

```

1 |#!/usr/bin/env python
2 |# Auteure: Eva Rolin
3 |# Date : Novembre 2019
4 |#*****
5 |# Nous devons pour ce module importer le parseur xml ainsi que la librairie spacy pour
lemmatiser
6 |# Enfin, au niveau des ressources, nous avons utilisé le fichier xml suivant:
http://rali.iro.umontreal.ca/rali/?q=fr/dem
7 |#*****
8 |import xml.etree.ElementTree as ET
9 |import spacy
10 |tree = ET.parse("DEM-1_0.xml")
11 |root = tree.getroot()
12 |nlp = spacy.load('fr_core_news_md')
13 |#*****
14 |# Cette fonction comme son nom l'indique a pour but de lemmatiser le mot complexe afin
15 |# qu'on puisse aisément le retrouver dans le fichier xml. De la même manière que ReSyf,
16 |# le Dictionnaire Electronique des Mots contient uniquement des lemmes.
17 |#*****
18 |def lemmatize(target):
19 |    doc = nlp(target)
20 |    for word in doc:
21 |        return word.lemma_
22 |#*****
23 |# Cette fonction retourne la partie du discours du mot complexe.
24 |#*****
25 |def pos_ok(target,phrase,ident):
26 |    liste = []
27 |    doc = nlp(phrase)
28 |    for word in doc:
29 |        liste.append(word.pos_)
30 |    return liste[ident]
31 |#*****
32 |# Cette fonction retourne pour un lemme, son sens. Cette fonction fonctionne pour toutes les
parties du discours
33 |# sauf pour les adjectifs pour lesquels nous avons créé une fonction spécifique.
34 |# Nous retenons uniquement les sens provenant des domaines qui sont proches de la médecine
35 |# pour éviter les ambiguïtés.
36 |# En sortie de notre module, nous recevons un mot assorti de sa paraphrase: mot(paraphrase).
37 |# Si le mot n'est pas trouvé alors on retourne le mot complexe.
38 |#*****
39 |def get_paraphrases(lemma,target):
40 |    string = ""
41 |    essai = ["anatomie","pathologie","biologie","médecine","pharmacie","bactérie,
virus","chirurgie","corps humain"]
42 |    try:
43 |        for entree in root.iter('entree'):
44 |            for mot in entree.iter('M'):
45 |                if mot.get('mot') == lemma:
46 |                    for m in entree.iter('DOM'):
47 |                        if m.get('nom') in essai:
48 |                            for ma in entree.iter('SENS'):
49 |                                string = ma.text
50 |                    return target+"("+string+)"
51 |    except:
52 |        return target
53 |#*****
54 |# Nous avons créé une fonction spécifique à l'adjectif suite aux constats que nous avons
fait.
55 |# En effet, exploiter le sens d'un adjectif n'est pas très informatif étant donné qu'il donne
56 |# la plupart du temps le sens suivant: adjectif(provient du nom).
57 |# Dès lors nous extrayons le nom dont l'adjectif provient et recherchons dans le fichier,
58 |# le sens de ce nom. Nous recevons en sortie le mot complexe assorti de sa paraphrase.
59 |# Si le mot n'a pas été trouvé, nous retournons le mot complexe.
60 |#*****
61 |def get_paraphrases_adj(lemma,target):

```

```

62     string = ""
63     paraphrase2 = ""
64     essai =
["anatomie", "pathologie", "biologie", "médecine", "pharmacie", "bactérie", "virus", "chirurgie", "corps
humain"]
65     essai2 =
["anatomie", "pathologie", "biologie", "médecine", "pharmacie", "bactérie", "virus", "chirurgie", "corps
humain"]
66     for entree in root.iter('entree'):
67         for mot in entree.iter('M'):
68             if mot.get('mot') == lemma:
69                 for m in entree.iter('DOM'):
70                     if m.get('nom') in essai:
71                         for ma in entree.iter('SENS'):
72                             string = ma.text
73     try:
74         paraphrase_cut = string.split()
75         nom_adj = paraphrase_cut[1]
76         for entree in root.iter('entree'):
77             for mot in entree.iter('M'):
78                 if mot.get('mot') == nom_adj:
79                     for m in entree.iter('DOM'):
80                         if m.get('nom') in essai:
81                             for ma in entree.iter('SENS'):
82                                 paraphrase2 = ma.text
83     return target+'('+paraphrase2+')'
84 except:
85     return target
86 *****
87 # Cette fonction principale prend en entrée un fichier du format victor et renvoie un fichier
88 # où chaque ligne correspond à une phrase et où chaque mot complexe est assorti de sa
paraphrase
89 # quand il en existe une.
90 *****
91 def get_subst(victor_corpus, test):
92     with open (victor_corpus, "r") as file:
93         with open (test, "w") as file2:
94             texte = file.readline()
95             mot = ""
96             new_phrase = ""
97             while texte:
98                 texte_cut = texte.strip().split("\t")
99                 phrase = texte_cut[0]
100                 target = texte_cut[1]
101                 ident = int(texte_cut[2])
102                 lemma_test = lemmatize(target)
103                 pos_essai = pos_ok(target, phrase, ident)
104                 if pos_essai == "ADJ":
105                     mot = get_paraphrases_adj(lemma_test, target)
106                     ok = str(phrase).replace(str(target), mot)
107                     file2.write(ok)
108                     file2.write("\n")
109                 else:
110                     mot = get_paraphrases(lemma_test, target)
111                     ok = str(phrase).replace(str(target), mot)
112                     file2.write(ok)
113                     file2.write("\n")
114                 texte = file.readline()
115
116

```