

1845479-HW01

Carmignani Federico

November 9, 2021

Abstract

The goal of this homework is to break a substitution cipher.

The solution has to be sent to `cns@diag.uniroma1.it` from the institutional email.

The password to unzip the file with the ciphertext published by Classroom is: 'Act0r1um['.

The deadline is November 9th at 11:59 pm.

1 What is the challenge

The challenge consists in trying to simulate an attack to a given ciphertext using frequencies. The objective is to determine the plaintext, a text in English, by analyzing the frequencies of symbols appearing in the ciphertext. The used cipher is a substitution cipher, a method of encrypting in which units of plaintext are replaced with the ciphertext, in a defined manner, with the help of a key, in this case based on a permutation of the symbols of the alphabet. The practice of a frequency-based attack consists in sorting the symbols in order of frequency, comparing this information to public information, that is a list of frequencies of English words, in order to understand symbols behind the ciphertext. A substitution cipher generates a random permutation and uses it for ciphering the information.

Example: HOUSE \rightarrow VUIMC.

It happens if the following matchings are true: H \rightarrow V, O \rightarrow U, U \rightarrow I, E \rightarrow C.

2 How to solve the challenge

In order to break the substitution cipher, I had to calculate the frequency of each symbol and then memorize it.

I did it using Python as programming language and the implementation will be added to the section 'How to use and run the program'.

It is a frequency analyzer, that stores the frequency in a dictionary, in which the keys are the letters and the values are the frequencies and then I had to compare with the known ones to find the right choices. However, it was not so easy to perform this attack since the text is not enough long to be compared with the frequency of a generic English text. So, I used a web-tool at <https://www.boxentriq.com/code-breaking/frequency-analysis> to analyze in parallel the frequency of the 'bigrams' and 'trigrams' in the ciphertext as a support to the first matchings created from the frequency of the single letters. I also had to analyze the most popular words in English of two and three letters, and, in this way, I could cross the evidences given by these analyses. To do everything in Python would have asked more time since an implementation also of an analysis of the words of two and three letters had to be done and then the program had to check also if a sequence of possible choices could lead to 'real' English words. So the program I created just computes the frequencies of all single letters of a ciphertext that can be given in input from a file, and then it asks to the user for replacing letters with others. I used first the frequency of the letters in a generic English text, taken from Wikipedia, as shown below in Figure 1. The ciphertext is without new lines, spaces and the line breaks are meaningless. Only alphabetic characters are encrypted, while punctuation and other special characters are unchanged, all letters are capitalised.



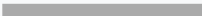
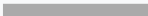






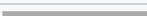
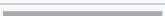
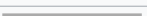
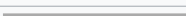


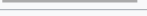
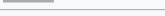


































Letter ↕	Relative frequency in the English language			
	Texts ▼		Dictionaries ↕	
E	13%		11%	
T	9.1%		6.7%	
A	8.2%		7.8%	
O	7.5%		6.1%	
I	7%		8.6%	
N	6.7%		7.2%	
S	6.3%		8.7%	
H	6.1%		2.3%	
R	6%		7.3%	
D	4.3%		3.8%	
L	4%		5.3%	
C	2.8%		4%	
U	2.8%		3.3%	
M	2.4%		2.7%	
W	2.4%		0.91%	
F	2.2%		1.4%	
G	2%		3%	
Y	2%		1.6%	
P	1.9%		2.8%	
B	1.5%		2%	
V	0.98%		1%	
K	0.77%		0.97%	
J	0.15%		0.21%	
X	0.15%		0.27%	
Q	0.095%		0.19%	
Z	0.074%		0.44%	

Figure 1: Frequency of single letters in English language.

2.1 The steps towards the solution

In order to achieve the solution of the text, which is a part of a famous text written by Shakespeare in the prologue of Romeo and Juliet, I started to make substitutions according to the frequency of the letters in the ciphertext, I sorted them and then I compared them with the similar frequency in the English language. In this way, I assigned the letters to the letters ‘C’, ‘R’ and ‘Q’, it was also supported by the web tool according to which the most frequent word of three letters is “THE” in English and in our text the trigram “RQC” was the most used.

After I analyzed that the most frequent sequence of two same letters in English is “SS” and in our text it was “XX”, so I tried to substitute them. Afterwards, I saw the first part of the ciphertext and also that our letters with frequency similar to the natural frequency of ‘Y’ were ‘A’ or ‘O’, seeing that the first phrase using ‘O’ was becoming understandable, I used it. Then, analyzing the “bigrams” more frequent in the English text and in our text I saw similarities between “RE” in English text and “HC” in the ciphertext, so I did the substitution ‘H’ in place of ‘R’.

I watched the text and a word “O?R” was being formed in the last part, so I decided to replace this symbol with ‘U’ in order to create the word “OUR”. From this moment all the text was more clear, so I continued just completing famous English words with other substitutions, like ‘children’, ‘strive’, ‘cross’ or ‘blood’ for example.

These steps led me to formulate a substitutional dictionary that is summarised by the Table 1 in the next page.

In order to perform all the substitutions and see the plaintext I used the program that I have written in Python.

3 How to use and run the program

The program asks for a file to be used as ciphertext and then produces the frequencies of every symbol included in it in a dictionary. Then it can be used to make substitutions to the ciphertext in order

Ciphertext	Plaintext
C	E
R	T
Y	O
H	R
J	I
Q	H
G	A
X	S
I	N
K	L
B	F
T	U
E	D
D	W
Z	C
V	V
A	M
F	G
U	P
M	B
L	K
O	Y

Table 1: The substitutions.

to obtain the plaintext, it is also possible to come back to the original ciphertext if something is not right and restart the procedure. It has to be executed using Spider or another IDE to run the program or also in the terminal using Python interpreter with this command: 'python 1845479-HW01.py', the important thing is that the file with the ciphertext is in the same folder of the Python script. If the file is not present, an error will be shown in the console.

3.1 The code

```
import sys

#initial prints
print('\nWelcome to the frequencies analyzer!')
print('NOTE: It assumes that the plaintext is in English and all in capital letters..')
ciphertext_filename = input('Enter the name of the file with the ciphertext: ')

#open text file in read mode
ciphertext_file = open(ciphertext_filename, 'r')

#read whole file to a string
ciphertext = ciphertext_file.read()

#close file
ciphertext_file.close()

#print the ciphertext
print('\nThis is the ciphertext uploaded:')
print(ciphertext)

#analyzing frequencies
num_chars=0
```

```

freq_dictionary = {}
for char in ciphertext:
    if char not in freq_dictionary:
        freq_dictionary[char] = 1
    else:
        freq_dictionary[char] += 1
num_chars+=1

print('\nNow these are the occurrences of all the characters in the ciphertext:')
print(freq_dictionary)
print('The number of chars is: ',num_chars)
print('The frequencies of the characters are shown below:')

for key in freq_dictionary:
    freq_dictionary[key] *= 100
    freq_dictionary[key] /= num_chars

sorted_freq_dictionary = {}
sorted_values = sorted(freq_dictionary.values())
for i in sorted_values:
    for k in freq_dictionary.keys():
        if freq_dictionary[k] == i:
            sorted_freq_dictionary[k] = freq_dictionary[k]

print('Dictionary sorted by frequency value',sorted_freq_dictionary)

#ask for replaces ('EXIT' to exit the program or 'NEW' to return to the original ciphertext)
original_one = ciphertext
while(True):
    char_to_replace = input('\nSelect a char to replace (or 'EXIT' to exit the program
    or 'NEW' to restart from the original ciphertext): ')
    if(char_to_replace == 'EXIT'):
        sys.exit()
    if(char_to_replace == 'NEW'):
        ciphertext = original_one
        print('\nRestart from the original one..',ciphertext)
        continue
    char_to_replace_with = input ('\nSelect a char to replace it with: ')
    print('\n')
    ciphertext = ciphertext.replace(char_to_replace,char_to_replace_with.lower())

    print('\nNow the text becomes: ',ciphertext)

```

3.2 The program in use

As told before, the program asks to insert the name of the file and then the dictionary with the sorted frequencies is printed.

An illustration is given in Figure 2, where the ciphertext is printed and then even all the occurrences of the symbols in the ciphertext.

The number of chars in the text is computed in order to have the frequencies of the symbols, and so these frequencies are calculated and the dictionary is printed sorted by the values. Instead, in Figure 2, it is shown the last step of the procedure, where, after that the substitutions of the table are performed, the ciphertext is modified continuously until the final plaintext is reached.

The plaintext is shown at the end of Figure 2 and it is the same of the original prologue of Romeo and Juliet, so it seems to be the right one.

```
HW01_CyberSec — python 1845479-HW01.py — 129x43
Last login: Mon Nov  8 17:55:02 on ttys000
(base) federico@MBP-di-Federico HW01_CyberSec % python 1845479-HW01.py

Welcome to the frequencies analyzer!
NOTE: It assumes that the plaintext is in English and all in capital letters..
Enter the name of the file with the ciphertext: ciphertext.txt

This is the ciphertext uploaded:
RDYQYTXCQYKEX,MYRQGKJLCJIEJFIJRO,JIBGJHVCHYIG,DQCHCDCKGOYTHXZCIC,BHYAGIZJCIRFHTEFCMHGGLRYICDATTRJIO,DQCHCZJVJMKMYEAGLCXZJVJKQGIEX
TIZKCGI.BHYABYHRQRCBGRGKKYJIXYBRQCXCRDYBYCXGUGJHYBXRGH-ZHYXX'EKYVCHXRGLCRQCJHKJBC;DQYXCAJXGEVCIRTHCEUJRCYTYVCHRHQYDXEYDJRQRCJH
ECGRQMTHORQCJHUGHCIRX'XRHJBC.RQCBGCBHTKUGXXGFCYBRQCJHECGRQ-AGHL'EKYVC,GIERQCZYIRJITGIZCYBRQCJHUGHCIRX'HGFC,DQJZQ,MTRRQCJHZQJKEHCI
'XCIE,IYTFQRZYTEKHCAVVC,JXIYDRQCXDYQYTHX'RHGBBJZYBYTHXRGFC;RQCDQJZQJBOYTDJRQUGRJCIRCGHXGRRCIE,DQGRQCCHXQKKAJXX,YTHRYJKXQKXKRHVJ
CRY

Now these are the occurrences of all the characters in the ciphertext:
{'R': 47, 'D': 15, 'Y': 40, 'Q': 36, 'T': 17, 'X': 31, 'C': 59, 'K': 20, 'E': 17, ' ': 12, 'M': 5, 'G': 35, 'J': 37, 'L': 5, 'I':
26, 'F': 7, 'O': 5, 'B': 18, 'H': 39, 'V': 9, 'Z': 13, 'A': 8, '.': 2, 'U': 6, '-': 2, '"': 6, ';': 2, '\n': 1}
The number of chars is: 520
The frequencies of the characters are shown below:
Dictionary sorted by frequency value {'\n': 0.19230769230769232, ' ': 0.38461538461538464, '-': 0.38461538461538464, ';': 0.38461
538461538464, 'M': 0.9615384615384616, 'L': 0.9615384615384616, 'O': 0.9615384615384616, 'U': 1.1538461538461537, '"': 1.15384615
38461537, 'F': 1.3461538461538463, 'A': 1.5384615384615385, 'V': 1.7307692307692308, ' ': 2.3076923076923075, 'Z': 2.5, 'D': 2.88
46153846153846, 'T': 3.269230769230769, 'E': 3.269230769230769, 'B': 3.4615384615384617, 'K': 3.8461538461538463, 'I': 5.0, 'X':
5.961538461538462, 'G': 6.730769230769231, 'Q': 6.923076923076923, 'J': 7.115384615384615, 'H': 7.5, 'Y': 7.6923076923076925, 'R':
9.038461538461538, 'C': 11.346153846153847}

Select a char to replace (or 'EXIT' to exit the program or 'NEW' to restart from the original ciphertext):
```

Figure 2: Demo 1.

```
HW01_CyberSec — python 1845479-HW01.py — 129x43
lmiss,ourtoilshallstriveto

Select a char to replace (or 'EXIT' to exit the program or 'NEW' to restart from the original ciphertext): M
Select a char to replace it with: B

Now the text becomes: twohouseholds,bothalileindignit0,infairverona,wherewelaOourscene,fromancientgrudgebrealtonewmutin0,wherci
vilbloodmalescivilhandsunclean.fromforththefatalloinsofthesetwofoesapairofstar-cross'dloverstaketheirlife;whosemisadventuredpito
usoverthrowsdowiththeirdeathburOtheirparents'strife.thefearfulpassageoftheirdeath-mark'dlove,andthecontinuanceoftheirparents'rage
,which,buttheirchildren'send,noughtcouldremove,isnowthetwohours'trafficofourstage;thewhichifOouwithpatientearsattend,whathereshal
lmiss,ourtoilshallstriveto

Select a char to replace (or 'EXIT' to exit the program or 'NEW' to restart from the original ciphertext): L
Select a char to replace it with: K

Now the text becomes: twohouseholds,bothalileindignit0,infairverona,wherewelayourscene,fromancientgrudgebreaktonewmutin0,wherci
vilbloodmakescivilhandsunclean.fromforththefatalloinsofthesetwofoesapairofstar-cross'dloverstaketheirlife;whosemisadventuredpito
usoverthrowsdowiththeirdeathburytheirparents'strife.thefearfulpassageoftheirdeath-mark'dlove,andthecontinuanceoftheirparents'rage
,which,buttheirchildren'send,noughtcouldremove,isnowthetwohours'trafficofourstage;thewhichifOouwithpatientearsattend,whathereshal
lmiss,ourtoilshallstriveto

Select a char to replace (or 'EXIT' to exit the program or 'NEW' to restart from the original ciphertext): O
Select a char to replace it with: Y

Now the text becomes: twohouseholds,bothalileindignit0,infairverona,wherewelayourscene,fromancientgrudgebreaktonewmutin0,wherci
vilbloodmakescivilhandsunclean.fromforththefatalloinsofthesetwofoesapairofstar-cross'dloverstaketheirlife;whosemisadventuredpito
usoverthrowsdowiththeirdeathburytheirparents'strife.thefearfulpassageoftheirdeath-mark'dlove,andthecontinuanceoftheirparents'rage
,which,buttheirchildren'send,noughtcouldremove,isnowthetwohours'trafficofourstage;thewhichifOouwithpatientearsattend,whathereshal
lmiss,ourtoilshallstriveto

Select a char to replace (or 'EXIT' to exit the program or 'NEW' to restart from the original ciphertext):
```

Figure 3: Demo 2.