Manipulation de mots pour analyser des corpus

Cours d'initiation au TAL - M1 SCA & SDL

28 septembre 2013

A l'aide d'un navigateur, affichez le tutoriel NLTK en allant à l'URL : *nltk.org/book*

Ouvrez le tutoriel au chapitre 3 *Processing Raw Text*. Lisez attentivement les passages qui vous seront recommandés en exécutant les exemples dans une fenêtre *Idle* ouverte en parallèle.

1 Utilisation de fichiers pour stocker des textes

Allez directement à la sous-section *Reading Local Files* de la section 3.1 qui aborde le problème de la lecture de fichiers sous Python.

Ouvrez une nouvelle fenêtre d'édition sous Idle en allant dans le menu *File* à l'aide de la commande *New Window*. Tapez un texte de votre choix sur plusieurs lignes et sauvegardez le fichier dans votre répertoire personnel U sous le nom *document.txt*. Vous pourrez alors exécuter la première instruction donnée en exemple mais en mettant le nom du chemin qui permet d'accéder au fichier. il vous faudra donc taper *f=open('U :\\document.txt')*. Continuez ensuite à lire la sous-section en exécutant les exemples.

Ensuite, vous compléterez l'expérimentation par la tâche d'écriture dans un fichier. Après avoir lu le fichier *document.txt* identifié à l'aide de la variable *f*, vous devrez le fermer à l'aide de l'instruction *f.close()*. Vous allez l'ouvrir de nouveau avec l'instruction *f= open('U :\\document.txt', 'a')*. Le fichier est ouvert en mode *append*. Vous allez pouvoir écrire à la fin de celui-ci. Par exemple, écrivez à la fin le texte *'C'est la fin'* à l'aide de l'instruction *f.write('Voici la fin')*. Fermez le fichier, puis ouvrez-le en mode lecture et vérifiez que le texte a bien été ajouté à la fin.

Recommencez l'opération en ouvrant cette fois le fichier en mode *write*. Pour cela, utilisez l'instruction *f= open('U :\\document.txt', 'w')*. Poursuivez ensuite de la même façon que pour le mode *append*. Que constatez-vous quand à l'effet du mode *write* ?

2 Utilisation d'expressions régulières pour segmenter un texte

Allez à la section 3.4 *Regular Expressions for Detecting Word Patterns*. La première sous-section *Using Basic Meta-Characters* présente certains méta-caractères utilisés dans

les expressions régulières.

Dans la sous-section suivante *Ranges and Closures*, vous trouvez quelques applications des expressions régulières. La première est le système T9 d'entrée d'un texte sur un téléphone portable. Les autres exemples qui suivent sont là uniquement pour mieux comprendre la syntaxe des expressions régulières.

Passez ensuite directement à la section 3.7 *Regular Expressions for Tokenizing Text*, qui montre comment "tokeniser" un texte, c'est-à-dire le découper en unités élémentaires, à l'aide d'expressions régulières. La première sous-section *Simple Approaches to Tokenization* explique plusieurs façons de "tokeniser" un texte en utilisant une expression élémentaire permettrant de reconnaître les séparateurs de "tokens".

La sous-section *NLTK's Regular Expression Tokenizer* présente ensuite un "tokeniseur" fourni par NLTK et enfin la dernière sous-section *Further Issues with Tokenization* évoque certaines difficultés de la tâche.

La section 3.8 *Segmentation* montre que la "tokenisation" est une instance de la tâche plus générale de segmentation. Lisez cette section sans nécessairement exécuter les exemples.

La tâche évoquée peut être de la segmentation en phrases, ce que décrit la sous-section *Sentence Segmentation*. Cette tâche peut aussi être de la segmentation en mots, décrite dans la sous-section *Word Segmentation*. Cette tâche est difficile pour des corpus où cette segmentation n'est pas déjà amorcée par l'utilisation d'espaces (corpus oraux ou langues particulières). Vous pouvez passer cette sous-section.

La section 3.9 *Formatting : From Lists to Strings* explique comment formater la sortie d'un programme qui est sous forme d'une liste pour la mettre sous forme d'une chaîne de caractères. C'est l'objet de la sous-section *From Lists to Strings*. La sous-section suivante *Strings and Formats* montre comment associer un format à une chaîne de caractères. Il n'est pas nécessaire d'étudier les sous-sections suivantes.

3 Exercices

Mettre les réponses à chacun des exercices ci-dessous dans un fichier désigné ainsi : *TP2<numéro de l'exercice>.py*. Les réponses qui sont des explications et non des programmes Python doivent être mises sous forme de commentaires. En Python un commentaire est une ligne précédée par le caractère #.

Exercice 3.1 *Dans le dossier* P :\UFRMI\Perrier\InitiationTAL*, vous trouverez deux fichiers que contiennent le livre de Jules Verne* le tour_du_monde en 80 jours*. Dans l'un, le texte est codé en utf8 et dans l'autre, en latin1. Choisissez-en un et copiez le dans votre répertoire personnel U avec le nom* tour du monde.txt.

Ecrivez un programme Python qui effectue les tâches suivantes :

1. *Ouvrir en lecture le fichier* tour du monde.txt *à l'aide de la fonction* open.
2. *Extraire le contenu du fichier pour le mettre dans une variable* texte *à l'aide de la méthode* read.

3. *Découper le texte en une liste de phrases stockée dans la variable* liste_phrases *à l'aide d'une expression régulière représentant les séparateurs de phrases (pour que la ponctuation soit conservée dans les phrases découpées, on peut mettre des parenthèses autour de la sous-expression représentant la ponctuation, mais leur interprétation dépend de la plateforme utilisée).*
4. *Ouvrir en écriture un nouveau fichier* phrases.txt *à l'aide de la fonction* open *avec deux paramètres : le chemin d'accès au fichier* phrases.txt *et* 'w' *(pour indiquer que le fichier est ouvert en écriture).*
5. *Ecrire la liste des phrases* liste_phrases *dans le fichier* phrases.txt *et fermer ensuite le fichier.*

Exercice 3.2 *Ecrivez ensuite un programme Python qui effectue les tâches suivantes :*

1. *Ouvrir en lecture le fichier* tour_du_monde.txt *à l'aide de la fonction* open.
2. *Extraire le contenu du fichier pour le mettre dans une variable* texte *à l'aide de la méthode* read.
3. *Découper le texte en une liste de mots stockée dans la variable* liste_mots *à l'aide d'une expression régulière représentant les séparateurs de mots. Dans la mesure du possible, on considérera qu'un signe de ponctuation est un mot.*
4. *Ouvrir en écriture un nouveau fichier* mots.txt *à l'aide de la fonction* open *avec deux paramètres : le chemin d'accès au fichier* mots.txt *et* 'w' *(pour indiquer que le fichier est ouvert en écriture).*
5. *Ecrire la liste des mots* liste_mots *dans le fichier* mots.txt *et fermer ensuite le fichier.*

Indiquez ensuite dans un commentaire les principales erreurs que vous constatez en utlisant cette méthode. Si vous avez une solution pour les corriger, indiquez-le aussi.

Exercice 3.3 *Ecrivez un programme Python qui effectue les tâches suivantes :*

1. *Ouvrir en lecture le fichier* mots.txt *de l'exercice précédent.*
2. *Extraire les mots du fichier et les mettre dans une variable* liste_mots *de type liste. Fermer le fichier.*
3. *Pour chacun des mots de la liste, calculer sa fréquence d'apparition dans* liste_mots *à l'aide de la fonction* nltk.FreqDist*. Stocker le résultat dans une variable* liste_frequences
4. *Afficher les mots dont la fréquence est au moins égale à 50 en évitant si possible les mots grammaticaux (articles, pronoms, conjonctions, ...).*

Indiquez dans un commentaire les informations que peuvent apporter ces mots sur le contenu du texte.

Réunir les différents fichiers .py et .txt contenant les réponses aux exercices dans une archive nommée *<nom-étudiant>.<prénom-étudiant>.TP2.zip* et envoyez-le par mail en pièce jointe à l'adresse *perrier@loria.fr* en mettant dans l'objet du message *initiation au TAL : TP2.*