RITAL : Chirac ou Mitterand ?

Etudiant : **Nils Barrellon 21401602**

Github associé : https://github.com/nbarrellon/RITAL2026

1 Formulation du problème

Chirac ou Mitterrand : on dispose de la transcription d'un débat entre les deux présidents de la République, MM Chirac et Mitterrand. Chaque phrase prononcée par l'un ou l'autre est labellisée. On attribue le label M pour celles proposées par M.Mitterrand que l'on convertit en -1 et le label C (converti en 1) si c'est M.Chirac qui parle.
Les phrases labellisées de ce débat constitueront donc les données d'entraînement.
L'objectif du projet est de pouvoir attribuer à son auteur une phrase quelconque prononcée dans un autre contexte.

2 Bag of words

On crée différents bag of words (bog) pour nos documents afin d'en tester la pertinence.

2.1 Pré-processing

On effectue en maont de la construction du bog un nettoyage du texte. Les options retenues sont :

- suppression de la ponctuation ;
- suppression des chiffres ;
- mise en minuscule ;

Ce nettoyage est opéré par la fonction personnalisée de pré-processing suivante :

```
import codecs
import re
import string
from unidecode import unidecode

def preprocessPerso(text):
    #suppression de la ponctuation
    punc = string.punctuation
    punc += '\n\r\t'
    text = text.translate(str.maketrans(punc, ' '*len(punc)))
    #suppression des chiffres
    text = re.sub(r"[0-9]","",text)
    #suppression des accents
    text = unidecode(text)
    #mise en minuscule
    return text.lower()
```

2.2 Stopwords

On supprime les mots blancs avant construction du bog. La liste des stopwords est obtenue par concaténation des stopwords français donnés par la bibliothèque nltk auxquels on ajoute une liste personnelle, établie à partir de la liste des mots les plus présents et non-discriminants.

```
# Mots blancs + mots les plus fréquents

nltk.download('stopwords')
from nltk.corpus import stopwords
final_stopwords_list = stopwords.words('french')
#mots les plus fréquents donc non-discriminants
final_stopwords_list += ['ai','au','aujourd','aussi','autres','aux','
   avec','avez' ,'avons',
 'bien', 'ce' ,'cela', 'ces', 'cette' ,'ceux', 'chacun' ,'comme', '
    dans' ,'date'
 'de', 'depuis', 'des' ,'deux' ,'dire' ,'doit' ,'dont' ,'du'
 'elle' ,'en' ,'encore', 'ensemble', 'entre', 'est' ,'et', 'etat', '
    ete' ,'etre',
 'europe' ,'faire' ,'fait' ,'faut', 'francais' ,'france', 'hui' ,'ici'
    , 'il' ,'ils',
 'je' ,'la', 'le' ,'les', 'leur' ,'leurs', 'mais', 'meme' ,'monde', '
    monsieur' ,'ne',
 'nom', 'nos', 'notre', 'nous', 'on', 'ont' ,'ou', 'paix', 'par', 'pas
    ', 'pays',
 'peut' ,'plus' ,'politique' ,'pour', 'president' ,'qu', 'que' ,'qui',
     'sa', 'sans',
 'se' ,'ses' ,'si' ,'son' ,'sont', 'sur', 'temps', 'tous', 'tout', '
    toute' ,'toutes',
 'tres', 'un' ,'une' ,'union', 'vie', 'vos' ,'votre', 'vous']
```

Cette liste a été obtenue à l'aide du script ci-dessous :

```
from sklearn.feature_extraction.text import TfidfVectorizer

use_idf=True
smooth_idf=True
sublinear_tf=False

vectorizer = TfidfVectorizer(preprocessor=preprocessPerso,use_idf=
   use_idf, smooth_idf=smooth_idf, sublinear_tf=sublinear_tf,
   max_features=100)
BOW1 = vectorizer.fit_transform(alltxts)
vocabulaire = vectorizer.get_feature_names_out()
```

2.3 Premier bog

Bag of words simple. Taille du vocabulaire = 26894

```
vectorizer = CountVectorizer(stop_words=final_stopwords_list,analyzer=
    "word", preprocessor=preprocessPerso)
BOW1 = vectorizer.fit_transform(alltxts)
```

2.4 Deuxième bog

Utilisation de TF-IDF. Nombre de mots du vocabulaire : 2000

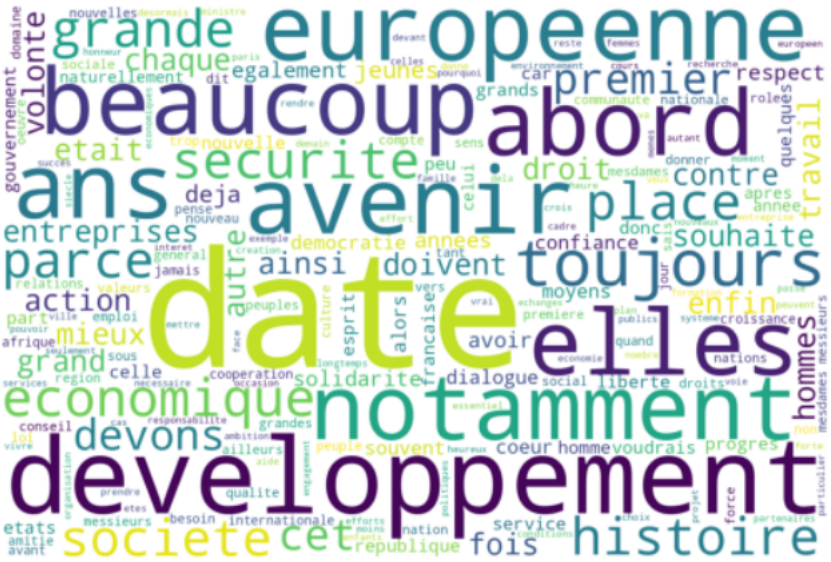
```
use_idf=True
smooth_idf=True
sublinear_tf=False

vectorizer = TfidfVectorizer(use_idf= use_idf, smooth_idf=smooth_idf,\
sublinear_tf=sublinear_tf,                    preprocessor=
   preprocessPerso,stop_words=final_stopwords_list,max_features=2000)
BOW2 = vectorizer.fit_transform(alltxts)
```

2.5 Troisième bog

Utilisation des bigrammes. Taille du vocabulaire = 328573

```
ngram_range = (1,2)
vectorizer = CountVectorizer(ngram_range=ngram_range,analyzer='word',
   stop_words=final_stopwords_list,preprocessor=preprocessPerso)
BOW3 = vectorizer.fit_transform(alltxts)
```



2.6 Quatrième bog

On utilise la racinisation avant construction du bog.

```
nlp = spacy.load("fr_core_news_sm", disable=["parser", "ner"])

def lemmatisation(text, stopwords):
    text = preprocessPerso(text)
    doc = nlp(text.lower())
    # Lemmatiser uniquement les mots qui ne sont pas des stopwords
    lemmatized = [token.lemma_ for token in doc if token.text not in
       stopwords]
    text = " ".join(lemmatized).strip()
    #on supprime les espaces surnuméraires créés par la racinisation
    text = re.sub(r'\s+', ' ', text)
    return text

# Pré-traiter tous les textes
alltxts_lemmatise = []
for text in alltxts:
    alltxts_lemmatise.append(lemmatisation(text, final_stopwords_list)
       )
```

Le résultat est très probant comme le montre l'exemple ci-dessous (1re phrase du débat avant et après racinisation) :

```
C'est toujours très émouvant de venir en Afrique car c'est probablement
l'une des rares terres du monde où l'on ait conservé cette convivialité,
cette amitié, ce respect de l'autre qui s'expriment avec chaleur, avec spontanéité
et qui réchauffent le coeur de ceux qui arrivent et de ceux qui reçoivent.

--------------------------------
être toujours tre emouver venir afrique car être probablement rare terre avoir
conserve convivialite amitie respect autre exprimer chaleur spontaneite rechauffer
coeur celui arriver celui recoiver
```

Taille du vocabulaire = 19049
On note que, sans surprise, les auxiliaires être et avoir arrivent en tête des mots les plus fréquents.
Il convient peut-être de les ajouter aux mots blancs.

2.7 Cinquième bog

On conjugue bigrammes et racinisation. Taille du vocabulaire = 328573

```
ngram_range = (1,2) # unigrams and bigrams
vectorizer = CountVectorizer(ngram_range=ngram_range,analyzer='word',\
stop_words=final_stopwords_list,preprocessor=preprocessPerso) # Maybe
   2-grams or 3-grams bring improvements ?
BOW5 = vectorizer.fit_transform(alltxts_lemm)
```

On note que les mots les plus fréquents sont des mnogrammes.

2.8 Sixième bog

On travaille avec un bog binaire. Taille du vocabulaire = 10000

```
min_df=5
max_df=0.5
max_features=10000
vectorizer = CountVectorizer(max_df=max_df,min_df=min_df,max_features=
    max_features,\
binary=True,stop_words=final_stopwords_list) #try out some values
BOW6 = vectorizer.fit_transform(alltxts)
```

2.9 Septième bog

On applique une racinisation avant un TF-IDF. Taille du vocabulaire = 4000

```
use_idf=True
smooth_idf=True
sublinear_tf=False
final_stopwords_list += ["avoir","être"]
vectorizer = TfidfVectorizer(use_idf= use_idf, smooth_idf=smooth_idf,
    sublinear_tf=sublinear_tf,\
                                stop_words=final_stopwords_list,
                                   max_features=4000)
BOW7 = vectorizer.fit_transform(alltxts_lemm)
```

2.10 Huitième bog

On applique une racinisation avant un TF-IDF sur des bigrammes. Taille du vocabulaire = 4000

```
use_idf=True
smooth_idf=True
sublinear_tf=False
final_stopwords_list += ["avoir","être"]
vectorizer = TfidfVectorizer(use_idf= use_idf, smooth_idf=smooth_idf,
   sublinear_tf=sublinear_tf,\
                             stop_words=final_stopwords_list,
                                max_features=4000)
BOW7 = vectorizer.fit_transform(alltxts_lemm)
```

3 Classificateurs

4 Métriques

Je me suis aperçu, en séparant les données par locuteur en 2 corpus distincts, que M.Chirac a beaucoup plus parlé que M.Mitterrand lors de ce débat.

```
corpus1 = [alltxts[i] for i in range(len(alltxts)) if alllabs[i]==1] #
   Chirac
corpus2 = [alltxts[i] for i in range(len(alltxts)) if alllabs[i]==-1]
   #Mitterand

#Taille des corpus
print("------------------Taille des corpus----------------------")
print("Chirac:",len(corpus1))
print("Mitterrand",len(corpus2))
```

```
------------------Taille des corpus----------------------
Chirac: 49890
Mitterrand 7523
```

Ceci a une influence considérable sur l'entrainement et les prédictions des différents modéles. La métrique de la précision, très bonne quelque soit le bog utilisé, n'est plus pertinente.