 'WBJGW RBGRC BRKHW RJKCK RZZDR CDZRW BJLGV TNTPT JKCIR LBERH JKCCE␣
↪IPRMR HPCBR JCARK VWBUK VTKBC CBRRM RHYBR NIRSC HRILK WBDCR KHPWK JKVRR␣
↪OTGKC DDCJW KR'
alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

We know that the text in encrypted using the method from exercise 1, so we need to find the key, that is a word in English without repeated letters.

First, we are going to define a function that counts the frequency of each letter:

[12]:
```
f=frequency(message,alphabet)
dict(sorted(f.items(), key=lambda x: x[1], reverse=True))
```

[12]:
```
{'R': 23,
 'C': 14,
 'K': 14,
 'B': 11,
 'J': 8,
 'W': 8,
 'H': 6,
 'D': 5,
 'T': 5,
 'G': 4,
 'I': 4,
 'P': 4,
 'V': 4,
 'L': 3,
 'Z': 3,
 'E': 2,
 'M': 2,
 'N': 2,
 'A': 1,
 'O': 1,
 'S': 1,
 'U': 1,
 'Y': 1,
 'F': 0,
 'Q': 0,
 'X': 0}
```

We know that the most common letter in English is e, so R should be e. If we suppose that, then our key should be - - - - R - ... with non-repeated letters. I used this page https://www.dictionary.com/e/word-finder/words-with-the-letter-r/ for looking up the words with R and using C with flex (language for regular expresions) we can get a list of words in English with R in the 5th position. Because the text is short and the list is no more of 350 words, we can introduce them in a text file a iterate the words (uppercasing them) using the algorithm from exercise 1 until we find a decrypted text with sense. After a few interations (it took like 5-6 minutes because I had to see if the text made sense or no before going to the next iteration), we get that is

using TIGER as key so:

[13]:
```
keyword = 'TIGER'
alphabet_permutation = permutation(keyword)
decrypt(alphabet,alphabet_permutation,message)
```

[13]:
```
'INSCI ENCEO NETRI ESTOT ELLPE OPLEI NSUCH AWAYA STOBE UNDER STOOD BYEVE RYONE
SOMET HINGT HATNO ONEEV ERKNE WBEFO REBUT INPOE TRYIT STHEE XACTO PPOSI TE'
```

Ordering it we get that:

[14]:
```
'In science one tries to tell peolpe in such a way as to be understood by␣
 ↪everyone, something that no one ever knew before but in poetry its the exact␣
 ↪opposite'
```

[14]:
```
'In science one tries to tell peolpe in such a way as to be understood by
everyone, something that no one ever knew before but in poetry its the exact
opposite'
```

0.3 Exercise 3

[15]:
```
message = 'NBPFR KISOQ NFRDB FKJFD XNOIN OJXIX NZXSI DJXIJ NYENO ISDSA SOFBY␣
 ↪REJRK IKSKI PFRAR DJZIJ RUSEE JXIZI KADFB JXIJK SODYI OGIOJ SEJIK ADSOG␣
 ↪UESOJ JXIAI VKPWX IKIPF RARDJ ENIRU FOJXI GSNDN IDSOG GNDYF RKDIN OOFVI␣
 ↪EUXKS DIDFB PFRKY FAUEN YSJIG DJSJI FBANO GJXIA ISONO ZGFID OJASJ JIKNB␣
 ↪NJDFO EPNGE IYXSJ JIKFB SJKSO DYIOG IOJSE LNOGS OGIVK PFOIW NEEDS PSDPF␣
 ↪RWSEL PFRKA PDJNY WSPNB JXNDP FROZA SOIQU KIDDI DXNAD IEBNO JIKAD JFFGI␣
 ↪IUBFK AIWXP WXSJS VIKPD NOZRE SKEPG IIUPF ROZAS OJXND GIIUP FROZA SOARD␣
 ↪JCICI IEFMR IOJNO UKSND IFBJX IVIKP GREEF EGGSP DWXNY XXSVI EFOZD NOYIU␣
 ↪SDDIG SWSPS OGYFO VNOYI IANBP FRYSO JXSJJ XIKIN ZOFBZ FFGMR IIOSO OIWSD␣
 ↪YREJR KIDUS EANID JGSPF BYFRK DIPFR WNEEU FFXUF FXWXS JIVIK DBKID XSOGO␣
 ↪IWSOG GIYES KINJD YKRGI SOGAI SOBFK SKJDJ FUUIG DXFKJ NOJXI YREJN VSJIG␣
 ↪YFRKJ FBJXI IAUKI DDHFD IUXNO ISOGI VKPFO IWNEE DSPSD PFRWS ELPFR KAPDJ␣
 ↪NYWSP NBJXS JDOFJ ZFFGI OFRZX BFKXN AWXNY XNDZF FGIOF RZXBF KAIWX PWXSJ␣
 ↪SVIKP YREJN VSJIG LNOGF BPFRJ XJXND LNOGF BPFRJ XARDJ CIJXI OSDIO JNAIO␣
 ↪JSEUS DDNFO FBSVI ZIJSC EIBSD XNFOA RDJIQ YNJIP FRKES OZRNG DUEII OSOSJ␣
 ↪JSYXA IOJSE SUESJ FBFKS CSDXB REPFR OZUFJ SJFFK SOFJJ FFBKI OYXBK IOYXC␣
 ↪ISOJX FRZXJ XIUXN ENDJN OIDAS PHFDJ EIPFR WNEEK SOLSD SOSUF DJEIN OJXIX␣
 ↪NZXSI DJXIJ NYCSO GNBPF RWSEL GFWOU NYYSG NEEPW NJXSU FUUPF KSENE PNOPF␣
 ↪RKAIG NIVSE XSOGS OGIVK PFOIW NEEDS PSDPF RWSEL PFRKB EFWKP WSPNB XIDYF␣
 ↪OJIOJ WNJXS VIZIJ SCEIE FVIWX NYXWF REGYI KJSNO EPOFJ DRNJA IWXPW XSJSA␣
 ↪FDJUS KJNYR ESKEP URKIP FROZA SOJXN DURKI PFROZ ASOAR DJCI'
alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

We are going to start studing the frequency of the letters:

[16]:
```
f=frequency(message,alphabet)
dict(sorted(f.items(), key=lambda x: x[1], reverse=True))
```

[16]:
```
{'I': 130,
 'S': 108,
 'F': 98,
 'J': 98,
 'O': 91,
 'N': 77,
 'D': 75,
 'X': 66,
 'E': 61,
 'K': 58,
 'R': 57,
 'P': 54,
 'G': 43,
 'A': 36,
 'Y': 34,
 'B': 32,
 'U': 31,
 'W': 31,
 'Z': 23,
 'V': 17,
 'C': 9,
 'L': 8,
 'Q': 3,
 'H': 2,
 'M': 2,
 'T': 0}
```

Also we are going to study frequency of bigrams and trigrams:

[17]:
```
message_no_space=''
for char in message:
    if char != ' ':
        message_no_space += char
```

[18]:
```
bigrams = count_bigrams(message_no_space)
bigrams
```

[18]:
```
{'FR': 32,
 'SO': 31,
 'PF': 29,
 'JX': 26,
 'NO': 20,
 'OJ': 19,
 'DJ': 18,
 'XI': 17,
 'OG': 17,
 'XN': 16,
```

```
'IO': 16,
'SJ': 16,
'RK': 14,
'KI': 14,
'XS': 14,
'FB': 14,
'ID': 13,
'IK': 13,
'JS': 13,
'SD': 12,
'SE': 12,
'GI': 12,
'JI': 12,
'JN': 11,
'WX': 11,
'OI': 10,
'NY': 10,
'KS': 10,
'FO': 10,
'OZ': 10,
'SP': 10,
'WS': 10,
'FK': 9,
'OF': 9,
'RE': 9,
'AI': 9,
'ND': 9,
'DI': 9,
'VI': 9,
'IW': 9,
'AS': 8,
'EE': 8,
'KA': 8,
'KP': 8,
'NE': 8,
'FF': 8,
'II': 8,
'NB': 7,
'IS': 7,
'RD': 7,
'JF': 7,
'IJ': 7,
'DS': 7,
'IP': 7,
'ES': 7,
'IV': 7,
'IG': 7,
```

```
'EP': 7,
'YX': 7,
'WN': 7,
'BP': 6,
'BF': 6,
'KJ': 6,
'DX': 6,
'EJ': 6,
'SK': 6,
'AR': 6,
'DY': 6,
'JJ': 6,
'PW': 6,
'UF': 6,
'GS': 6,
'YF': 6,
'NJ': 6,
'EI': 6,
'RW': 6,
'RO': 6,
'IU': 6,
'FD': 5,
'IN': 5,
'ZX': 5,
'EN': 5,
'YR': 5,
'US': 5,
'BJ': 5,
'YI': 5,
'IE': 5,
'JD': 5,
'PN': 5,
'EL': 5,
'ZA': 5,
'SV': 5,
'CI': 5,
'EF': 5,
'OS': 5,
'ZI': 4,
'AD': 4,
'UE': 4,
'IA': 4,
'VK': 4,
'NI': 4,
'DN': 4,
'GN': 4,
'GF': 4,
```

```
'PS': 4,
'DP': 4,
'PD': 4,
'DD': 4,
'FG': 4,
'OY': 4,
'FJ': 4,
'XB': 4,
'NF': 3,
'NZ': 3,
'JR': 3,
'IZ': 3,
'DF': 3,
'JE': 3,
'SN': 3,
'GG': 3,
'KD': 3,
'EU': 3,
'UX': 3,
'YS': 3,
'GD': 3,
'AN': 3,
'IY': 3,
'BS': 3,
'LN': 3,
'ED': 3,
'LP': 3,
'UK': 3,
'NA': 3,
'XP': 3,
'KE': 3,
'UP': 3,
'OA': 3,
'JC': 3,
'GY': 3,
'ZF': 3,
'DU': 3,
'BK': 3,
'VS': 3,
'RZ': 3,
'SC': 3,
'SU': 3,
'DB': 2,
'IX': 2,
'SI': 2,
'YE': 2,
'SA': 2,
```

```
'BY': 2,
'RA': 2,
'JZ': 2,
'RU': 2,
'JK': 2,
'OD': 2,
'OO': 2,
'FV': 2,
'AU': 2,
'IF': 2,
'DO': 2,
'JA': 2,
'BN': 2,
'OE': 2,
'NG': 2,
'AP': 2,
'YW': 2,
'IQ': 2,
'ZR': 2,
'PG': 2,
'MR': 2,
'RI': 2,
'OU': 2,
'EG': 2,
'FX': 2,
'XW': 2,
'FU': 2,
'UU': 2,
'XF': 2,
'NV': 2,
'HF': 2,
'RJ': 2,
'XJ': 2,
'XA': 2,
'CE': 2,
'RN': 2,
'CS': 2,
'FW': 2,
'UR': 2,
'OQ': 1,
'QN': 1,
'GU': 1,
'IR': 1,
'XK': 1,
'KY': 1,
'FA': 1,
'BA': 1,
```

```
'GJ': 1,
'ON': 1,
'ZG': 1,
'FI': 1,
'KN': 1,
'GE': 1,
'KF': 1,
'QU': 1,
'EB': 1,
'UB': 1,
'DG': 1,
'IC': 1,
'FM': 1,
'GR': 1,
'FE': 1,
'DW': 1,
'XX': 1,
'ZD': 1,
'SW': 1,
'OV': 1,
'VN': 1,
'RY': 1,
'ZO': 1,
'BZ': 1,
'GM': 1,
'EA': 1,
'JG': 1,
'XU': 1,
'GO': 1,
'YK': 1,
'KR': 1,
'RG': 1,
'GA': 1,
'OB': 1,
'UI': 1,
'DH': 1,
'KX': 1,
'AW': 1,
'DZ': 1,
'PY': 1,
'GL': 1,
'DL': 1,
'IB': 1,
'QY': 1,
'YN': 1,
'SY': 1,
'BR': 1,
```

```
 'ZU': 1,
 'XC': 1,
 'DA': 1,
 'PH': 1,
 'EK': 1,
 'OL': 1,
 'LS': 1,
 'YC': 1,
 'LG': 1,
 'WO': 1,
 'UN': 1,
 'YY': 1,
 'SG': 1,
 'OP': 1,
 'EX': 1,
 'KB': 1,
 'BE': 1,
 'WK': 1,
 'BX': 1,
 'JW': 1,
 'WF': 1,
 'PO': 1,
 'DR': 1,
 'AF': 1,
 'JU': 1,
 'PU': 1}
```

[19]:
```
trigrams = count_trigrams(message_no_space)
trigrams
```

[19]:
```
{'PFR': 24,
 'JXI': 15,
 'SOG': 11,
 'FRK': 10,
 'OJX': 8,
 'IPF': 7,
 'IOJ': 7,
 'XSJ': 7,
 'BPF': 6,
 'ISO': 6,
 'BFK': 6,
 'ASO': 6,
 'ARD': 6,
 'RDJ': 6,
 'NEE': 6,
 'FRW': 6,
 'FRO': 6,
```

```
'ROZ': 6,
'RKI': 5,
'JNY': 5,
'YRE': 5,
'BJX': 5,
'OGI': 5,
'SOJ': 5,
'SJI': 5,
'SEL': 5,
'OIW': 5,
'WNE': 5,
'XND': 5,
'OZA': 5,
'ZAS': 5,
'SVI': 5,
'NOI': 4,
'NOJ': 4,
'REJ': 4,
'KIP': 4,
'KSO': 4,
'GIO': 4,
'OJS': 4,
'JSE': 4,
'JIK': 4,
'IVK': 4,
'VKP': 4,
'PWX': 4,
'NOG': 4,
'SJJ': 4,
'SPS': 4,
'DPF': 4,
'RWS': 4,
'WSE': 4,
'WSP': 4,
'JXN': 4,
'KID': 4,
'FFG': 4,
'IWX': 4,
'WXS': 4,
'VIK': 4,
'JXS': 4,
'FKS': 4,
'NBP': 3,
'DXN': 3,
'INO': 3,
'IDJ': 3,
'XIJ': 3,
```

```
'OFB': 3,
'ZIJ': 3,
'IZI': 3,
'IKA': 3,
'KAD': 3,
'FBJ': 3,
'DSO': 3,
'DJE': 3,
'YFR': 3,
'FBP': 3,
'JIG': 3,
'LNO': 3,
'GIV': 3,
'KPF': 3,
'PFO': 3,
'FOI': 3,
'IWN': 3,
'EED': 3,
'EDS': 3,
'DSP': 3,
'PSD': 3,
'SDP': 3,
'ELP': 3,
'LPF': 3,
'RKA': 3,
'DJN': 3,
'SPN': 3,
'PNB': 3,
'JFF': 3,
'FGI': 3,
'GII': 3,
'IIU': 3,
'KAI': 3,
'AIW': 3,
'WXP': 3,
'XPW': 3,
'SJS': 3,
'IKP': 3,
'ESK': 3,
'UPF': 3,
'OAR': 3,
'DJC': 3,
'JCI': 3,
'IEF': 3,
'JNO': 3,
'WXN': 3,
'XNY': 3,
```

```
'NYX': 3,
'ZFF': 3,
'IOS': 3,
'BKI': 3,
'OFJ': 3,
'FRZ': 3,
'RZX': 3,
'FDJ': 3,
'FKJ': 2,
'KJF': 2,
'XNO': 2,
'XIX': 2,
'IXN': 2,
'XNZ': 2,
'NZX': 2,
'ZXS': 2,
'XSI': 2,
'SID': 2,
'DJX': 2,
'IJN': 2,
'OIS': 2,
'SDS': 2,
'SOF': 2,
'FBY': 2,
'EJR': 2,
'JRK': 2,
'KSK': 2,
'SKI': 2,
'FRA': 2,
'RAR': 2,
'USE': 2,
'DFB': 2,
'JKS': 2,
'SOD': 2,
'ODY': 2,
'DYI': 2,
'YIO': 2,
'IOG': 2,
'UES': 2,
'ESO': 2,
'JJX': 2,
'XIA': 2,
'IAI': 2,
'KPW': 2,
'XIK': 2,
'IKI': 2,
'FOJ': 2,
```

```
'IGS': 2,
'SND': 2,
'NID': 2,
'OGG': 2,
'DYF': 2,
'RKD': 2,
'KDI': 2,
'FVI': 2,
'VIE': 2,
'SDI': 2,
'DID': 2,
'IGD': 2,
'JSJ': 2,
'IFB': 2,
'AIS': 2,
'NOZ': 2,
'JJI': 2,
'NJD': 2,
'OEP': 2,
'EPN': 2,
'FBS': 2,
'OGS': 2,
'GSO': 2,
'KAP': 2,
'APD': 2,
'PDJ': 2,
'NYW': 2,
'YWS': 2,
'NBJ': 2,
'UKI': 2,
'IDD': 2,
'DDI': 2,
'IDX': 2,
'XNA': 2,
'OJI': 2,
'DJF': 2,
'FKA': 2,
'JSV': 2,
'DNO': 2,
'OZR': 2,
'RES': 2,
'SKE': 2,
'KEP': 2,
'IUP': 2,
'SOA': 2,
'MRI': 2,
'OJN': 2,
```

```
'IVI': 2,
'GSP': 2,
'XSV': 2,
'NOY': 2,
'OYI': 2,
'USD': 2,
'SDD': 2,
'GYF': 2,
'YFO': 2,
'IIA': 2,
'KIN': 2,
'IIO': 2,
'OSO': 2,
'IWS': 2,
'RWN': 2,
'UFF': 2,
'FFX': 2,
'XSO': 2,
'SKJ': 2,
'FUU': 2,
'KJN': 2,
'EJN': 2,
'JNV': 2,
'NVS': 2,
'VSJ': 2,
'JFB': 2,
'HFD': 2,
'IUX': 2,
'UXN': 2,
'IOF': 2,
'OFR': 2,
'ZXB': 2,
'XBF': 2,
'OGF': 2,
'GFB': 2,
'FRJ': 2,
'RJX': 2,
'XJX': 2,
'AIO': 2,
'NFO': 2,
'VIZ': 2,
'IJS': 2,
'JSC': 2,
'SCE': 2,
'CEI': 2,
'SDX': 2,
'SOS': 2,
```

```
'SJF': 2,
'KIO': 2,
'IOY': 2,
'OYX': 2,
'JEI': 2,
'SUF': 2,
'WNJ': 2,
'NJX': 2,
'URK': 2,
'KIS': 1,
'SOQ': 1,
'OQN': 1,
'QNF': 1,
'NFR': 1,
'FRD': 1,
'RDB': 1,
'DBF': 1,
'JFD': 1,
'FDX': 1,
'OIN': 1,
'NYE': 1,
'YEN': 1,
'ENO': 1,
'ISD': 1,
'DSA': 1,
'SAS': 1,
'BYR': 1,
'KIK': 1,
'IKS': 1,
'DJZ': 1,
'JZI': 1,
'IJR': 1,
'JRU': 1,
'RUS': 1,
'SEE': 1,
'EEJ': 1,
'EJX': 1,
'XIZ': 1,
'ZIK': 1,
'ADF': 1,
'IJK': 1,
'SEJ': 1,
'EJI': 1,
'ADS': 1,
'OGU': 1,
'GUE': 1,
'OJJ': 1,
```

```
'AIV': 1,
'WXI': 1,
'JEN': 1,
'ENI': 1,
'NIR': 1,
'IRU': 1,
'RUF': 1,
'UFO': 1,
'XIG': 1,
'GSN': 1,
'NDN': 1,
'DNI': 1,
'IDS': 1,
'GGN': 1,
'GND': 1,
'NDY': 1,
'DIN': 1,
'NOO': 1,
'OOF': 1,
'OFV': 1,
'IEU': 1,
'EUX': 1,
'UXK': 1,
'XKS': 1,
'KSD': 1,
'IDF': 1,
'RKY': 1,
'KYF': 1,
'YFA': 1,
'FAU': 1,
'AUE': 1,
'UEN': 1,
'ENY': 1,
'NYS': 1,
'YSJ': 1,
'GDJ': 1,
'DJS': 1,
'JIF': 1,
'FBA': 1,
'BAN': 1,
'ANO': 1,
'OGJ': 1,
'GJX': 1,
'SON': 1,
'ONO': 1,
'OZG': 1,
'ZGF': 1,
```

```
'GFI': 1,
'FID': 1,
'IDO': 1,
'DOJ': 1,
'OJA': 1,
'JAS': 1,
'ASJ': 1,
'IKN': 1,
'KNB': 1,
'NBN': 1,
'BNJ': 1,
'JDF': 1,
'DFO': 1,
'FOE': 1,
'PNG': 1,
'NGE': 1,
'GEI': 1,
'EIY': 1,
'IYX': 1,
'YXS': 1,
'IKF': 1,
'KFB': 1,
'BSJ': 1,
'SJK': 1,
'ELN': 1,
'NDP': 1,
'SOI': 1,
'OIQ': 1,
'IQU': 1,
'QUK': 1,
'NAD': 1,
'ADI': 1,
'DIE': 1,
'IEB': 1,
'EBN': 1,
'BNO': 1,
'ADJ': 1,
'IUB': 1,
'UBF': 1,
'KPD': 1,
'PDN': 1,
'ZRE': 1,
'EPG': 1,
'PGI': 1,
'NDG': 1,
'DGI': 1,
'CIC': 1,
```

```
'ICI': 1,
'CII': 1,
'IIE': 1,
'EFM': 1,
'FMR': 1,
'RIO': 1,
'NOU': 1,
'OUK': 1,
'UKS': 1,
'KSN': 1,
'NDI': 1,
'DIF': 1,
'XIV': 1,
'KPG': 1,
'PGR': 1,
'GRE': 1,
'REE': 1,
'EEF': 1,
'EFE': 1,
'FEG': 1,
'EGG': 1,
'GGS': 1,
'SPD': 1,
'PDW': 1,
'DWX': 1,
'YXX': 1,
'XXS': 1,
'EFO': 1,
'FOZ': 1,
'OZD': 1,
'ZDN': 1,
'YIU': 1,
'IUS': 1,
'DIG': 1,
'GSW': 1,
'SWS': 1,
'PSO': 1,
'OGY': 1,
'FOV': 1,
'OVN': 1,
'VNO': 1,
'YII': 1,
'IAN': 1,
'ANB': 1,
'FRY': 1,
'RYS': 1,
'YSO': 1,
```

```
'INZ': 1,
'NZO': 1,
'ZOF': 1,
'FBZ': 1,
'BZF': 1,
'FGM': 1,
'GMR': 1,
'RII': 1,
'SOO': 1,
'OOI': 1,
'WSD': 1,
'SDY': 1,
'DYR': 1,
'IDU': 1,
'DUS': 1,
'SEA': 1,
'EAN': 1,
'ANI': 1,
'DJG': 1,
'JGS': 1,
'SPF': 1,
'PFB': 1,
'BYF': 1,
'DIP': 1,
'EEU': 1,
'EUF': 1,
'FXU': 1,
'XUF': 1,
'FXW': 1,
'XWX': 1,
'JIV': 1,
'IKD': 1,
'KDB': 1,
'DBK': 1,
'DXS': 1,
'OGO': 1,
'GOI': 1,
'WSO': 1,
'GGI': 1,
'GIY': 1,
'IYE': 1,
'YES': 1,
'INJ': 1,
'JDY': 1,
'DYK': 1,
'YKR': 1,
'KRG': 1,
```

```
'RGI': 1,
'GIS': 1,
'OGA': 1,
'GAI': 1,
'SOB': 1,
'OBF': 1,
'KJD': 1,
'JDJ': 1,
'JFU': 1,
'UUI': 1,
'UIG': 1,
'GDX': 1,
'DXF': 1,
'XFK': 1,
'XIY': 1,
'IYR': 1,
'IGY': 1,
'RKJ': 1,
'XII': 1,
'IAU': 1,
'AUK': 1,
'DDH': 1,
'DHF': 1,
'FDI': 1,
'DIU': 1,
'SJD': 1,
'JDO': 1,
'DOF': 1,
'FJZ': 1,
'JZF': 1,
'FKX': 1,
'KXN': 1,
'NAW': 1,
'AWX': 1,
'YXN': 1,
'NDZ': 1,
'DZF': 1,
'KPY': 1,
'PYR': 1,
'IGL': 1,
'GLN': 1,
'JXJ': 1,
'NDL': 1,
'DLN': 1,
'JXA': 1,
'XAR': 1,
'CIJ': 1,
```

```
'IJX': 1,
'XIO': 1,
'OSD': 1,
'DIO': 1,
'JNA': 1,
'NAI': 1,
'SEU': 1,
'EUS': 1,
'DDN': 1,
'DNF': 1,
'FOF': 1,
'BSV': 1,
'EIB': 1,
'IBS': 1,
'BSD': 1,
'XNF': 1,
'FOA': 1,
'DJI': 1,
'JIQ': 1,
'IQY': 1,
'QYN': 1,
'YNJ': 1,
'NJI': 1,
'JIP': 1,
'RKE': 1,
'KES': 1,
'SOZ': 1,
'ZRN': 1,
'RNG': 1,
'NGD': 1,
'GDU': 1,
'DUE': 1,
'UEI': 1,
'EII': 1,
'OSJ': 1,
'JJS': 1,
'JSY': 1,
'SYX': 1,
'YXA': 1,
'XAI': 1,
'SES': 1,
'ESU': 1,
'SUE': 1,
'ESJ': 1,
'FBF': 1,
'KSC': 1,
'SCS': 1,
```

```
'CSD': 1,
'DXB': 1,
'XBR': 1,
'BRE': 1,
'REP': 1,
'EPF': 1,
'OZU': 1,
'ZUF': 1,
'UFJ': 1,
'FJS': 1,
'FFK': 1,
'FJJ': 1,
'JJF': 1,
'FFB': 1,
'FBK': 1,
'YXB': 1,
'XBK': 1,
'YXC': 1,
'XCI': 1,
'CIS': 1,
'JXF': 1,
'XFR': 1,
'ZXJ': 1,
'XIU': 1,
'XNE': 1,
'NEN': 1,
'END': 1,
'NDJ': 1,
'OID': 1,
'IDA': 1,
'DAS': 1,
'ASP': 1,
'SPH': 1,
'PHF': 1,
'EIP': 1,
'EEK': 1,
'EKS': 1,
'SOL': 1,
'OLS': 1,
'LSD': 1,
'OSU': 1,
'UFD': 1,
'EIN': 1,
'NYC': 1,
'YCS': 1,
'CSO': 1,
'OGN': 1,
```

```
'GNB': 1,
'ELG': 1,
'LGF': 1,
'GFW': 1,
'FWO': 1,
'WOU': 1,
'OUN': 1,
'UNY': 1,
'NYY': 1,
'YYS': 1,
'YSG': 1,
'SGN': 1,
'GNE': 1,
'EEP': 1,
'EPW': 1,
'PWN': 1,
'XSU': 1,
'UFU': 1,
'UUP': 1,
'PFK': 1,
'KSE': 1,
'SEN': 1,
'ENE': 1,
'NEP': 1,
'PNO': 1,
'NOP': 1,
'OPF': 1,
'AIG': 1,
'IGN': 1,
'GNI': 1,
'NIV': 1,
'IVS': 1,
'VSE': 1,
'SEX': 1,
'EXS': 1,
'RKB': 1,
'KBE': 1,
'BEF': 1,
'EFW': 1,
'FWK': 1,
'WKP': 1,
'PWS': 1,
'NBX': 1,
'BXI': 1,
'XID': 1,
'IDY': 1,
'JIO': 1,
```

```
 'OJW': 1,
 'JWN': 1,
 'EIE': 1,
 'EFV': 1,
 'VIW': 1,
 'YXW': 1,
 'XWF': 1,
 'WFR': 1,
 'FRE': 1,
 'REG': 1,
 'EGY': 1,
 'GYI': 1,
 'YIK': 1,
 'IKJ': 1,
 'KJS': 1,
 'JSN': 1,
 'SNO': 1,
 'NOE': 1,
 'EPO': 1,
 'POF': 1,
 'FJD': 1,
 'JDR': 1,
 'DRN': 1,
 'RNJ': 1,
 'NJA': 1,
 'JAI': 1,
 'JSA': 1,
 'SAF': 1,
 'AFD': 1,
 'DJU': 1,
 'JUS': 1,
 'USK': 1,
 'NYR': 1,
 'EPU': 1,
 'PUR': 1,
 'NDU': 1,
 'DUR': 1}
```

[20]:
```python
def count_quadgrams(string):
    quadgrams = {}
    for i in range(len(string)-3):
        quadgram = string[i:i+4]
        if quadgram in quadgrams:
            quadgrams[quadgram] += 1
        else:
            quadgrams[quadgram] = 1
```

```
    quadgram_counts = dict(sorted(quadgrams.items(), key=lambda x: x[1],␣
↪reverse=True))
    return quadgram_counts
```

[21]:
```
quadgrams = count_quadgrams(message_no_space)
quadgrams
```

[21]:
```
{'PFRK': 7,
 'IPFR': 7,
 'BPFR': 6,
 'ARDJ': 6,
 'PFRW': 6,
 'PFRO': 6,
 'FROZ': 6,
 'WNEE': 5,
 'ROZA': 5,
 'OZAS': 5,
 'ZASO': 5,
 'OJXI': 4,
 'YREJ': 4,
 'KIPF': 4,
 'IOJS': 4,
 'OJSE': 4,
 'IVKP': 4,
 'DPFR': 4,
 'FRWS': 4,
 'RWSE': 4,
 'WSEL': 4,
 'JXND': 4,
 'WXSJ': 4,
 'SOJX': 4,
 'NBPF': 3,
 'NOJX': 3,
 'JXIJ': 3,
 'IKAD': 3,
 'FBJX': 3,
 'BJXI': 3,
 'YFRK': 3,
 'FBPF': 3,
 'SJIG': 3,
 'LNOG': 3,
 'SOGI': 3,
 'OGIV': 3,
 'GIVK': 3,
 'VKPF': 3,
 'KPFO': 3,
 'PFOI': 3,
```

```
'FOIW': 3,
'OIWN': 3,
'IWNE': 3,
'NEED': 3,
'EEDS': 3,
'EDSP': 3,
'DSPS': 3,
'SPSD': 3,
'PSDP': 3,
'SDPF': 3,
'SELP': 3,
'ELPF': 3,
'LPFR': 3,
'FRKA': 3,
'WSPN': 3,
'SPNB': 3,
'FFGI': 3,
'GIIU': 3,
'AIWX': 3,
'IWXP': 3,
'WXPW': 3,
'XPWX': 3,
'PWXS': 3,
'XSJS': 3,
'VIKP': 3,
'OARD': 3,
'RDJC': 3,
'DJCI': 3,
'WXNY': 3,
'XNYX': 3,
'ZFFG': 3,
'FRZX': 3,
'XNOI': 2,
'INOJ': 2,
'JXIX': 2,
'XIXN': 2,
'IXNZ': 2,
'XNZX': 2,
'NZXS': 2,
'ZXSI': 2,
'XSID': 2,
'SIDJ': 2,
'IDJX': 2,
'DJXI': 2,
'XIJN': 2,
'IJNY': 2,
'NOIS': 2,
```

```
'REJR': 2,
'EJRK': 2,
'JRKI': 2,
'PFRA': 2,
'FRAR': 2,
'RARD': 2,
'JKSO': 2,
'KSOD': 2,
'SODY': 2,
'ODYI': 2,
'DYIO': 2,
'YIOG': 2,
'IOGI': 2,
'OGIO': 2,
'GIOJ': 2,
'JIKA': 2,
'DSOG': 2,
'JJXI': 2,
'JXIA': 2,
'XIAI': 2,
'XIKI': 2,
'SOGG': 2,
'FRKD': 2,
'RKDI': 2,
'AISO': 2,
'SJJI': 2,
'JJIK': 2,
'XSJJ': 2,
'OGSO': 2,
'GSOG': 2,
'RKAP': 2,
'KAPD': 2,
'APDJ': 2,
'PDJN': 2,
'DJNY': 2,
'JNYW': 2,
'NYWS': 2,
'YWSP': 2,
'PNBJ': 2,
'NBJX': 2,
'UKID': 2,
'KIDD': 2,
'BFKA': 2,
'FKAI': 2,
'KAIW': 2,
'SJSV': 2,
'JSVI': 2,
```

```
'SVIK': 2,
'RESK': 2,
'ESKE': 2,
'SKEP': 2,
'IIUP': 2,
'IUPF': 2,
'UPFR': 2,
'ASOJ': 2,
'OJXN': 2,
'ASOA': 2,
'SOAR': 2,
'IOJN': 2,
'IVIK': 2,
'XSVI': 2,
'NOYI': 2,
'USDD': 2,
'JXSJ': 2,
'IIOS': 2,
'IOSO': 2,
'OIWS': 2,
'FRWN': 2,
'RWNE': 2,
'UFFX': 2,
'XSOG': 2,
'ISOG': 2,
'BFKS': 2,
'REJN': 2,
'EJNV': 2,
'JNVS': 2,
'NVSJ': 2,
'VSJI': 2,
'IUXN': 2,
'FGIO': 2,
'GIOF': 2,
'IOFR': 2,
'OFRZ': 2,
'RZXB': 2,
'ZXBF': 2,
'XBFK': 2,
'NOGF': 2,
'OGFB': 2,
'GFBP': 2,
'PFRJ': 2,
'FRJX': 2,
'AIOJ': 2,
'SVIZ': 2,
'VIZI': 2,
```

```
'IZIJ': 2,
'ZIJS': 2,
'IJSC': 2,
'JSCE': 2,
'SCEI': 2,
'BKIO': 2,
'KIOY': 2,
'IOYX': 2,
'FDJE': 2,
'DJEI': 2,
'WNJX': 2,
'NJXS': 2,
'URKI': 2,
'RKIP': 2,
'FRKI': 1,
'RKIS': 1,
'KISO': 1,
'ISOQ': 1,
'SOQN': 1,
'OQNF': 1,
'QNFR': 1,
'NFRD': 1,
'FRDB': 1,
'RDBF': 1,
'DBFK': 1,
'BFKJ': 1,
'FKJF': 1,
'KJFD': 1,
'JFDX': 1,
'FDXN': 1,
'DXNO': 1,
'NOIN': 1,
'OINO': 1,
'JNYE': 1,
'NYEN': 1,
'YENO': 1,
'ENOI': 1,
'OISD': 1,
'ISDS': 1,
'SDSA': 1,
'DSAS': 1,
'SASO': 1,
'ASOF': 1,
'SOFB': 1,
'OFBY': 1,
'FBYR': 1,
'BYRE': 1,
```

```
'RKIK': 1,
'KIKS': 1,
'IKSK': 1,
'KSKI': 1,
'SKIP': 1,
'RDJZ': 1,
'DJZI': 1,
'JZIJ': 1,
'ZIJR': 1,
'IJRU': 1,
'JRUS': 1,
'RUSE': 1,
'USEE': 1,
'SEEJ': 1,
'EEJX': 1,
'EJXI': 1,
'JXIZ': 1,
'XIZI': 1,
'IZIK': 1,
'ZIKA': 1,
'KADF': 1,
'ADFB': 1,
'DFBJ': 1,
'XIJK': 1,
'IJKS': 1,
'JSEJ': 1,
'SEJI': 1,
'EJIK': 1,
'KADS': 1,
'ADSO': 1,
'SOGU': 1,
'OGUE': 1,
'GUES': 1,
'UESO': 1,
'ESOJ': 1,
'SOJJ': 1,
'OJJX': 1,
'IAIV': 1,
'AIVK': 1,
'VKPW': 1,
'KPWX': 1,
'PWXI': 1,
'WXIK': 1,
'IKIP': 1,
'RDJE': 1,
'DJEN': 1,
'JENI': 1,
```

```
'ENIR': 1,
'NIRU': 1,
'IRUF': 1,
'RUFO': 1,
'UFOJ': 1,
'FOJX': 1,
'JXIG': 1,
'XIGS': 1,
'IGSN': 1,
'GSND': 1,
'SNDN': 1,
'NDNI': 1,
'DNID': 1,
'NIDS': 1,
'IDSO': 1,
'OGGN': 1,
'GGND': 1,
'GNDY': 1,
'NDYF': 1,
'DYFR': 1,
'KDIN': 1,
'DINO': 1,
'INOO': 1,
'NOOF': 1,
'OOFV': 1,
'OFVI': 1,
'FVIE': 1,
'VIEU': 1,
'IEUX': 1,
'EUXK': 1,
'UXKS': 1,
'XKSD': 1,
'KSDI': 1,
'SDID': 1,
'DIDF': 1,
'IDFB': 1,
'DFBP': 1,
'FRKY': 1,
'RKYF': 1,
'KYFA': 1,
'YFAU': 1,
'FAUE': 1,
'AUEN': 1,
'UENY': 1,
'ENYS': 1,
'NYSJ': 1,
'YSJI': 1,
```

```
'JIGD': 1,
'IGDJ': 1,
'GDJS': 1,
'DJSJ': 1,
'JSJI': 1,
'SJIF': 1,
'JIFB': 1,
'IFBA': 1,
'FBAN': 1,
'BANO': 1,
'ANOG': 1,
'NOGJ': 1,
'OGJX': 1,
'GJXI': 1,
'IAIS': 1,
'ISON': 1,
'SONO': 1,
'ONOZ': 1,
'NOZG': 1,
'OZGF': 1,
'ZGFI': 1,
'GFID': 1,
'FIDO': 1,
'IDOJ': 1,
'DOJA': 1,
'OJAS': 1,
'JASJ': 1,
'ASJJ': 1,
'JIKN': 1,
'IKNB': 1,
'KNBN': 1,
'NBNJ': 1,
'BNJD': 1,
'NJDF': 1,
'JDFO': 1,
'DFOE': 1,
'FOEP': 1,
'OEPN': 1,
'EPNG': 1,
'PNGE': 1,
'NGEI': 1,
'GEIY': 1,
'EIYX': 1,
'IYXS': 1,
'YXSJ': 1,
'JIKF': 1,
'IKFB': 1,
```

```
'KFBS': 1,
'FBSJ': 1,
'BSJK': 1,
'SJKS': 1,
'JSEL': 1,
'SELN': 1,
'ELNO': 1,
'NOGS': 1,
'BJXN': 1,
'XNDP': 1,
'NDPF': 1,
'ASOI': 1,
'SOIQ': 1,
'OIQU': 1,
'IQUK': 1,
'QUKI': 1,
'IDDI': 1,
'DDID': 1,
'DIDX': 1,
'IDXN': 1,
'DXNA': 1,
'XNAD': 1,
'NADI': 1,
'ADIE': 1,
'DIEB': 1,
'IEBN': 1,
'EBNO': 1,
'BNOJ': 1,
'NOJI': 1,
'OJIK': 1,
'KADJ': 1,
'ADJF': 1,
'DJFF': 1,
'JFFG': 1,
'FGII': 1,
'IIUB': 1,
'IUBF': 1,
'UBFK': 1,
'IKPD': 1,
'KPDN': 1,
'PDNO': 1,
'DNOZ': 1,
'NOZR': 1,
'OZRE': 1,
'ZRES': 1,
'KEPG': 1,
'EPGI': 1,
```

```
'PGII': 1,
'XNDG': 1,
'NDGI': 1,
'DGII': 1,
'JCIC': 1,
'CICI': 1,
'ICII': 1,
'CIIE': 1,
'IIEF': 1,
'IEFM': 1,
'EFMR': 1,
'FMRI': 1,
'MRIO': 1,
'RIOJ': 1,
'OJNO': 1,
'JNOU': 1,
'NOUK': 1,
'OUKS': 1,
'UKSN': 1,
'KSND': 1,
'SNDI': 1,
'NDIF': 1,
'DIFB': 1,
'IFBJ': 1,
'JXIV': 1,
'XIVI': 1,
'IKPG': 1,
'KPGR': 1,
'PGRE': 1,
'GREE': 1,
'REEF': 1,
'EEFE': 1,
'EFEG': 1,
'FEGG': 1,
'EGGS': 1,
'GGSP': 1,
'GSPD': 1,
'SPDW': 1,
'PDWX': 1,
'DWXN': 1,
'NYXX': 1,
'YXXS': 1,
'XXSV': 1,
'SVIE': 1,
'VIEF': 1,
'IEFO': 1,
'EFOZ': 1,
```

```
'FOZD': 1,
'OZDN': 1,
'ZDNO': 1,
'DNOY': 1,
'OYIU': 1,
'YIUS': 1,
'IUSD': 1,
'SDDI': 1,
'DDIG': 1,
'DIGS': 1,
'IGSW': 1,
'GSWS': 1,
'SWSP': 1,
'WSPS': 1,
'SPSO': 1,
'PSOG': 1,
'SOGY': 1,
'OGYF': 1,
'GYFO': 1,
'YFOV': 1,
'FOVN': 1,
'OVNO': 1,
'VNOY': 1,
'OYII': 1,
'YIIA': 1,
'IIAN': 1,
'IANB': 1,
'ANBP': 1,
'PFRY': 1,
'FRYS': 1,
'RYSO': 1,
'YSOJ': 1,
'OJXS': 1,
'SJJX': 1,
'JXIK': 1,
'IKIN': 1,
'KINZ': 1,
'INZO': 1,
'NZOF': 1,
'ZOFB': 1,
'OFBZ': 1,
'FBZF': 1,
'BZFF': 1,
'FFGM': 1,
'FGMR': 1,
'GMRI': 1,
'MRII': 1,
```

```
'RIIO': 1,
'OSOO': 1,
'SOOI': 1,
'OOIW': 1,
'IWSD': 1,
'WSDY': 1,
'SDYR': 1,
'DYRE': 1,
'RKID': 1,
'KIDU': 1,
'IDUS': 1,
'DUSE': 1,
'USEA': 1,
'SEAN': 1,
'EANI': 1,
'ANID': 1,
'NIDJ': 1,
'IDJG': 1,
'DJGS': 1,
'JGSP': 1,
'GSPF': 1,
'SPFB': 1,
'PFBY': 1,
'FBYF': 1,
'BYFR': 1,
'KDIP': 1,
'DIPF': 1,
'NEEU': 1,
'EEUF': 1,
'EUFF': 1,
'FFXU': 1,
'FXUF': 1,
'XUFF': 1,
'FFXW': 1,
'FXWX': 1,
'XWXS': 1,
'XSJI': 1,
'SJIV': 1,
'JIVI': 1,
'VIKD': 1,
'IKDB': 1,
'KDBK': 1,
'DBKI': 1,
'BKID': 1,
'KIDX': 1,
'IDXS': 1,
'DXSO': 1,
```

```
'SOGO': 1,
'OGOI': 1,
'GOIW': 1,
'IWSO': 1,
'WSOG': 1,
'OGGI': 1,
'GGIY': 1,
'GIYE': 1,
'IYES': 1,
'YESK': 1,
'ESKI': 1,
'SKIN': 1,
'KINJ': 1,
'INJD': 1,
'NJDY': 1,
'JDYK': 1,
'DYKR': 1,
'YKRG': 1,
'KRGI': 1,
'RGIS': 1,
'GISO': 1,
'SOGA': 1,
'OGAI': 1,
'GAIS': 1,
'ISOB': 1,
'SOBF': 1,
'OBFK': 1,
'FKSK': 1,
'KSKJ': 1,
'SKJD': 1,
'KJDJ': 1,
'JDJF': 1,
'DJFU': 1,
'JFUU': 1,
'FUUI': 1,
'UUIG': 1,
'UIGD': 1,
'IGDX': 1,
'GDXF': 1,
'DXFK': 1,
'XFKJ': 1,
'FKJN': 1,
'KJNO': 1,
'JNOJ': 1,
'JXIY': 1,
'XIYR': 1,
'IYRE': 1,
```

```
'JIGY': 1,
'IGYF': 1,
'GYFR': 1,
'FRKJ': 1,
'RKJF': 1,
'KJFB': 1,
'JFBJ': 1,
'JXII': 1,
'XIIA': 1,
'IIAU': 1,
'IAUK': 1,
'AUKI': 1,
'IDDH': 1,
'DDHF': 1,
'DHFD': 1,
'HFDI': 1,
'FDIU': 1,
'DIUX': 1,
'UXNO': 1,
'OISO': 1,
'BJXS': 1,
'XSJD': 1,
'SJDO': 1,
'JDOF': 1,
'DOFJ': 1,
'OFJZ': 1,
'FJZF': 1,
'JZFF': 1,
'BFKX': 1,
'FKXN': 1,
'KXNA': 1,
'XNAW': 1,
'NAWX': 1,
'AWXN': 1,
'NYXN': 1,
'YXND': 1,
'XNDZ': 1,
'NDZF': 1,
'DZFF': 1,
'IKPY': 1,
'KPYR': 1,
'PYRE': 1,
'JIGL': 1,
'IGLN': 1,
'GLNO': 1,
'RJXJ': 1,
'JXJX': 1,
```

```
'XJXN': 1,
'XNDL': 1,
'NDLN': 1,
'DLNO': 1,
'RJXA': 1,
'JXAR': 1,
'XARD': 1,
'JCIJ': 1,
'CIJX': 1,
'IJXI': 1,
'JXIO': 1,
'XIOS': 1,
'IOSD': 1,
'OSDI': 1,
'SDIO': 1,
'DIOJ': 1,
'OJNA': 1,
'JNAI': 1,
'NAIO': 1,
'JSEU': 1,
'SEUS': 1,
'EUSD': 1,
'SDDN': 1,
'DDNF': 1,
'DNFO': 1,
'NFOF': 1,
'FOFB': 1,
'OFBS': 1,
'FBSV': 1,
'BSVI': 1,
'CEIB': 1,
'EIBS': 1,
'IBSD': 1,
'BSDX': 1,
'SDXN': 1,
'DXNF': 1,
'XNFO': 1,
'NFOA': 1,
'FOAR': 1,
'RDJI': 1,
'DJIQ': 1,
'JIQY': 1,
'IQYN': 1,
'QYNJ': 1,
'YNJI': 1,
'NJIP': 1,
'JIPF': 1,
```

```
'FRKE': 1,
'RKES': 1,
'KESO': 1,
'ESOZ': 1,
'SOZR': 1,
'OZRN': 1,
'ZRNG': 1,
'RNGD': 1,
'NGDU': 1,
'GDUE': 1,
'DUEI': 1,
'UEII': 1,
'EIIO': 1,
'OSOS': 1,
'SOSJ': 1,
'OSJJ': 1,
'SJJS': 1,
'JJSY': 1,
'JSYX': 1,
'SYXA': 1,
'YXAI': 1,
'XAIO': 1,
'JSES': 1,
'SESU': 1,
'ESUE': 1,
'SUES': 1,
'UESJ': 1,
'ESJF': 1,
'SJFB': 1,
'JFBF': 1,
'FBFK': 1,
'FKSC': 1,
'KSCS': 1,
'SCSD': 1,
'CSDX': 1,
'SDXB': 1,
'DXBR': 1,
'XBRE': 1,
'BREP': 1,
'REPF': 1,
'EPFR': 1,
'ROZU': 1,
'OZUF': 1,
'ZUFJ': 1,
'UFJS': 1,
'FJSJ': 1,
'JSJF': 1,
```

```
'SJFF': 1,
'JFFK': 1,
'FFKS': 1,
'FKSO': 1,
'KSOF': 1,
'SOFJ': 1,
'OFJJ': 1,
'FJJF': 1,
'JJFF': 1,
'JFFB': 1,
'FFBK': 1,
'FBKI': 1,
'OYXB': 1,
'YXBK': 1,
'XBKI': 1,
'OYXC': 1,
'YXCI': 1,
'XCIS': 1,
'CISO': 1,
'ISOJ': 1,
'OJXF': 1,
'JXFR': 1,
'XFRZ': 1,
'RZXJ': 1,
'ZXJX': 1,
'XJXI': 1,
'JXIU': 1,
'XIUX': 1,
'UXNE': 1,
'XNEN': 1,
'NEND': 1,
'ENDJ': 1,
'NDJN': 1,
'DJNO': 1,
'JNOI': 1,
'NOID': 1,
'OIDA': 1,
'IDAS': 1,
'DASP': 1,
'ASPH': 1,
'SPHF': 1,
'PHFD': 1,
'HFDJ': 1,
'JEIP': 1,
'EIPF': 1,
'NEEK': 1,
'EEKS': 1,
```

```
'EKSO': 1,
'KSOL': 1,
'SOLS': 1,
'OLSD': 1,
'LSDS': 1,
'SDSO': 1,
'DSOS': 1,
'SOSU': 1,
'OSUF': 1,
'SUFD': 1,
'UFDJ': 1,
'JEIN': 1,
'EINO': 1,
'JNYC': 1,
'NYCS': 1,
'YCSO': 1,
'CSOG': 1,
'SOGN': 1,
'OGNB': 1,
'GNBP': 1,
'SELG': 1,
'ELGF': 1,
'LGFW': 1,
'GFWO': 1,
'FWOU': 1,
'WOUN': 1,
'OUNY': 1,
'UNYY': 1,
'NYYS': 1,
'YYSG': 1,
'YSGN': 1,
'SGNE': 1,
'GNEE': 1,
'NEEP': 1,
'EEPW': 1,
'EPWN': 1,
'PWNJ': 1,
'JXSU': 1,
'XSUF': 1,
'SUFU': 1,
'UFUU': 1,
'FUUP': 1,
'UUPF': 1,
'UPFK': 1,
'PFKS': 1,
'FKSE': 1,
'KSEN': 1,
```

```
'SENE': 1,
'ENEP': 1,
'NEPN': 1,
'EPNO': 1,
'PNOP': 1,
'NOPF': 1,
'OPFR': 1,
'RKAI': 1,
'KAIG': 1,
'AIGN': 1,
'IGNI': 1,
'GNIV': 1,
'NIVS': 1,
'IVSE': 1,
'VSEX': 1,
'SEXS': 1,
'EXSO': 1,
'SOGS': 1,
'FRKB': 1,
'RKBE': 1,
'KBEF': 1,
'BEFW': 1,
'EFWK': 1,
'FWKP': 1,
'WKPW': 1,
'KPWS': 1,
'PWSP': 1,
'PNBX': 1,
'NBXI': 1,
'BXID': 1,
'XIDY': 1,
'IDYF': 1,
'DYFO': 1,
'YFOJ': 1,
'FOJI': 1,
'OJIO': 1,
'JIOJ': 1,
'IOJW': 1,
'OJWN': 1,
'JWNJ': 1,
'JXSV': 1,
'CEIE': 1,
'EIEF': 1,
'IEFV': 1,
'EFVI': 1,
'FVIW': 1,
'VIWX': 1,
```

```
'IWXN': 1,
'NYXW': 1,
'YXWF': 1,
'XWFR': 1,
'WFRE': 1,
'FREG': 1,
'REGY': 1,
'EGYI': 1,
'GYIK': 1,
'YIKJ': 1,
'IKJS': 1,
'KJSN': 1,
'JSNO': 1,
'SNOE': 1,
'NOEP': 1,
'OEPO': 1,
'EPOF': 1,
'POFJ': 1,
'OFJD': 1,
'FJDR': 1,
'JDRN': 1,
'DRNJ': 1,
'RNJA': 1,
'NJAI': 1,
'JAIW': 1,
'SJSA': 1,
'JSAF': 1,
'SAFD': 1,
'AFDJ': 1,
'FDJU': 1,
'DJUS': 1,
'JUSK': 1,
'USKJ': 1,
'SKJN': 1,
'KJNY': 1,
'JNYR': 1,
'NYRE': 1,
'YRES': 1,
'KEPU': 1,
'EPUR': 1,
'PURK': 1,
'XNDU': 1,
'NDUR': 1,
'DURK': 1}
```

We know that the most common letter in English is e. Also we know that the most common bigram is th and the most common trigram is the, so using this we can suppose that I->e so JXI->the

JX->th J->t X->h. Also t is the second most common letter in English and the frequence of J is high, so it makes sense. Also a common word in English is that and a is the third most common letter in English, so we look for quadgrams that starts by JX and finnish by J:

[22]:
```
import re
quadgramsJX = {}
pattern = r'JX[A-Z]J'
matches = re.findall(pattern, message_no_space)
for q in matches:
    quadgramsJX[q]=quadgrams[q]
quadgramsJX
```

[22]:
```
{'JXIJ': 3, 'JXSJ': 2}
```

Since S has a high frequence it makes sense that S->a. We are going to do this sustituions in the text and look what we have:

[23]:
```
message1 = ''
for char in message:
    if char == 'J':
        message1 += 't'
    elif char == 'X':
        message1+= 'h'
    elif char == 'I':
        message1 += 'e'
    elif char == 'S':
        message1 += 'a'
    else:
        message1 += char
message1
```

[23]:
```
'NBPFR KeaOQ NFRDB FKtFD hNOeN Otheh NZhae Dthet NYENO eaDaA aOFBY REtRK eKaKe
PFRAR DtZet RUaEE theZe KADFB thetK aODYe OGeOt aEteK ADaOG UEaOt theAe VKPWh
eKePF RARDt ENeRU FOthe GaNDN eDaOG GNDYF RKDeN OOFVe EUhKa DeDFB PFRKY FAUEN
YateG Dtate FBANO GtheA eaONO ZGFeD OtAat teKNB NtDFO EPNGE eYhat teKFB atKaO
DYeOG eOtaE LNOGa OGeVK PFOeW NEEDa PaDPF RWaEL PFRKA PDtNY WaPNB thNDP FROZA
aOeQU KeDDe DhNAD eEBNO teKAD tFFGe eUBFK AeWhP Whata VeKPD NOZRE aKEPG eeUPF
ROZAa OthND GeeUP FROZA aOARD tCeCe eEFMR eOtNO UKaND eFBth eVeKP GREEF EGGaP
DWhNY hhaVe EFOZD NOYeU aDDeG aWaPa OGYFO VNOYe eANBP FRYaO thatt heKeN ZOFBZ
FFGMR eeOaO OeWaD YREtR KeDUa EANeD tGaPF BYFRK DePFR WNEEU FFhUF FhWha teVeK
DBKeD haOGO eWaOG GeYEa KeNtD YKRGe aOGAe aOBFK aKtDt FUUeG DhFKt NOthe YREtN
VateG YFRKt FBthe eAUKe DDHFD eUhNO eaOGe VKPFO eWNEE DaPaD PFRWa ELPFR KAPDt
NYWaP NBtha tDOFt ZFFGe OFRZh BFKhN AWhNY hNDZF FGeOF RZhBF KAeWh PWhat aVeKP
YREtN VateG LNOGF BPFRt hthND LNOGF BPFRt hARDt Cethe OaDeO tNAeO taEUa DDNFO
FBaVe ZetaC EeBaD hNFOA RDteQ YNteP FRKEa OZRNG DUEee OaOat taYhA eOtaE aUEat
FBFKa CaDhB REPFR OZUFt atFFK aOFtt FFBKe OYhBK eOYhC eaOth FRZht heUhN ENDtN
OeDAa PHFDt EePFR WNEEK aOLaD aOaUF DtEeN Otheh NZhae Dthet NYCaO GNBPF RWaEL
GFWOU NYYaG NEEPW NthaU FUUPF KaENE PNOPF RKAeG NeVaE haOGa OGeVK PFOeW NEEDa
```

```
PaDPF RWaEL PFRKB EFWKP WaPNB heDYF OteOt WNtha VeZet aCEeE FVeWh NYhWF REGYe
KtaNO EPOFt DRNtA eWhPW hataA FDtUa KtNYR EaKEP URKeP FROZA aOthN DURKe PFROZ
AaOAR DtCe'
```

In English, two commons digraphs are an and nt, so we are going to look for this pattern. Also, a common trigram is ent:

[24]:
```
bi = {}
pattern1 = r'[A-Z]J'
pattern2 = r'S[A-Z]'
matches = re.findall(pattern1, message_no_space)
for b in matches:
    bi[b] = bigrams[b]
matches = re.findall(pattern2, message_no_space)
for b in matches:
    bi[b] = bigrams[b]
dict(sorted(bi.items(), key=lambda x: x[1], reverse=True))
```

[24]:
```
{'SO': 31,
 'OJ': 19,
 'DJ': 18,
 'SJ': 16,
 'SD': 12,
 'SE': 12,
 'SP': 10,
 'IJ': 7,
 'KJ': 6,
 'EJ': 6,
 'NJ': 6,
 'SK': 6,
 'BJ': 5,
 'SV': 5,
 'FJ': 4,
 'SN': 3,
 'SC': 3,
 'SU': 3,
 'RJ': 2,
 'XJ': 2,
 'SI': 2,
 'SA': 2,
 'GJ': 1,
 'SW': 1,
 'SY': 1,
 'SG': 1}
```

[25]:
```
tri = {}
pattern = r'I[A-Z]J'
```

```
matches = re.findall(pattern, message_no_space)
for t in matches:
    tri[t] = trigrams[t]
dict(sorted(tri.items(), key=lambda x: x[1], reverse=True))
```

[25]:
```
{'IOJ': 7, 'IDJ': 3, 'INJ': 1, 'IKJ': 1}
```

Look that SO=aO and OJ=Ot are the most common ones and IOJ=eOt is also very common, so we can suppose that O->n. On the other hand, another usual digraphs are st and as so, since SD=aD and DJ=Dt have high frequence, I am going to suppose that D->s

Let's do this changes in our text:

[26]:
```
message2 = ''
for char in message1:
    if char == 'O':
        message2 += 'n'
    elif char == 'D':
        message2 += 's'
    else:
        message2 += char
message2
```

[26]:
```
'NBPFR KeanQ NFRsB FKtFs hNneN ntheh NZhae sthet NYENn easaA anFBY REtRK eKaKe
PFRAR stZet RUaEE theZe KAsFB thetK ansYe nGent aEteK AsanG UEant theAe VKPWh
eKePF RARst ENeRU Fnthe GaNsN esanG GNsYF RKseN nnFVe EUhKa sesFB PFRKY FAUEN
YateG state FBANn GtheA eanNn ZGFes ntAat teKNB NtsFn EPNGE eYhat teKFB atKan
sYenG entaE LNnGa nGeVK PFneW NEEsa PasPF RWaEL PFRKA PstNY WaPNB thNsP FRnZA
aneQU Kesse shNAs eEBNn teKAs tFFGe eUBFK AeWhP Whata VeKPs NnZRE aKEPG eeUPF
RnZAa nthNs GeeUP FRnZA anARs tCeCe eEFMR entNn UKaNs eFBth eVeKP GREEF EGGaP
sWhNY hhaVe EFnZs NnYeU asseG aWaPa nGYFn VNnYe eANBP FRYan thatt heKeN ZnFBZ
FFGMR eenan neWas YREtR KesUa EANes tGaPF BYFRK sePFR WNEEU FFhUF FhWha teVeK
sBKes hanGn eWanG GeYEa KeNts YKRGe anGAe anBFK aKtst FUUeG shFKt Nnthe YREtN
VateG YFRKt FBthe eAUKe ssHFs eUhNn eanGe VKPFn eWNEE saPas PFRWa ELPFR KAPst
NYWaP NBtha tsnFt ZFFGe nFRZh BFKhN AWhNY hNsZF FGenF RZhBF KAeWh PWhat aVeKP
YREtN VateG LNnGF BPFRt hthNs LNnGF BPFRt hARst Cethe nasen tNAen taEUa ssNFn
FBaVe ZetaC EeBas hNFnA RsteQ YNteP FRKEa nZRNG sUEee nanat taYhA entaE aUEat
FBFKa CashB REPFR nZUFt atFFK anFtt FFBKe nYhBK enYhC eanth FRZht heUhN ENstN
nesAa PHFst EePFR WNEEK anLas anaUF stEeN ntheh NZhae sthet NYCan GNBPF RWaEL
GFWnU NYYaG NEEPW NthaU FUUPF KaENE PNnPF RKAeG NeVaE hanGa nGeVK PFneW NEEsa
PasPF RWaEL PFRKB EFWKP WaPNB hesYF ntent WNtha VeZet aCEeE FVeWh NYhWF REGYe
KtaNn EPnFt sRNtA eWhPW hataA FstUa KtNYR EaKEP URKeP FRnZA anthN sURKe PFRnZ
AanAR stCe'
```

The 4th most common letter in English is o. Also, common digrams in English are on and to. And common word are so, no, to... So we are going to look up for this kind of digraphs:

[27]:
```
bi = {}
pattern = r'[A-Z]O'
matches = re.findall(pattern, message_no_space)
for b in matches:
    if 'S' not in b and 'X' not in b and 'I' not in b:
        bi[b] = bigrams[b]
pattern = r'J[A-Z]'
matches = re.findall(pattern, message_no_space)
for b in matches:
    if 'S' not in b and 'X' not in b and 'I' not in b:
        bi[b] = bigrams[b]
dict(sorted(bi.items(), key=lambda x: x[1], reverse=True))
```

[27]:
```
{'NO': 20,
 'JN': 11,
 'FO': 10,
 'JF': 7,
 'RO': 6,
 'JJ': 6,
 'JD': 5,
 'JR': 3,
 'JE': 3,
 'JC': 3,
 'DO': 2,
 'JZ': 2,
 'JK': 2,
 'JA': 2,
 'ZO': 1,
 'GO': 1,
 'WO': 1,
 'PO': 1,
 'JG': 1,
 'JW': 1,
 'JU': 1}
```

We cannot suppose that N is O because i is also a common letter and NO=in and JN=ti. So looking at this, the point is if N is i or N is o. In that case, looking at the frequence of digraphs, the other letter should be F. Since we know that ion and tio are common trigraphs, let's try to make a more accurate assumption. Suppose that o is N, so i is F, then:

[28]:
```
print(trigrams['JFN']) # tio
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_18370/2191245834.py in <module>
----> 1 print(trigrams['JFN']) # tio
```

```
KeyError: 'JFN'
```

[29]:
```
print(trigrams['FNO']) # ion
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_18370/3178128955.py in <module>
----> 1 print(trigrams['FNO']) # ion

KeyError: 'FNO'
```

Suppose the inverse case:

[30]:
```
print(trigrams['JNF']) # tio
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_18370/2592903843.py in <module>
----> 1 print(trigrams['JNF']) # tio

KeyError: 'JNF'
```

[31]:
```
print(trigrams['NFO']) # ion
```

```
2
```

So we are going to suppose that N->i and F->o. So our text looks now as follows:

[32]:
```
message3 = ''
for char in message2:
    if char == 'N':
        message3 += 'i'
    elif char == 'F':
        message3 += 'o'
    else:
        message3 += char
message3
```

[32]:
```
'iBPoR KeanQ ioRsB oKtos hinei ntheh iZhae sthet iYEin easaA anoBY REtRK eKaKe
PoRAR stZet RUaEE theZe KAsoB thetK ansYe nGent aEteK AsanG UEant theAe VKPWh
eKePo RARst EieRU onthe Gaisi esanG GisYo RKsei nnoVe EUhKa sesoB PoRKY oAUEi
YateG state oBAin GtheA eanin ZGoes ntAat teKiB itson EPiGE eYhat teKoB atKan
sYenG entaE LinGa nGeVK PoneW iEEsa PasPo RWaEL PoRKA PstiY WaPiB thisP oRnZA
aneQU Kesse shiAs eEBin teKAs tooGe eUBoK AeWhP Whata VeKPs inZRE aKEPG eeUPo
RnZAa nthis GeeUP oRnZA anARs tCeCe eEoMR entin UKais eoBth eVeKP GREEo EGGaP
sWhiY hhaVe EonZs inYeU asseG aWaPa nGYon VinYe eAiBP oRYan thatt heKei ZnoBZ
```

```
ooGMR eenan neWas YREtR KesUa EAies tGaPo BYoRK sePoR WiEEU oohUo ohWha teVeK
sBKes hanGn eWanG GeYEa Keits YKRGe anGAe anBoK aKtst oUUeG shoKt inthe YREti
VateG YoRKt oBthe eAUKe ssHos eUhin eanGe VKPon eWiEE saPas PoRWa ELPoR KAPst
iYWaP iBtha tsnot ZooGe noRZh BoKhi AWhiY hisZo oGeno RZhBo KAeWh PWhat aVeKP
YREti VateG LinGo BPoRt hthis LinGo BPoRt hARst Cethe nasen tiAen taEUa ssion
oBaVe ZetaC EeBas hionA RsteQ YiteP oRKEa nZRiG sUEee nanat taYhA entaE aUEat
oBoKa CashB REPoR nZUot atooK anott ooBKe nYhBK enYhC eanth oRZht heUhi Eisti
nesAa PHost EePoR WiEEK anLas anaUo stEei ntheh iZhae sthet iYCan GiBPo RWaEL
GoWnU iYYaG iEEPW ithaU oUUPo KaEiE PinPo RKAeG ieVaE hanGa nGeVK PoneW iEEsa
PasPo RWaEL PoRKB EoWKP WaPiB hesYo ntent Witha VeZet aCEeE oVeWh iYhWo REGYe
Ktain EPnot sRitA eWhPW hataA ostUa KtiYR EaKEP URKeP oRnZA anthi sURKe PoRnZ
AanAR stCe'
```

Another usual trigraphs in English are and and nde:

[33]:
```
tri = {}
pattern = r'SO[A-Z]'
matches = re.findall(pattern, message_no_space)
for t in matches:
    tri[t] = trigrams[t]
pattern = r'O[A-Z]I'
matches = re.findall(pattern, message_no_space)
for t in matches:
    tri[t] = trigrams[t]
dict(sorted(tri.items(), key=lambda x: x[1], reverse=True))
```

[33]:
```
{'SOG': 11,
 'SOJ': 5,
 'OGI': 5,
 'SOF': 2,
 'SOD': 2,
 'SOA': 2,
 'SOS': 2,
 'OJI': 2,
 'OYI': 2,
 'SOQ': 1,
 'SON': 1,
 'SOI': 1,
 'SOO': 1,
 'SOB': 1,
 'SOZ': 1,
 'SOL': 1,
 'OOI': 1}
```

So we get that d must be G.

[34]:
```
message4 = ''
for char in message3:
    if char == 'G':
        message4 += 'd'
    else:
        message4 += char
message4
```

[34]:
```
'iBPoR KeanQ ioRsB oKtos hinei ntheh iZhae sthet iYEin easaA anoBY REtRK eKaKe
PoRAR stZet RUaEE theZe KAsoB thetK ansYe ndent aEteK Asand UEant theAe VKPWh
eKePo RARst EieRU onthe daisi esand disYo RKsei nnoVe EUhKa sesoB PoRKY oAUEi
Yated state oBAin dtheA eanin Zdoes ntAat teKiB itson EPidE eYhat teKoB atKan
sYend entaE Linda ndeVK PoneW iEEsa PasPo RWaEL PoRKA PstiY WaPiB thisP oRnZA
aneQU Kesse shiAs eEBin teKAs toode eUBoK AeWhP Whata VeKPs inZRE aKEPd eeUPo
RnZAa nthis deeUP oRnZA anARs tCeCe eEoMR entin UKais eoBth eVeKP dREEo EddaP
sWhiY hhaVe EonZs inYeU assed aWaPa ndYon VinYe eAiBP oRYan thatt heKei ZnoBZ
oodMR eenan neWas YREtR KesUa EAies tdaPo BYoRK sePoR WiEEU oohUo ohWha teVeK
sBKes handn eWand deYEa Keits YKRde andAe anBoK aKtst oUUed shoKt inthe YREti
Vated YoRKt oBthe eAUKe ssHos eUhin eande VKPon eWiEE saPas PoRWa ELPoR KAPst
iYWaP iBtha tsnot Zoode noRZh BoKhi AWhiY hisZo odeno RZhBo KAeWh PWhat aVeKP
YREti Vated Lindo BPoRt hthis Lindo BPoRt hARst Cethe nasen tiAen taEUa ssion
oBaVe ZetaC EeBas hionA RsteQ YiteP oRKEa nZRid sUEee nanat taYhA entaE aUEat
oBoKa CashB REPoR nZUot atooK anott ooBKe nYhBK enYhC eanth oRZht heUhi Eisti
nesAa PHost EePoR WiEEK anLas anaUo stEei ntheh iZhae sthet iYCan diBPo RWaEL
doWnU iYYad iEEPW ithaU oUUPo KaEiE PinPo RKAed ieVaE handa ndeVK PoneW iEEsa
PasPo RWaEL PoRKB EoWKP WaPiB hesYo ntent Witha VeZet aCEeE oVeWh iYhWo REdYe
Ktain EPnot sRitA eWhPW hataA ostUa KtiYR EaKEP URKeP oRnZA anthi sURKe PoRnZ
AanAR stCe'
```

In the first line we have tos hinei ntheh iZhae sthet iYEin easaA. Ordering it we have to shine in the hiZh aesthetiY EneasaA... So it looks that Z->g and Y->c. So we have that our text is:

[35]:
```
message5 = ''
for char in message4:
    if char == 'Z':
        message5 += 'g'
    elif char == 'Y':
        message5 += 'c'
    else:
        message5 += char
message5
```

[35]:
```
'iBPoR KeanQ ioRsB oKtos hinei ntheh ighae sthet icEin easaA anoBc REtRK eKaKe
PoRAR stget RUaEE thege KAsoB thetK ansce ndent aEteK Asand UEant theAe VKPWh
eKePo RARst EieRU onthe daisi esand disco RKsei nnoVe EUhKa sesoB PoRKc oAUEi
cated state oBAin dtheA eanin gdoes ntAat teKiB itson EPidE echat teKoB atKan
scend entaE Linda ndeVK PoneW iEEsa PasPo RWaEL PoRKA Pstic WaPiB thisP oRngA
```

```
aneQU Kesse shiAs eEBin teKAs toode eUBoK AeWhP Whata VeKPs ingRE aKEPd eeUPo
RngAa nthis deeUP oRngA anARs tCeCe eEoMR entin UKais eoBth eVeKP dREEo EddaP
sWhic hhaVe Eongs inceU assed aWaPa ndcon Vince eAiBP oRcan thatt heKei gnoBg
oodMR eenan neWas cREtR KesUa EAies tdaPo BcoRK sePoR WiEEU oohUo ohWha teVeK
sBKes handn eWand decEa Keits cKRde andAe anBoK aKtst oUUed shoKt inthe cREti
Vated coRKt oBthe eAUKe ssHos eUhin eande VKPon eWiEE saPas PoRWa ELPoR KAPst
icWaP iBtha tsnot goode noRgh BoKhi AWhic hisgo odeno RghBo KAeWh PWhat aVeKP
cREti Vated Lindo BPoRt hthis Lindo BPoRt hARst Cethe nasen tiAen taEUa ssion
oBaVe getaC EeBas hionA RsteQ citeP oRKEa ngRid sUEee nanat tachA entaE aUEat
oBoKa CashB REPoR ngUot atooK anott ooBKe nchBK enchC eanth oRght heUhi Eisti
nesAa PHost EePoR WiEEK anLas anaUo stEei ntheh ighae sthet icCan diBPo RWaEL
doWnU iccad iEEPW ithaU oUUPo KaEiE PinPo RKAed ieVaE handa ndeVK PoneW iEEsa
PasPo RWaEL PoRKB EoWKP WaPiB hesco ntent Witha Veget aCEeE oVeWh ichWo REdce
Ktain EPnot sRitA eWhPW hataA ostUa KticR EaKEP URKeP oRngA anthi sURKe PoRng
AanAR stCe'
```

Also, a common trigraphs in English is men:

[36]:
```
tri = {}
pattern = r'[A-Z]IO'
matches = re.findall(pattern, message_no_space)
for t in matches:
    tri[t] = trigrams[t]
dict(sorted(tri.items(), key=lambda x: x[1], reverse=True))
```

[36]:
```
{'GIO': 4,
 'YIO': 2,
 'IIO': 2,
 'AIO': 2,
 'KIO': 2,
 'RIO': 1,
 'XIO': 1,
 'DIO': 1,
 'JIO': 1}
```

It seems is not so useful since m could be A, K..., but if we look at 4th line we have state oBAin dtheA eanin gdoes ntAat teKiB. Reordering we have that state oB Aind the Aeaning doesnt AatteK. If we change B->f A->m and K->r we have state of mind the meaning doesnt matter, what looks really good.

Let's try to make this changes on the text:

[37]:
```
message6 = ''
for char in message5:
    if char == 'B':
        message6 += 'f'
    elif char == 'A':
        message6 += 'm'
```

```
    elif char == 'K':
        message6 += 'r'
    else:
        message6 += char
message6
```

[37]:
```
'ifPoR reanQ ioRsf ortos hinei ntheh ighae sthet icEin easam anofc REtRr erare
PoRmR stget RUaEE thege rmsof thetr ansce ndent aEter msand UEant theme VrPWh
erePo RmRst EieRU onthe daisi esand disco Rrsei nnoVe EUhra sesof PoRrc omUEi
cated state ofmin dthem eanin gdoes ntmat terif itson EPidE echat terof atran
scend entaE Linda ndeVr PoneW iEEsa PasPo RWaEL PoRrm Pstic WaPif thisP oRngm
aneQU resse shims eEfin terms toode eUfor meWhP Whata VerPs ingRE arEPd eeUPo
Rngma nthis deeUP oRngm anmRs tCeCe eEoMR entin Urais eofth eVerP dREEo EddaP
sWhic hhaVe Eongs inceU assed aWaPa ndcon Vince emifP oRcan thatt herei gnofg
oodMR eenan neWas cREtR resUa Emies tdaPo fcoRr sePoR WiEEU oohUo ohWha teVer
sfres handn eWand decEa reits crRde andme anfor artst oUUed short inthe cREti
Vated coRrt ofthe emUre ssHos eUhin eande VrPon eWiEE saPas PoRWa ELPoR rmPst
icWaP iftha tsnot goode noRgh forhi mWhic hisgo odeno Rghfo rmeWh PWhat aVerP
cREti Vated Lindo fPoRt hthis Lindo fPoRt hmRst Cethe nasen timen taEUa ssion
ofaVe getaC Eefas hionm RsteQ citeP oRrEa ngRid sUEee nanat tachm entaE aUEat
ofora Cashf REPoR ngUot atoor anott oofre nchfr enchC eanth oRght heUhi Eisti
nesma PHost EePoR WiEEr anLas anaUo stEei ntheh ighae sthet icCan difPo RWaEL
doWnU iccad iEEPW ithaU oUUPo raEiE PinPo Rrmed ieVaE handa ndeVr PoneW iEEsa
PasPo RWaEL PoRrf EoWrP WaPif hesco ntent Witha Veget aCEeE oVeWh ichWo REdce
rtain EPnot sRitm eWhPW hatam ostUa rticR EarEP URreP oRngm anthi sURre PoRng
manmR stCe'
```

In the first line we had to shine in the hiZh aesthetiY EneasaA... with the new changes we have to shine in the high asthetic Ein easam anofc REtRr erare. It seems to be to shine in the high asthetic line as a man of cREtRr erare. So we change E->l and that implies we have the high asthetic line as a man of cRltRr erare so R should be u to form culture.

[38]:
```
message7 = ''
for char in message6:
    if char == 'E':
        message7 += 'l'
    elif char == 'R':
        message7 += 'u'
    else:
        message7 += char
message7
```

[38]:
```
'ifPou reanQ iousf ortos hinei ntheh ighae sthet iclin easam anofc ultur erare
Poumu stget uUall thege rmsof thetr ansce ndent alter msand Ulant theme VrPWh
erePo umust lieuU onthe daisi esand disco ursei nnoVe lUhra sesof Pourc omUli
cated state ofmin dthem eanin gdoes ntmat terif itson lPidl echat terof atran
scend ental Linda ndeVr PoneW illsa PasPo uWalL Pourm Pstic WaPif thisP oungm
```

```
aneQU resse shims elfin terms toode eUfor meWhP Whata VerPs ingul arlPd eeUPo
ungma nthis deeUP oungm anmus tCeCe eloMu entin Urais eofth eVerP dullo lddaP
sWhic hhaVe longs inceU assed aWaPa ndcon Vince emifP oucan thatt herei gnofg
oodMu eenan neWas cultu resUa lmies tdaPo fcour sePou WillU oohUo ohWha teVer
sfres handn eWand decla reits crude andme anfor artst oUUed short inthe culti
Vated court ofthe emUre ssHos eUhin eande VrPon eWill saPas PouWa lLPou rmPst
icWaP iftha tsnot goode nough forhi mWhic hisgo odeno ughfo rmeWh PWhat aVerP
culti Vated Lindo fPout hthis Lindo fPout hmust Cethe nasen timen talUa ssion
ofaVe getaC lefas hionm usteQ citeP ourla nguid sUlee nanat tachm ental aUlat
ofora Cashf ulPou ngUot atoor anott oofre nchfr enchC eanth ought heUhi listi
nesma PHost lePou Willr anLas anaUo stlei ntheh ighae sthet icCan difPo uWalL
doWnU iccad illPW ithaU oUUPo ralil PinPo urmed ieVal handa ndeVr PoneW illsa
PasPo uWalL Pourf loWrP WaPif hesco ntent Witha Veget aClel oVeWh ichWo uldce
rtain lPnot suitm eWhPW hatam ostUa rticu larlP UureP oungm anthi sUure Poung
manmu stCe'
```

Now, in the first line we have ifPou reanQ iousf ortos hinei that seems to be if Pou re anQious for to... Change P->y and Q->x and you get you're anxious. Also, since z is the most uncommon letter in English, we suppose T is z (it does not appear in the text)

[39]:
```
message8 = ''
for char in message7:
    if char == 'P':
        message8 += 'y'
    elif char == 'Q':
        message8 += 'x'
    else:
        message8 += char
message8
```

[39]:
```
'ifyou reanx iousf ortos hinei ntheh ighae sthet iclin easam anofc ultur erare
youmu stget uUall thege rmsof thetr ansce ndent alter msand Ulant theme VryWh
ereyo umust lieuU onthe daisi esand disco ursei nnoVe lUhra sesof yourc omUli
cated state ofmin dthem eanin gdoes ntmat terif itson lyidl echat terof atran
scend ental Linda ndeVr yoneW illsa yasyo uWalL yourm ystic Wayif thisy oungm
anexU resse shims elfin terms toode eUfor meWhy Whata Verys ingul arlyd eeUyo
ungma nthis deeUy oungm anmus tCeCe eloMu entin Urais eofth eVery dullo ldday
sWhic hhaVe longs inceU assed aWaya ndcon Vince emify oucan thatt herei gnofg
oodMu eenan neWas cultu resUa lmies tdayo fcour seyou WillU oohUo ohWha teVer
sfres handn eWand decla reits crude andme anfor artst oUUed short inthe culti
Vated court ofthe emUre ssHos eUhin eande Vryon eWill sayas youWa lLyou rmyst
icWay iftha tsnot goode nough forhi mWhic hisgo odeno ughfo rmeWh yWhat aVery
culti Vated Lindo fyout hthis Lindo fyout hmust Cethe nasen timen talUa ssion
ofaVe getaC lefas hionm ustex citey ourla nguid sUlee nanat tachm ental aUlat
ofora Cashf ulyou ngUot atoor anott oofre nchfr enchC eanth ought heUhi listi
nesma yHost leyou Willr anLas anaUo stlei ntheh ighae sthet icCan difyo uWalL
doWnU iccad illyW ithaU oUUyo ralil yinyo urmed ieVal handa ndeVr yoneW illsa
```

```
yasyo uWalL yourf loWry Wayif hesco ntent Witha Veget aClel oVeWh ichWo uldce
rtain lynot suitm eWhyW hatam ostUa rticu larly Uurey oungm anthi sUure young
manmu stCe'
```

In the second line we have you must get uU all the germs of... so U->p and, with this change,in the last line we have this pure young man must Ce so C->b.

[40]:
```
message9 = ''
for char in message8:
    if char == 'C':
        message9 += 'b'
    elif char == 'U':
        message9 += 'p'
    else:
        message9 += char
message9
```

[40]:
```
'ifyou reanx iousf ortos hinei ntheh ighae sthet iclin easam anofc ultur erare
youmu stget upall thege rmsof thetr ansce ndent alter msand plant theme VryWh
ereyo umust lieup onthe daisi esand disco ursei nnoVe lphra sesof yourc ompli
cated state ofmin dthem eanin gdoes ntmat terif itson lyidl echat terof atran
scend ental Linda ndeVr yoneW illsa yasyo uWalL yourm ystic Wayif thisy oungm
anexp resse shims elfin terms toode epfor meWhy Whata Verys ingul arlyd eepyo
ungma nthis deepy oungm anmus tbebe eloMu entin prais eofth eVery dullo ldday
sWhic hhaVe longs incep assed aWaya ndcon Vince emify oucan thatt herei gnofg
oodMu eenan neWas cultu respa lmies tdayo fcour seyou Willp oohpo ohWha teVer
sfres handn eWand decla reits crude andme anfor artst opped short inthe culti
Vated court ofthe empre ssHos ephin eande Vryon eWill sayas youWa lLyou rmyst
icWay iftha tsnot goode nough forhi mWhic hisgo odeno ughfo rmeWh yWhat aVery
culti Vated Lindo fyout hthis Lindo fyout hmust bethe nasen timen talpa ssion
ofaVe getab lefas hionm ustex citey ourla nguid splee nanat tachm ental aplat
ofora bashf ulyou ngpot atoor anott oofre nchfr enchb eanth ought hephi listi
nesma yHost leyou Willr anLas anapo stlei ntheh ighae sthet icban difyo uWalL
doWnp iccad illyW ithap oppyo ralil yinyo urmed ieVal handa ndeVr yoneW illsa
yasyo uWalL yourf loWry Wayif hesco ntent Witha Veget ablel oVeWh ichWo uldce
rtain lynot suitm eWhyW hatam ostpa rticu larly purey oungm anthi spure young
manmu stbe'
```

Analyzing the text, we see that H->j L->k V->v and W->w so we get:

[41]:
```
result = ''
for char in message9:
    if char == 'H':
        result += 'j'
    elif char == 'L':
        result += 'k'
    elif char == 'V':
        result += 'v'
```

```
    elif char == 'W':
        result += 'w'
    else:
        result += char
result
```

[41]:
```
'ifyou reanx iousf ortos hinei ntheh ighae sthet iclin easam anofc ultur erare
youmu stget upall thege rmsof thetr ansce ndent alter msand plant theme vrywh
ereyo umust lieup onthe daisi esand disco ursei nnove lphra sesof yourc ompli
cated state ofmin dthem eanin gdoes ntmat terif itson lyidl echat terof atran
scend ental kinda ndevr yonew illsa yasyo uwalk yourm ystic wayif thisy oungm
anexp resse shims elfin terms toode epfor mewhy whata verys ingul arlyd eepyo
ungma nthis deepy oungm anmus tbebe eloMu entin prais eofth every dullo ldday
swhic hhave longs incep assed awaya ndcon vince emify oucan thatt herei gnofg
oodMu eenan newas cultu respa lmies tdayo fcour seyou willp oohpo ohwha tever
sfres handn ewand decla reits crude andme anfor artst opped short inthe culti
vated court ofthe empre ssjos ephin eande vryon ewill sayas youwa lkyou rmyst
icway iftha tsnot goode nough forhi mwhic hisgo odeno ughfo rmewh ywhat avery
culti vated kindo fyout hthis kindo fyout hmust bethe nasen timen talpa ssion
ofave getab lefas hionm ustex citey ourla nguid splee nanat tachm ental aplat
ofora bashf ulyou ngpot atoor anott oofre nchfr enchb eanth ought hephi listi
nesma yjost leyou willr ankas anapo stlei ntheh ighae sthet icban difyo uwalk
downp iccad illyw ithap oppyo ralil yinyo urmed ieval handa ndevr yonew illsa
yasyo uwalk yourf lowry wayif hesco ntent witha veget ablel ovewh ichwo uldce
rtain lynot suitm ewhyw hatam ostpa rticu larly purey oungm anthi spure young
manmu stbe'
```

Using the correct spaces we get:

If you're anxious for to shine in the high aesthetic line as a man of culture rare you must get up all the germs of the transcendent alterms and plant them evrywhere you must lie up on the daisies and discourse in novel phrases of your coplicated state of mind the meaning doesn't matter if its only idle chatter of a transcendental mind and evryone will say as you walk your mystic way if this young man expresses himself in terms too deep for me why what a very singularly deep young man this deep young man must be be eloquent in praise of the very dull old days which have long since passed away and convince em if you can that the reign of good queen anne was cultures palmiest day of course you will pooh pooh whatevers fresh and new and declare its crude and mean for art stopped short in the cultivated court of the empress josephine and evryone will say as you walm your mystic way if thats not good enough for him which is good enough for me why what a very cultivated mind of youth this mind of youth must be then a sentimental passion of a vegetable fashion must excite your languid spleen an attachmental a plato for a bashful young potato or a not too french french bean thought he philistines may jostle you will rank as an apostle in the high aesthetic band if you walk down piccadilly with a poppy or a lily in your medieval hand and evryone will say as you walk your flowry way if hes content with a vegetable love which would certainly not suit me why what a most particularly pure young man this pure young man must be

0.4 Exercise 4

[42]:
```
message = 'TYJNI WXWXZ NIGXN IWBMM XTDWN ZJXBF XBFBR XBEVT NOWND WBOFI VYNZT␣
 ↪WWNDX NHBFN IUWTO GHNZM FWXBW BRXTM FRBOW XTOKB OFVND DTUMJ FNTWV ZBRWT␣
 ↪RBMMJ JNIHN IMFOW RNODW BOWMJ ZIOBR ZNDDY NMKDW NFBJH XNRMB TEWXB WBRXT␣
 ↪MFFNO WKONH BOJWX TOGBR XTMFD UZBTO DWBZW DYIOR WTNOT OGBWU TZWXB OFXBD␣
 ↪BENOG DWTWD EBOJT OYBOW RNOSN MIWTN ODWXN IDBOF DNYFN ZEBOW BWNED TOWNH␣
 ↪XTRXG NFXBD VIWBE JDWTR VNDDT UTMTW JYNZO NWTRT OGBOB FIMWD BRWBO FYTGI␣
 ↪ZTOGN IWTWD VIZVN ZWIVW NBUNI WTWDV ZTEBZ JDRXN NMFBJ DBRXT MFWXT OKDOB␣
 ↪WIZBM MJNOM JNYVM BJUIW EBOJB YNZEN YVMBJ RNOWB TODFT DRTVM TOBZJ YBRWN␣
 ↪ZDJNI RBOWF NWXTD NZWXB WVIWD JNINI WDXNH DBRXT MFWXB WTWEI DWWXT OKVZB␣
 ↪RWTRB MMJNZ YBTMO NHTYW XZNIG XNIWR XTMFX NNFBU ZBTOX BDONN VVNDT WTNOT␣
 ↪WTDVM BTOWX BWTWH TMMBW WBTOB VNDTW TNONY DWBWI DCINB DHTWX NIZNZ FTOBZ␣
 ↪JBOTE BMDEB OKONH DONWH XJBRN HFNGN ZMTNO HBDON WUNZO HTWXB UZBTO NOBVB␣
 ↪ZHTWX NIZDH XJDIR XBOTE BMDRB OONWB FFDIU WZBRW NZNUW BTOYZ NEUNN KDBOF␣
 ↪DRXNN MTOGW XBWVB ZBENI OWVND TWTNO HXTRX EBOXN MFDWN FBJ'
alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

We are going to start studing the frequency of the letters:

[43]:
```
f = frequency(message,alphabet)
dict(sorted(f.items(), key=lambda x: x[1], reverse=True))
```

[43]:
```
{'N': 93,
 'B': 90,
 'W': 90,
 'T': 72,
 'O': 68,
 'D': 55,
 'X': 48,
 'Z': 38,
 'M': 37,
 'I': 34,
 'F': 33,
 'R': 32,
 'J': 26,
 'V': 22,
 'H': 18,
 'E': 17,
 'Y': 16,
 'U': 13,
 'G': 12,
 'K': 7,
 'C': 1,
 'S': 1,
 'A': 0,
 'L': 0,
 'P': 0,
```

```
 'Q': 0}
```

Like the previous exercise, we look up for bigraphs and trigraphs:

[44]:
```
message_no_space=''
for char in message:
    if char != ' ':
        message_no_space += char
```

[45]:
```
bigrams = count_bigrams(message_no_space)
bigrams
```

[45]:
```
{'BO': 20,
 'TO': 20,
 'WX': 19,
 'NI': 17,
 'WT': 16,
 'XB': 15,
 'TW': 15,
 'WB': 14,
 'XT': 14,
 'NZ': 14,
 'NO': 14,
 'BR': 13,
 'XN': 12,
 'RX': 12,
 'OW': 12,
 'DW': 11,
 'BW': 11,
 'IW': 10,
 'WN': 10,
 'MF': 10,
 'ON': 10,
 'TM': 9,
 'BT': 9,
 'ND': 8,
 'ZB': 8,
 'EB': 8,
 'JN': 7,
 'TN': 7,
 'NH': 7,
 'OG': 7,
 'OB': 7,
 'WD': 7,
 'ZN': 6,
 'BM': 6,
 'OF': 6,
```

```
'FN': 6,
'VN': 6,
'DT': 6,
'MJ': 6,
'RW': 6,
'TR': 6,
'BZ': 6,
'DB': 6,
'MM': 5,
'BF': 5,
'FB': 5,
'RB': 5,
'NM': 5,
'BJ': 5,
'HX': 5,
'MB': 5,
'FD': 5,
'BD': 5,
'NW': 5,
'IZ': 5,
'NN': 5,
'HT': 5,
'TD': 4,
'ZJ': 4,
'FX': 4,
'BE': 4,
'YN': 4,
'OK': 4,
'WV': 4,
'RN': 4,
'OD': 4,
'NF': 4,
'TE': 4,
'ZW': 4,
'OT': 4,
'NY': 4,
'DV': 4,
'JD': 4,
'MT': 4,
'DR': 4,
'DO': 4,
'VM': 4,
'ZT': 3,
'WW': 3,
'HB': 3,
'UW': 3,
'FW': 3,
```

```
'DD': 3,
'VZ': 3,
'WR': 3,
'IO': 3,
'KD': 3,
'OJ': 3,
'GB': 3,
'UZ': 3,
'EN': 3,
'YB': 3,
'ID': 3,
'GN': 3,
'VI': 3,
'WI': 3,
'BU': 3,
'UN': 3,
'JB': 3,
'OH': 3,
'TY': 2,
'XZ': 2,
'IG': 2,
'GX': 2,
'FI': 2,
'IV': 2,
'DX': 2,
'IU': 2,
'HN': 2,
'ZM': 2,
'TU': 2,
'IM': 2,
'DY': 2,
'FF': 2,
'KO': 2,
'WU': 2,
'UT': 2,
'DE': 2,
'OY': 2,
'DN': 2,
'ZE': 2,
'NE': 2,
'JY': 2,
'ZO': 2,
'RT': 2,
'NB': 2,
'YV': 2,
'WE': 2,
'FT': 2,
```

```
'ZD': 2,
'DJ': 2,
'IR': 2,
'IN': 2,
'HD': 2,
'OX': 2,
'WH': 2,
'BV': 2,
'DH': 2,
'MD': 2,
'XJ': 2,
'VB': 2,
'DI': 2,
'YJ': 1,
'XW': 1,
'MX': 1,
'JX': 1,
'EV': 1,
'VT': 1,
'VY': 1,
'GH': 1,
'FR': 1,
'KB': 1,
'FV': 1,
'UM': 1,
'JF': 1,
'NT': 1,
'JJ': 1,
'IH': 1,
'FO': 1,
'WM': 1,
'JZ': 1,
'ZI': 1,
'RZ': 1,
'MK': 1,
'JH': 1,
'NR': 1,
'RM': 1,
'EW': 1,
'WK': 1,
'JW': 1,
'DU': 1,
'YI': 1,
'OR': 1,
'TZ': 1,
'GD': 1,
'JT': 1,
```

```
'OS': 1,
'SN': 1,
'MI': 1,
'YF': 1,
'ED': 1,
'XG': 1,
'EJ': 1,
'RV': 1,
'WJ': 1,
'MW': 1,
'FY': 1,
'YT': 1,
'TG': 1,
'GI': 1,
'ZV': 1,
'VW': 1,
'OM': 1,
'JU': 1,
'UI': 1,
'BY': 1,
'JR': 1,
'DF': 1,
'TV': 1,
'WF': 1,
'EI': 1,
'KV': 1,
'ZY': 1,
'MO': 1,
'YW': 1,
'NV': 1,
'VV': 1,
'YD': 1,
'DC': 1,
'CI': 1,
'ZF': 1,
'HF': 1,
'NG': 1,
'ZH': 1,
'OO': 1,
'WZ': 1,
'NU': 1,
'YZ': 1,
'EU': 1,
'NK': 1,
'GW': 1,
'XE': 1}
```

[46]:
```
trigrams = count_trigrams(message_no_space)
trigrams
```

[46]:
```
{'WXB': 8,
 'TNO': 7,
 'BTO': 7,
 'NIW': 6,
 'BRX': 6,
 'BOF': 6,
 'TOG': 6,
 'XBW': 6,
 'RXT': 6,
 'XTM': 6,
 'TMF': 6,
 'XNI': 5,
 'BOW': 5,
 'WXT': 5,
 'VND': 5,
 'BRW': 5,
 'WTN': 5,
 'WTW': 5,
 'EBO': 5,
 'JNI': 4,
 'BMM': 4,
 'DWB': 4,
 'XTO': 4,
 'WTR': 4,
 'ONW': 4,
 'TWT': 4,
 'DWN': 3,
 'WNZ': 3,
 'FXB': 3,
 'NOW': 3,
 'WBO': 3,
 'YNZ': 3,
 'MFW': 3,
 'FWX': 3,
 'RBO': 3,
 'TOK': 3,
 'NDD': 3,
 'ZBR': 3,
 'RWT': 3,
 'MMJ': 3,
 'RNO': 3,
 'ODW': 3,
 'NFB': 3,
 'FBJ': 3,
```

```
'ONH': 3,
'BOJ': 3,
'OGB': 3,
'UZB': 3,
'ZBT': 3,
'XBD': 3,
'TWD': 3,
'IWT': 3,
'WXN': 3,
'DBR': 3,
'TEB': 3,
'BZJ': 3,
'XNN': 3,
'MJN': 3,
'VMB': 3,
'WBT': 3,
'TOB': 3,
'DON': 3,
'NDT': 3,
'DTW': 3,
'HTW': 3,
'TWX': 3,
'WXZ': 2,
'XZN': 2,
'ZNI': 2,
'NIG': 2,
'IGX': 2,
'GXN': 2,
'IWB': 2,
'XTD': 2,
'XBF': 2,
'RXB': 2,
'OWN': 2,
'WND': 2,
'DXN': 2,
'XNH': 2,
'NHB': 2,
'IUW': 2,
'NZM': 2,
'BWB': 2,
'WBR': 2,
'OWX': 2,
'DDT': 2,
'DTU': 2,
'VZB': 2,
'TRB': 2,
'RBM': 2,
```

```
'OWR': 2,
'WRN': 2,
'NOD': 2,
'WNF': 2,
'MBT': 2,
'KON': 2,
'MFD': 2,
'TOD': 2,
'NOT': 2,
'ZWX': 2,
'XBO': 2,
'BEN': 2,
'DWT': 2,
'DEB': 2,
'TOY': 2,
'DBO': 2,
'OFD': 2,
'NZE': 2,
'OWB': 2,
'WBW': 2,
'TOW': 2,
'HXT': 2,
'XTR': 2,
'TRX': 2,
'DVI': 2,
'VIW': 2,
'NZO': 2,
'WDV': 2,
'NZW': 2,
'DRX': 2,
'RXN': 2,
'NNM': 2,
'NMF': 2,
'BWI': 2,
'NYV': 2,
'YVM': 2,
'MBJ': 2,
'MTO': 2,
'OBZ': 2,
'RWN': 2,
'DJN': 2,
'BWV': 2,
'IWD': 2,
'NHD': 2,
'BWT': 2,
'BUZ': 2,
'BDO': 2,
```

```
'OBV': 2,
'NIZ': 2,
'BOT': 2,
'OTE': 2,
'EBM': 2,
'BMD': 2,
'HXJ': 2,
'NOH': 2,
'VBZ': 2,
'TYJ': 1,
'YJN': 1,
'IWX': 1,
'WXW': 1,
'XWX': 1,
'WBM': 1,
'MMX': 1,
'MXT': 1,
'TDW': 1,
'NZJ': 1,
'ZJX': 1,
'JXB': 1,
'BFX': 1,
'BFB': 1,
'FBR': 1,
'XBE': 1,
'BEV': 1,
'EVT': 1,
'VTN': 1,
'NDW': 1,
'OFI': 1,
'FIV': 1,
'IVY': 1,
'VYN': 1,
'NZT': 1,
'ZTW': 1,
'TWW': 1,
'WWN': 1,
'NDX': 1,
'HBF': 1,
'BFN': 1,
'FNI': 1,
'NIU': 1,
'UWT': 1,
'WTO': 1,
'OGH': 1,
'GHN': 1,
'HNZ': 1,
```

```
'ZMF': 1,
'MFR': 1,
'FRB': 1,
'OKB': 1,
'KBO': 1,
'OFV': 1,
'FVN': 1,
'TUM': 1,
'UMJ': 1,
'MJF': 1,
'JFN': 1,
'FNT': 1,
'NTW': 1,
'TWV': 1,
'WVZ': 1,
'MJJ': 1,
'JJN': 1,
'NIH': 1,
'IHN': 1,
'HNI': 1,
'NIM': 1,
'IMF': 1,
'MFO': 1,
'FOW': 1,
'OWM': 1,
'WMJ': 1,
'MJZ': 1,
'JZI': 1,
'ZIO': 1,
'IOB': 1,
'OBR': 1,
'BRZ': 1,
'RZN': 1,
'ZND': 1,
'DDY': 1,
'DYN': 1,
'YNM': 1,
'NMK': 1,
'MKD': 1,
'KDW': 1,
'BJH': 1,
'JHX': 1,
'HXN': 1,
'XNR': 1,
'NRM': 1,
'RMB': 1,
'BTE': 1,
```

```
'TEW': 1,
'EWX': 1,
'MFF': 1,
'FFN': 1,
'FNO': 1,
'OWK': 1,
'WKO': 1,
'HBO': 1,
'OJW': 1,
'JWX': 1,
'GBR': 1,
'FDU': 1,
'DUZ': 1,
'WBZ': 1,
'BZW': 1,
'ZWD': 1,
'WDY': 1,
'DYI': 1,
'YIO': 1,
'IOR': 1,
'ORW': 1,
'OTO': 1,
'GBW': 1,
'BWU': 1,
'WUT': 1,
'UTZ': 1,
'TZW': 1,
'OFX': 1,
'BDB': 1,
'DBE': 1,
'ENO': 1,
'NOG': 1,
'OGD': 1,
'GDW': 1,
'WDE': 1,
'OJT': 1,
'JTO': 1,
'OYB': 1,
'YBO': 1,
'NOS': 1,
'OSN': 1,
'SNM': 1,
'NMI': 1,
'MIW': 1,
'DWX': 1,
'NID': 1,
'IDB': 1,
```

```
'FDN': 1,
'DNY': 1,
'NYF': 1,
'YFN': 1,
'FNZ': 1,
'ZEB': 1,
'BWN': 1,
'WNE': 1,
'NED': 1,
'EDT': 1,
'DTO': 1,
'WNH': 1,
'NHX': 1,
'RXG': 1,
'XGN': 1,
'GNF': 1,
'NFX': 1,
'BDV': 1,
'WBE': 1,
'BEJ': 1,
'EJD': 1,
'JDW': 1,
'TRV': 1,
'RVN': 1,
'TUT': 1,
'UTM': 1,
'TMT': 1,
'MTW': 1,
'TWJ': 1,
'WJY': 1,
'JYN': 1,
'ZON': 1,
'NWT': 1,
'TRT': 1,
'RTO': 1,
'GBO': 1,
'BOB': 1,
'OBF': 1,
'BFI': 1,
'FIM': 1,
'IMW': 1,
'MWD': 1,
'WDB': 1,
'RWB': 1,
'OFY': 1,
'FYT': 1,
'YTG': 1,
```

```
'TGI': 1,
'GIZ': 1,
'IZT': 1,
'ZTO': 1,
'OGN': 1,
'GNI': 1,
'VIZ': 1,
'IZV': 1,
'ZVN': 1,
'VNZ': 1,
'ZWI': 1,
'WIV': 1,
'IVW': 1,
'VWN': 1,
'WNB': 1,
'NBU': 1,
'BUN': 1,
'UNI': 1,
'DVZ': 1,
'VZT': 1,
'ZTE': 1,
'EBZ': 1,
'ZJD': 1,
'JDR': 1,
'MFB': 1,
'BJD': 1,
'JDB': 1,
'OKD': 1,
'KDO': 1,
'DOB': 1,
'OBW': 1,
'WIZ': 1,
'IZB': 1,
'ZBM': 1,
'JNO': 1,
'NOM': 1,
'OMJ': 1,
'JNY': 1,
'BJU': 1,
'JUI': 1,
'UIW': 1,
'IWE': 1,
'WEB': 1,
'OJB': 1,
'JBY': 1,
'BYN': 1,
'ZEN': 1,
```

```
'ENY': 1,
'BJR': 1,
'JRN': 1,
'ODF': 1,
'DFT': 1,
'FTD': 1,
'TDR': 1,
'DRT': 1,
'RTV': 1,
'TVM': 1,
'VMT': 1,
'ZJY': 1,
'JYB': 1,
'YBR': 1,
'NZD': 1,
'ZDJ': 1,
'NIR': 1,
'IRB': 1,
'OWF': 1,
'WFN': 1,
'FNW': 1,
'NWX': 1,
'TDN': 1,
'DNZ': 1,
'WVI': 1,
'WDJ': 1,
'NIN': 1,
'INI': 1,
'WDX': 1,
'HDB': 1,
'TWE': 1,
'WEI': 1,
'EID': 1,
'IDW': 1,
'DWW': 1,
'WWX': 1,
'OKV': 1,
'KVZ': 1,
'JNZ': 1,
'NZY': 1,
'ZYB': 1,
'YBT': 1,
'BTM': 1,
'TMO': 1,
'MON': 1,
'NHT': 1,
'HTY': 1,
```

```
'TYW': 1,
'YWX': 1,
'IWR': 1,
'WRX': 1,
'MFX': 1,
'FXN': 1,
'NNF': 1,
'FBU': 1,
'TOX': 1,
'OXB': 1,
'ONN': 1,
'NNV': 1,
'NVV': 1,
'VVN': 1,
'OTW': 1,
'WTD': 1,
'TDV': 1,
'DVM': 1,
'TWH': 1,
'WHT': 1,
'HTM': 1,
'TMM': 1,
'MMB': 1,
'MBW': 1,
'BWW': 1,
'WWB': 1,
'BVN': 1,
'NON': 1,
'ONY': 1,
'NYD': 1,
'YDW': 1,
'WID': 1,
'IDC': 1,
'DCI': 1,
'CIN': 1,
'INB': 1,
'NBD': 1,
'BDH': 1,
'DHT': 1,
'IZN': 1,
'ZNZ': 1,
'NZF': 1,
'ZFT': 1,
'FTO': 1,
'ZJB': 1,
'JBO': 1,
'MDE': 1,
```

```
'BOK': 1,
'OKO': 1,
'HDO': 1,
'NWH': 1,
'WHX': 1,
'XJB': 1,
'JBR': 1,
'BRN': 1,
'RNH': 1,
'NHF': 1,
'HFN': 1,
'FNG': 1,
'NGN': 1,
'GNZ': 1,
'ZMT': 1,
'MTN': 1,
'OHB': 1,
'HBD': 1,
'NWU': 1,
'WUN': 1,
'UNZ': 1,
'ZOH': 1,
'OHT': 1,
'XBU': 1,
'TON': 1,
'ONO': 1,
'NOB': 1,
'BVB': 1,
'BZH': 1,
'ZHT': 1,
'IZD': 1,
'ZDH': 1,
'DHX': 1,
'XJD': 1,
'JDI': 1,
'DIR': 1,
'IRX': 1,
'MDR': 1,
'DRB': 1,
'BOO': 1,
'OON': 1,
'NWB': 1,
'WBF': 1,
'BFF': 1,
'FFD': 1,
'FDI': 1,
'DIU': 1,
```

```
 'UWZ': 1,
 'WZB': 1,
 'NZN': 1,
 'ZNU': 1,
 'NUW': 1,
 'UWB': 1,
 'OYZ': 1,
 'YZN': 1,
 'ZNE': 1,
 'NEU': 1,
 'EUN': 1,
 'UNN': 1,
 'NNK': 1,
 'NKD': 1,
 'KDB': 1,
 'FDR': 1,
 'NMT': 1,
 'OGW': 1,
 'GWX': 1,
 'WVB': 1,
 'BZB': 1,
 'ZBE': 1,
 'ENI': 1,
 'NIO': 1,
 'IOW': 1,
 'OWV': 1,
 'WVN': 1,
 'OHX': 1,
 'RXE': 1,
 'XEB': 1,
 'BOX': 1,
 'OXN': 1,
 'XNM': 1,
 'FDW': 1}
```

I tried to make the same assumptions as before (that e is the most common letter and th and the the most common bigraphs and trigraphs) but after some time reasoning this way I didn't get anything, so I am going to try another assumptions. (I tried also supossing that N=a and I didn't get anything, so I am going to try something else)

I am going to suppose that N->o because is a common letter in English. We know that on, of, or and to are common bigraphs in English

[47]:
```
bi = {}
pattern = r'[A-Z]N'
matches = re.findall(pattern, message_no_space)
for b in matches:
    bi[b] = bigrams[b]
```

```
pattern = r'N[A-Z]'
matches = re.findall(pattern, message_no_space)
for b in matches:
    bi[b] = bigrams[b]
dict(sorted(bi.items(), key=lambda x: x[1], reverse=True))
```

[47]:
```
{'NI': 17,
 'NZ': 14,
 'NO': 14,
 'XN': 12,
 'WN': 10,
 'ON': 10,
 'ND': 8,
 'JN': 7,
 'TN': 7,
 'NH': 7,
 'ZN': 6,
 'FN': 6,
 'VN': 6,
 'NM': 5,
 'NW': 5,
 'NN': 5,
 'YN': 4,
 'RN': 4,
 'NF': 4,
 'NY': 4,
 'EN': 3,
 'GN': 3,
 'UN': 3,
 'HN': 2,
 'DN': 2,
 'IN': 2,
 'NE': 2,
 'NB': 2,
 'SN': 1,
 'NT': 1,
 'NR': 1,
 'NG': 1,
 'NU': 1}
```

Because most of the most common trigraphs in English contain n (and, ent, nde, men, nce…) I am going to suppose that O->n since O appears in more trigraphs than I or Z and, also, the most common bigraphs in English contain n (on, in, nd…) and O appears in more bigraphs than the others.

We know that t and a are very common letters in English, so probably they are B and W.

[48]:
```
bi = {}
pattern = r'BN'
matches = re.findall(pattern, message_no_space)
for b in matches:
    bi[b] = bigrams[b]
pattern = r'NB'
matches = re.findall(pattern, message_no_space)
for b in matches:
    bi[b] = bigrams[b]
pattern = r'WN'
matches = re.findall(pattern, message_no_space)
for b in matches:
    bi[b] = bigrams[b]
pattern = r'NW'
matches = re.findall(pattern, message_no_space)
for b in matches:
    bi[b] = bigrams[b]
bi
```

[48]:
```
{'NB': 2, 'WN': 10, 'NW': 5}
```

Since to is more common than oa, I am going to suppose that W->t, so B should be a. In that case, our text goes as follows:

[49]:
```
message1 = ''
for char in message:
    if char == 'N':
        message1 += 'o'
    elif char == 'O':
        message1 += 'n'
    elif char == 'B':
        message1 += 'a'
    elif char == 'W':
        message1 += 't'
    else:
        message1 += char
message1
```

[49]:
```
'TYJoI tXtXZ oIGXo ItaMM XTDto ZJXaF XaFaR XaEVT ontoD tanFI VYoZT ttoDX oHaFo
IUtTn GHoZM FtXat aRXTM FRant XTnKa nFVoD DTUMJ FoTtV ZaRtT RaMMJ JoIHo IMFnt
RonDt antMJ ZInaR ZoDDY oMKDt oFaJH XoRMa TEtXa taRXT MFFon tKnoH anJtX TnGaR
XTMFD UZaTn DtaZt DYInR tTonT nGatU TZtXa nFXaD aEonG DtTtD EanJT nYant RonSo
MItTo nDtXo IDanF DoYFo ZEant atoED TntoH XTRXG oFXaD VItaE JDtTR VoDDT UTMTt
JYoZn otTRT nGana FIMtD aRtan FYTGI ZTnGo ItTtD VIZVo ZtIVt oaUoI tTtDV ZTEaZ
JDRXo oMFaJ DaRXT MFtXT nKDna tIZaM MJonM JoYVM aJUIt EanJa YoZEo YVMaJ Ronta
TnDFT DRTVM TnaZJ YaRto ZDJoI RantF otXTD oZtXa tVItD JoIoI tDXoH DaRXT MFtXa
tTtEI DttXT nKVZa RtTRa MMJoZ YaTMn oHTYt XZoIG XoItR XTMFX ooFaU ZaTnX aDnoo
```

```
VVoDT tTonT tTDVM aTntX atTtH TMMat taTna VoDTt TonoY DtatI DCIoa DHTtX oIZoZ
FTnaZ JanTE aMDEa nKnoH DnotH XJaRo HFoGo ZMTon HaDno tUoZn HTtXa UZaTn onaVa
ZHTtX oIZDH XJDIR XanTE aMDRa nnota FFDIU tZaRt oZoUt aTnYZ oEUoo KDanF DRXoo
MTnGt XatVa ZaEoI ntVoD TtTon HXTRX EanXo MFDto FaJ'
```

[50]:
```
bi = {}
pattern = r'W[A-Z]'
matches = re.findall(pattern, message_no_space)
for b in matches:
    bi[b] = bigrams[b]
dict(sorted(bi.items(), key=lambda x: x[1], reverse=True))
```

[50]:
```
{'WX': 19,
 'WT': 16,
 'WB': 14,
 'WN': 10,
 'WD': 7,
 'WV': 4,
 'WW': 3,
 'WR': 3,
 'WI': 3,
 'WU': 2,
 'WE': 2,
 'WH': 2,
 'WM': 1,
 'WK': 1,
 'WJ': 1,
 'WF': 1,
 'WZ': 1}
```

[51]:
```
tri = {}
pattern = r'W[A-Z][A-Z]'
matches = re.findall(pattern, message_no_space)
for t in matches:
    tri[t] = trigrams[t]
dict(sorted(tri.items(), key=lambda x: x[1], reverse=True))
```

[51]:
```
{'WXB': 8,
 'WXT': 5,
 'WTN': 5,
 'WTW': 5,
 'WTR': 4,
 'WNZ': 3,
 'WBO': 3,
 'WXN': 3,
 'WBT': 3,
 'WND': 2,
```

```
 'WBR': 2,
 'WRN': 2,
 'WNF': 2,
 'WBW': 2,
 'WXZ': 2,
 'WXW': 1,
 'WBM': 1,
 'WWN': 1,
 'WTO': 1,
 'WVZ': 1,
 'WMJ': 1,
 'WKO': 1,
 'WBZ': 1,
 'WDY': 1,
 'WUT': 1,
 'WNH': 1,
 'WBE': 1,
 'WJY': 1,
 'WDB': 1,
 'WIV': 1,
 'WNB': 1,
 'WIZ': 1,
 'WEB': 1,
 'WFN': 1,
 'WVI': 1,
 'WDJ': 1,
 'WDX': 1,
 'WWX': 1,
 'WRX': 1,
 'WTD': 1,
 'WWB': 1,
 'WHX': 1,
 'WUN': 1,
 'WBF': 1,
 'WZB': 1,
 'WVB': 1,
 'WVN': 1}
```

Since th is the most common bigraph in English and tha (tha=W(A-Z)B)) is one of the most common trigraph in English we suppose that WXB->tha and WX->th so X->h.

On the other hand, we know that in, ti and io are common bigraphs in English and that ion and tio are common trigraphs.

[52]:
```
bi = {}
pattern = r'W[A-Z]'
matches = re.findall(pattern, message_no_space)
for b in matches:
```

```
    bi[b] = bigrams[b]
pattern = r'[A-Z]O'
matches = re.findall(pattern, message_no_space)
for b in matches:
    bi[b] = bigrams[b]
pattern = r'[A-Z]N'
matches = re.findall(pattern, message_no_space)
for b in matches:
    bi[b] = bigrams[b]
dict(sorted(bi.items(), key=lambda x: x[1], reverse=True))
```

[52]:
```
{'BO': 20,
 'TO': 20,
 'WX': 19,
 'WT': 16,
 'WB': 14,
 'NO': 14,
 'XN': 12,
 'WN': 10,
 'ON': 10,
 'WD': 7,
 'JN': 7,
 'TN': 7,
 'ZN': 6,
 'FN': 6,
 'VN': 6,
 'WV': 4,
 'DO': 4,
 'YN': 4,
 'RN': 4,
 'WW': 3,
 'WR': 3,
 'WI': 3,
 'IO': 3,
 'EN': 3,
 'GN': 3,
 'UN': 3,
 'WU': 2,
 'WE': 2,
 'WH': 2,
 'KO': 2,
 'ZO': 2,
 'HN': 2,
 'DN': 2,
 'IN': 2,
 'WM': 1,
 'WK': 1,
```

```
 'WJ': 1,
 'WF': 1,
 'WZ': 1,
 'FO': 1,
 'MO': 1,
 'SN': 1}
```

[53]:
```
tri = {}
pattern = r'[A-Z]NO'
matches = re.findall(pattern, message_no_space)
for t in matches:
    tri[t] = trigrams[t]
pattern = r'W[A-Z]N'
matches = re.findall(pattern, message_no_space)
for t in matches:
    tri[t] = trigrams[t]
dict(sorted(tri.items(), key=lambda x: x[1], reverse=True))
```

[53]:
```
{'TNO': 7,
 'WTN': 5,
 'RNO': 3,
 'WXN': 3,
 'WRN': 2,
 'FNO': 1,
 'ENO': 1,
 'JNO': 1,
 'ONO': 1,
 'WWN': 1,
 'WFN': 1,
 'WUN': 1,
 'WVN': 1}
```

So we can see that T seems to be i (T->i). If we change our text it looks like this:

[54]:
```
message2 = ''
for char in message1:
    if char == 'T':
        message2 += 'i'
    elif char == 'X':
        message2 += 'h'
    else:
        message2 += char
message2
```

[54]:
```
'iYJoI ththZ oIGho ItaMM hiDto ZJhaF haFaR haEVi ontoD tanFI VYoZi ttoDh oHaFo
IUtin GHoZM Fthat aRhiM FRant hinKa nFVoD DiUMJ FoitV ZaRti RaMMJ JoIHo IMFnt
RonDt antMJ ZInaR ZoDDY oMKDt oFaJH hoRMa iEtha taRhi MFFon tKnoH anJth inGaR
```

```
hiMFD UZain DtaZt DYInR tioni nGatU iZtha nFhaD aEonG DtitD EanJi nYant RonSo
MItio nDtho IDanF DoYFo ZEant atoED intoH hiRhG oFhaD VItaE JDtiR VoDDi UiMit
JYoZn otiRi nGana FIMtD aRtan FYiGI ZinGo ItitD VIZVo ZtIVt oaUoI titDV ZiEaZ
JDRho oMFaJ DaRhi MFthi nKDna tIZaM MJonM JoYVM aJUIt EanJa YoZEo YVMaJ Ronta
inDFi DRiVM inaZJ YaRto ZDJoI RantF othiD oZtha tVItD JoIoI tDhoH DaRhi MFtha
titEI Dtthi nKVZa RtiRa MMJoZ YaiMn oHiYt hZoIG hoItR hiMFh ooFaU Zainh aDnoo
VVoDi tioni tiDVM ainth atitH iMMat taina VoDit ionoY DtatI DCIoa DHith oIZoZ
FinaZ JaniE aMDEa nKnoH DnotH hJaRo HFoGo ZMion HaDno tUoZn Hitha UZain onaVa
ZHith oIZDH hJDIR haniE aMDRa nnota FFDIU tZaRt oZoUt ainYZ oEUoo KDanF DRhoo
MinGt hatVa ZaEoI ntVoD ition HhiRh Eanho MFDto FaJ'
```

Another common bigraphs in English are of and or and common trigraphs are oft and for.

[55]:
```
bi = {}
pattern = r'N[A-Z]'
matches = re.findall(pattern, message_no_space)
for b in matches:
    bi[b] = bigrams[b]
dict(sorted(bi.items(), key=lambda x: x[1], reverse=True))
```

[55]:
```
{'NI': 17,
 'NZ': 14,
 'NO': 14,
 'ND': 8,
 'NH': 7,
 'NM': 5,
 'NW': 5,
 'NN': 5,
 'NF': 4,
 'NY': 4,
 'NE': 2,
 'NB': 2,
 'NT': 1,
 'NR': 1,
 'NG': 1,
 'NU': 1}
```

[56]:
```
tri = {}
pattern = r'N[A-Z]W'
matches = re.findall(pattern, message_no_space)
for t in matches:
    tri[t] = trigrams[t]
pattern = r'[A-Z]N[A-Z]'
matches = re.findall(pattern, message_no_space)
for t in matches:
    tri[t] = trigrams[t]
dict(sorted(tri.items(), key=lambda x: x[1], reverse=True))
```

[56]: {'TNO': 7,
 'NIW': 6,
 'XNI': 5,
 'VND': 5,
 'JNI': 4,
 'ONW': 4,
 'NOW': 3,
 'WNZ': 3,
 'YNZ': 3,
 'RNO': 3,
 'ONH': 3,
 'XNN': 3,
 'NZW': 2,
 'ZNI': 2,
 'WND': 2,
 'XNH': 2,
 'WNF': 2,
 'NDW': 1,
 'NTW': 1,
 'NUW': 1,
 'FNI': 1,
 'HNZ': 1,
 'FNT': 1,
 'HNI': 1,
 'ZND': 1,
 'YNM': 1,
 'XNR': 1,
 'FNO': 1,
 'ENO': 1,
 'SNM': 1,
 'DNY': 1,
 'FNZ': 1,
 'WNE': 1,
 'WNH': 1,
 'GNF': 1,
 'GNI': 1,
 'VNZ': 1,
 'WNB': 1,
 'UNI': 1,
 'JNO': 1,
 'JNY': 1,
 'ENY': 1,
 'FNW': 1,
 'DNZ': 1,
 'JNZ': 1,
 'ONN': 1,
 'INB': 1,

```
 'ZNZ': 1,
 'RNH': 1,
 'FNG': 1,
 'UNZ': 1,
 'ONO': 1,
 'ZNE': 1,
 'UNN': 1,
 'ENI': 1,
 'XNM': 1}
```

Since NZ has high frequence but NZW has not high frequence, we suppose that Z->r (so it forms or).

[57]:
```
message3 = ''
for char in message2:
    if char == 'Z':
        message3 += 'r'
    else:
        message3 += char
message3
```

[57]:
```
'iYJoI ththr oIGho ItaMM hiDto rJhaF haFaR haEVi ontoD tanFI VYori ttoDh oHaFo
IUtin GHorM Fthat aRhiM FRant hinKa nFVoD DiUMJ FoitV raRti RaMMJ JoIHo IMFnt
RonDt antMJ rInaR roDDY oMKDt oFaJH hoRMa iEtha taRhi MFFon tKnoH anJth inGaR
hiMFD Urain Dtart DYInR tioni nGatU irtha nFhaD aEonG DtitD EanJi nYant RonSo
MItio nDtho IDanF DoYFo rEant atoED intoH hiRhG oFhaD VItaE JDtiR VoDDi UiMit
JYorn otiRi nGana FIMtD aRtan FYiGI rinGo ItitD VIrVo rtIVt oaUoI titDV riEar
JDRho oMFaJ DaRhi MFthi nKDna tIraM MJonM JoYVM aJUIt EanJa YorEo YVMaJ Ronta
inDFi DRiVM inarJ YaRto rDJoI RantF othiD ortha tVItD JoIoI tDhoH DaRhi MFtha
titEI Dtthi nKVra RtiRa MMJor YaiMn oHiYt hroIG hoItR hiMFh ooFaU rainh aDnoo
VVoDi tioni tiDVM ainth atitH iMMat taina VoDit ionoY DtatI DCIoa DHith oIror
Finar JaniE aMDEa nKnoH DnotH hJaRo HFoGo rMion HaDno tUorn Hitha Urain onaVa
rHith oIrDH hJDIR haniE aMDRa nnota FFDIU traRt oroUt ainYr oEUoo KDanF DRhoo
MinGt hatVa raEoI ntVoD ition HhiRh Eanho MFDto FaJ'
```

In the first line we have iYJoI ththr oIGho. We have to th together, so that indicates we have 2 diferrents words. iYoIth and ThoI... I search for words in English starting i and ending th (https://www.worddb.com/words/starting-with/i/ending-with/th) and no ones fits in iYJoIth, so we can suppose there are 2 words, one ending th and other starting i. Since the only word starting o and finishing th is oath (https://www.worddb.com/words/starting-with/o/ending-with/th), that does not fit in oIth because B->a and a word with a vocal ending in th does not exists, we suppose JoIth is a word, so iY is another. It makes sense that Y->f, so we have if. On the other hand, if we search for 5-letterswords in English ending th (https://www.thefreedictionary.com/words-that-end-in-th) we see that the only one that fits in JoIth is youth, so J->y and I->u.

[58]:
```
message4 = ''
for char in message3:
    if char == 'J':
```

```
        message4 += 'y'
    elif char == 'I':
        message4 += 'u'
    elif char == 'Y':
        message4 += 'f'
    else:
        message4 += char
message4
```

[58]:
```
'ifyou ththr ouGho utaMM hiDto ryhaF haFaR haEVi ontoD tanFu Vfori ttoDh oHaFo
uUtin GHorM Fthat aRhiM FRant hinKa nFVoD DiUMy FoitV raRti RaMMy youHo uMFnt
RonDt antMy runaR roDDf oMKDt oFayH hoRMa iEtha taRhi MFFon tKnoH anyth inGaR
hiMFD Urain Dtart DfunR tioni nGatU irtha nFhaD aEonG DtitD Eanyi nfant RonSo
Mutio nDtho uDanF DofFo rEant atoED intoH hiRhG oFhaD VutaE yDtiR VoDDi UiMit
yforn otiRi nGana FuMtD aRtan FfiGu rinGo utitD VurVo rtuVt oaUou titDV riEar
yDRho oMFay DaRhi MFthi nKDna turaM MyonM yofVM ayUut Eanya forEo fVMay Ronta
inDFi DRiVM inary faRto rDyou RantF othiD ortha tVutD youou tDhoH DaRhi MFtha
titEu Dtthi nKVra RtiRa MMyor faiMn oHift hrouG houtR hiMFh ooFaU rainh aDnoo
VVoDi tioni tiDVM ainth atitH iMMat taina VoDit ionof Dtatu DCuoa DHith ouror
Finar yaniE aMDEa nKnoH DnotH hyaRo HFoGo rMion HaDno tUorn Hitha Urain onaVa
rHith ourDH hyDuR haniE aMDRa nnota FFDuU traRt oroUt ainfr oEUoo KDanF DRhoo
MinGt hatVa raEou ntVoD ition HhiRh Eanho MFDto Fay'
```

Again,in the first sentence we have if youth throuGho utaMM hiDto ryhaF. Reordering we have if youth throuGh out aMM hiDtory haF... It seems to be if youth through out all history, so M->l, G->g and D->s.

[59]:
```
message5 = ''
for char in message4:
    if char == 'D':
        message5 += 's'
    elif char == 'G':
        message5 += 'g'
    elif char == 'M':
        message5 += 'l'
    else:
        message5 += char
message5
```

[59]:
```
'ifyou ththr ougho utall histo ryhaF haFaR haEVi ontos tanFu Vfori ttosh oHaFo
uUtin gHorl Fthat aRhil FRant hinKa nFVos siUly FoitV raRti Rally youHo ulFnt
Ronst antly runaR rossf olKst oFayH hoRla iEtha taRhi lFFon tKnoH anyth ingaR
hilFs Urain start sfunR tioni ngatU irtha nFhas aEong stits Eanyi nfant RonSo
lutio nstho usanF sofFo rEant atoEs intoH hiRhg oFhas VutaE ystiR Vossi Uilit
yforn otiRi ngana Fults aRtan Ffigu ringo utits VurVo rtuVt oaUou titsV riEar
ysRho olFay saRhi lFthi nKsna tural lyonl yofVl ayUut Eanya forEo fVlay Ronta
insFi sRiVl inary faRto rsyou RantF othis ortha tVuts youou tshoH saRhi lFtha
```

```
titEu stthi nKVra RtiRa llyor failn oHift hroug houtR hilFh ooFaU rainh asnoo
VVosi tioni tisVl ainth atitH illat taina Vosit ionof statu sCuoa sHith ouror
Finar yaniE alsEa nKnoH snotH hyaRo HFogo rlion Hasno tUorn Hitha Urain onaVa
rHith oursH hysuR haniE alsRa nnota FFsuU traRt oroUt ainfr oEUoo KsanF sRhoo
lingt hatVa raEou ntVos ition HhiRh Eanho lFsto Fay'
```

Again, in first line we have ...history haF haFaR haEVi... We see we have 2-times haF so we have 2-times the same word and, since D->s, it should be F->d.

[60]:
```
message6 = ''
for char in message5:
    if char == 'F':
        message6 += 'd'
    else:
        message6 += char
message6
```

[60]:
```
'ifyou ththr ougho utall histo ryhad hadaR haEVi ontos tandu Vfori ttosh oHado
uUtin gHorl dthat aRhil dRant hinKa ndVos siUly doitV raRti Rally youHo uldnt
Ronst antly runaR rossf olKst odayH hoRla iEtha taRhi lddon tKnoH anyth ingaR
hilds Urain start sfunR tioni ngatU irtha ndhas aEong stits Eanyi nfant RonSo
lutio nstho usand sofdo rEant atoEs intoH hiRhg odhas VutaE ystiR Vossi Uilit
yforn otiRi ngana dults aRtan dfigu ringo utits VurVo rtuVt oaUou titsV riEar
ysRho olday saRhi ldthi nKsna tural lyonl yofVl ayUut Eanya forEo fVlay Ronta
insdi sRiVl inary faRto rsyou Rantd othis ortha tVuts youou tshoH saRhi ldtha
titEu stthi nKVra RtiRa llyor failn oHift hroug houtR hildh oodaU rainh asnoo
VVosi tioni tisVl ainth atitH illat taina Vosit ionof statu sCuoa sHith ouror
dinar yaniE alsEa nKnoH snotH hyaRo Hdogo rlion Hasno tUorn Hitha Urain onaVa
rHith oursH hysuR haniE alsRa nnota ddsuU traRt oroUt ainfr oEUoo Ksand sRhoo
lingt hatVa raEou ntVos ition HhiRh Eanho ldsto day'
```

If we see oHift hroug houtR hildh oodaU rainh asnoo VVosi tioni tisVl and we order it we have throughout Rhildhood Urain has no oVVosition ... so it seems to be throughout childhood, a brain has no opposition. We have then that R->c, U->b and V->p.

[61]:
```
message7 = ''
for char in message6:
    if char == 'R':
        message7 += 'c'
    elif char == 'U':
        message7 += 'b'
    elif char == 'V':
        message7 += 'p'
    else:
        message7 += char
message7
```

[61]:
```
'ifyou ththr ougho utall histo ryhad hadac haEpi ontos tandu pfori ttosh oHado
ubtin gHorl dthat achil dcant hinKa ndpos sibly doitp racti cally youHo uldnt
const antly runac rossf olKst odayH hocla iEtha tachi lddon tKnoH anyth ingac
hilds brain start sfunc tioni ngatb irtha ndhas aEong stits Eanyi nfant conSo
lutio nstho usand sofdo rEant atoEs intoH hichg odhas putaE ystic possi bilit
yforn otici ngana dults actan dfigu ringo utits purpo rtupt oabou titsp riEar
yscho olday sachi ldthi nKsna tural lyonl yofpl aybut Eanya forEo fplay conta
insdi scipl inary facto rsyou cantd othis ortha tputs youou tshoH sachi ldtha
titEu stthi nKpra ctica llyor failn oHift hroug houtc hildh oodab rainh asnoo
pposi tioni tispl ainth atitH illat taina posit ionof statu sCuoa sHith ouror
dinar yaniE alsEa nKnoH snotH hyaco Hdogo rlion Hasno tborn Hitha brain onapa
rHith oursH hysuc haniE alsca nnota ddsub tract orobt ainfr oEboo Ksand schoo
lingt hatpa raEou ntpos ition Hhich Eanho ldsto day'
```

Now, looking at the text, is easy to see that K->e, C->j, E->m, S->v and H->w.

[62]:
```
result = ''
for char in message7:
    if char == 'M':
        result += 'e'
    elif char == 'C':
        result += 'j'
    elif char == 'E':
        result += 'm'
    elif char == 'H':
        result += 'w'
    else:
        result += char
result
```

[62]:
```
'ifyou ththr ougho utall histo ryhad hadac hampi ontos tandu pfori ttosh owado
ubtin gworl dthat achil dcant hinKa ndpos sibly doitp racti cally youwo uldnt
const antly runac rossf olKst odayw hocla imtha tachi lddon tKnow anyth ingac
hilds brain start sfunc tioni ngatb irtha ndhas among stits manyi nfant conSo
lutio nstho usand sofdo rmant atoms intow hichg odhas putam ystic possi bilit
yforn otici ngana dults actan dfigu ringo utits purpo rtupt oabou titsp rimar
yscho olday sachi ldthi nKsna tural lyonl yofpl aybut manya formo fplay conta
insdi scipl inary facto rsyou cantd othis ortha tputs youou tshow sachi ldtha
titmu stthi nKpra ctica llyor failn owift hroug houtc hildh oodab rainh asnoo
pposi tioni tispl ainth atitw illat taina posit ionof statu sjuoa swith ouror
dinar yanim alsma nKnow snotw hyaco wdogo rlion wasno tborn witha brain onapa
rwith oursw hysuc hanim alsca nnota ddsub tract orobt ainfr omboo Ksand schoo
lingt hatpa ramou ntpos ition which manho ldsto day'
```

Reordering we have:

if youth throughout all history had had a champion to stand up for it to show a doubting world that a child can think and possibly do it practically you wouldnt constantly run across folks today who

claim that a child dont know anything a childs brain starts functioning at birth and has amongst its many infant convolutions thousands of dormant atoms into which god has put a mystic possibility for noticing an adults act and figuring out its purport up to about its primary school days a child thinks naturally only of play but many a form of play contains disciplinary factors you cant do this or that puts you out shows a child that it must think practically or fail now if throughout childhood a brain has no opposition it is plain that it will attain a position of status quo as with our ordinary animals man knows not why a cow dog or lion was not born with a brain on a par with ours why such animals cannot add subtract or obtain from books and schooling that paramount position which man hold to day

[ ]: