

```

1  #!/usr/bin/env python
2  # Auteure: Eva Rolin
3  # Date: Novembre 2019
4  #*****
5  # Nous devons d'abord charger les modules suivants: spacy pour les pos, xml pour parser notre
fichier.
6  # Nous avons téléchargé la version xml de ReSyf que nous avons obtenue à l'adresse suivante:
7  # https://cental.uclouvain.be/resyf/download.html
8  #*****
9  import spacy
10 nlp = spacy.load("fr_core_news_md")
11 import xml.etree.ElementTree as ET
12 tree = ET.parse("ReSyf.xml")
13 root = tree.getroot()
14 #*****
15 # Cette fonction nettoie la phrase des virgules et des parenthèses
16 #*****
17 def clean_phrase(phrase):
18     exclude = [",", "(", ")"]
19     phrase = ''.join(ch for ch in phrase if ch not in exclude)
20     return phrase
21 #*****
22 # Cette fonction lemmatise le mot cible puisque ReSyf fonctionne avec des formes lemmatisées
23 # à l'aide de la librairie spaCy
24 #*****
25 def lemmatize(target, phrase, ident):
26     liste = []
27     doc = nlp(phrase)
28     for word in doc:
29         liste.append(word.lemma_)
30     return liste[ident]
31 #*****
32 # Cette fonction détermine la partie du discours du mot cible à l'aide du module spaCy
33 #*****
34 def pos(target, phrase, ident):
35     liste = []
36     doc = nlp(phrase)
37     for word in doc:
38         liste.append(word.pos_)
39     return liste[ident]
40 #*****
41 # Cette fonction détermine la partie du discours de chaque candidat généré par reSyf.
42 # Puisque pour avoir la bonne pos, il faut absolument la phrase et donc un contexte,
43 # nous avons créé une fonction expressement pour les candidats: nous remplaçons temporairement
44 # le mot complexe par le candidat dont nous voulons le pos dans la phrase.
45 #*****
46 def pos_candidat(candidat, phrase, ident, target):
47     liste = []
48     phrase = phrase.replace(target, candidat)
49     doc2 = nlp(phrase)
50     for element in doc2:
51         liste.append(element.pos_)
52     return liste[ident]
53 #*****
54 # Cette fonction prend en entrée une liste et la transforme en chaînes de caractères
55 # pour les imprimer dans le format victor. Nous retenons uniquement les termes qui partagent
les
56 # mêmes pos que le mot cible et les mots qui sont différents du mot cible et de son lemme.
57 #*****
58 def conversion_string(liste, target, phrase, ident, lemma):
59     st = ""
60     pos_target = pos(target, phrase, ident)
61     for elt in liste:
62         if elt != target:
63             if elt != lemma:
64                 if pos_target == pos_candidat(elt, phrase, ident, target):

```

```

65         st += elt+" "
66     return st
67 *****
68 # Cette fonction permet à l'aide du parseur xml de parser notre fichier de sorte à obtenir
69 # les candidats de chaque sens du lemme. Nous obtenons en sortie une liste de synonymes.
70 # Si le lemme n'a pas de synonymes alors nous renvoyons une liste vide
71 *****
72 def get_synonyms_RESYF(lemma):
73     syn = ""
74     liste = []
75     for entree in root.iter('LexicalEntry'):
76         for elt in entree.iter('Lemma'):
77             for i in elt.iter('feat'):
78                 if i.get('att') == 'lexeme':
79                     if i.get('val') == lemma:
80                         for sens in entree.iter('Sense'):
81                             for s in sens.iter('SenseExample'):
82                                 for ok in sens.iter('feat'):
83                                     if ok.get('att') == "word":
84                                         syn = ok.get('val')
85                                         liste.append(syn)
86
87
88     return set(liste)
89 *****
90 # La fonction principale prend en argument un fichier en format victor et donne en sortie
91 # un fichier également au format victor avec les candidats générés par la ressource:
92 # phrase mot indice candidat1 candidat2 candidat3
93 *****
94 def get_subst(victor_corpus,victor_corpus_bis):
95     with open (victor_corpus, "r") as file:
96         with open(victor_corpus_bis,"w") as file2:
97             texte = file.readline()
98             while texte:
99                 texte_cut = texte.strip().split("\t")
100                 phrase = texte_cut[0]
101                 target = texte_cut[1]
102                 ident = int(texte_cut[2])
103                 phrase_cl = clean_phrase(phrase)
104                 lemma = lemmatize(target,phrase_cl,ident)
105                 liste_subs = get_synonyms_RESYF(lemma)
106                 candidat = conversion_string(liste_subs,target,phrase,ident,lemma)
107                 file2.write("{}\t{}\t{}\t{}".format(phrase,target,ident,candidat))
108                 file2.write("\n")
109                 texte = file.readline()
110     return liste_subs
111
112

```