TP d'algorithmique

Décryptage

Matteo Wei

1 Le chiffre de Vigenère

Le chiffre de Vigenère est une méthode cryptographique pour coder des messages décrite pour la première fois au XVIe siècle par l'Italien Giovan Battista Bellaso, puis par le Français Blaise de Vigenère. L'idée est de camoufler un message en décalant les lettres à partir d'une clé (une description plus précise sera donnée en dessous).
Le but de ce problème est de développer un algorithme permettant de casser le chiffre de Vigenère.
Un certain nombres de choses sont dans le fichier Vigenere.py, il faut le télécharger et l'importer dans le fichier utilisé, par exemple en mettant `from Vigenere import *` au début.

1.1 Première approche : le chiffre de César

On commence par étudier un chiffre plus simple, le chiffre de César.
Pour deux entiers a et b avec $b \neq 0$, on note $a\%b$ le reste de a dans la divison euclidienne par b (notation cohérente avec Python).
La méthode est la suivante : on a un message, écrit sur un alphabet (ici l'alphabet latin minuscule auquel on a rajouté le caractère ' '), et une clé, qui est une lettre de l'alphabet (disons la i-ème, avec la convention que a est la 0-ème lettre).
On remplace alors chaque lettre du message par celle qu'on obtient en la décalant de i crans dans l'alphabet, en revenant au début si on dépasse 27. Cela revient à dire que l'on remplace une occurrence de la j-ème lettre de l'alphabet par une occurrence de la $(j+i)\%27$-ème.
Par exemple, `abc xyz` encodé par `c` devient `cdebz a` (on décale chaque lettre de deux crans, en se souvenant que le blanc est le 26-ème caractère).
Une autre manière de le voir est que la clé est la lettre que devient `a`.

Question 1. *Quel est le résultat du chiffre de César, pour le mot* `cryptographie`*, et la clé* `e` *?*

Une chose importante à remarquer est que le chiffre de César remplace toujours une lettre donnée par la même lettre : par exemple, tous les `a` du message sont remplacés par

la clé. Ainsi, le message codé conserve une trace du message d'origine : les fréquences d'apparitions des lettres.

Question 2. *Soit c une lettre. On note $f(c)$ sa fréquence dans le message d'origine, et $f'(c)$ sa fréquence dans le message encodé. On note k le numéro dans l'alphabet de la clé. Montrer que pour toute lettre c, on*

$$f'((c+k)\%27) = f(c).$$

On peut se reposer sur ces observations pour dégager une stratégie d'attaque, dite fréquentielle. Si le message est assez long, on peut s'attendre à ce que les fréquences des lettres dans le message soient proches de leurs fréquence d'apparition en français. On peut donc essayer de trouver le décalage qui permet d'approximer au mieux ces fréquences du français, à partir des fréquences mesurées dans le message crypté. Cela devrait permettre de calculer la clé, puis de décrypter le message.
Pour essayer de rendre précis l'idée de bonne approximation, on introduit une notion de distance entre listes de nombres de même longueur. Si $u = [u_0, \dots, u_{n-1}]$ et $v = [v_0, \dots, v_{n-1}]$ sont deux listes de longueur, on note $d(u,v) = (u_0 - v_0)^2 + \cdots + (u_{n-1} - v_{n-1})^2$ (c'est la généralisation à n dimensions du (carré de) la distance dans le plan). Le but va donc être de trouver le décalage qui minimise la distance aux fréquences théoriques.

Question 3. *Compléter le code ci-dessous pour que la fonction* `dist` *calcule la distance entre deux listes.*

```
def dist(u, v):
  n, d = len(u), 0
  for i in range(n):
    d += ??
  return d
```

Question 4. *Compléter le code ci-dessous pour que la fonction* `break_cesar` *casse le chiffre de César.*

```
def break_cesar(cypher):
  # cypher est le message codé à décrypter.
  n = len(cypher)
  occurences = [0 for a in alpha]
  for c in cypher:
    occurences[letter_to_number[c]] += 1
    # On calcule le nombre d'occurences de chaque lettre,
    # en itèrant sur le message codé.
    # Pour chaque lettre de cypher,
    # on incrémente de 1 la case correspondante dans occurences.
  frequences = [1 / n * o for o in occurences]
  ?? # Compléter, il faut trouver le décalage i qui minimise la distance.
  return [alpha[(letter_to_number[c] - i) % len(alpha)] for c in cypher]
  # Renvoie le message décrypté.
```

1.2 Le chiffre de Vigenère

Le principe du chiffre de Vigenère ressemble à celui du chiffre de César, seulement cette fois-ci la clé est un mot et pas une lettre.
L'idée est la suivante : on écrit en dessous du message à encoder la clé, autant de fois qu'il faut pour recouvrir le message d'origine. Ensuite, on code chaque lettre du message avec la méthode du chiffre de César, en prenant pour clé la lettre écrite en dessous.
Notamment, si deux indices diffèrent d'un multiple de la longueur de la clé, alors les lettres du message correspondant sont encodés avec la même lettre de la clé.
Avec des formules, le chiffre consiste, en notant m la longueur de la clé k, et pour tout $l \leqslant m$, k_l le numéro de la l-ème lettre de la clé, à remplacer la i-ème lettre du message, qui est la n_i-ème dans l'alphabet, par la $(n_i + k_{i\%m})\%27$-ème.
Par exemple, pour le message `abc xyz`, et la clé `abc`, on procède (à la main) de la manière suivante :
On commence par écrire la clé sous le message en la répétant :
`abc xyz`
`abcabca`
Ensuite on ne fait rien sur la première lettre car le `a` de la clé correspond à l'absence de décalage ; en revanche, la deuxième est au dessus d'un `b`, donc elle est remplacée par la suivante (dans l'alphabet), donc par `c`. La troisième est remplacée par celle deux rangs plus loin, puis on revient sur un `a`, donc la quatrième n'est pas changée, *etc.*
Au final, on obtient `ace y z`.

Question 5. *Quel est le résultat du chiffre de Vigenère, pour le mot* `cryptographie`*, et la clé* `abc` *?*

On ne peut plus faire d'attaque fréquentielle directe : comme les lettres sont encryptées différemment selon leur position, il n'a pas de raison que les fréquences soient conservées. En revanche, si on connaît la longueur m de la clé, on peut séparer le message en m bandes, où la i-ème bande est constituée des lettres encodées à l'aide de la i-ème lettre de la clé. Mais alors cette bande est encodée avec le chiffre de César, et on peut la casser avec l'approche d'avant. Il ne reste plus qu'à recoller le tout pour reconstituer le message.

Question 6. *Écrire une fonction* `break_vigenere`*, qui prend deux arguments, un message à décoder et une longeur de clé, telle que* `break_vigenere(cypher, m)` *essaie de décoder* `cypher`*, en supposant une longueur de clé* `m`*.*

Il ne reste donc plus qu'à trouver la longueur de la clé.

1.3 Trouver la longueur de la clé

Là encore, on se repose sur une analyse fréquentielle. L'idée est que le français a tendance à souvent produire certaines suites de symboles (par exemple, ' `a` ', ' `est` ', ' `et` ', *etc.*), et que d'une manière générale un texte en français a plus de répétition de motifs qu'un texte aléatoire. Mais si un motif apparaît souvent dans le message, on devrait raisonnablement souvent trouver des paires d'occurences du motif dont les débuts sont

séparés d'un multiple de la longueur de la clé. Or cela implique que ces deux occurrences sont encodées de la même manière ; ainsi, on peut s'attendre à ce qu'on ait une surreprésentation de motifs séparés d'un multiple de la longueur de la clé. Ce ne sont bien sûr pas la seule source de répétition de motifs dans le texte chiffré, mais on peut s'attendre à ce que ça engendre quelque chose de détectable d'un point de vue fréquentiel.
Cela suggère l'approche suivante : regarder toutes les paires de motifs de trois lettres qui se répètent, puis noter l'écart entre leurs positions. Ensuite, compter pour chaque longueur de clé raisonnable combien de ces écarts elle divise, puis utiliser cette information pour isoler des bons candidats, qu'on pourra utiliser avec la fonction de la partie précédente.

Question 7. *Écrire une fonction* `patterns_rep`*, qui prend en argument un message à décoder, et un entier, telle que* `patterns_rep(cypher, l)` *renvoie une liste de longueur* `l-1`*, dont le* `i`*-ème élément est le nombre de paires d'occurences d'un motif de longueur* `i+2`*.*
S'en servir pour déterminer à la main la longueur de la clé, en sachant qu'elle est de longueur au plus 20*.*