Cryptanalyse à l'aide d'un dictionnaire

Sujet proposé par François Morain

morain@lix.polytechnique.fr

URL de suivi : `http://www.enseignement.polytechnique.fr/profs/informatique/Francois.Morain/INF431/projetX05.html`

1 Préambule

Le but de ce projet est de décrypter *automatiquement* des textes chiffrés à l'aide d'une substitution simple. L'algorithme proposé repose sur l'emploi judicieux d'un dictionnaire compilé à l'avance.

2 Détail du sujet

On considère un texte clair $\mathcal{C}$ écrit en français, sans accents, ni signes de ponctuation et en lettres minuscules choisies dans $\mathcal{A} = \{a, b, c, \ldots, z\} = \{\alpha_1, \alpha_2, \ldots, \alpha_{26}\}$. Le texte est formé de mots séparés par des espaces : $\mathcal{C} = C_1\ C_2\ \ldots\ C_n$.

Chiffrer $\mathcal{C}$ à l'aide d'une substitution simple revient à construire une permutation Π de $\mathcal{A}$ dans lui-même et à remplacer chaque mot $C_i = \beta_1\beta_2\ldots\beta_{n(i)}$ de $\mathcal{C}$ par $M_i = \Pi(\beta_1)\Pi(\beta_2)\ldots\Pi(\beta_{n(i)})$. Par exemple, on peut chiffrer le texte

```
les sanglots longs des violons de l automne
```

en

```
mft tbohmput mpoht eft wjpmpot ef m bvupnof
```

par application de la permutation circulaire $\Pi : a \mapsto b \mapsto c \mapsto \cdots \mapsto z \mapsto a$.

Partant d'un message chiffré (dans lequel on a conservé les espaces d'origine), $\mathcal{M} = \Pi(\mathcal{C})$, le but du projet est de retrouver $\mathcal{C}$ sans connaître Π à l'avance.

Dans une première phase, on constitue une base de données de mots de la façon suivante. À partir d'un dictionnaire de référence, on construit la matrice D dans laquelle l'élément $D(i, j, k)$ contient la liste des mots du dictionnaire qui ont i lettres et qui contiennent la lettre α_j à la position k (cette liste peut éventuellement être vide). Une fois cette base constituée, on trouve les mots correspondants à des critères complexes par intersection des éléments de D. Par exemple, si l'on veut connaître tous les mots de 8 lettres ayant un a en deuxième position et un l en quatrième, on calculera $D(8, 1, 2) \cap D(8, 12, 4)$.

Une fois cette base construite, on inspecte le message chiffré $\mathcal{M} = M_1\ M_2\ \cdots\ M_n$ et on construit pour chaque mot M_i la liste des mots $L(M_i)$ du dictionnaire qui peuvent en être le correspondant clair. On sélectionne le premier mot de $L(M_1)$, soit $E_1 = \beta_1\beta_2\ldots\beta_{n(1)}$ et on en déduit un début de déchiffrement : si $M_1 = \gamma_1\gamma_2\ldots\gamma_{n(1)}$, on suppose donc que $\Pi(\beta_1) = \gamma_1$, $\Pi(\beta_2) = \gamma_2$, …, $\Pi(\beta_{n(1)}) = \gamma_{n(1)}$. On regarde alors si ce début de décryptage est compatible avec les déchiffrements possibles des autres mots. On continue l'exploration de l'arbre des possibilités en testant à chaque fois si toute nouvelle lettre déchiffrée est compatible avec celles déjà déchiffrées.

Pour rendre efficace ce parcours, il convient de trouver des critères de sélection des mots à chaque étape et de pouvoir reconnaître les incohérences du décryptage très tôt (attention : il peut arriver que des mots ne

se trouvent pas dans le dictionnaire, par exemple les noms propres). Certaines optimisations sont suggérées dans l'article donné en référence et pourront être testées.

3 Travail demandé

3.1 Prérequis

Dans un premier temps, il faut récupérer un dictionnaire et le traiter comme indiqué ci-dessus. Cela fait, il faut pouvoir prendre en entrée un fichier texte contenant un message chiffré en respectant les indications ci-dessus (mots non accentués en minuscules séparés par des blancs ou des `<retour-chariot>`).

Dans un second temps, il faut programmer l'exploration de l'arbre des possibilités et afficher le message décrypté.

On prendra soin d'utiliser des structures de données adapatées au problème. En particulier, le choix de la représentation de la matrice D sera capital.

3.2 Données fournies

Je me charge de fournir des fichiers tests dans le format décrit ci-dessus, ainsi qu'un dictionnaire. Le jour de l'oral, des fichiers répondant aux mêmes critères seront fournis.

3.3 Extensions possibles

On pourra essayer plusieurs langues pour le décryptement, avec des dictionnaires adéquates à chaque fois. On peut aussi laisser le programme déterminer la langue tout seul.

Références

[1] Michael Lucks. A constraint satisfaction algorithm for the automated decryption of simple substitution ciphers. In S. Goldwasser, editor, *Advances in Cryptology, Crypto '88*, volume 403 of *Lect. Notes in Computer Science*, pages 132–144, 1990. Proceedings of Crypto '88, University of California, Santa Barbara, August 21–25, 1988.