Coursework 1 - Exercise 2

November 9, 2023

The following text was encrypted using the substitution cipher. Please decode it using any method you find adequate. After you found a solution, please describe how you analyzed the text.

The first thing we notice is that the text seems to be in MORSE code. By a little further exploration, we also realise there is only letters, so we proceed to obtain the characters

[1]:
```
from util import *

data = open('ex2_raw.txt', 'r').read()
data = data.split('\n')
data = [x.split() for x in data]
data = [[morse_code[i] for i in x] for x in data]
data = ["".join(x) for x in data]
for x in data:
    print(x)
```

```
KQAVDUWRJVXUHDXYZGYWKWZWKRQAKUHFVKYXPFWHRERIFQAVDUWKQJKQSHKAHZQKWYRIUNXKQWFOWXVF
VFUNXAFESKWHWHFAKUHFVWFOWKQXEFIKQFEPXQQFVSKWHWHFHFNURIXMFDWHFZQKWYPXDGFYKQJNFNFW
WFVYWHFPRYWARPPRQUXKVYRINFWWFVYWVKUNFWYRINFWWFVYPKOWZVFYRIWHFXGRBFXQEYRIRVWHWHFV
FAFKBFVEFAKUHFVYWHFWFOWGDUFVIRVPKQJWHFKQBFVYFYZGYWKWZWKRQUVRAFYYWRFOWVXAWWHFRVKJ
KQXNPFYYXJF
YZGYWKWZWKRQAKUHFVYAXQGFARPUXVFESKWHWVXQYURYKWKRQAKUHFVYKQXWVXQYURYKWKRQAKUHFVWH
FZQKWYRIWHFUNXKQWFOWXVFVFXVVXQJFEKQXEKIIFVFQWXQEZYZXNNDCZKWFARPUNFORVEFVGZWWHFZQ
KWYWHFPYFNBFYXVFNFIWZQAHXQJFEGDARQWVXYWKQXYZGYWKWZWKRQAKUHFVWHFZQKWYRIWHFUNXKQWF
OWXVFVFWXKQFEKQWHFYXPFYFCZFQAFKQWHFAKUHFVWFOWGZWWHFZQKWYWHFPYFNBFYXVFXNWFVFE
UNFXYFEFAREFWHKYWFOWZYKQJXQDPFWHREDRZIKQEXEFCZXWFDRZSKNNIKJZVFRZWWHXWWHFWFOWSXYF
QAVDUWFEZYKQJWHFYZGYWKWZWKRQAKUHFVXIWFVDRZIRZQEXYRNZWKRQUNFXYFEFYAVKGFHRSDRZXQXN
DTFEWHFWFOWHKQWDRZPXDZYFXQDUVRJVXPRVYKPUNDARZQWWHFIVFCZFQAKFYRIXNNWHFUKAWZVFYWHX
WXUUFXVSKWHKQWHFWFOWHXBFIZQ
```

We can see how we apparently have three sentences of random characters. We first try to decypher the texts with the caesar cypher by attempting all 26 combinations on one sentence

[2]:
```
def cipher(message, K):
    message = [ord(x) for x in message]
    return "".join([chr(
        ((x - (65 if (65 <= x <= 90) else 97) + K) % 26 + (65 if (65 <= x <=␣
 ↪90) else 97))
```

```
        if (65 <= x <= 90 or 97 <= x <= 122) else x)
        for x in message])

for i in range(26):
    print(i, cipher(data[2], -i), sep=":\t")
```

```
0:      UNFXYFEFAREFWHKYWFOWZYKQJXQDPFWHREDRZIKQEXEFCZXWFDRZSKNNIKJZVFRZWWHXWWHF
WFOWSXYFQAVDUWFEZYKQJWHFYZGYWKWZWKRQAKUHFVXIWFVDRZIRZQEXYRNZWKRQUNFXYFEFYAVKGFHR
SDRZXQXNDTFEWHFWFOWHKQWDRZPXDZYFXQDUVRJVXPRVYKPUNDARZQWWHFIVFCZFQAKFYRIXNNWHFUKA
WZVFYWHXWXUUFXVSKWHKQWHFWFOWHXBFIZQ
1:      TMEWXEDEZQDEVGJXVENVYXJPIWPCOEVGQDCQYHJPDWDEBYWVECQYRJMMHJIYUEQYVVGWVVGE
VENVRWXEPZUCTVEDYXJPIVGEXYFXVJVYVJQPZJTGEUWHVEUCQYHQYPDWXQMYVJQPTMEWXEDEXZUJFEGQ
RCQYWPWMCSEDVGEVENVGJPVCQYOWCYXEWPCTUQIUWOQUXJOTMCZQYPVVGEHUEBYEPZJEXQHWMMVGETJZ
VYUEXVGWVWTTEWURJVGJPVGEVENVGWAEHYP
2:      SLDVWDCDYPCDUFIWUDMUXWIOHVOBNDUFPCBPXGIOCVCDAXVUDBPXQILLGIHXTDPXUUFVUUFD
UDMUQVWDOYTBSUDCXWIOHUFDWXEWUIUXUIPOYISFDTVGUDTBPXGPXOCVWPLXUIPOSLDVWDCDWYTIEDFP
QBPXVOVLBRDCUFDUDMUFIOUBPXNVBXWDVOBSTPHTVNPTWINSLBYPXOUUFDGTDAXDOYIDWPGVLLUFDSIY
UXTDWUFVUVSSDVTQIUFIOUFDUDMUFVZDGXO
3:      RKCUVCBCXOBCTEHVTCLTWVHNGUNAMCTEOBAOWFHNBUBCZWUTCAOWPHKKFHGWSCOWTTEUTTEC
TCLTPUVCNXSARTCBWVHNGTECVWDVTHTWTHONXHRECSUFTCSAOWFOWNBUVOKWTHONRKCUVCBCVXSHDCEO
PAOWUNUKAQCBTECTCLTEHNTAOWMUAWVCUNARSOGSUMOSVHMRKAXOWNTTECFSCZWCNXHCVOFUKKTECRHX
TWSCVTEUTURRCUSPHTEHNTECTCLTEUYCFWN
4:      QJBTUBABWNABSDGUSBKSVUGMFTMZLBSDNAZNVEGMATABYVTSBZNVOGJJEGFVRBNVSSDTSSDB
SBKSOTUBMWRZQSBAVUGMFSDBUVCUSGSVSGNMWGQDBRTESBRZNVENVMATUNJVSGNMQJBTUBABUWRGCBDN
OZNVTMTJZPBASDBSBKSDGMSZNVLTZVUBTMZQRNFRTLNRUGLQJZWNVMSSDBERBYVBMWGBUNETJJSDBQGW
SVRBUSDTSTQQBTROGSDGMSDBSBKSDTXBEVM
5:      PIASTAZAVMZARCFTRAJRUTFLESLYKARCMZYMUDFLZSZAXUSRAYMUNFIIDFEUQAMURRCSRRCA
RAJRNSTALVQYPRAZUTFLERCATUBTRFRURFMLVFPCAQSDRAQYMUDMULZSTMIURFMLPIASTAZATVQFBACM
NYMUSLSIYOAZRCARAJRCFLRYMUKSYUTASLYPQMEQSKMQTFKPIYVMULRRCADQAXUALVFATMDSIIRCAPFV
RUQATRCSRSPPASQNFRCFLRCARAJRCSWADUL
6:      OHZRSZYZULYZQBESQZIQTSEKDRKXJZQBLYXLTCEKYRYZWTRQZXLTMEHHCEDTPZLTQQBRQQBZ
QZIQMRSZKUPXOQZYTSEKDQBZSTASQEQTQELKUEOBZPRCQZPXLTCLTKYRSLHTQELKOHZRSZYZSUPEAZBL
MXLTRKRHXNZYQBZQZIQBEKQXLTJRXTSZRKXOPLDPRJLPSEJOHXULTKQQBZCPZWTZKUEZSLCRHHQBZOEU
QTPZSQBRQROOZRPMEQBEKQBZQZIQBRVZCTK
7:      NGYQRYXYTKXYPADRPYHPSRDJCQJWIYPAKXWKSBDJXQXYVSQPYWKSLDGGBDCSOYKSPPAQPPAY
PYHPLQRYJTOWNPYXSRDJCPAYRSZRPDPSPDKJTDNAYOQBPYOWKSBKSJXQRKGSPDKJNGYQRYXYRTODZYAK
LWKSQJQGWMYXPAYPYHPADJPWKSIQWSRYQJWNOKCOQIKORDINGWTKSJPPAYBOYVSYJTDYRKBQGGPAYNDT
PSOYRPAQPQNNYQOLDPADJPAYPYHPAQUYBSJ
8:      MFXPQXWXSJWXOZCQOXGORQCIBPIVHXOZJWVJRACIWPWXURPOXVJRKCFFACBRNXJROOZPOOZX
OXGOKPQXISNVMOXWRQCIBOZXQRYQOCOROCJISCMZXNPAOXNVJRAJRIWPQJFROCJIMFXPQXWXQSNCYXZJ
KVJRPIPFVLXWOZXOXGOZCIOVJRHPVRQXPIVMNJBNPHJNQCHMFVSJRIOOZXANXURXISCXQJAPFFOZXMCS
ORNXQOZPOPMMXPNKCOZCIOZXOXGOZPTXARI
9:      LEWOPWVWRIVWNYBPNWFNQPBHAOHUGWNYIVUIQZBHVOVWTQONWUIQJBEEZBAQMWIQNNYONNYW
NWFNJOPWHRMULNWVQPBHANYWPQXPNBNQNBIHRBLYWMOZNWMUIQZIQHVOPIEQNBIHLEWOPWVWPRMBXWYI
JUIQOHOEUKWVNYWNWFNYBHNUIQGOUQPWOHULMIAMOGIMPBGLEURIQHNNYWZMWTQWHRBWPIZOEENYWLBR
NQMWPNYONOLLWOMJBNYBHNYWNWFNYOSWZQH
10:     KDVNOVUVQHUVMXAOMVEMPOAGZNGTFVMXHUTHPYAGUNUVSPNMVTHPIADDYAZPLVHPMMXNMMXV
MVEMINOVGQLTKMVUPOAGZMXVOPWOMAMPMAHGQAKXVLNYMVLTHPYHPGUNOHDPMAHGKDVNOVUVOQLAWVXH
```

ITHPNGNDTJVUMXVMVEMXAGMTHPFNTPOVNGTKLHZLNFHLOAFKDTQHPGMMXVYLVSPVGQAVOHYNDDMXVKAQ
MPLVOMXNMNKKVNLIAMXAGMXVMVEMXNRVYPG
11: JCUMNUTUPGTULWZNLUDLONZFYMFSEULWGTSGOXZFTMTUROMLUSGOHZCCXZYOKUGOLLWMLLWU
LUDLHMNUFPKSJLUTONZFYLWUNOVNLZLOLZGFPZJWUKMXLUKSGOXGOFTMNGCOLZGFJCUMNUTUNPKZVUWG
HSGOMFMCSIUTLWULUDLWZFLSGOEMSONUMFSJKGYKMEGKNZEJCSPGOFLLWUXKUROUFPZUNGXMCCLWUJZP
LOKUNLWMLMJJUMKHZLWZFLWULUDLWMQUXOF
12: IBTLMTSTOFSTKVYMKTCKNMYEXLERDTKVFSRFNWYESLSTQNLKTRFNGYBBWYXNJTFNKKVLKKVT
KTCKGLMTEOJRIKTSNMYEXKVTMNUMKYKNKYFEOYIVTJLWKTJRFNWFNESLMFBNKYFEIBTLMTSTMOJYUTVF
GRFNLELBRHTSKVTKTCKVYEKRFNDLRNMTLERIJFXJLDFJMYDIBROFNEKKVTWJTQNTEOYTMFWLBBKVTIYO
KNJTMKVLKLIITLJGYKVYEKVTKTCKVLPTWNE
13: HASKLSRSNERSJUXLJSBJMLXDWKDQCSJUERQEMVXDRKRSPMKJSQEMFXAAVXWMISEMJJUKJJUS
JSBJFKLSDNIQHJSRMLXDWJUSLMTLJXJMJXEDNXHUSIKVJSIQEMVEMDRKLEAMJXEDHASKLSRSLNIXTSUE
FQEMKDKAQGSRJUSJSBJUXDJQEMCKQMLSKDQHIEWIKCEILXCHAQNEMDJJUSVISPMSDNXSLEVKAAJUSHXN
JMISLJUKJKHHSKIFXJUXDJUSJSBJUKOSVMD
14: GZRJKRQRMDQRITWKIRAILKWCVJCPBRITDQPDLUWCQJQROLJIRPDLEWZZUWVLHRDLIITJIITR
IRAIEJKRCMHPGIRQLKWCVITRKLSKIWILIWDCMWGTRHJUIRHPDLUDLCQJKDZLIWDCGZRJKRQRKMHWSRTD
EPDLJCJZPFRQITRIRAITWCIPDLBJPLKRJCPGHDVHJBDHKWBGZPMDLCIITRUHROLRCMWRKDUJZZITRGWM
ILHRKITJIJGGRJHEWITWCITRIRAITJNRULC
15: FYQIJQPQLCPQHSVJHQZHKJVBUIBOAQHSCPOCKTVBPIPQNKIHQOCKDVYYTVUKGQCKHHSIHHSQ
HQZHDIJQBLGOFHQPKJVBUHSQJKRJHVHKHVCBLVFSQGITHQGOCKTCKBPIJCYKHVCBFYQIJQPQJLGVRQSC
DOCKIBIYOEQPHSQHQZHSVBHOCKAIOKJQIBOFGCUGIACGJVAFYOLCKBHHSQTGQNKQBLVQJCTIYYHSQFVL
HKGQJHSIHIFFQIGDVHSVBHSQHQZHSIMQTKB
16: EXPHIPOPKBOPGRUIGPYGJIUATHANZPGRBONBJSUAOHOPMJHGPNBJCUXXSUTJFPBJGGRHGGRP
GPYGCHIPAKFNEGPOJIUATGRPIJQIGUGJGUBAKUERPFHSGPFNBJSBJAOHIBXJGUBAEXPHIPOPIKFUQPRB
CNBJHAHXNDPOGRPGPYGRUAGNBJZHNJIPHANEFBTFHZBFIUZEXNKBJAGGRPSFPMJPAKUPIBSHXXGRPEUK
GJFPIGRHGHEEPHFCUGRUAGRPGPYGRHLPSJA
17: DWOGHONOJANOFQTHFOXFIHTZSGZMYOFQANMAIRTZNGNOLIGFOMAIBTWWRTSIEOAIFFQGFFQO
FOXFBGHOZJEMDFONIHTZSFQOHIPHFTFIFTAZJTDQOEGRFOEMAIRAIZNGHAWIFTAZDWOGHONOHJETPOQA
BMAIGZGWMCONFQOFOXFQTZFMAIYGMIHOGZMDEASEGYAEHTYDWMJAIZFFQOREOLIOZJTOHARGWWFQODTJ
FIEOHFQGFGDDOGEBTFQTZFQOFOXFQGKORIZ
18: CVNFGNMNIZMNEPSGENWEHGSYRFYLXNEPZMLZHQSYMFMNKHFENLZHASVVQSRHDNZHEEPFEEPN
ENWEAFGNYIDLCENMHGSYREPNGHOGESEHESZYISCPNDFQENDLZHQZHYMFGZVHESZYCVNFGNMNGIDSONPZ
ALZHFYFVLBNMEPNENWEPSYELZHXFLHGNFYLCDZRDFXZDGSXCVLIZHYEEPNQDNKHNYISNGZQFVVEPNCSI
EHDNGEPFEFCCNFDASEPSYEPNENWEPFJNQHY
19: BUMEFMLMHYLMDORFDMVDGFRXQEXKWMDOYLKYGPRXLELMJGEDMKYGZRUUPRQGCMYGDDOEDDOM
DMVDZEFMXHCKBDMLGFRXQDOMFGNFDRDGDRYXHRBOMCEPDMCKYGPYGXLEFYUGDRYXBUMEFMLMFHCRNMOY
ZKYGEXEUKAMLDOMDMVDORXDKYGWEKGFMEXKBCYQCEWYCFRWBUKHYGXDDOMPCMJGMXHRMFYPEUUDOMBRH
DGCMFDOEDEBBMECZRDORXDOMDMVDOEIMPGX
20: ATLDELKLGXKLCNQECLUCFEQWPDWJVLCNXKJXFOQWKDKLIFDCLJXFYQTTOQPFBLXFCCNDCCNL
CLUCYDELWGBJACLKFEQWPCNLEFMECQCFCQXWGQANLBDOCLBJXFOXFWKDEXTFCQXWATLDELKLEGBQMLNX
YJXFDWDTJZLKCNLCLUCNQWCJXFVDJFELDWJABXPBDVXBEQVATJGXFWCCNLOBLIFLWGQLEXODTTCNLAQG
CFBLECNDCDAALDBYQCNQWCNLCLUCNDHLOFW
21: ZSKCDKJKFWJKBMPDBKTBEDPVOCVIUKBMWJIWENPVJCJKHECBKIWEXPSSNPOEAKWEBBMCBBMK
BKTBXCDKVFAIZBKJEDPVOBMKDELDBPBEBPWVFPZMKACNBKAIWENWEVJCDWSEBPWVZSKCDKJKDFAPLKMW
XIWECVCSIYKJBMKBKTBMPVBIWEUCIEDKCVIZAWOACUWADPUZSIFWEVBBMKNAKHEKVFPKDWNCSSBMKZPF
BEAKDBMCBCZZKCAXPBMPVBMKBKTBMCGKNEV
22: YRJBCJIJEVIJALOCAJSADCOUNBUHTJALVIHVDMOUIBIJGDBAJHVDWORRMONDZJVDAALBAALJ
AJSAWBCJUEZHYAJIDCOUNALJCDKCAOADAOVUEOYLJZBMAJZHVDMVDUIBCVRDAOVUYRJBCJIJCEZOKJLV

```
WHVDBUBRHXJIALJAJSALOUAHVDTBHDCJBUHYZVNZBTVZCOTYRHEVDUAALJMZJGDJUEOJCVMBRRALJYOE
ADZJCALBABYYJBZWOALOUALJAJSALBFJMDU
23:     XQIABIHIDUHIZKNBZIRZCBNTMATGSIZKUHGUCLNTHAHIFCAZIGUCVNQQLNMCYIUCZZKAZZKI
ZIRZVABITDYGXZIHCBNTMZKIBCJBZNZCZNUTDNXKIYALZIYGUCLUCTHABUQCZNUTXQIABIHIBDYNJIKU
VGUCATAQGWIHZKIZIRZKNTZGUCSAGCBIATGXYUMYASUYBNSXQGDUCTZZKILYIFCITDNIBULAQQZKIXND
ZCYIBZKAZAXXIAYVNZKNTZKIZIRZKAEILCT
24:     WPHZAHGHCTGHYJMAYHQYBAMSLZSFRHYJTGFTBKMSGZGHEBZYHFTBUMPPKMLBXHTBYYJZYYJH
YHQYUZAHSCXFWYHGBAMSLYJHABIAYMYBYMTSCMWJHXZKYHXFTBKTBSGZATPBYMTSWPHZAHGHACXMIHJT
UFTBZSZPFVHGYJHYHQYJMSYFTBRZFBAHZSFWXTLXZRTXAMRWPFCTBSYYJHKXHEBHSCMHATKZPPYJHWMC
YBXHAYJZYZWWHZXUMYJMSYJHYHQYJZDHKBS
25:     VOGYZGFGBSFGXILZXGPXAZLRKYREQGXISFESAJLRFYFGDAYXGESATLOOJLKAWGSAXXIYXXIG
XGPXTYZGRBWEVXGFAZLRKXIGZAHZXLXAXLSRBLVIGWYJXGWESAJSARFYZSOAXLSRVOGYZGFGZBWLHGIS
TESAYRYOEUGFXIGXGPXILRXESAQYEAZGYREVWSKWYQSWZLQVOEBSARXXIGJWGDAGRBLGZSJYOOXIGVLB
XAWGZXIYXYVVGYWTLXILRXIGXGPXIYCGJAR
```

We get no results with the caesar method os we are going to try to obtain the substitutions with frequency analysis

[3]:
```python
import pandas as pd

def analizeFreq(sentence):
    freq = {}
    num_letters = 0
    for letter in sentence:
        if ord(letter) < 65 or ord(letter) > 90:
            continue
        num_letters += 1
        if letter in freq:
            freq[letter] += 1
        else:
            freq[letter] = 1
    # normalize the frequencies
    for letter in freq:
        freq[letter] /= num_letters
    return freq

# analizing the frequencies of the letters in each ciphered text
freqs = [analizeFreq(x) for x in data]
# creating a dataframe with the frequencies rows as letters and columns as␣
 ↪frequency of each letter in each text
df = pd.DataFrame(freqs, index=[f"Text {i}" for i in range(len(freqs))])
# printing the dataframe
print(df)

combined_freq = analizeFreq("".join(data))
combined_freq = sorted(combined_freq.items(), key=lambda x: (len(x[0]),␣
 ↪-len(x[0]), -x[1]), reverse=False)[:10]
```

```
# creating a dataframe to compare the top 10 most frequent letters of the texts␣
 ↪with the top 10 most frequent letters in the english language
df_1letter = pd.DataFrame()
df_1letter["T1l"] = [x[0] for x in combined_freq]
df_1letter["Txt 1 frq"] = [x[1] for x in combined_freq]

english_freq = sorted(ENGLISH_FREQ.items(), key=lambda x: (len(x[0]),␣
 ↪-len(x[0]), -x[1]), reverse=False)[:10]

df_1letter["E1l"] = [x[0] for x in english_freq]
df_1letter["Eng 1 frq"] = [x[1] for x in english_freq]
```

```
                K         Q         A         V         D         U         W  \
Text 0   0.084592  0.057402  0.033233  0.072508  0.018127  0.039275  0.126888
Text 1   0.088608  0.082278  0.034810  0.066456  0.006329  0.034810  0.126582
Text 2   0.063670  0.056180  0.026217  0.041199  0.041199  0.033708  0.119850

                R         J         X  …         P         E         I  \
Text 0   0.060423  0.018127  0.048338  …  0.027190  0.018127  0.027190
Text 1   0.037975  0.006329  0.069620  …  0.015823  0.025316  0.015823
Text 2   0.063670  0.014981  0.071161  …  0.014981  0.033708  0.026217

                S         N         O         M         B         C         T
Text 0   0.009063  0.027190  0.015106  0.003021  0.009063       NaN       NaN
Text 1   0.003165  0.028481  0.012658       NaN  0.006329  0.006329       NaN
Text 2   0.014981  0.033708  0.014981       NaN  0.003745  0.007491  0.003745

[3 rows x 25 columns]
```

From this we can notice how there is a pretty similar character frequency on all three sentences, which then leeds us to think that fortunetly the substitutions are the same for the three sentences. We are now going to analize letter touples to try to match the $\mathfrak{C} \leftrightarrow \mathfrak{M}$

[4]:
```
df_2letter = pd.DataFrame()

def find_repeated_sequences(input_str):
    sequence_counts = {}
    max_sequence_length = len(input_str)

    for length in range(max_sequence_length, 0, -1):
        for start in range(max_sequence_length - length + 1):
            sequence = input_str[start:start + length]
            count = input_str.count(sequence)
            if count >= 2 and len(sequence) > 1 and len(sequence) < 10:
                sequence_counts[sequence] = count/
 ↪(len(input_str)-len(sequence)+1)
```

```
    sorted_sequences = sorted(sequence_counts.items(), key=lambda x:␣
 ↪(len(x[0]), -len(x[0]), -x[1]), reverse=False)

    for sequence, count in sorted_sequences[:10]:
        print(f'{sequence}: {round(count, 4)}')
    df_2letter["T2l"] = [x[0] for x in sorted_sequences[:10]]
    df_2letter["Txt 2 frq"] = [x[1] for x in sorted_sequences[:10]]

find_repeated_sequences("".join(data))
print(len("".join(data)))
```

```
WH: 0.0405
HF: 0.0383
FV: 0.0263
KQ: 0.0252
KW: 0.0197
WF: 0.0197
WK: 0.0175
VF: 0.0175
YW: 0.0164
FY: 0.0153
914
```

To compare this data with real data we proceed to analyze a large amount of text that also contains no spaces

[5]:
```
import re



with open("sample_text.txt", encoding='utf-8') as f:
    text = f.read()
    # convert all to caps
    text = text.upper()
    # remove all non-alphabetic characters
    text = re.sub(r'[^A-Z]', '', text)
    # save the text
    with open("sample_text_clean.txt", "w") as f2:
        f2.write(text)
# analizing the frequencies of couples of  letters in text
text_freqs = {}
for i in range(len(text) - 1):
    couple = text[i:i + 2]
    if couple in text_freqs:
        text_freqs[couple] += 1
    else:
        text_freqs[couple] = 1
# normalize the frequencies
```

```
for couple in text_freqs:
    text_freqs[couple] /= len(text)
# sort the frequencies
sorted_text_freqs = sorted(text_freqs.items(), key=lambda x: x[1], reverse=True)

df_2letter["E2l"] = [x[0] for x in sorted_text_freqs[:10]]
df_2letter["Eng 2 frq"] = [x[1] for x in sorted_text_freqs[:10]]
print(df_1letter, "\n")
print(df_2letter)
```

```
  T1l  Txt 1 frq E1l  Eng 1 frq
0   F   0.137856   E      0.122
1   W   0.124726   T      0.088
2   K   0.079869   A      0.079
3   Q   0.065646   O      0.072
4   Y   0.063457   H      0.069
5   X   0.062363   I      0.068
6   V   0.061269   N      0.065
7   H   0.056893   S      0.061
8   R   0.053611   R      0.058
9   Z   0.044858   D      0.041

  T2l  Txt 2 frq E2l  Eng 2 frq
0  WH   0.040526  TH   0.031541
1  HF   0.038335  HE   0.029873
2  FV   0.026287  IN   0.019233
3  KQ   0.025192  ER   0.019003
4  KW   0.019715  AN   0.018187
5  WF   0.019715  RE   0.014442
6  WK   0.017525  ND   0.014059
7  VF   0.017525  ED   0.013032
8  YW   0.016429  ES   0.011904
9  FY   0.015334  HA   0.011834
```

With this results and after performing some analysis and research we make the following assignments

- $\mathfrak{C}[KQAVDUWRJXHYZGFPEISNOMBCT] \equiv \mathfrak{M}[INCRYPTOGAHSUBEMDFWLXKVQZ]$

[16]:
```
import random
from collections import OrderedDict

def substitute(sentence, map):
    return "".join(map[letter] if letter in map else letter for letter in␣
 ↪sentence)
freqs = analizeFreq("".join(data))

def force_match(map, diff_map, l1, l2):
```

```
    del diff_map[l1]
    for k in diff_map:
        del diff_map[k][l2]
    map[l2] = l1
    return map, diff_map

change_to   = "INCRYPTOGAHSUBEMDFWLXKVQZ"
change_from = "KQAVDUWRJXHYZGFPEISNOMBCT"

def find_double_letters(sentence):
    # returns indexes
    return [i for i in range(len(sentence)) if (i != len(sentence) - 1 and␣
 ↪sentence[i] == sentence[i + 1]) or (i != 0 and sentence[i] == sentence[i -␣
 ↪1])]

def matchFreq_prob_with_noise(freq1, freq2, noise_factor=0.01):
    map = {}
    for letter in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
        if letter not in freq1:
            freq1[letter] = 0
        if letter not in freq2:
            freq2[letter] = 0
    freq1 = OrderedDict(sorted(freq1.items(), key=lambda x: x[0]))
    freq2 = OrderedDict(sorted(freq2.items(), key=lambda x: x[0]))
    diff_map = {}
    for k1, v1 in freq1.items():
        diff_map[k1] = {}
        for k2, v2 in freq2.items():
            # Add random noise to the frequency differences
            noise = random.uniform(-noise_factor, noise_factor)
            diff = abs(v1 - v2) + noise
            diff_map[k1][k2] = diff
        diff_map[k1] = OrderedDict(sorted(diff_map[k1].items(), key=lambda x:␣
 ↪x[1]))


    for f, t in zip(change_from, change_to):
        map, diff_map = force_match(map, diff_map, t, f)


    for _ in range(len(diff_map)):
        l1, l2 = None, None
        # selecting the letter with the smallest difference
        for k, v in diff_map.items():
            if l1 is None:
                l1 = k
                l2 = list(v.keys())[0]
```

```
            else:
                if diff_map[l1][l2] > v[list(v.keys())[0]]:
                    l1 = k
                    l2 = list(v.keys())[0]
        # deleting the selected letter from the map
        del diff_map[l1]
        for k in diff_map:
            del diff_map[k][l2]
        map[l2] = l1
    return map
for ms in range(len(data)):
    sentence = substitute(data[ms], matchFreq_prob_with_noise(ENGLISH_FREQ,␣
 ↪analizeFreq("".join(data)), noise_factor=0))
    repeated_letters = find_double_letters(data[ms])
    print(sentence)
    # print("".join([(x if x in change_to else ("*" if i in repeated_letters␣
 ↪else "_")) for i, x in enumerate(sentence)]))
    # print(data[ms], "\n")
```

```
INCRYPTOGRAPHYASUBSTITUTIONCIPHERISAMETHODOFENCRYPTINGINWHICHUNITSOFPLAINTEXTARE
REPLACEDWITHTHECIPHERTEXTINADEFINEDMANNERWITHTHEHELPOFAKEYTHEUNITSMAYBESINGLELET
TERSTHEMOSTCOMMONPAIRSOFLETTERSTRIPLETSOFLETTERSMIXTURESOFTHEABOVEANDSOFORTHTHER
ECEIVERDECIPHERSTHETEXTBYPERFORMINGTHEINVERSESUBSTITUTIONPROCESSTOEXTRACTTHEORIG
INALMESSAGE
SUBSTITUTIONCIPHERSCANBECOMPAREDWITHTRANSPOSITIONCIPHERSINATRANSPOSITIONCIPHERTH
EUNITSOFTHEPLAINTEXTAREREARRANGEDINADIFFERENTANDUSUALLYQUITECOMPLEXORDERBUTTHEUN
ITSTHEMSELVESARELEFTUNCHANGEDBYCONTRASTINASUBSTITUTIONCIPHERTHEUNITSOFTHEPLAINTE
XTARERETAINEDINTHESAMESEQUENCEINTHECIPHERTEXTBUTTHEUNITSTHEMSELVESAREALTERED
PLEASEDECODETHISTEXTUSINGANYMETHODYOUFINDADEQUATEYOUWILLFIGUREOUTTHATTHETEXTWASE
NCRYPTEDUSINGTHESUBSTITUTIONCIPHERAFTERYOUFOUNDASOLUTIONPLEASEDESCRIBEHOWYOUANAL
YZEDTHETEXTHINTYOUMAYUSEANYPROGRAMORSIMPLYCOUNTTHEFREQUENCIESOFALLTHEPICTURESTHA
TAPPEARWITHINTHETEXTHAVEFUN
```