Le chiffre de Vigenere

TP de Python – Octobre 2012

Le chiffre de Vigenère

Chapitre: X. Chiffres polyalphabétiques Prérequis: Carré de Vigenère Aller vers :

Blaise de Vigenère (1523-1596), diplomate français, se familiarisa avec les écrits d'Alberti, Trithème et Porta à Rome, où, âgé de vingt-six ans, il passa deux années en mission diplomatique. Au début, son intérêt pour la cryptographie était purement pratique et lié à son activité diplomatique. Une dizaine d'années plus tard, vers 1560, Vigenère considéra qu'il avait mis de côté assez d'argent pour abandonner sa carrière et se consacrer à l'étude. C'est seulement à ce moment-là qu'il examina en détail les idées de ses prédécesseurs, tramant grâce à elles un nouveau chiffre, cohérent et puissant. Bien qu'Alberti, Trithème, Bellaso et Porta en aient fourni les bases, c'est du nom de Vigenère que ce nouveau chiffre fut baptisé, en l'honneur de l'homme qui lui donna sa forme finale.
Le **chiffre de Vigenère** est une amélioration décisive du chiffre de César. Sa force réside dans l'utilisation non pas d'un, mais de 26 alphabets décalés pour chiffrer un message. On peut résumer ces décalages avec un carré de Vigenère. Ce chiffre utilise une clef qui définit le décalage pour chaque lettre du message (A: décalage de 0 cran, B: 1 cran, C: 2 crans, ..., Z: 25 crans).

Exemple: chiffrons le texte "CHIFFRE DE VIGENERE" avec la clef "BACHELIER" (cette clef est éventuellement répétée plusieurs fois pour être aussi longue que le texte clair).

Clair	C	H	I	F	F	R	E	D	E	V	I	G	E	N	E	R	E
Clef	B	A	C	H	E	L	I	E	R	B	A	C	H	E	L	I	E
Décalage	1	0	2	7	4	11	8	4	17	1	0	2	7	4	11	8	4
Chiffré	D	H	K	M	J	C	M	H	V	W	I	I	L	R	P	Z	I

(d'après http://www.apprendre-en-ligne.net)

1. Première partie : lecture, codage, décodage, Indice de coïncidence

Il est utile dans cette première partie de faire le codage et le décodage de Vigenere. En Python cela se fait en quelque lignes seulement . Voici par exemple le codage de Vigenere si le texte est dans la chaîne texte. On suppose que l'on travaille sur le caractère i.

1. `On cherche le décalage de l'alphabet correspondant a la position i :`
   `di = ord(cle[i % lgCle]) - ord('A')`
2. `On applique ce décalage`
   `code[i] =  chr((ord(texte[i]) – ord('A') + di) % 26 + ord('A'))`

Cela peut paraître compliqué, mais c'est bien d'expliciter le passage du caractère au rang du caractère. Un autre moyen serait de convertir tout le message en une suite numérique, mais c'est moins clair, je trouve !!

C'est dans cette première partie également que l'on va lire un fichier et également calculer l'indice de coïncidence d'un texte :

L'indice se calcule avec la formule suivante : $IC = \sum_{q=A}^{q=Z} \frac{n_q(n_q-1)}{n(n-1)}$ avec n le nombre de lettres total du message, n_A le nombre de A, n_B le nombre de B, etc.

En français, l'indice de coïncidence vaut environ 0,0746. Dans le cas de lettres uniformément distribuées (contenu aléatoire sans biais), l'indice se monte à 0,0385.

2. Deuxième partie : Cryptanalyse du code de Vigenère

Ce code secret est très facilement cassé avec un petit programme. Néanmoins il est assez difficile à casser à la main surtout si la clé est assez longue. Nous allons nous placer dans le cadre ou la clé est comprise entre 5 et 20 caractères et le texte est suffisamment long.

2.1 Calcul de la longueur de la clé (Méthode de Kasiskki et Babbage)

En analysant le texte crypté, on s'aperçoit qu'il y a des suites de lettre qui se répètent. Cela provient :

- Soit du fait que c'est la même séquence en clair - donc ces deux séquences répétées sont distantes d'un multiple de la clé
- Soit cela provient de 2 séquences différentes qui par hasard ont produits la même séquence codée (peu probable).

Il faut donc chercher dans le texte crypté toutes les séquences de 3 ou 4 caractères (ou plus) qui se répètent deux fois. La distance entre ces séquences est probablement (mais ce n'est pas sur) un multiple de la taille de la clé.

Texte chiffré :

KQOWE FVJPU JUUNU KGLME KJINM WUXFQ MKJBG WRLFN FGHUD **WUU**MB SVLPS NCMUE
KQCTE SWR**EE** **K**OYSS IWCTU AXYOT APXPL WPNTC GOJBG FQHTD **WXIZA** **YG**FFN SXCSE
YNCTS SPNTU JNYTG GWZGR **WUU**NE JUUQE APYME KQHUI DUXFP GUYTS MTFFS H**NUOC**
ZGMRU WEYTR GKMEE DCTVR ECFBD JQCUS WVBPN LGOYL SKMTE FVJJT WWMFM WPNME
MTMHR SPXFS SKFFS T**NUOC** **ZGMDO** **EOYEE** **K**CPJR GPMUR SKHFR SEIUE VGOYC **WXIZA**
YGOSA ANY**DO** **EOY**JL WUNHA MEBFE LXYVL WNOJN SIOFR WUCCE SWKVI D**GMU**C GOCRU
WGNMA AFFVN SIUDE KQHCE UCPFC MPVSU DGAVE MNYMA MVLFM AOYFN TQCUA FVFJN
XKLNE IWCWO DCCUL WRIFT W**GMU**S WOVMA TNYBU HTCOC WFYTN MGYTQ MKBBN LGFBT
WOJFT WGNTE JKNEE DCLDH WTVBU VGFBI JG

Il faut d'abord chercher des séquences de lettres qui apparaissent plus d'une fois dans le texte.

		Longueurs de clef possibles			
Séquence répétée	Espace de répétition	2	3	5	19
WUU	95			x	x
EEK	200	x		x	
WXIZAYG	190	x		x	x
NUOCZGM	80	x		x	
DOEOY	45		x	x	
GMU	90	x	x	x	

Les facteurs premiers du nombre de caractères entre deux débuts de séquences figurent dans le tableau (ex. 95 = 5 x 19). Il apparaît dans le tableau que toutes les périodes sont divisibles par 5. Tout se cale parfaitement sur un mot-clef de 5 lettres.

Vous construirez une liste des distances entre 2 chaînes répétées. Ensuite à l'aide d'une boucle, vous balayerez les longueur de clé (entre 5 et 20) en testant si la distance est un multiple de la longueur de la clé. Vous choisirez la longueur qui a le meilleur score.

2.2 Découverte de la clé

Quand on connaît la longueur *n* de la clé, il faut trouver le mot clé. On commence par découper le texte T en n sous textes : T_0, T_1, … T_{n-1}.

T_i = T[i], T[i +n], T[i + 2n], ….

La particularité des textes T_i est qu'ils sont codés avec le même alphabet (qui correspond a un décalage de k_i de l'alphabet). Trouvez la clé est équivalent à trouver les décalages k_0, k_1, … k_{n-1}.

Une première méthode (simple) consiste a calculer pour chacun des textes la lettre la plus fréquentes (On suppose alors que c'est un « E » et on en déduit le décalage correspondant). Un défaut de cette méthode est que si le texte est trop court ou si la clé est trop longue, on ne peut pas assurer que cela soit toujours le « E » qui soit la lettre la plus fréquente.

Nous allons utiliser une deuxième méthode : Pour cela nous allons calculer successivement : $d_1 = k_1 - k_0$, $d_2 = k_2 - k_0$, …, $d_{n-1} = k_{n-1} - k_0$

Attention, ce point est un peu délicat à comprendre : Pour calculer di, nous allons boucler sur tous les décalages possible : de 0 à 25 :

```
indice = 0
decalage = 0
pour d variant de 0 a 25 faire
        t = concaténation de T0 avec Ti auquel on a appliqué le décalage d
        Calcul de l'indice de coïncidence du texte t
        If (IC(t) > indice)
                indice = IC(t)
                decalage = d
di = decalage
```

En effet quand on trouve le bon décalage alors cela veut dire que les deux textes T_0 et Ti sont codés avec exactement le même alphabet et correspondent donc a du français : l'indice de coïncidence est alors maximal.

Il ne reste plus maintenant qu'a fixer l'origine pour cela on concatène tous les textes Ti chacun avec son décalage di et on cherche la lettre la plus fréquente qui va correspondre alors à « E » ce qui nous permet de fixer le décalage global.

Il suffit maintenant qu'a afficher le texte en clair.

Voyons si votre programme marche avec le texte suivant :

```
TJAEZ JMTQT NRSQR YBGWE WRERH ULVLR NMOFV JCTBJ XTVYH TLGXE RXPNH PVKPR LRTII
CPEQG EDVLR XMPGS DGHRG RVOQF EYKXT UYSCD YIDWV BSCRH DRJBJ VUYQI PPESQ VIJRG
IEKZN TQCPE VEIFD RDRUR CCXSC JRTDJ IARDN TQHVS RHMRE VFGWF LBGQI OEIET TIVVQ
CCESR CVEZD RYRJV ESFIY WTEHS IKYMW PPGJY BEIQP FFILR EHGRI BEIFE SLIYH ZGOIV
JZIFP MVCTR PLXSC JGHRG GYVZH YBGGI BKAGT ELOKB FPVWI CHOQJ WRGXR WCUTI CMTRH
QRUMY PGPIW ZYIFT QSCIV PLVEZ XMRRI IDFCY PCUHE WWLNK ECMME LGPYV NIDHC ITFYH
TJNIH NWAVC XARKD FCUIX KMEAI SKDIP SGPEP NQEAI ETTIO WCREV UEMBG RVAWH CLGII
CPACT VJGMP EGXIH DRTEX WKVTR YBGQE RRJRE SIKIV LKGWP NZRRH YEVKH TJNIV NIDHI
LVFCW LTCMW UEIFH IJRUB WJKVY WQOER IRLLR XYFIP NMNRB EZJIY TLUXE WXMRB IFLTN
RMTKI NQEYT IUVAZ TCVXI BHUTP XVRCG ZSELE VSNCP PRZAW PRTIW BEIYA MJRBG PLVMJ
JGEDJ MJVXN DQCMX MIXGG EFILV YYKVI NRMBX YEGTN TQKVH NPIPX ILOUN GYKXI WZAUX
MJFTR DYPWP JROGX SEUMF LACYW NMLZP ZRZBN FQUMX XXRRC HLCMF GGEMW BMTHS IJUMY
LTKIM WHISU IIVVG PQUIW MISNH XIVAV YMHJI WWISH WRSZV PTGXI RPLHH SZIMQ PJCQI
VIFNR SEHCB ACTIP JQOHG IEDME PKRPM BWAAI HLEMR DQGRG NTRRR MVLAR ZSRPY CSTPT
XKVMF DCPGI WITNX XGRAR YKQMI UPERI EZKUB THCZE RWCRH WVUMZ PQGRX RVMRS MFTZR
NMPXM WKEAI QFIBR WBQYE EEIGE YDVDR YGTGI CXECJ MJJIA ECLSM NNEFT RKRQF BSGPP
NITNX XCZMR LSISY CHUGW IVKLH RYVIE DQAVH ULVTY PJGHI YESFP MKZVS TLKQI WXNRS
IMRQG AYUIX AIDRB IDVVN ESTIH XYVRC EZKMY WCSYI BMGAX JZRQG PJNIS DPACE VVYMA
OCTNI KSIFJ RVJMP ZLFIK XVGRT SLAMA PRTSY EIRVT RUVXY FQSYI MENFA EGIMZ TCTIY
WITED MJZMZ POWMQ JTPBG XVLVC PSOSM WWQHT PRJMP ZLFIM UISGI IDGAD FCLIQ JVRRI
ICRDR CRWHY KVEHK EXVAR XZNIH RQIAJ IIZTR DREPE RVQHT PRMME TRGUY NNEPW IITPR
YCUXT JWEAA YZDIV DCPQS RMLYN EVMMV WJGIQ JMSAT PRTWA YYKXT JWEGC IGVCG BSGVI
YITRG MEUMS TLKQI WXAIT GUVUB TLUIR VSIAH HVWWE NCEIQ NQEGT QFZOA LEGUY NNEAT
WRZAC LQKRX NVPET XVIMG BSGNI EIUKP YDFQA DNQYZ XMRYJ MIVLR XYPHI AITET XIFCI
PPKRX JGTNB EUZAC ZQKXM XRTBJ XRCPR FPGTS DVUAT GCRQE NGUWI VINGS ITZAV QHGTS
BILNI EJJMR EKGXS DVNRK IIJUB YCUTV RXCRH XRCCV OCVVS DZEEA EMVZV ECOEM BGOZB
IEKOE LTGMR LIRGX XLUMG ZSVIW UISSD MJHCR WCUTV RXSRH IEKLR AYUWI YERYJ MDVUR
BSCRH UYIYT GYVZP SCWVI BXTBJ XVEAR XZNIP NTALH SSJKH CMWMP MSIGR LVIKU PPGXS
DXOHI WFEJN RYIIR NPUVH IIRLR CGGRG QIRPW IIGIF DCWPI VINGR VVVZV WCUXI WJAPT
HVHCR WOWIG QSSRF YZEMF ENCWI WGOET IKHCR DCWPM UTEHI VVRTV DCTTY RWFNX VVVVG
CCTHE WWSNA YDZME P
```

Ci joint également le programme correspondant en Python3. Je suis loin d'être un expert en Python, mais le programme marche !!

Bon courage Michel Van Caneghem

```
# CRYPANALYSE DE VIGNERE
# Michel Van Caneghem -- Octobre 2012

# Lecture du fichier crypté
#-------------------------
def lireTexte(nomFichier) :
    try:
        fichier = open(nomFichier, 'r')
    except IOError:
        print (nomFichier + " : ce fichier n'existe pas")
        return None
    texte = ''
    while True:
        ligne = fichier.readline()
        if ligne == '':
            break
        for mot in ligne.split():
            texte = texte + mot
    fichier.close()
    return texte

# On va travailler avec l'alphabet : ALPHABET
ALPHABET = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

# Calcul de l'indice de coincidence
#---------------------------------
def IC(texte) :
       occ = { }
       n = len(texte)
       for car in texte:
              if car in occ:
                     occ[car] = occ[car] + 1
              else:
                     occ[car] = 1
       ic = 0.0
       for car in occ.keys(): ic += occ[car]*(occ[car] - 1)/n/(n-1)
       return ic

def decal(car, d) :
       return chr((ord(car) - ord('A') + d) % 26 + ord('A'))

def convertCarCode(car) :
       return ord(car) - ord('A')

# Construction d'un texte avec un alphabet décalé de d
#-----------------------------------------------------
def cesar(texte, d) :
       texteDecale = ''
       for car in texte : texteDecale = texteDecale + decal(car, d)
       return texteDecale

# Calcul de la longueur de la cle
#--------------------------------
def longueurCle(texte) :
       distances = []
       n = len(texte)
       for lgSC in range(4, 10) :
              for i in range(0, n - lgSC) :
                     sc = texte[i : i + lgSC]
                     j = texte[i+1 :].find(sc) + 1
                     if j > 0 : distances.append(j)
       # Maintenant que l'on a le tableau des distances, on va boucler sur la longueur de la clé
       # On va s'intéresser a la longueur qui donne le meilleur score
       # mais cela n'est pas sur surtout quand la longueur de la clé a des petits diviseurs
       nbMax = 0
       for lg in range(5,20) :
              nb = 0
              for d in distances :
                     if d % lg == 0 : nb = nb + 1
              if nb >= nbMax :
```

```
                        nbMax = nb
                        lgCle = lg
        return lgCle

# Recherche de la cle
#--------------------
def rechercheCle(texte, lgCle) :

        # 1 - on va découper le texte en lgCle morceaux
        # on profite d'une fonction de python sur les sequences
        texte0 = texte[0:len(texte):lgCle]
        decalages = [0]
        for k in range(1, lgCle) :
                IcMax = 0
                decalage = -1
                textek = texte[k:len(texte):lgCle]
                for d in range(0,26) :
                        Ic = IC(texte0 + cesar(textek, d))
                        if Ic > IcMax :
                                IcMax = Ic
                                decalage = d
                decalages.append(decalage)

        # 2 - maintenant on va reprendre le texte et appliquer tous les décalages
        # et chercher la lettre la plus fréquente

        occ = { }
        for i in range(0, len(texte)) :
                car = cesar(texte[i], decalages[i % lgCle])
                if car in occ:
                        occ[car] = occ[car] + 1
                else:
                        occ[car] = 1
        max = 0;
        for car in occ.keys():
                if occ[car] > max :
                        max = occ[car];
                        carE = car;
        delta = (26 + ord(carE) - ord('E') ) % 26

        # 3 - maintenant on connait la cle
        cle = ''
        for d in decalages :
                cle = cle + ALPHABET[(delta - d) % 26]
        return cle

# On decrypte Vigenere avec la cle
#---------------------------------
def decrypte(texte, cle) :
        lgCle = len(cle)
        clair = ''
        for i in range(0, len(texte)) :
                code = (convertCarCode(texte[i]) - convertCarCode(cle[i % lgCle])) % 26
                clair = clair + ALPHABET[code]
        return clair

# Debut du programme de cryptanalyse de Vigenere
#-----------------------------------------------
def casse(texte) :
        lgCle = longueurCle(texte)
        print("Longueur de la clé = ",lgCle)
        cle = rechercheCle(texte, lgCle)
        print("Cle = ", cle)
        clair = decrypte(texte, cle)
        print(clair)

texte = lireTexte('code.txt')
casse(texte)
```