

```

1  #!/usr/bin/env python
2  # coding: utf-8
3  # *****
4  # Auteure: Eva Rolin
5  # Date: Novembre 2019
6  #*****
7  # Importation des modules et chargement du corpus d'apprentissage
8  # Nous entraînons notre modèle sur les prescripteurs de sens longueur et fréquence
9  # à partir du module Features Ranking
10 #*****
11 import spacy
12 nlp = spacy.load("fr_core_news_md")
13 import pandas as pd
14 import numpy as np
15 from sklearn import svm
16 from Features_ranking import *
17 simplification = pd.read_csv('train_2.csv')
18 features = simplification[['Longueur', 'Sens', 'Fréquence']].as_matrix()
19 type_label = np.where(simplification['Type']=='complexe', 0, 1)
20 model = svm.SVC(kernel='linear')
21 model.fit(features, type_label)
22 #*****
23 # Cette fonction permet de déterminer si notre mot est complexe ou pas
24 #*****
25 def complexe_or_simple(longueur,sens,fréquence):
26     if(model.predict([[longueur, sens,fréquence]])==0:
27         return "complexe"
28     else:
29         return "simple"
30 #*****
31 # L'ensemble de ces fonctions permettent de convertir nos dictionnaires
32 # en liste puis en chaine de caractères afin d'imprimer chaque phrase, mot, indice et
candidats
33 # dans le bon format
34 #*****
35 def lemmatize_fr_candidat(candidat,phrase,ident,target):
36     liste = []
37     phrase = phrase.replace(target,candidat)
38     doc2 = nlp(phrase)
39     for element in doc2:
40         liste.append(element.lemma_)
41     return liste[ident]
42 def test_dico_2(dico,target):
43     dicot = {}
44     liste = []
45     result = {}
46     for key,val in dico.items():
47         if key == target:
48             for i,j in val:
49                 if i not in dicot:
50                     dicot[i] = []
51                 try:
52                     dicot[i].append(float(j))
53                 except:
54                     dicot[i].append(j)
55     for key,value in dicot.items():
56         if value == [None]:
57             value = [0]
58         result[key] = value
59     for w in sorted(result, key= result.get, reverse=True):
60         liste.append(w)
61     return liste
62 def conversion_string(liste,target):
63     st = ""
64     try:
65         for elt in liste:

```

```

66         if elt!= target:
67             st += elt+" "
68     return st
69 except:
70     return st
71 *****
72 # Cette fonction principale permet de classer les candidats en fonction du résultat obtenu
73 # par la machine de vecteurs couplée au prescripteur de fréquence
74 *****
75 def main_SVM(victor_corpus,vic):
76     with open(victor_corpus,"r") as file:
77         with open(vic,"w") as file2:
78             texte = file.readline()
79             dictionnaire = {}
80             while texte:
81                 texte_cut = texte.strip().split("\t")
82                 phrase = texte_cut[0]
83                 target = texte_cut[1]
84                 index = int(texte_cut[2])
85                 try:
86                     candidat_cut = texte_cut[3]
87                 except:
88                     candidat_cut = ""
89                 cdt = candidat_cut.split()
90                 for candidat in cdt:
91                     longueur = prescripteur_longueur(candidat)
92                     cd = lemmatize_fr_candidat(candidat,phrase,index,target)
93                     sens = prescripteur_sens(cd)
94                     freq = prescripteur_frequence(candidat)
95                     if freq is None:
96                         freq = 0
97                     result = complexe_or_simple(longueur,sens,freq)
98                     if result == "simple":
99                         if target not in dictionnaire:
100                             dictionnaire[target] = []
101                             dictionnaire[target].append((candidat,freq))
102                 liste = test_dico_2(dictionnaire,target)
103                 result = conversion_string(liste,target)
104                 file2.write("{}\t{}\t{}\t{}".format(phrase,target,index,result))
105                 file2.write("\n")
106             texte = file.readline()
107

```