expose_vigenere

January 1, 2023

1 1A.algo - Casser le code de Vigenère

La lettre la plus fréquente en français est la lettre E. Cette information permet de casser le code de César en calculant le décalage entre la lettre la plus fréquente du message codé et E. Mais cette même méthode ne marchera pas pour casser le code de Vigenère. Babbage a contourné cet obstacle en étudiant la fréquence des groupes de trois lettres.
Charles Babbage s'est dit qu'un groupe de trois lettres consécutives avaient toutes les chances, à chaque fois qu'il apparaissait dans le message chiffré, d'être la conséquence du chiffrement des mêmes lettres du message avec les mêmes lettres de la clé (voir Cryptanalyse du chiffre de Vigenère). Pour un groupe de quatre lettres, c'est encore plus probable. Par conséquent, l'espacement entre deux mêmes groupes de lettres chiffrées est un multiple de la longueur de la clé. Par exemple, si la répétition d'un groupe est espacée de 30 lettres, puis celle d'un autre de 25, le plus grand diviseur commun de 25 et 30 est 5. La clé possède donc dans ce cas 5 lettres.
La première fonction crypte et décrypte le code de Vigenère connaissant la clé.

[1]:
```
def code_vigenere ( message, cle, decode = False) :
    message_code = ""
    for i,c in enumerate(message) :
        d = cle[ i % len(cle) ]
        d = ord(d) - 65
        if decode : d = 26 - d
        message_code += chr((ord(c)-65+d)%26+65)
    return message_code

def DecodeVigenere(message, cle):
    return code_vigenere(message, cle, True)

def CodeVigenere(message, cle):
    return code_vigenere(message, cle)
```

Les deux fonctions suivantes estime la longueur de la clé.

[2]:
```
def PGCD (m,n) :
    if m <= 0 or n <= 0 : raise Exception("impossible de calculer le PGCD")
    if m == 1 or n == 1 : return 1
    if m == n : return m
    if m < n : return PGCD (m, n-m)
    return PGCD (n, m-n)

def DecodeVigenereLongueurCle (message, mot = 3) :
    """
    cette fonction determine la longueur de la clé, elle
    repère les groupes de trois lettres qui se répète dans le message codé
```

```
    et suppose qu'il y a une très forte probabilité qu'un même groupe de trois
    lettres soit codé avec les mêmes trois lettres du message et les mêmes trois
    lettres de la clé

    message  : .....DES...........DES...........DES.........DES....DES
    cle      : ABCDABCDABCDABCDABCDABCDABCDABCDABCDABCDABCDABCDABCDABCDABCD
    code     : .....EGV.........................EGV.........EGV..........
    distance :      <----------24--------------><----8----->

    la longueur de la clé divise le PGCD de 24 et 8
    """
    al = "".join([ chr(97+i) for i in range(0,26) ]) # l'alphabet
    al = al.upper ()

    # parcours du message pour recenser toutes les positions
    dico = {}
    for i in range (0, len (message)-2) :
        t = message [i:i+mot]
        if t in dico : dico [t].append (i)
        else : dico [t] = [i]

    # on va garder toutes les distances entre
    # entre deux occurrences du meme mot de n lettres
    dis = []
    for d in dico :
        p = dico [d]
        if len (p) > 1 :
            for i in range (0, len (p)-1) :
                #print d, p [i+1] - p [i], " --- ", float (p [i+1] - p [i]) / 8
                dis.append ( p [i+1] - p [i] )

    # on extrait le PGCD
    if len (dis) == 0 :
        raise Exception("impossible de determiner la clé")

    if len (dis) == 1 : return dis [0]

    longueur = PGCD (dis [0], dis [1])
    for d in dis :
        longueur = PGCD (longueur, d)

    if longueur > 5 :
        # si la longueur est suffisante, le resultat a des chances d'etre bon
        return longueur
    else :
        # sinon, on relance l'algorithme avec des mots plus grand
        return DecodeVigenereLongueurCle (message, mot+1)
```

La fonction suivante casse le code de Vigenère connaissance la longueur de la clé.

[3]:
```
def DecodeVigenereCle (code, l) :
    """
    Détermine la cle du message code, connaissant sa longueur,
    on suppose que la lettre E est la lettre la plus fréquente
```

```
    @param      code        message codé
    @param      l           longueur probable de la clé
    @return                 message décodé
    """
    al  = "".join([ chr(97+i) for i in range(0,26) ])
    al  = al.upper ()
    cle = ""
    for i in range (0, l) :
        nombre = [ 0 for a in al]
        sous   = code [i:len (code):l]  # on extrait toutes les lettres
                                        # i, i+l, i+2l; i+3l, ...

        # on compte les lettres
        for k in sous : nombre [ al.find (k) ] += 1

        # on cherche le maximum
        p = 0
        for k in range (0, len (nombre)) :
            if nombre [k] > nombre [p] : p = k

        # on suppose que al [p] est la lettre E code,
        # il ne reste plus qu'a trouver la lettre de la cle
        # qui a permis de coder E en al [p]
        cle += al [ (p + 26 - al.find ("E")) % 26 ]

    return cle
```

Enfin, la dernière fonction qui casse le code en appelant toutes les autres :

[4]:
```
def CasseVigenere(message):
    """
    appelle les deux fonctions @see fn DecodeVigenereLongueurCle et
    @see fn DecodeVigenereCle pour casser le code de Vigenère

    @param      message     message codé
    @return                 message décodé (sans la clé)
    """
    l = DecodeVigenereLongueurCle(message)
    cle = DecodeVigenereCle(message,l)
    decode = DecodeVigenere(message, cle)
    return decode
```

Un petit example avec le dernier jour d'un condamné qu'on récupère depuis le site Gutenberg :

[5]:
```
from ensae_teaching_cs.data import gutenberg_name
text = gutenberg_name("condamne", load=True)
len(text)
```

[5]: 224060

On enlève les caractères indésirables :

[6]:
```
message = text.replace ("\n", "").replace ("\r", "").replace ("\t", "").replace (" ",␣
↪"").replace (",", "")
```

```
message = message.replace (";", "").replace (":", "").replace (".", "").replace ("'",␣
↪"").replace ("\"", "")
message = message.replace ("-", "").replace ("!", "").replace ("?", "").replace ("(",␣
↪"").replace (")", "")
message = message.upper ()
```

On le code une clé :

[7]:
```
message = message [5000:7000]  # on réduit la taille du message
code = CodeVigenere (message, "VIGENERES")
```

Puis on essaye de retrouver la clé :

[8]:
```
cle_code = DecodeVigenereCle (code, DecodeVigenereLongueurCle (code))
cle_code
```

[8]: 'VIGENERES'

[9]:
```

```