2 - Traitement de textes bruts

1. Systèmes de codage des caractères
2. Expressions régulières
3. Segmentation d'un texte

Références : *livre en ligne sur NLTK* *http://nltk.org/book/ch03.html*

Jurafsky & Martin - Speech and Language Processing - chapitre 2

2.1 - Systèmes de codage des caractères

- Un texte est représenté comme une suite de caractères : des caractères alphanumériques, des signes de ponctuation, des caractères d'espacement (blanc, tabulation, passage à la ligne).

- Chaque caractère est traduit par un nombre selon un système de codage.

- Historiquement, un des premiers systèmes de codage a été ASCII. Puis se sont développés des systèmes par famille de langues (ISO 8859-1 ou latin-1 pour l'alphabet latin).

- Unicode, et une de ses variantes UTF-8, est un système universel de codage valable pour toutes les langues.

2.1 - Systèmes de codage des caractères

- Dans le système ISO 8859-1, chaque caractère est codé sur un octet (8 bits). Cela permet de coder 256 caractères : caractères de contrôle, caractères de l'alphabet latin, diacritiques, ligatures, chiffres, signes de ponctuation et autres symboles.

- Dans UTF-8, chaque caractère est représenté par un entier (son point de code) codé sur un nombre variable d'octets allant de 1 à 4. Les premiers bits de chaque octet nous renseigne sur le nombre d'octets utilisés pour un caractère donné.

Caractère	point de code décimal	point de code binaire	code UTF-8
A	65	0011111	00011111
é	233	00011 101001	11000011 10101001
€	8364	0010 000010 101100	11100010 10000010 10101100

2.1 - Systèmes de codage des caractères

- Python distingue deux types de chaînes de caractères selon le système de codage : ***Str*** pour le système ISO 8859-1, et ***Unicode*** pour UTF-8.

- Par défaut, une chaîne est de type Str. Pour indiquer qu'elle est de type unicode, c'est codée en UTF-8, il faut placer la lettre *u* devant la chaîne. Par exemple : *u"tête"*.

- Python dispose d'un opérateur + de **concaténation** de chaînes de caractères et d'un opérateur de **tranchage** : si *s* est une chaîne de caractères et si *n* et *m* sont deux entiers, *s[n:m]* a comme valeur la sous-chaîne de *s* qui commence à la position *n* et qui termine à la position *m-1*.

2.1 - Systèmes de codage des caractères

1. En UTF-8, le début d'un octet nous renseigne soit sur le nombre d'octets codant un caractères, soit sur le rôle de l'octet dans le code du caractère :

   - Si l'octet commence par 0, cela signifie que le caractère est codé sur un octet, les 7 bits restant donnant le point de code du caractère.
   - Si l'octet commence par 110, le caractère est codé sur deux octets et les 5 bits restant représentent le début du point de code du caractère.
   - Si l'octet commence par 1110, le caractère est codé sur trois octets et les 4 bits restant représentent le début du point de code du caractère.
   - Si l'octet commence par 11110, le caractère est codé sur quatre octets et les 3 bits restant représentent le début du point de code du caractère.
   - Si l'octet commence par 10, le caractère est codé sur plus d'un octet et les 6 bits qui suivent représentent la suite du point de code du caractère.

   Déterminez le nombre de caractères contenus dans la chaîne codée de la façon suivante en UTF-8 et déterminez le point de code décimal de chaque caractère :
   11101101 10111100 10000110 01100001 11000010 10000101

2.1 - Systèmes de codage des caractères

2. Soit en Python la chaîne de caractères *s* de type *Str* qui a comme valeur *"je vis en France"*. Donnez la valeur retournée par chacune des expressions Python ci-dessous.

   a) *s[3:7]*

   b) *s[0:1] + s[0:4]*

   c) *s[-8 : 10]*

   d) *s[4:4]*

   e) *2 ⋆ s[7:9]*

2.2 - Expressions régulières : motivation

- Dans le traitement de textes écrits, il est souvent nécessaire de rechercher dans ces textes des segments ayant une caractéristique particulière : les phrases, les mots, les entités nommées, un mot particulier, les formes fléchies d'un lemme particulier …

- Un moyen d'effectuer une telle tâche est d'utiliser des **expressions régulières**. Une expression régulière est une formule permettant de caractériser un ensemble de chaînes de caractères.

2.2 - Expressions régulières : définition

- Formellement, une expression régulière (ou rationnelle) sur un alphabet de symboles Σ est une expression construite à partir de symboles de Σ et du mot vide ε, à l'aide de 3 opérations : la **concaténation** (les expressions concaténées sont simplement juxtaposées), la **disjonction** « | » et la **clôture de Kleene** « * ».

- Une expression régulière définit un ensemble de chaînes de symboles de Σ qui forment un **langage régulier**.

- A partir des trois opérations primitives, on définit habituellement 2 opérations complexes qu'on exprime à l'aide des opérateurs ? et + :

Si E est une expression régulière, $E? = E \mid \varepsilon$ *et* $E^+ = E\,E^*$

2.2 - Expressions régulières : définition

- Dans certains langages formels comme Python, on utilise d'autres opérations complexes :

.	Remplace n'importe quel symbole (sauf éventuellement le caractère spécial \n de passage à la ligne)
[ ]	Remplace l'un quelconque des symboles placés entre les crochets
[^]	Remplace l'un quelconque des symboles qui ne sont pas entre les crochets
{m,n}	m et n sont des entiers qui indiquent respectivement le nombre minimum et maximum de fois que la sous-expression située juste avant est répétée
\w	Remplace n'importe quel caractère alphanumérique (lettre ou chiffre) plus le caractère _
\W	Remplace n'importe quel symbole qui n'est ni un caractère alphanumérique, ni le caractère _
\d	Remplace n'importe quel chiffre
\D	Remplace n'importe quel symbole qui n'est pas un chiffre
\s	Remplace l'un quelconque des caractères d'espacement ' ', \t, \n, \f, \r \v
\S	Remplace n'importe quel symbole qui n'est pas un caractère d'espacement

2.2 - Expressions régulières : définition

- Dans des expressions régulières complexes, les opérations unaires qui sont toutes postfixées, sont effectuées en premier. Ensuite, ce sont les concaténations et enfin les disjonctions. Les parenthèses permettent de forcer les priorités.

2.2 - Expressions régulières : utilisation

- L'application la plus répandue des expressions régulières en TAL consiste à rechercher dans un texte des chaînes de caractères ayant une caractéristique donnée.

- La caractéristique donnée est exprimée par une expression régulière.

- La plupart des algorithmes utilisés parcourt le texte du début à la fin en essayant de trouver la plus longue chaîne de caractères correspondant à l'expression régulière. Dès qu'une chaîne est trouvée, le processus est réitéré à partir du caractère suivant la fin de la chaîne trouvée.

2.2 – Expressions régulières : utilisation

- **Algorithme de recherche d'expressions régulières dans une chaîne**

```
Fonction RECHERCHE_EXPR (expr, chaine)
index ← 0
resultat ← [ ]
tantque index < longueur(chaine) faire
    si  chaine[index] est le début d'un mot du langage L(expr) représenté par  expr  alors
            fin_pref = index
            fin_mot = -1
            tantque chaine[index : fin_pref + 1] est le début d'un mot du langage L(expr)
              et  fin_pref < longueur(chaine) faire
                    si  chaine[index : fin_pref + 1] ∈ L(expr)  alors
                            fin_mot = fin_pref
                    fin_pref  = fin_pref + 1
            si fin_mot > -1 alors
                    resultat ← resultat + [ (index, fin_mot)]
                    index ← fin_mot + 1
            sinon
                    index ← index + 1
    sinon
            index ← index + 1
retourner resultat
```

2.2 - Expressions régulières : utilisation

- Pour exprimer certaines contraintes sur la position des chaînes de caractères recherchées, la syntaxe des expressions régulières est étendue par des constantes de contrôle ou **ancres**. Les ancres ne représentant aucune chaîne de caractères (le mot vide) mais elles indiquent des positions particulières dans le texte.

^	Début d'une ligne
$	Fin d'une ligne
\b	Début ou fin d'un mot (un mot est une suite continue de caractères alphanumériques ou de caractères _)
\B	Position qui n'est ni un début, ni une fin de mot
\A	Début du texte
\Z	Fin du texte

2.2 - Expressions régulières : exercices

1. On considère l'alphabet usuel des caractères utilisés pour écrire un texte en français. Pour chacune des expressions régulières ci-dessous, décrivez le langage qu'elle représente, c'est-à-dire l'ensemble des chaînes de caractères qui correspondent à l'expression.

a) ab^+| b?a

b) a\d*\s$^+$

c) \s |[ea] \s

d) $[0\text{-}7]^+$(,[0-7]*[1-7])?

2. On considère l'alphabet usuel des caractères utilisés pour écrire un texte en français. Trouver une expression régulière caractérisant chacun des langages suivants :

a) Les sommes d'argent en euros avec au maximum deux chiffres après la virgule,

b) Les formes fléchies du verbe *finir* à l'indicatif présent,

c) Les chaînes de caractères comportant au moins deux fois le mot *cher* (mais ni en début, ni en fin de chaîne).

d) Les chaînes de caractères ne comportant pas la sous-chaîne *du*.

2.2 - Expressions régulières : exercices

3. On considère l'alphabet usuel des caractères utilisés pour écrire un texte en français. Trouver une expression régulière permettant de rechercher dans un texte en français les chaînes de caractères ayant chacune des caractéristiques suivantes :

a) Le mot « cheval » ou « chevaux »,

b) Les lignes se terminant par un point d'interrogation (le point d'interrogation est représenté par « \? » dans une expression régulière),

c) Les phrases interrogatives (on suppose qu'une phrase est caractérisée comme se terminant par un point, un point-virgule, un point d'exclamation ou un point d'interrogation)

4. On applique l'algorithme de recherche des chaînes de caractères dans un texte correspondant à une expression régulière donnée. Dans chacun des cas, donnez le résultat de la recherche

	Expression régulière	texte
a)	'a\w*a'	'acddcceeacvvv bbaa'
b)	'[^a-z]+ba'	'Aaa Abbca bac'
c)	'^ .\b'	'Je viens.\n Et toi?'
d)	'\b[A-Z].*'	'Tu connais Paul et Marie ?'
e)	'[\s .] [A-Z]'	'Tu connais l'O.N.U. C'est difficile.'

2.3 - Segmentation d'un texte

- La segmentation d'un texte consiste à le découper en unités plus petites : phrases, mots, tokens...

- La tokenisation consiste à découper un texte en unités élémentaires ou tokens. La définition d'un token est variable et dépend du traitement que l'on veut ensuite effectuer sur le texte. Un token peut être un mot, un signe de ponctuation, une entité nommée...

- Pour tokeniser un texte, on peut utiliser une expression régulière représentant soit la forme générale des tokens, soit celle des séparateurs de tokens.

- La segmentation d'un texte en phrases est un préalable nécessaire à l'analyse syntaxique. Pour cela, on peut utliser des expressions régulières représentant la forme générale des séparateurs de phrases.