

```

1 #!/usr/bin/env python
2 #Auteur: Eva Rolin
3 #Date: Novembre 2019
4 #####
5 #Pour ce module devons importer le module frenetic qui permet un accès à l'api de Wonef.
6 #Nous avons trouvé ce module à l'adresse suivante:
7 # https://github.com/hardik-vala/FreNetic/blob/master/README.md
8 from frenetic import *
9 fwn = FreNetic("C:/Users/evaro/OneDrive/Documents/Github/LExenstein/EVAstein/wolf-1.0b4.xml")
10 import spacy
11 nlp = spacy.load("fr_core_news_md")
12 nlp2 = spacy.load("en")
13 from nltk.corpus import stopwords
14 from nltk.tokenize import word_tokenize
15 from googletrans import Translator
16 translator = Translator()
17 #####
18 # Cette fonction nous permet d'avoir les définitions d'un mot à partir
19 # son identifiant que nous avons extrait à l'aide d'une expression régulière.
20 # Si nous trouvons une définition pour le mot lemmatisé alors nous renvoyons une liste
21 # avec les différentes définitions. Sinon dans le cas contraire, nous renvoyons une liste
22 # contenant le mot rien (si nous renvoyons une liste vide, la suite du code est compliquée.
23 # Par conséquent nous contournons le problème de cette manière. Nous verrons que ce n'est pas
24 # un problème puisque si deux mots ont en commun le fait de ne pas avoir de définitions et
donc
25 # le mot "rien", nous les sélectionnons pas.
26 #####
27 import re
28 from string import punctuation
29 def get_list_of_definitions(word):
30     liste_definition_bis = ["rien"]
31     try:
32         synsets = fwn.synsets(word)
33         element = re.compile("eng-[0-9]+-[0-9]+-[a-z]")
34         resultat = element.findall(str(synsets))
35         nb = len(resultat)
36         liste_definition = []
37         compteur = 0
38         if nb > 1:
39             while nb != 0:
40                 for r in resultat:
41                     syn = fwn.synset(r)
42                     definition = syn.defn()
43                     liste_definition.append(definition)
44                     nb = nb - 1
45                     compteur += 1
46             return liste_definition
47         if nb == 1:
48             syn = fwn.synset(r)
49             definition = syn.defn()
50             liste_definition.append(definition)
51             return liste_definition
52     except:
53         return liste_definition_bis
54 #####
55 # Nous comparons chaque mot de la liste du mot target avec celle du candidat.
56 # Au préalable, nous supprimons les mots vides et la ponctuation
57 #####
58 def comparaison(list_of_definitionA, list_of_definitionB):
59     compteur = 0
60     try:
61         listeB = []
62         listeA = []
63         punctuation = [ "?", ",", ".", ";", "'", "[", "]", ")", "(" ]
64         EN_STOPWORDS = set(stopwords.words('english'))
65         FR_STOPWORDS = [ "ou", "en", "se", "un", "en", "ou", "est", "le", "la", "de", "des" ]

```

```

66         for element in list_of_definitionB:
67             word = word_tokenize(element)
68             for elt in word:
69                 if elt not in punctuation:
70                     if elt not in EN_STOPWORDS:
71                         if elt not in FR_STOPWORDS:
72                             listeB.append(elt)
73         for element in listeB:
74             if element in list_of_definitionA:
75                 compteur+=1
76     except:
77         pass
78     return compteur
79 *****
80 # Nous lemmatisons notre mot cible à l'aide du module spacy
81 *****
82 def lemmatize_fr(target,phrase,ident):
83     liste = []
84     doc = nlp(phrase)
85     for word in doc:
86         liste.append(word.lemma_)
87     return liste[ident]
88 *****
89 # Cette fonction nous permet transforme la liste de candidats obtenue en une suite de
caractères
90 # afin d'obtenir le bon format vector où chaque ligne est composée de:
91 # phrase mot indice candidat sélectionné candidat sélectionné etc
92 *****
93 def conversion_string(liste,target,lemma):
94     st = ""
95     try:
96         for elt in liste:
97             elt = elt.lower()
98             if elt!=target:
99                 if elt!=lemma:
100                     st += elt+" "
101     return st
102 except:
103     return st
104 *****
105 # Cette fonction permet de lemmatiser les candidats.
106 # Elle est différente de la précédente parce que nous remplaçons temporairement
107 # le mot complexe par son candidat dans la phrase afin d'obtenir un contexte pour
108 # les candidats et une meilleure lemmatisation
109 *****
110 def lemmatize_fr_candidat(candidat,phrase,ident,target):
111     liste = []
112     phrase = phrase.replace(target,candidat)
113     doc2 = nlp(phrase)
114     for element in doc2:
115         liste.append(element.lemma_)
116     return liste[ident]
117 *****
118 # Cette fonction permet, comme son nom l'indique, de nettoyer les phrases
119 *****
120 def clean_phrase(phrase):
121     exclude = [",","(",")"]
122     phrase = ''.join(ch for ch in phrase if ch not in exclude)
123     return phrase
124 *****
125 # Cette fonction permet de lemmatiser
126 # Cette fois-ci nous avons opté pour une lemmatisation anglaise étant donné
127 # que les définitions sont en anglais
128 *****
129 def lemmatize(target):
130     doc = nlp2(target)
131     for elt in doc:

```

```

132     return elt.lemma_
133 *****
134 # Cette fonction implémente l'algorithme de Lesk Simple.
135 # Elle renvoie le meilleur sens du mot
136 *****
137 def LeskSimplified(mot,phrase):
138     meilleur_sens = (get_list_of_definitions(mot))[0]
139     max_overlap = 0
140     contexte = contexte(mot,phrase)
141     for definition in get_list_of_definitions(mot):
142         sign = signature(definition)
143         overlap1 = overlap(contexte,sign)
144         if overlap1 > max_overlap:
145             max_overlap = overlap1
146             meilleur_sens = definition
147     return meilleur_sens
148 *****
149 # Nous récupérons les mots pleins de chaque définition afin d'obtenir une liste
150 # de tokens relative aux définitions
151 *****
152 def signature(definition):
153     EN_STOPWORDS = set(stopwords.words('english'))
154     punctuation = ["?", ",", ".", ";"]
155     liste_ok = []
156     definition_lemma = lemmatize(definition)
157     tokenize = word_tokenize(definition_lemma)
158     for element in tokenize:
159         if element not in EN_STOPWORDS:
160             if element not in punctuation:
161                 liste_ok.append(element)
162     return liste_ok
163 *****
164 # Nous déterminons le chevauchement entre les lemmes du contexte et ceux de la signature
165 *****
166 def overlap(contexte,signature):
167     compteur = 0
168     for element in signature:
169         if element in contexte:
170             compteur += 1
171     return compteur
172 *****
173 # Nous déterminons le contexte autour du mot.
174 # Puisque les définitions sont en anglais, nous avons décidé de passer par le module
175 # Googletrans pour traduire le contexte en anglais et pour avoir assuré une comparaison
176 *****
177 def contexte(mot,phrase):
178     liste_ok = []
179     contexte = phrase.replace(mot,"")
180     contexte = contexte.strip("")
181     traduction = str(translator.translate(contexte))
182     trad_cut = traduction.split(',')
183     trad_good = trad_cut[2]
184     trad_text = trad_good.split("=")
185     trad_text_ok = trad_text[1]
186     tokenize = word_tokenize(trad_text_ok)
187     for element in tokenize:
188         element = lemmatize(element)
189         if element.lower() not in EN_STOPWORDS:
190             liste_ok.append(element)
191     return liste_ok
192 *****
193 # Cette fonction permet de récupérer sous forme de liste les candidats de chaque clé
194 # c'est-à-dire de chaque mot complexe
195 *****
196 def dico(dic,target):
197     for key,val in dic.items():
198         if key == target:

```

```

199         return val
200     #####
201     # Cette fonction prend en argument un fichier au format victor et
202     # retourne un fichier également au format victor toujours
203     # au même format que pour les autres modules de sélection c'est-à-dire:
204     # phrase mot indice candidat sélectionné candidat sélectionné
205     #####
206     def get_selection(victor_corpus,victor_bis):
207         with open (victor_corpus, "r", encoding='utf-8') as file:
208             with open(victor_bis, "w",encoding='utf-8') as file2:
209                 texte = file.readline()
210                 data = []
211                 dictionnaire = {}
212                 while texte:
213                     texte_cut = texte.strip().split("\t")
214                     phrase = texte_cut[0].strip()
215                     target = texte_cut[1].strip()
216                     index_ok = (texte_cut[2]).strip()
217                     index = int(index_ok)
218                     try:
219                         candidat_cut = texte_cut[3].strip()
220                     except:
221                         candidat_cut = ""
222                     cdt = candidat_cut.split()
223                     ph = clean_phrase(phrase)
224                     lemme = lemmatize_fr(target,ph,index)
225                     liste_def_target = LeskSimplified(lemme,phrase)
226                     for c in cdt:
227                         phr_clean = clean_phrase(phrase)
228                         lemme = lemmatize_fr_candidat(c,phr_clean,index,target)
229                         liste_def_candidat = get_list_of_definitions(lemme)
230                         overlap = comparaison(liste_def_target,liste_def_candidat)
231                         if overlap >= 4:
232                             if target not in dictionnaire:
233                                 dictionnaire[target] = []
234                                 dictionnaire[target].append(c)
235                     liste_candidat = dico(dictionnaire,target)
236                     string = conversion_string(liste_candidat,target,lemme)
237                     file2.write("{}\t{}\t{}\t{}".format(phrase,target,index,string))
238                     file2.write("\n")
239                     texte = file.readline()
240

```