

```

1 | #!/usr/bin/env python
2 | # coding: utf-8
3 | #*****
4 | # Auteure: Eva Rolin
5 | # Date: Novembre 2019
6 | # Ce script réutilise le code suivant: https://pypi.org/project/pytorch-pretrained-bert/
7 | # Nous téléchargeons tous les modules dont nous avons besoin à partir de pytorch
8 | # Au préalable, nous devons installer pytorch: pip install torch
9 | # Ensuite nous devons installer PyTorch version of Google AI BERT model with script to load
Google pre-trained models:
10 | # pip install pytorch-pretrained-bert
11 | # Nous téléchargeons le modèle pré-entraîné pour le tokeniseur. Il n'existe pas de modèle pré-
entraîné pour le français.
12 | # Nous avons donc opté pour le modèle multilingue proposé par Google (104 languages) contenant
le français.
13 | #*****
14 | import torch
15 | from pytorch_pretrained_bert import BertTokenizer, BertModel, BertForMaskedLM
16 | tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased')
17 | model = BertForMaskedLM.from_pretrained('bert-base-multilingual-cased')
18 | model.eval()
19 | #*****
20 | # A l'aide de cette fonction, nous transformons les phrase dans le format demandé par le
modèle
21 | # c'est-à-dire que les phrases doivent respecter le format: [CLS] Première phrase [SEP]
Deuxième phrase [SEP]
22 | #*****
23 | def transformation_sentences(sentences):
24 |     try:
25 |         true_sent = sentences.replace("/", " [SEP] ")
26 |         true_sent = "[CLS] " + true_sent
27 |         return true_sent
28 |     except:
29 |         pass
30 | #*****
31 | # Ensuite nous avons converti nos phrases en une suite d'indices étant donné
32 | # que ce format est demandé par le modèle.
33 | # Pour la première phrase, nous devons remplacer chaque élément par 0.
34 | # Pour la deuxième phrase, nous devons remplacer chaque élément par 1.
35 | # *****
36 | def convert(sentences):
37 |     liste_convert = [0]
38 |     compteur = 0
39 |     for element in sentences:
40 |         if element != '[SEP]':
41 |             liste_convert.append(compteur)
42 |         if element == '[SEP]':
43 |             compteur += 1
44 |             liste_convert.append(compteur)
45 |     return liste_convert[0:len(liste_convert)-1]
46 | #*****
47 | # Cette fonction masque le mot complexe et donne en sortie le mot que BERT a prédit
48 | # en fonction du contexte de gauche et de droite
49 | #*****
50 | def tokeniseur(sentences,target):
51 |     tokenized_text = tokenizer.tokenize(sentences)
52 |     try:
53 |         if target in tokenized_text:
54 |             masked_index = tokenized_text.index(target)
55 |             tokenized_text[masked_index] = '[MASK]'
56 |             indexed_tokens = tokenizer.convert_tokens_to_ids(tokenized_text)
57 |             segments_ids = convert(tokenized_text)
58 |             tokens_tensor = torch.tensor([indexed_tokens])
59 |             segments_tensors = torch.tensor([segments_ids])
60 |             with torch.no_grad():
61 |                 predictions = model(tokens_tensor, segments_tensors)

```

