

```

1 | #!/usr/bin/env python
2 | # Auteure: Eva Rolin
3 | # Date: Novembre 2019
4 | #*****
5 | # Pour faire fonctionner ce module, nous avons besoin des modules gensim et spacy.
6 | # Nous avons également besoin du modèle préentraîné que nous avons téléchargé à l'adresse
suivante:
7 | # https://fasttext.cc/docs/en/crawl-vectors.html
8 | # Puisque nous avons opté pour un format texte nous devons télécharger le modèle de FastText à
l'aide
9 | # de Word2Vec
10 | # Pour l'écriture de ce script nous nous sommes aidés des conseils provenant de:
11 | # https://radimrehurek.com/gensim/models/keyedvectors.html
12 | #*****
13 | import gensim
14 | model =
gensim.models.KeyedVectors.load_word2vec_format('C:/Users/evaro/OneDrive/Documents/Github/LEXenstein/
EVAstein/cc.fr.300.vec/cc.fr.300.vec')
15 | import spacy
16 | nlp = spacy.load("fr_core_news_md")
17 | #*****
18 | # Sous les conseils du site internet https://radimrehurek.com/gensim/models/keyedvectors.html,
19 | # nous avons comparé chaque mot cible avec ses candidats et décidé de garder uniquement les
candidats
20 | # dont la similarité dépasse le seuil 0.5 sur base de plusieurs tests effectués auparavant.
21 | #*****
22 | def comparaison_vectors(target,synonym):
23 |     try:
24 |         pas_ok = ""
25 |         resultat_comparaison = model.similarity(target,synonym)
26 |         if resultat_comparaison > 0.5:
27 |             return synonym
28 |     except:
29 |         pass
30 | #*****
31 | # Cette fonction, comme dans chacun des autres scripts, permet de convertir chaque liste de
candidat
32 | # en une chaîne de caractères afin de pouvoir les écrire dans le format vector adéquat.
33 | # Nous sélectionnons uniquement les candidats qui ne sont pas le mot complexe ou le lemme de
celui-ci
34 | #*****
35 | def conversion_string(liste,target,lemma):
36 |     st = ""
37 |     try:
38 |         for elt in liste:
39 |             if elt != target:
40 |                 if elt != lemma:
41 |                     st += elt+" "
42 |         return st
43 |     except:
44 |         return st
45 | #*****
46 | # Cette fonction permet de lemmatiser le mot cible. Nous utilisons le lemme dans la fonction
précédente
47 | # afin de sélectionner uniquement les mots différents du mot complexe ou du lemme de celui-ci
48 | #*****
49 | def lemmatize_fr(target,phrase,ident):
50 |     liste = []
51 |     doc = nlp(phrase)
52 |     for word in doc:
53 |         liste.append(word.lemma_)
54 |     return liste[ident]
55 | #*****
56 | # Cette fonction nettoie la phrase des virgules et des parenthèses
57 | #*****
58 | def clean_phrase(phrase):

```

```

59     exclude = [",", "(", ")", "\""]
60     phrase = ''.join(ch for ch in phrase if ch not in exclude)
61     return phrase
62     *****
63     # Cette fonction permet pour chaque clé (mot complexe) de retourner la liste de candidats
64     *****
65     def dico(dic,target):
66         for key,val in dic.items():
67             if key == target:
68                 return val
69     *****
70     # Cette fonction prend en argument un fichier au format victor et donne en sortie
71     # un fichier également au format victor où pour chaque ligne, nous pouvons lire:
72     # phrase mot candidat sélectionné candidat sélectionné candidat sélectionné
73     *****
74     def final(victor_corpus,vic):
75         with open (victor_corpus, "r",encoding="utf-8") as file:
76             with open(vic,"w",encoding="utf-8") as file2:
77                 texte = file.readline()
78                 data = {}
79                 while texte:
80                     texte_cut = texte.strip().split("\t")
81                     phrase = texte_cut[0]
82                     target = texte_cut[1]
83                     index = int(texte_cut[2])
84                     try:
85                         candidat_cut = texte_cut[3]
86                     except:
87                         candidat_cut = ""
88                     phr_cl = clean_phrase(phrase)
89                     lemma = lemmatize_fr(target,phr_cl,index)
90                     cdt = candidat_cut.split()
91                     for c in cdt:
92                         if c!= target:
93                             comp = comparaison_vectors(target,c)
94                             if comp != None:
95                                 if target not in data:
96                                     data[target] = []
97                                     data[target].append(comp)
98                     liste = dico(data,target)
99                     string = conversion_string(liste,target,lemma)
100                     file2.write("{}\t{}\t{}\t{}".format(phrase,target,index,string))
101                     file2.write("\n")
102                     texte = file.readline()
103
104

```