



Université Mohammed V
Faculté des Sciences
Rabat

NLP Projet

Traitement automatique des langues

Réalisé par :
M. MOHAMMED SAMIR
M. BOUZAIDI MOUAD

Encadré par :
Abdelhak MAHMOUDI

FSR RABAT – IDDL0 – 2019/2020

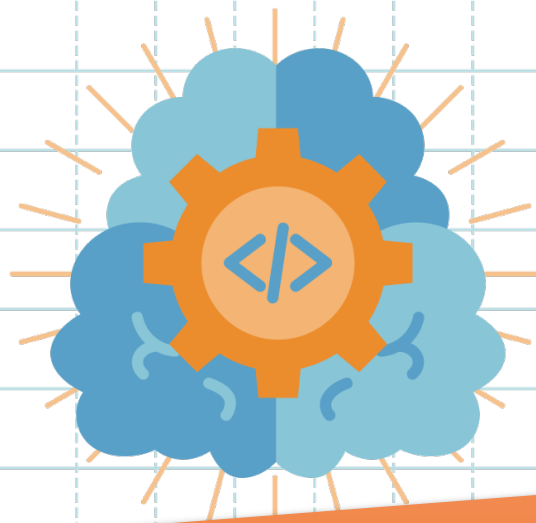


Table des matières

INTRODUCTION	3
INSTALLATION	4
CLASSIFICATION DES DONNÉES TEXTUELLES	7
<i>Analyse de données :.....</i>	<i>7</i>
<i>EXTRACTION DE FEATURES</i>	<i>10</i>
<i>CLASSIFICATION NAÏVE BAYESIENNE.....</i>	<i>10</i>
<i>RESULTATS</i>	<i>11</i>
CONCLUSION	12
Bibliographie	13

INTRODUCTION

Le traitement du langage naturel est un des domaines les plus en vue de l'intelligence artificielle. Il s'agit principalement de la parole et du texte qui sont produits, à des quantités faramineuses sous forme d'emails, de SMS, de pages web : blogs, réseaux sociaux.

Le langage naturel est la façon dont nous, humains, communiquons les uns avec les autres. L'analyse des sentiments ou l'analyse des opinions est l'étude computationnelle des opinions, des sentiments, des attitudes et des émotions des personnes exprimées dans le langage naturel écrit. [1]

Le Natural Language Toolkit est relativement mature (il est en développement depuis 2001) et s'est positionné comme l'une des principales ressources en matière de traitement du langage naturel. NLTK possède des outils pour presque toutes les tâches NLP. [2]

Avec l'avènement du web 2.0, la manière dont les personnes expriment leur opinion a beaucoup changée : nous postons des critiques de produits de consommation sur des sites marchands et exposons nos points de vue sur presque tous les sujets, sur des forums, des groupes de discussion ou des blogs.

Tout cela constitue une importante source d'information avec de nombreuses applications. C'est pourquoi, au cours des dernières années, de nombreux travaux ont pris pour objet la fouille d'opinion.

Actuellement, l'un des meilleurs exemples de réseaux sociaux permettant d'observer l'évolution des opinions est Twitter, Dans notre projet, on va analyser des différents opinions à propos d'un film écrites dans des tweets sur le réseau social Twitter.

INSTALLATION

Nous avons commencé par l'installation de pip3 qui nous a permis d'installer la bibliothèque logicielle de python NLTK dans un ordinateur OSX :

Téléchargement du fichier d'installation : curl <https://bootstrap.pypa.io/get-pip.py> -o get-pip.py

```
MacDeMouad:source Mouad$ ls
get-pip.py      iddlo.py
MacDeMouad:source Mouad$ python3 get-pip.py
Collecting pip
  Downloading pip-20.1.1-py2.py3-none-any.whl (1.5 MB)
    |#####| 1.5 MB 1.1 MB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 19.2.3
    Uninstalling pip-19.2.3:
      Successfully uninstalled pip-19.2.3
Successfully installed pip-20.1.1
MacDeMouad:source Mouad$ which pip3
/usr/local/bin/pip3
MacDeMouad:source Mouad$ pip3 install -U nltk
Collecting nltk
  Using cached nltk-3.5.zip (1.4 MB)
Collecting click
  Using cached click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting joblib
  Downloading joblib-0.16.0-py3-none-any.whl (300 kB)
    |#####| 300 kB 434 kB/s
Collecting regex
  Using cached regex-2020.7.14.tar.gz (690 kB)
Collecting tqdm
  Using cached tqdm-4.48.0-py2.py3-none-any.whl (67 kB)
Building wheels for collected packages: nltk, regex
  Building wheel for nltk (setup.py) ... done
  Created wheel for nltk: filename=nltk-3.5-py3-none-any.whl size=1434674 sha256=fd47e733bd5770e335ce819ea31fb466cedd7f41499d8645ff7b0a9bf9f9e5a5
  Stored in directory: /Users/Mouad/Library/Caches/pip/wheels/45/6c/46/a1865e7ba706b3817f5d1b2ff7ce8996aabd0d03d47ba0266
  Building wheel for regex (setup.py) ... done
  Created wheel for regex: filename=regex-2020.7.14-cp37m-macosx_10_11_x86_64.whl size=286421 sha256=9c02f35809c45459d169382f21d46a24df180891e337eb07eb871a58dd84996
  Stored in directory: /Users/Mouad/Library/Caches/pip/wheels/5b/68/ce/2508b5a5afc13bd96566c62d3ffebca7b401477c2ead3e8cc0
Successfully built nltk regex
Installing collected packages: click, joblib, regex, tqdm, nltk
Successfully installed click-7.1.2 joblib-0.16.0 nltk-3.5 regex-2020.7.14 tqdm-4.48.0
MacDeMouad:source Mouad$
```

Ensuite : pip3 install --user -U nltk, et nous avons importé la bibliothèque NLTK dans notre mini projet

```
1 |#!/usr/local/bin/python
2 |#Importation de NLTK
3 |import nltk
```

Après, dans python 3, nous avons téléchargé le tokenizer punkt pour la division du text en phrases :

```
>>> import nltk
>>> nltk.download('punkt')
[nltk_data] Downloading package punkt to /Users/Mouad/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.
True
>>>
```

Installation de 'sklearn' : pip3 install -U scikit-learn scipy matplotlib

```
MacDeMouad:~ Mouad$ pip3 install -U scikit-learn scipy matplotlib
Collecting scikit-learn
  Downloading scikit-learn-0.23.1-cp37m-macosx_10_9_x86_64.whl (7.2 MB)
    |#####| 7.2 MB 1.1 MB/s
Collecting scipy
  Downloading scipy-1.5.2-cp37m-macosx_10_9_x86_64.whl (28.7 MB)
    |#####| 28.7 MB 1.2 MB/s
Collecting matplotlib
  Downloading matplotlib-3.3.0-1-cp37m-macosx_10_9_x86_64.whl (11.4 MB)
    |#####| 11.4 MB 1.2 MB/s
Requirement already satisfied, skipping upgrade: joblib>=0.11 in /usr/local/lib/python3.7/site-packages (from scikit-learn) (0.16.0)
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-2.1.0-py3-none-any.whl (12 kB)
Requirement already satisfied, skipping upgrade: numpy>=1.13.3 in /usr/local/lib/python3.7/site-packages (from scikit-learn) (1.18.1)
Requirement already satisfied, skipping upgrade: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/site-packages (from matplotlib) (1.1.0)
Requirement already satisfied, skipping upgrade: pyparsing>=2.0.4,!=2.1.2,!=2.1.6,!=2.0.3 in /usr/local/lib/python3.7/site-packages (from matplotlib) (2.4.6)
Requirement already satisfied, skipping upgrade: cycler>=0.10 in /usr/local/lib/python3.7/site-packages (from matplotlib) (0.10.0)
Collecting pillow>=6.2.0
  Downloading Pillow-7.2.0-cp37m-macosx_10_9_x86_64.whl (2.2 MB)
    |#####| 2.2 MB 1.1 MB/s
Requirement already satisfied, skipping upgrade: python-dateutil>=2.1 in /usr/local/lib/python3.7/site-packages (from matplotlib) (2.8.1)
Requirement already satisfied, skipping upgrade: setuptools in /usr/local/lib/python3.7/site-packages (from kiwisolver=1.0.1->matplotlib) (48.0.0)
Requirement already satisfied, skipping upgrade: six in /Library/Python/3.7/lib/python/site-packages (from cycler>=0.10->matplotlib) (1.12.0)
Installing collected packages: scipy, threadpoolctl, scikit-learn, pillow, matplotlib
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.1.3
    Uninstalling matplotlib-3.1.3:
      Successfully uninstalled matplotlib-3.1.3
Successfully installed matplotlib-3.3.0 pillow-7.2.0 scikit-learn-0.23.1 scipy-1.5.2 threadpoolctl-2.1.0
MacDeMouad:~ Mouad$
```

Nous avons fait un petit test :

```
1  #!/usr/local/bin/python
2  #Importation de NLTK
3  import nltk
4  from nltk.tokenize import sent_tokenize
5  text="""La Faculté des sciences-Rabat (FSR) a été créée en 1952.
   Elle fait partie de l'ensemble des facultés de l'Université
   Mohammed V de Rabat. Elle est la faculté où les étudiants
   bacheliers poursuivent leurs études supérieures dans le domaine
   scientifique (mathématiques, informatique, physique, science de
   la vie et de l'univers, chimie)."""
6  tokenized_text=sent_tokenize(text)
7  print(tokenized_text)
```

MacDeMouad:source Mouad\$ python3 iddlo.py
['La Faculté des sciences-Rabat (FSR) a été créée en 1952.', 'Elle fait partie de l'ensemble des facultés de l'Université Mohammed V de Rabat.', 'Elle est la faculté où les étudiants bacheliers poursuivent leurs études supérieures dans le domaine scientifique (mathématiques, informatique, physique, science de la vie et de l'univers, chimie).']
MacDeMouad:source Mouad\$

Ou bien la division en mots :

```
1  #!/usr/local/bin/python
2  #Importation de NLTK
3  import nltk
4  from nltk.tokenize import sent_tokenize
5  from nltk.tokenize import word_tokenize
6  text="""La Faculté des sciences-Rabat (FSR) a été créée en 1952.
   Elle fait partie de l'ensemble des facultés de l'Université
   Mohammed V de Rabat. Elle est la faculté où les étudiants
   bacheliers poursuivent leurs études supérieures dans le domaine
   scientifique (mathématiques, informatique, physique, science de
   la vie et de l'univers, chimie)."""
7  #Division en phrases
8  #tokenized_text=sent_tokenize(text)
9  #print(tokenized_text)
10 #Division en mots
11 tokenized_word=word_tokenize(text)
12 print(tokenized_word)
```

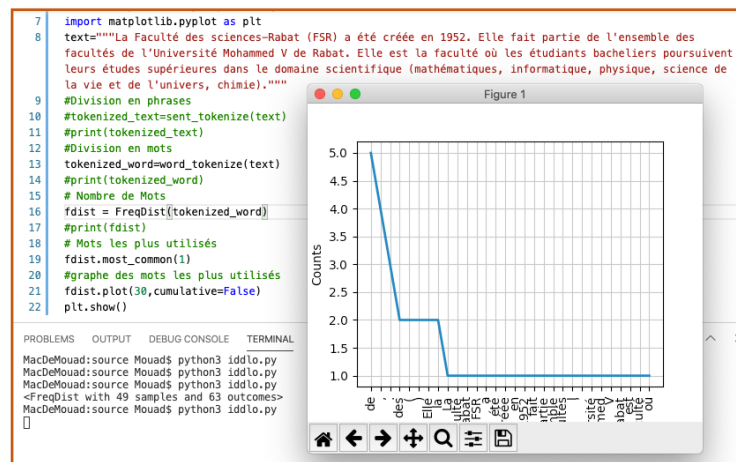
MacDeMouad:source Mouad\$ python3 iddlo.py
['La', 'Faculté', 'des', 'sciences-Rabat', '(', 'FSR', ')', 'a', 'été', 'créée', 'en', '1952', '.', 'Elle', 'fait', 'partie', 'de', 'l\'ensemble', 'des', 'facultés', 'de', 'l\'', 'Université', 'Mohammed', 'V', 'de', 'Rabat', '.', 'Elle', 'est', 'la', 'faculté', 'où', 'les', 'étudiants', 'bacheliers', 'poursuivent', 'leurs', 'études', 'supérieures', 'dans', 'le', 'domaine', 'scientifique', '(', 'mathématiques', ',', 'informatique', ',', 'physique', 'et', 'de', 'la', 'vie', 'et', 'de', 'l\'univers', ',', 'chimie', ')', '.']
MacDeMouad:source Mouad\$

Pour savoir combien de mots on a:

```
6  from nltk.probability import FreqDist
7  text="""La Faculté des sciences-Rabat (FSR) a été créée en 1952.
   Elle fait partie de l'ensemble des facultés de l'Université
   Mohammed V de Rabat. Elle est la faculté où les étudiants
   bacheliers poursuivent leurs études supérieures dans le domaine
   scientifique (mathématiques, informatique, physique, science de
   la vie et de l'univers, chimie)."""
8  #Division en phrases
9  #tokenized_text=sent_tokenize(text)
10 #print(tokenized_text)
11 #Division en mots
12 tokenized_word=word_tokenize(text)
13 #print(tokenized_word)
14 fdist = FreqDist(tokenized_word)
15 print(fdist)
```

MacDeMouad:source Mouad\$ python3 iddlo.py
<FreqDist with 49 samples and 63 outcomes>
MacDeMouad:source Mouad\$

Et pour présenter les mots les plus utilisés à l'aide d'un graphe :



Vous pouvez remarquer que le mot le plus utilisé c'est 'de'.

CLASSIFICATION DES DONNÉES TEXTUELLES

Dans cette partie nous allons comprendre les méthodes qui permettent de transformer le texte en features exploitables par des algorithmes de machine learning, et les architectures et modèles qui correspondent le mieux à ce type de données.

Dans notre cas, nous allons essayer de créer un algorithme d'analyse des sentiments. Pour ce faire, nous allons commencer par l'utilisation de la base de données 'movie_reviews' qui fait partie du NLTK. À partir de là, nous essaierons d'utiliser des mots comme 'features' qui font partie d'une critique de film positive ou négative.

L'ensemble de données movie_reviews du NLTK contient les opinions et elles sont déjà étiquetées comme positives ou négatives. Cela signifie que nous pouvons nous entraîner et tester avec ces données.

Analyse de données :

Pour pouvoir utiliser les algorithmes de Machine et Deep Learning avec le NLP, il nous faut tout d'abord savoir préparer notre corpus avant de connaître quelles sont les méthodes qui permettent de transformer ce dernier en features exploitables par ces programmes.

```
Python 3.6.5 Shell
> In [1]: import nltk
import random
from nltk.corpus import movie_reviews

> In [2]: movie_reviews

<CategorizedPlainTextCorpusReader in '/Users/Mouad/nltk_data/corpora/movie_reviews'>
```

Notre base de données se trouve dans :

/Users/Mouad/nltk_data/corpora/movie_reviews

La commande `movie_reviews.fileids()` affiche la liste des fichiers constituant notre base de données :

```
Python 3.6.5 Shell
> In [3]: movie_reviews.fileids()

t',
'neg/cv160_10848.txt',
'neg/cv161_12224.txt',
'neg/cv162_10977.txt'
```

```
Python 3.6.5 Shell
> In [4]: movie_reviews.categories()

['neg', 'pos']
```

Ces fichiers sont divisés en deux catégories/class qui sont : neg (négative) et pos (positive)

On a regroupé tous les documents (neg et pos) dans une seule grande liste:

```
documents = [(list(movie_reviews.words(fileid)), category)
              for category in movie_reviews.categories()
              for fileid in movie_reviews.fileids(category)]
random.shuffle(documents)
```

La fonction shuffle va mélanger les éléments de notre liste, par défaut la liste commence par les négatives ensuite les positives, nous on va utiliser cette liste pour l'entraînement et le test.

Vérifions la taille de la liste :

len(documents)

[illegible]

Les mots du première
élément de la liste,
catégorie positive (pos)

Les mots du deuxième
élément de la liste, catégorie
négative (neg) :

```
print(documents[1])
```


Maintenant nous allons regrouper tous les mots dans une seule liste, et afficher 100 mots les plus fréquents:

```

> MI
all_words = []
for w in movie_reviews.words():
    all_words.append(w.lower())
all_words = nltk.FreqDist(all_words)
print(all_words.most_common(100))

[(',', 77717), ('the', 76529), ('.', 65876), ('a', 38106), ('and', 35576), ('of', 34123), ('to', 31937), ('"', 30585), ('is', 25195), ('in', 21822), ('s', 18513), ('"', 17612), ('it', 16107), ('that', 15924), ('-', 15595), ('', 11781), ('(', 11664), ('as', 11378), ('with', 10792), ('for', 9961), ('his', 9587), ('this', 9578), ('film', 9517), ('i', 8889), ('he', 8864), ('but', 8634), ('on', 7385), ('are', 6949), ('t', 6410), ('by', 6261), ('be', 6174), ('one', 5852), ('movie', 5771), ('an', 5744), ('who', 5692), ('not', 5577), ('you', 5316), ('from', 4999), ('at', 4986), ('was', 4940), ('have', 4901), ('they', 4825), ('has', 4719), ('her', 4522), ('all', 4373), ('?', 3771), ('there', 3770), ('like', 3690), ('so', 3683), ('out', 3637), ('about', 3523), ('up', 3405), ('more', 3347), ('what', 3322), ('when', 3258), ('which', 3161), ('or', 3148), ('she', 3141), ('their', 3122), (':', 3042), ('some', 2985), ('just', 2905), ('can', 2882), ('if', 2799), ('we', 2775), ('him', 2633), ('into', 2623), ('even', 2565), ('only', 2495), ('than', 2474), ('no', 2472), ('good', 2411), ('time', 2411), ('most', 2306), ('its', 2270), ('will', 2216), ('story', 2169), ('would', 2109), ('been', 2050), ('much', 2049), ('character', 2020), ('also', 1967), ('get', 1949), ('other', 1948), ('do', 1915), ('two', 1911), ('well', 1906), ('them', 1877), ('very', 1863), ('characters', 1859), (';', 1850), ('first', 1836), ('--', 1815), ('after', 1762), ('see', 1749), ('!', 1713), ('way', 1693), ('because', 1684), ('make', 1642), ('life', 1586)]

```

Les premières mots les plus populaires sont en fait des ponctuations, mais nous arrivons rapidement à des mots légitimes, cela ne devrait pas poser de problème parceque nous arrivons rapidement à des mots légitimes et nous avons l'intention de stocker des milliers des mots.

Mais quand même nous allons faire par la suite un nettoyage en supprimant les stopwords.

Après l'ajout à l'entête :

```

> MI
stop = stopwords.words("english")
print(stop)

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'you're', 'you've', 'you'll', 'you'd', 'your', 'your's', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'she's', 'her', 'hers', 'herself', 'it', 'it's', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'that'll', 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did',

```

En affiche la liste des stopword.

Ce sont les mots très courants mais ils n'apportent pas de valeur informative pour la compréhension du "sens" d'un document et corpus.

Après la suppression des stopward qui sont très fréquents et ralentissent notre travail :

```

> MI
all_words = []
for w in movie_reviews.words():
    if w not in stop and len(w)>1:
        all_words.append(w.lower())
all_words = nltk.FreqDist(all_words)
print(all_words.most_common(100))

[('film', 9517), ('one', 5852), ('movie', 5771), ('like', 3690), ('even', 2565), ('good', 2411), ('time', 2411), ('story', 2169), ('would', 2109), ('much', 2049), ('character', 2020), ('also', 1967), ('get', 1949), ('two', 1911), ('well', 1906), ('characters', 1859), ('first', 1836), ('--', 1815), ('see', 1749), ('characters', 1859), ('make', 1642), ('life', 1586), ('really', 1558), ('films', 1536), ('plot', 1513), ('little', 1501), ('people', 1455), ('could', 1427), ('scene', 1397), ('man', 1396), ('bad', 1395), ('never', 1374), ('best', 1333), ('new',

```

```

> MI
len(all_words)

39768

```

```

> MI
len(all_words)

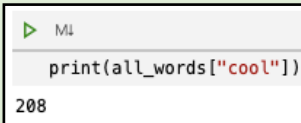
39554

```

Les mots avec 1 seul caractère sont aussi supprimés.

L'ancien nombre de mots était 39768, après nettoyage il deviendra 39554

On a exécuté la commande `print(all_words["cool"])` pour savoir combien de fois le mot ‘good’ est répété :



```
ML
print(all_words["cool"])
208
```

Nous avons effectué quelques étapes essentielles du prétraitement du texte, A partir de ces mots on va extraire les ‘features’.

EXTRACTION DE FEATURES

Nous pouvons maintenant passer à la création de notre ensemble de features représentatives de notre corpus de texte.

```
word_features = list(all_words.keys())[:3000]
def find_features(document):
    words = set(document)
    features = {}
    for w in word_features:
        features[w] = (w in words)
    return features
featuresets = [(find_features(rev), category) for
                (rev, category) in documents]
print(featuresets)

aise, 'crappy': False, 'law': False, 'crowd': False, 'pleasi
ng': False, 'regular': False, 'reader': False, 'mine': Fals
e, 'heard': False, 'hart': False, 'abc': False, 'sabrina': F
alse, 'stud': False, 'basketball': False, 'adrien': False,
'grenier': False, 'grungy': False, 'broken': False, 'activi
st': False, 'wants': False, 'dance': False, 'date': False,
'popularity': False, 'scheme': False, 'resurgence': False,
'terms': False, 'hardly': False, 'wait': False, 'lesser': F
alse, 'extent': False, 'kissed': False, 'realize': False, 's
cripts': False, 'accordingly': False, 'intention': False, 's
mart': True, 'teens': False, 'whining': False, 'shut': Fals
e, 'honesty': False, 'normally': False, 'followed': False,
'obligatory': False, 'britney': False, 'spears': False, 'ti
tular': False, 'song': False, 'jingle': False, 'burger': Fal
se, 'joint': False, 'rendition': False, 'preceded': False,
'appearances': False, 'collins': False, 'trek': False, 'mot
ion': False, 'faye': False, 'tv': False, 'miniseries': Fals
e, 'mighty': False, 'joe': False, 'blunders': False, 'gorill
a': False, 'entrance': False, 'trees': False, 'leaps': Fals
e, 'gargantuan': False, 'imposing': False, 'volkswagen': Fal
se, 'feet': False, 'larger': False, 'pro': False, 'player':
False, 'bellows': False, 'poachers': False, 'millions': Fal
se, 'dollars': False, 'ape': False, 'fields': False, 'cars':
False, 'stares': False, 'pensively': False, 'breaks': False,
'clumsiness': False, 'hold': False, 'register': False, 'lim
p': False, 'entertainment': False, 'stamp': False, 'qualit
y': False, 'pretending': False, 'remake': False, '1949': Fal
se, 'strictly': False, 'formality': False, 'jill': False),
'neg']]
```

FreqDist, déjà exécuté, a ordonné notre ‘all_words’ liste a partir des mots les plus fréquents jusqu’aux moins fréquents.

Dans la liste `word_features` on stock les 3000 première mots les plus utilisés.

Ensuite nous allons vérifier l’existence des mots de `word_features` dans les listes de `movie_reviews` fournie par NLTK et stocker les résultats dans un dictionnaire `features`

CLASSIFICATION NAÏVE BAYESIENNE

La classification naïve bayésienne est un type de classification bayésienne probabiliste simple basée sur le théorème de Bayes avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un classifieur bayésien naïf, ou classifieur naïf de Bayes, appartenant à la famille des classifieurs linéaires.

‘Naive Bayes classifieur’ est l’algorithme qu’on va travailler avec, il nécessite des données pour l’entraînement et d’autres pour le test.

```
ML
len(featuresets)

2000

ML
# les données d'entraînement
training_set = featuresets[:1000]

# les données de test
testing_set = featuresets[1000:]
```

La taille des ‘features’ est 2000, il faut toujours prendre un nombre intéressant des données d’entraînement par rapport au nombre des données de test.

Dans notre cas les premières 1000 pour l’entraînement et les dernières 100 pour le test.

Avant de faire le test on a commencé par entraîner notre classifieur avec les données d’entraînements qu’on a préparées

Sa précision est 87%

```
ML
classifier = nltk.NaiveBayesClassifier.train(
    training_set)
print("Pourcentage de précision du classifieur : ",
      (nltk.classify.accuracy(classifier, testing_set))*100)

Pourcentage de précision du classifieur : 87.05263157894737
```

```
ML
classifier.show_most_informative_features(15)

Most Informative Features
turkey = True          neg : pos = 10.7 : 1.0
chick = True           neg : pos = 8.8 : 1.0
sucks = True           neg : pos = 8.8 : 1.0
awful = True           neg : pos = 7.7 : 1.0
whatsoever = True      neg : pos = 7.2 : 1.0
atrocious = True       neg : pos = 6.8 : 1.0
inane = True           neg : pos = 6.8 : 1.0
schumacher = True      neg : pos = 6.8 : 1.0
bore = True            neg : pos = 6.5 : 1.0
presumably = True      neg : pos = 6.4 : 1.0
uninspired = True      neg : pos = 6.4 : 1.0
lame = True            neg : pos = 6.4 : 1.0
border = True          neg : pos = 6.2 : 1.0
justin = True          neg : pos = 6.2 : 1.0
employ = True          pos : neg = 5.8 : 1.0
```

La liste des meilleurs ‘Features’ qui donne de bonnes informations .

Exp : le mot ‘awful’ apparaît dans les opinions négatives 7 fois plus que dans les opinions positives.

RESULTATS

Nous avons tester notre classifieur sur un commentaire négative:

```
ML
def sentiment(text):
    feats = find_features(text)
    return classifier.classify(feats)
print(sentiment("Bad film"))

neg
```

CONCLUSION

L'analyse de sentiment est de différencier et de classer les points de vue ou des sentiments ou des évaluations dans le contenu composé. Les sentiments des gens peuvent être exprimés en positif, façons négatives ou neutres.

La tâche de l'analyse des sentiments, en particulier dans le domaine du micro-blogging, est encore en développement et loin d'être complète, en fait il s'agit d'un important domaine de recherche actuel. Et ce n'est pas seulement le microblogging, Il peut être utilisé avec une variété des domaines d'applications comme le domaine politique, le marketing ou prédiction de stock.

En effet, jusqu'à présent, aucun outil n'est encore arrivé au point de faire une analyse de sentiments parfaite, même pas les êtres humains à cause de la subjectivité et transparence de certains discours. Par raison de difficulté de l'analyse des opinions (humour, ironie, sens caché, etc..). Plusieurs recherches en ingénierie linguistique montrent qu'il est possible de le faire avec la méthode des dictionnaires de sentiments (comme wordnet), la méthode de filtrage et autres étapes de prétraitement à effectuées pour optimiser l'analyse des opinions. Nous avons spécifiquement implémenté les étapes qui ont le plus impact sur le traitement. Nous nous appuyons également sur ces technologies pour construire notre propre méthode.

La mise en œuvre et la conception de notre projet visent à déterminer la polarité des commentaires écrite en langue anglaise.

Le concept fondamental de notre programme est de traiter un commentaire en utilisant les principales étapes de prétraitement pour une meilleure analyse. Pour que le traitement fonctionne correctement, il doit être basé sur un bon dictionnaire.

Bibliographie

- [1] I. Abdeljaoued-Tej, «Besoins et avantages de la fouille de données textuelles en sciences agronomiques,» 2020. [En ligne]. Available: <https://hal.archives-ouvertes.fr/hal-02126728/document>.
- [2] Y. SELMI, «Le NLP au cœur de la data science !,» [En ligne]. Available: <https://blog.soat.fr/2020/01/nlp-au-coeur-de-la-data-science/>.