IF202 CRYPTOLOGIE	TP1	22/03/2024

Préambule

Le but de ce TP est de mettre au point des méthodes de décryptage des codes monoalphabétiques (partie 1) puis du code de Vigenère (partie 2) en s'appuyant sur la distribution de probabilité des lettres de la langue utilisée. On pourra aussi vérifier la "loi de Zipf" (partie 3).

Le choix du langage est Python. Les fichiers à étudier sont disponibles à l'adresse :
`https://dept-info.labri.fr/~ges/ENSEIGNEMENT/CRYPTOLOGIE/crypto.html`
Les squelettes des principales fonctions à écrire se trouvent dans le fichier `start.py`.
Ne modifiez pas les prototypes des fonctions.
Il est fortement recommandé de tester unitairement vos fonctions sur des entrées judicieusement choisies au fur et à mesure que vous les écrivez.

Dans les parties 1,2, les messages en clair sont formés de 28 symboles :
- les 26 lettres minuscules
- les deux symboles `" "` (l'espace) et `"\n"` (le saut de ligne) que nous appellerons des séparateurs.

NB : l'apostrophe est absente. Elle a été éliminée par un prétraitement (qui a donc raccourci le message initial). La fonction d'encodage laisse les séparateurs invariants et remplace chaque lettre minuscule par une lettre minuscule.

1. Encryptage monoalphabétique

Alice envoie à Bob un message, encrypté selon le protocole suivant : elle utilise une clé `KEY` qui est un mot de longueur 26, composé de lettres minuscules et comportant exactement une fois chaque lettre minuscule. Elle applique au message en clair `clear.txt` la permutation définie par `KEY`, écrit le résultat dans `cipher.txt` et l'envoie à BOB.
Q1 Ecrire la fonction `cipher_mono(source:str,dest:str,keydico:dict)` permettant de réaliser cet encodage.

Pour deviner la clé `KEY` vous comparez la fréquence des lettres dans `cipher.txt` avec la fréquence des lettres en Francais.
Q2 Ecrire la fonction `decrypt_on_the_fly(cipher:str)` et l'utiliser pour progresser vers la découverte de `KEY`.
Q3 Eventuellement, ecrire la fonction `superp_on_the_fly(cipher:str)` et l'utiliser pour mieux vsualiser le placement des lettres encore non-décodées dans les mots courts.
Q4 Calculer la clé `KEY` et le message `clear.txt`

2. Encryptage de Vigenère

Maintenant Alice utilise le protocole suivant (dit "de Vigenère") : elle définit une clé `KEY_poly` qui est un mot, sur l'alphabet minuscule, de longueur $<= 20$. Elle traduit cette clé en suite de nombres, en remplaçant chaque lettre par son rang, et encode chaque facteur f de longueur `KEY` par la somme

de f et de `KEY`, lettre a lettre, modulo 26 (en traduisant les lettres du message en entiers). Le résultat est mis sous forme de mot sur les lettres minuscules.

2.1 Entropie

On rappelle que l'entropie d'une mesure de probabilité $\Pr : \mathcal{P}(\Omega) \to [0,1]$ sur un ensemble fini d'événements élémentaires Ω est définie par :

$$S(\Pr) := \sum_{\omega \in \Omega} -\Pr(\{\omega\}) \log_2(\Pr(\{\omega\})).$$

Q5 Montrer que la fonction $x \mapsto -xlog(x)$ est concave.
Q6 Montrer que, si $\Pr_1, \Pr_2$ sont des mesures de probabilité sur le même ensemble Ω et si $\alpha_1, \alpha_2 \in [0,1]$ avec $\alpha_1 + \alpha_2 = 1$ alors

$$S(\alpha_1 \Pr_1 + \alpha_2 \Pr_2) \geq \alpha_1 S(\Pr_1) + \alpha_2 S(\Pr_2).$$

Q7 Supposons que m est une suite de symboles dans $[0, 25]$, de longueur ℓ. Soient $I, J \subseteq [0, \ell - 1]$ tels que

$$I \cup J = [0, \ell - 1] \quad \text{et } I \cap J = \emptyset. \tag{1}$$

On note Pr_I (resp $\Pr_J$) les lois de probabilité sur $[0, 25]$:

$$\Pr_I(\omega) = \frac{\#\{i \in I \mid m[i] = \omega\}}{\#(I)}, \ \ \Pr_J(\omega) = \frac{\#\{j \in J \mid m[j] = \omega\}}{\#(J)}.$$

Vérifier que $\Pr = \alpha_I \Pr_I + \alpha_J \Pr_J$ pour des nombres $\alpha_I, \alpha_J \in [0,1]$ avec $\alpha_I + \alpha_J = 1$.
Q8 Montrer que, si I, J vérifient les hypothèses (1) alors

$$\min\{S(\Pr_I), S(\Pr_j)\} \leq S(\Pr).$$

2.2 Cryptanalyse du code de Vigenère

Remarquons que l'entropie d'un message en langue naturelle est significativement plus basse que l'entropie de la loi uniforme. Déduire de ces considérations une stratégie de recherche de la longueur de la clé d'Alice à partir du cryptogramme c.
Aide : on tentera de découper $[0, \ell]$ en parties $I_0, I_1, \ldots, I_r, \ldots, I_{\lambda-1}$ où I_r est formé des positions de $[0, \ell]$ qui ont la même classe r modulo la longueur de la clé λ. On calculera l'entropie des lois empiriques sur $[0, 25]$ associées aux sous-mots c obtenus par restriction à $I_0, I_1, \ldots, I_r, \ldots, I_{\lambda-1}$.
Q9 Ecrire les fonctions esquissés dans `start.py`. En déduire le calcul de la longueur de la clé.
Q10 Trouver la clé, par comparaison des lois de probabilité des lettres en Francais et dans chaque sous-mot de c restreint à un sous-ensemble I_r ($r \in [0, \lambda - 1]$).

3. Loi de Zipf

Cette loi empirique est décrite sur wikipedia. Est-elle valide pour la loi de probabilité empirique définie par `clear.txt` (ou par `clear_poly.txt`) ?