TD1 : analyse morpho-syntaxique

LO17

Fatma Chamekh

2021-2022

Objectifs :

L'objectif de ce TD est d'explorer les principales fonctions de la librairie NLTK pour réaliser une analyse morpho-syntaxique. Il comporte deux parties :

- Partie 1 : tester/expliquer des notebooks afin de se familiariser avec les fonctions usuelles qui permettent d'effectuer les tâches d'une analyse morphosyntaxique et l'usage des corpus.
- Partie 2 : Analyser un corpus.

Environnement de travail :

Qu'est-ce que c'est NLTK ?

Natural Language Toolkit (NLTK)[1] est une boîte-à-outil permettant la création de programmes pour l'analyse de texte. Cet ensemble a été créé à l'origine par Steven Bird et Edward Loper, en relation avec des cours de linguistique informatique à l'Université de Pennsylvanie en 2001. Il existe un manuel d'apprentissage pour cet ensemble titré Natural Language Processing with Python [2](en anglais).

Installation

Linux	Windows

`sudo pip install nltk`
 	
ou bien pour le python3
 	
`sudo pip3  install nltk`
	C:\python2.7\Scripts\pip.exe install nltk
	ou bien installer le fichier attaché

Travailler avec NLTK

La première chose à faire pour utiliser NLTK est de télécharger ce qui se nomme le *NLTK corpora*. On va télécharger quelques corpus.

Dans votre éditeur Python IDLE, écrivez ceci :

```
import nltk
nltk.download()
```

Dans ce cas précis, une interface graphique s'affiche, vous permettant de définir la destination des fichiers et de sélectionner ce dont vous avez besoin:

Sélectionner d'installer les fichiers suivants :

- models : punkt, averaged_perceptron_tagger
- corpora:stopwords

Lister les mots vides :

```
from nltk.corpus import stopwords
print('mots vides en  anglais')
print(stopwords.words("english"))
print('mots vides en  français')
print(stopwords.words("french"))
```

Eliminer les mots vides

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
text = 'In this tutorial, I\'m learning NLTK. It is an interesting platform.'
stop_words = set(stopwords.words('english'))
words = word_tokenize(text)
new_sentence = []
for word in words:
    if word not in stop_words:
    new_sentence.append(word)
print(new_sentence)
```

Disons que dans le fichier texte suivant (fichier attaché). Nous désirerions rechercher (fouiner) le mot ‘language’. Nous pourrions utiliser la librairie NLTK comme suit :

```
file2 = codecs.open('NLTK.txt', 'r', encoding='utf-8')
read_file = file2.read()
text = nltk.Text(nltk.word_tokenize(read_file))
```

Stemming

```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
words = ["game","gaming","gamed","games"]
ps = PorterStemmer()
for word in words:
    print(ps.stem(word))
```

PS : le fichier est nommé NLTK.txt

Partie 1 : exploration de NLTK

Exercice 1 :

Exécutez les jupyter notebooks : méthode NLP et analyse Corpus

Exercice 2 :

1- Utilisez le fichier NLTK.txt pour exécuter le code suivant et :

```
def treat_line(line):
    """
    traiter une ligne de texte.
    """
    return line

def read_file(filename):
    try:
        myfile = open(filename)
    except:
        print "Can't open file ", filename
        sys.exit()
    # lecture des lignes de texte
    return myfile.readlines();

def treat_text(lines):
    for line in lines:
        print line


# La fonction main
if __name__ == '__main__':
    DATA_FILE = "text.txt"
    lines = read_file(DATA_FILE)
    treat_text(lines);
```

2- Testez la fonction tokenize() :

```
# regular expression
import re
def tokenize(text):
    """ extraire les mots """
    # diviser la lignes par les espaces
    list_word =  text.split(" ")
    return list_word

# La fonction main
if __name__ == '__main__':
    text = "I'm Very Hungry, I want to eat something."
    tokens = tokenize(text);
    print tokens
```

3- On veut proposer plusieurs implémentations de la fonction tokenize(), tester les différentes fonctions :

```
# tokenize par des expression régulière simple
def tokenize_regex_punct(text):
    """ extraire les mots"""
    tokens = re.split("[.,:; ]+", text)
    return tokens


# tokenize par des expression régulière simple, en gardant la ponctuation
def tokenize_regex_punct_keep(text):
    """ extraire les mots"""
    tokens = re.split("([.,:; ]+)", text)
    return tokens

# tokenize par des expression régulière
def tokenize_regex(text):
    """ extraire les mots"""
    tokens = re.split("\W+", text)
    return tokens

# tokenize par des expression régulière, en gardant la ponctuation
def tokenize_regex_keep_punct(text):
    """ extraire les mots"""
    tokens = re.split("(\W+)", text)
    return tokens
```

Noter les différences entre les différentes implémentations de la fonction tokenize.

4- Testez le code de stemming suivant :

```
import re
# la liste des suffixes
SUFFIX_LIST=['ique', 'ation', 'tion', 'é', 'er','eur', 'ien']

def stemming(word, suffix_list):
    """ stem a word"""
    stem_list =[]
    for suffix in suffix_list:
        if word.endswith(suffix):
            stem = re.sub(suffix+'$','', word)
            stem_list.append((word, stem, suffix))
    return stem_list;


if __name__ == "__main__":
    text = """
    Soundex est un algorithme phonétique d'indexation de noms par
leur prononciation en anglais britannique"""
    list_word = text.split(' ')
    for word in  list_word:
        print stemming(word, SUFFIX_LIST)
```

On a un dictionnaire de la forme

```
lemme;suffix1;suffix2,suffix3;suffix3 ;*
```

Exemple

```
donner;é;ée;eur;euse;ant
```

- On peut générer les mots donné, donnée, donneur, donneuse, donnant.
- Ecrire le programme qui permet de générer les différentes formes d'un mots à partir d'un dictionnaire.

Partie 2 : analyse d'un corpus de sous-titres de séries

Vous disposez d'un corpus de sous-titres de séries en anglais, aux formats :

- srt: sous-titres avec indications temporelles.
- xml: fichier XML regroupant les phrases, tokenisé.
- txt: fichier texte (une phrase par ligne).

I faut privilégier le format txt.

À partir du corpus proposé, vous devrez définir un objectif d'analyse de ce corpus, et déterminer quelles sont les analyses TAL pertinentes pour le réaliser. L'objectif doit bien évidemment pouvoir s'appuyer sur l'analyse du corpus.

Voici quelques exemples d'objectifs :

- Déterminer quels sont les qualificatifs principaux attribués au personnage de Sheldon Cooper ;
- Déterminer quelle est la couleur de vêtements la plus souvent citée ;
- Établir les relations sociales ou familiales (ami, soeur…) entre les personnages évoqués…

Il n'est pas (nécessairement) demandé d'implémenter une application fonctionnelle, mais seulement de déterminer quelles sont les informations utiles pour l'objectif choisi en utilisant les fonctions de la librairie NLTK (optionnellement l'interface Core NLP https://corenlp.run/).

Vous testerez donc les analyses suivantes :

- Segmentation en mots (tokenisation)
- Segmentation en phrases
- Annotation en parties du discours ou étiquetage morpho-syntaxique (PoS tagging)
- Lemmatisation
- Reconnaissance d'entités nommées
- Analyse syntaxique en constituants (constituency parsing)
- Analyse syntaxique en dépendances (dependency parsing)
- Résolution de coréférence (avec CoreNLP uniquement)

Cet exercice fera l'objet d'un mini-compte rendu sous forme d'un jupyter notebook. Il faut prévoir une partie texte dans laquelle vous expliquez vos objectifs. Il faut détailler les différentes étapes de l'analyse morpho syntaxique adoptées avec des éventuelles interprétations.