Probabilité et modèle de langues
Estimation et lissage

Guillaume Wisniewski
Université Paris Sud & LIMSI
guillaume.wisniewski@limsi.fr

14 janvier 2016

1 Chiffrement par substitution

Le chiffrement est un procédé de cryptographie qui permet de rendre la compréhension d'un document impossible à toute personne qui n'a pas accès à une information additionnelle : la clé de chiffrement. La méthode de chiffrement la plus simple est le chiffrement *par substitution* qui consiste à remplacer chaque lettre par une autre lettre de l'alphabet. Par exemple, si l'alphabet ne comporte 3 lettres (α, β, γ) on peut choisir de remplacer tous les α par des γ, tous les β par des γ et tous les γ par des α.

Un chiffrement par substitution peut être implémenter de manière triviale en python à l'aide des fonctions `maketrans` et `translate` :

```
import string

def encode(msg, key):
    """
    Encode a message with a substitution cipher.

    Parameters
    ----------

    - msg, a string
           the message to encode
    - key, a string of 27 characters
           the i-the character of the alphabet is replaced by the i-th
           character of the key
    """
    key = key + key.upper()
    return msg.translate(msg.maketrans(string.ascii_letters, key))
```

```
# simple shift cipher
text = "I'm sorry Dave, I'm afraid I can't do that"
key = string.ascii_lowercase[13:] + string.ascii_lowercase[:13]
print(encode(text, key))
```

Dans cet exemple, on utilise un chiffrement par substitution particulier : le chiffrement par décalage ou chiffre de César dans lequel la clé est obtenue par une permutation circulaire de l'alphabet de départ.

2 Déchiffrement

L'objectif de ce TP est déterminer automatiquement la clé utilisée pour chiffrer un message. La méthode est très simple :

1. on génère toutes les clés possibles ;
2. pour chaque clé, on décode le message et on calcule sa probabilité selon le modèle de langue ;
3. on détermine la clé qui permet de générer le texte le plus probable (selon le modèle de langue).

Pour des raisons évidentes de complexité, on s'intéressera, dans un premier temps, uniquement aux chiffrements par décalage avant de considérer l'ensemble des chiffrements par substitution. On supposera également que les espaces sont conservés.

Le modèle de langue sera estimé à partir du corpus de 1 trillions de mots [1]. Ce corpus est diffusé sous une forme « résumée » qui contient pour chaque séquence de 1, 2, 3, 4 et 5 mots le nombre d'occurrences de la séquence de mots [2]. Le corpus original a été pré-traité pour supprimer tous les mots contenant des lettres hors de l'alphabet ASCII, mettre en minuscule tous les mots et ajouter, au début de toutes les phrases, le symbole `<S>`. Une version réduite (qui ne contient que 1/3 des données) est téléchargeable sur le site du cours.

1. Écrivez une fonction capable d'estimer un modèle de langue [3] 2-gram et un modèle de langue 1-gram à partir d'un fichier passer en paramètre suivant la syntaxe décrite dans le paragraphe précédent.
2. Comment s'écrit la probabilité qu'une phrase $s = w_1, ..., w_n$ ait été générée par $\mathcal{M}$ un modèle de langue 2-gram ? par un modèle de langue 1-gram ?
3. Écrire une fonction qui prend en entrée une phrase chiffrée et qui calcule la probabilité d'observer cette phrase. Testez là pour les trois phrases suivantes :

1. https://catalog.ldc.upenn.edu/LDC2006T13
2. Pour limiter la taille du corpus (qui fait quand même 24Go dans sa forme compressée) et l'impact des mots mal orthographiés les mots apparaissant moins de 200 fois sont remplacés par le symbole `UNK` et les n-gram apparaissant moins de 40 fois sont supprimés.
3. Pour mémoire : un modèle de langue d'ordre n, donne la probabilité d'un mot connaissant les $n-1$ mots précédents.

— `I love probabilities very much` ;
— `text = "I love probabilities " + "very " * 100 + "much"` (la même phrase que précédément mais `very` est répété 100 fois) ;
— `I 'm sorry Dave, I 'm afraid I can't do that.`
Qu'en concluez-vous ?

Le mise en œuvre du principe décrit dans le paragraphe précédent se heurte à deux problèmes pratiques :
— La probabilité de générer une phrase décroît avec la longueur de la phrase et l'on peut (très) vite atteindre les limites de représentation ;
— La phrase à déchiffrer comporte des mots qui n'ont pas été vus dans le corpus sur lequel a été estimé le modèle de langue (on parle de *mot hors-vocabulaire*).

Le premier problème se résout très facilement en ne considérant non plus la probabilité qu'une phrase s ait été générée par un modèle de langue $\mathcal{M}$, mais le logarithme de cette probabilité. Concrètement, les différents déchiffrements possibles de la phrase sont ordonnés selon le critère :

$$\log_{10} p(s|\mathcal{M}) \tag{1}$$

4. Pourquoi considérer une log-probabilité est absolument équivalent à considérer une probabilité ?

Le second problème nécessite de recourir à des techniques de *lissage* des probabilités estimées.

5. Que se passe-t-il lorsque l'on calcule la probabilité d'une phrase qui contient un mot hors-vocabulaire ? Dans quelle(s) autre(s) situation(s) observe-t-on un problème similaire ?

Pour éviter ce problème, il est nécessaire d'attribuer une partie de la masse de probabilité aux événements qui n'ont pas été vus. La manière la plus simple de le faire consiste à fixer la probabilité d'un n-gram inconnu à une constante. Par exemple, pour un modèle 1-gram, on définit généralement la probabilité d'un mot comme :

$$P(w) = \begin{cases} \frac{\#w}{\#\text{mots}} & \text{si } \#w \neq 0 \\ \frac{1}{N} & \text{sinon} \end{cases} \tag{2}$$

où $\#w$ est le nombre de fois où le mot w est observé, #mots le nombre de mots dans le corpus et N est la taille du vocabulaire (c.-à-d. le nombre de mots différents du corpus).

6. Avec ce lissage, la « quantité » $p(w)$ n'est plus une probabilité. Pourquoi ? Est-ce gênant ?
7. Comment généraliser ce lissage à un modèle 2-gram ?
8. Implémentez cette stratégie. Commentez.
9. Écrire une fonction qui prend, en entrée, une phrase chiffrée et affiche l'ensemble des phrases déchiffrées possibles classées par ordre de probabilité croissante.