TP : Chiffre de Vigenère

Le codage de Vigenère est un système de chiffrement polyalphabétique, c'est à dire que chaque lettre du texte clair subira un décalage différent.

Plus précisément, on utilisera un texte en tant que *clé*, chaque lettre étant associée à un nombre entier. Dans tout ce qui suit on utilisera la correspondance suivante :

A=0, B=1, C=2, ... , Z=25

Et on ne considérera que des textes formées d'une des 26 lettres majuscules sans espaces.

On définit la variable globale :

```
alphabet=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
          'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
```

et la fonction

```
def charNum(c):
    if c in alphabet:
        return ord(c)-ord('A')
    else:
        return 23 #X pour les caractères inconnus
```

qui renvoie la position dans la liste `alphabet` du caractère `c`

Pour décaler la première lettre du texte à coder on utilisera la première lettre de la clé, pour la seconde lettre du texte initial, la seconde lettre de la clé, etc. Lorsque toutes les lettres de la clé sont épuisées on recommence avec la première lettre de la clé. Le décalage des lettres est circulaire, c'est à dire que lorsqu'on dépasse la dernière lettre on retourne à la première.

Par exemple le chiffrement avec la clé `ABCD` du texte `CECIESTUNEPHRASEDETEST` donnera

```
CECIESTUNEPHRASEDETEST
ABCDABCDABCDABCDABCDAB
CFELETVXNFRKRBUHDFVHSU
```

Le but de ce TP est d'écrire des fonctions de chiffrement et déchiffrement utilisant cet algorithme puis de mettre en œuvre une méthode permettant de trouver la clé d'un texte chiffré par une analyse statistique.

1 Chiffrement et déchiffrement

1. Écrire une fonction `decalageDroite(car,cle)` qui prend en paramètre deux caractères `car` et `cle` et qui renvoie le caractère `Car` décalé vers la droite par la clé `cle`. Par exemple `decalage('F','C')` doit retourner le caractère `'H'`.
2. Écrire une fonction `decalageGauche(car,cle)` qui prend en paramètre deux caractères `car` et `cle` et qui renvoie le caractère `Car` décalé vers la gauche par la clé `cle`. Par exemple `decalage('F','C')` doit retourner le caractère `'D'`.
3. Écrire une fonction `chiffrer(texteClair,cle)` qui prend en paramètre deux chaînes de caractères `texteClair` et `cle` et qui renvoie la version de `texteClair` chiffrée avec la clé `cle`.
4. Écrire une fonction `dechiffrer(texteChiffre,cle)` qui prend en paramètre deux chaînes de caractères `texteChiffre` et `cle` et qui renvoie la version de `texteChiffre` déchiffrée avec la clé `cle`.

2 Cryptanalyse

2.1 Fonctions utiles

5. Écrire une fonction `apparitions(texte)` qui prend en paramètre une chaîne de caractères `texte` et qui renvoie une liste `L` dont chaque élément `L[i]` est le nombre d'apparitions de la lettre `i` dans le texte.

2.2 Détermination de la taille de la clé

Pour déterminer la taille de la clé, nous allons utiliser la technique de l'*indice de coïncidence*.

Pour un texte donné, l'indice de coïncidence correspond à la probabilité, lorsqu'on choisit deux lettres au hasard dans le texte de choisir deux fois la même lettre. On peut montrer que si n_i est le nombre d'apparitions de la lettre numéro i dans un texte contenant n caractères au total, alors l'indice de coïncidence IC est donnée par :

$$IC = \sum_{i=0}^{25} \frac{n_i(n_i-1)}{n(n-1)} \quad (1)$$

6. En utilisant la fonction `apparitions(texte)`, écrire une fonction `IC(texte)` qui renvoie l'indice de coïncidence de la chaîne de caractères `texte`.

L'indice de coïncidence à plusieurs propriétés intéressantes dans l'analyse d'un texte chiffré.

- Il reste le même lorsque toutes les lettres d'un texte subissent un décalage identique (un décalage ne fait que changer l'ordre des termes de la somme de l'équation (1))
- Pour un texte aléatoire la valeur de l'indice de coïncidence doit tendre vers $\frac{1}{26} \simeq 0.0385$ et chaque langue a un indice de coïncidence moyen, pour le français c'est environ 0.0778.

Donc un texte français dont toutes les lettres ont subit le même décalage (chiffre de César) aura un indice de coïncidence proche de 0.0778 alors que si les lettres ont subi un décalage différent, l'indice de coïncidence sera inférieur.

Dans un texte chiffré avec une clé de longueur N, si on ne conserve qu'une lettre sur N alors chaque lettre a subit le même décalage et l'indice de coïncidence doit être proche de 0.0778.

Pour déterminer la longueur de la clé, on utilisera l'algorithme suivant :

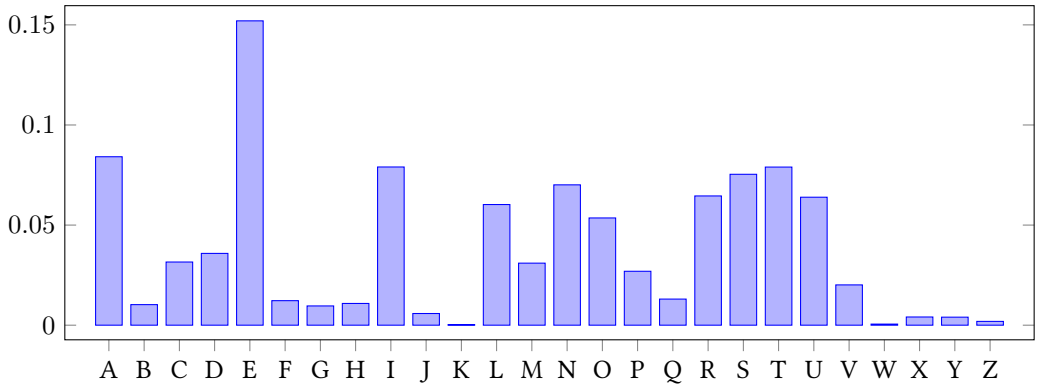
$N_{max} = 10$
pour $i = 0$ **à** N_{max} **faire**
  $x =$indice de coïncidence moyen du texte en ne prenant qu'un caractère sur i
  si $x > 0.06$ **alors**
    renvoyer i
  fin si
fin pour
renvoyer -1

7. Écrire une fonction `trouveLongueurCle(texte)` qui prend en paramètre la chaine de caractère `texte` et qui renvoie la longueur probable de la clé utilisée pour le chiffrer.

2.3 Trouver la clé

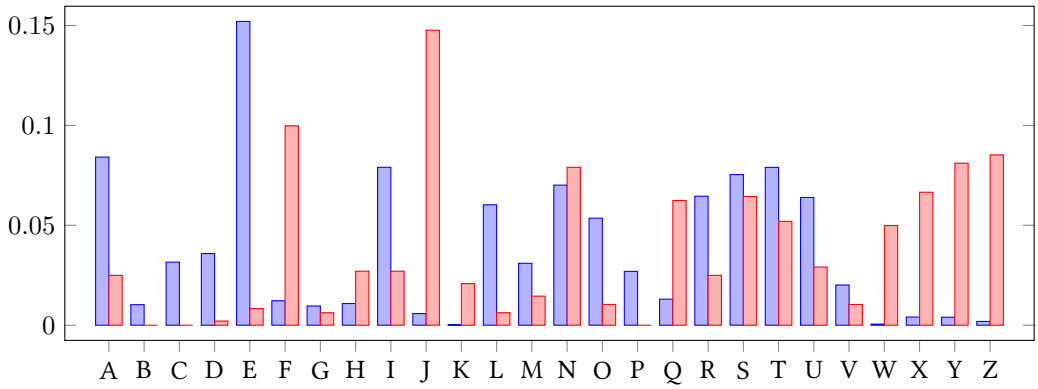
Connaissant la longueur N de la clé, on se ramène à devoir déchiffrer un nombre N de messages codés avec une clé de longueur N, chaque message étant constitué du texte chiffré dont on n'a conservé qu'un caractère sur N, en commençant pour chacun par un caractère différent.

Pour déterminer le décalage d'un texte nous allons en faire une analyse fréquentielle. En effet, pour une langue donnée, les fréquences d'apparition de chaque lettre dans un texte sont assez stables. Par exemple on peut représenter l'*histogramme* des fréquences d'apparition des lettres en français :



Si toutes les lettres ont subit le même décalage, l'histogramme des fréquences d'apparition de chaque lettre sera décalé de la même manière. On peut donc déterminer le décalage utilisé dans un texte chiffré en déterminant la valeur dont il faut décaler l'histogramme d'apparition des lettres pour qu'il ressemble le plus à celui de la langue du texte d'origine.

Sur le graphique ci-dessous on représente les fréquences d'apparition des lettres dans un texte français de référence ainsi que celles d'un texte chiffré avec un décalage égal à 5 :



8. Utiliser la fonction `apparitions(texte)`, écrire une fonction `freqences(texte)` qui prend en paramètre une chaîne de caractères `texte` et renvoie la fréquence d'apparition de chaque lettre. La fréquence d'apparition f_i de la lettre i est $f_i = \frac{n_i}{n}$ où n_i est le nombre d'occurrences de la lettre i dans le texte et n le nombre total de caractères.

Les fréquences d'apparition des lettres dans la langue française peuvent être déterminée en analysant un texte en français assez long. On souhaite pouvoir analyser un texte contenu dans un fichier texte.

9. Écrire une fonction `litFichier(nomFichier)` qui prend en paramètre un nom de fichier contenu dans la chaîne de caractères `nomFichier` et qui renvoie une chaîne de caractères contenant le texte du fichier. Lire la documentation de python concernant les fichiers si nécessaire.

Pour déterminer si deux histogrammes sont semblables on peut essayer de construire une grandeur qui sera faible pour deux histogrammes proches et élevée lorsqu'ils sont très différents. Par exemple on peut utiliser la formule suivante pour calculer la distance d entre deux distributions de fréquences F_1 et F_2 :

$$d(F_1, F_2) = \sum_{i=0}^{25} (F_2[i] - F_1[i])^2 \tag{2}$$

Il suffit alors de tester tous les décalages possibles entre 0 et 25 et garder celui qui donne la distance d la plus faible. En répétant la même opération pour chaque caractère de la clé on peut déterminer la clé de chiffrement et le texte déchiffré.

10. Écrire une fonction `decalage(F,i)` qui prend en paramètre une liste `F` de longueur `len(F)`=26 contenant les fréquences d'apparition des lettres de l'alphabet ainsi qu'un entier `i` et qui renvoie la liste `F` dont les valeurs ont été décalées de `i` places vers la gauche.

11. Écrire une fonction `dist(F1,F2)` qui prend en paramètre deux histogrammes de fréquences `F1` et `F2` et qui renvoie la distance entre les deux calculée à l'aide de l'équation (2).

12. Écrire une fonction `trouveDecalage(texte)` qui prend en paramètre un texte en français dont toutes les lettres ont subit un décalage identique (chiffre de César) et qui en détermine le décalage à l'aide des fonctions précédentes.

13. Écrire une fonction `trouveCle(texteChiffre)` qui utilise les fonctions prend en paramètre une chaîne de caractères `texteChiffre` chiffré avec la méthode de Vigenère et qui en détermine automatiquement la clé.