TD5._sujet

February 9, 2023

1 TD5 Stemming, lemmatizing, tokenization

1.1 Partie 1 Stemming et lemmatizing

On souhaite comparer des segmenteurs. Il en existe plus pour l'anglais que pour le français, nous allons commencer avec un string en anglais.

On va prendre le readme dans le corpus reuters. Pour importer reuters, il faut importer nltk et download ce corpus :

```
>>> import nltk
>>> nltk.download('reuters')
```

[ ]:
```
import nltk
from nltk.text import Text
```

[ ]:
```
from nltk.corpus import reuters
readme = reuters.readme().replace('\n', ' ').strip()

print(readme)
```

1.1.1 Exo1 Comparer 3 stemmers existants dans NLTK

- PorterStemmer
- LancasteStemmer
- SnowballStemmer

Importer ces 3 stemmers, et puis créer pour chacun une instance (`porter`, `lancaster`, `snowball`) Pour SnowballStemmer, préciser la langue "english".

[ ]:
```
from nltk.stem import PorterStemmer, LancasterStemmer
from nltk.stem.snowball import SnowballStemmer

# TODO
```

- Stemmer le mot "Re-testing" par ces trois stemmers, observer les résultats.

[ ]:
```
# TODO
```

- Ecrire une fonction qui cherche les N mots les plus fréquent dans le readme du corpus Reuter, print ces mots et leur fréquence

- Utiliser les trois stemmers que vous venez de créer pour stemmer les 15 mots. Print-les et comparér les résultats.

[ ]:
```
def most_frequent_word(N):
    pass

# test code
top15 = most_frequent_word(N=15)

# TODO
for mot in top15:
    pass
```

SnowballStemmer peut stemmer plusieurs langues, vous pouvez donc créer plusieurs snowballstemmer en fonction de la langue. - Créer une instance `snowball_fr` pour le français, et tester avec le mot “rerereconfinement” - Est-ce que le résultat est satisfisant ?

[ ]:
```
# TODO

snowball_fr = ...
```

1.1.2 Partie 2 Tokenizer avec regex

[ ]:
```
from nltk.tokenize import RegexpTokenizer
```

[ ]:
```
ding1 = "0091 R   Euh bon après de toute façon on va à la limite on va faire 1␣
 ↪tour pour euh alors oui si par contre au départ on va devoir à tour de rôle␣
 ↪placer notre première ville de départ plus 1 route euh adjacente à notre␣
 ↪ville"
ding2 = "0119 R   Euh faut que tu places aussi une route dans la direction qui␣
 ↪t'intéresse euh alors tout ce qui est construit normalement ne peut plus␣
 ↪être euh déconstruit ouais on peut pas détruire quelque chose par contre␣
 ↪ouais il faut peut-être qu'on attende un tout petit peu parce que c'est on␣
 ↪peut considérer que c'est le début du jeu"
ding3 = "0133 R   Alors ouais pour bien faire donc là au le premier tour de jeu␣
 ↪ça va être vraiment ça ça va être on va chacun mettre 2 moins on met une␣
 ↪ville chacun puis une deuxième ville avec une route adjacente et ensuite à␣
 ↪chaque tour on commence par lancer les dés ensuite on peut faire du commerce␣
 ↪euh et ensuite on fait nos achats et quand on a fini bah c'est le suivant␣
 ↪qui va lancer les dés et euh les les on va distribuer les ressources qui␣
 ↪correspondent et ensuite on va euh on va faire les différentes actions"
```

Note
Vous pouvez utiliser votre code du dernier fois pour obtenir le string `my_text` ou exécuter le code proposé. `my_text` combine les 3 extraits de DinG présentés au dessus. Faite en sorte que le résultat ne contienne que le texte, et non les indexations ni le nom du locuteur.

[ ]:
```
# Une proposition

my_text = ding1.split(' ', 2)[-1].strip() + ' ' + \
ding2.split(' ', 2)[-1].strip() + ' ' + ding3.split(' ', 2)[-1].strip()
my_text
```

Exo2 Utiliser le `regexpTokenizer`pour tokenizer le string `my_text`, et stocker le résultat dans une variable `text_tokenized`. Afficher le.

[ ]:
```
# TODO

regex_tokenizer = ...
```

1.1.3 Partie 3 Lemmatizer avec `WordNetLemmatizer`

Importer `WordNetLemmatizer` et créer une instance.
De la même manière qu'avec reuters, il faut d'abord importer wordnet dans Python :

```
>>> import nltk
>>> nltk.download('wordnet')
```

Exo 3 - Créer une instance `wnl`. - Pour obtenir le lemme, vous pouvez utiliser la syntaxe `wnl.lemmatize('mot', pos=?)`. Il existe un paramètre pour indiquer la catégorie grammaticale du mot (`pos=?`), cela sert à quoi ? Par exemple, pour le mot "brightening", tester avec deux catégories 'n' et 'v'. Qu'est-ce que vous observez ?

[ ]:
```
# TODO

from nltk import WordNetLemmatizer

wnl = ...
```

- Maintenant tester avec le mot "better" et "best", n'oublier pas de préciser la catégorie `pos=?`.

[ ]:
```
# TODO
```

- Discuter : À partir des différents exemples, quelle est la différence entre stemming et lemmatizing ?

[ ]:
```
# TODO
```

1.1.4 Partie 4 Un pipeline

Pour cet exercise, nous allons faire un pipeline de traitement du texte : on va extraire le vocabulaire du texte, et puis obtenir le lemme de chaque mot. Pour cela, vous allez faire :

- Ecrire une fonction `get_vocab(a_string, tokenizer)` qui prend deux arguments: un texte et une instance du tokenzier (e.g. word_tokenize) et retourne le vocabulaire de ce bout de texte. Pour ce dernier, vous pouvez utiliser `Text(tokenized_text).vocab()`. Vérifier le type de l'argument pour bien prendre en compte `word_tokenize` et `regextokenizer`.

- Ensuite, érire une fonction `get_lemme(alist, lemmatizer)` qui prend la sortie de la première fonction (i.e. : une liste des mots), et retourne le lemme en fonction du lemmatizer. Comparer les différents lemmatizers/stemmers que nous venons de voir (PorterStemmer, LancasterStemmer, SnowballStemmer, WordNetLemmatizer).

[ ]:
```python
from nltk.tokenize import word_tokenize, RegexpTokenizer

# TODO

def get_vocab(a_string: str, tokenizer):
    pass

def get_lemme(a_list: list, lemmatizer):
    pass


# test code, uncomment pour tester
# wt = get_vocab(a_string=readme, tokenizer=word_tokenize)
# reg = get_vocab(a_string=readme, tokenizer=RegexpTokenizer("[a-zA-Z0-9']+"))
# print(list(wt))
# print(list(reg))

# print(get_lemme(wt, porter))
# print(get_lemme(wt, lancaster))
# print(get_lemme(wt, snowball))
# print(get_lemme(wt, wnl))
```

1.1.5 /! Submission instructions

You will need to submit this lab on Arche before 9:59am on **Friday, 17th February**. Submit either a `.py` or an `.ipynb` file containing the functions you wrote for all the exercises and name it `td5_firstname_lastname.py` or `td5_firstname_lastname.ipynb` accordingly, where `firstname` should be your first name and `lastname` should be your last name (e.g. Jane Doe's submission should be called `td5_jane_doe.py` or `td5_jane_doe.ipynb`).