Projet N°16
Avril 2022

RAPPORT

NLP pour l'analyse de critiques de films

IA pour analyser le sentiment à travers des critiques en français de spectateurs sur des films.

Auteurs : **Maïna LE DEM**

Indice: A
Date d'émission: 02/03/2022

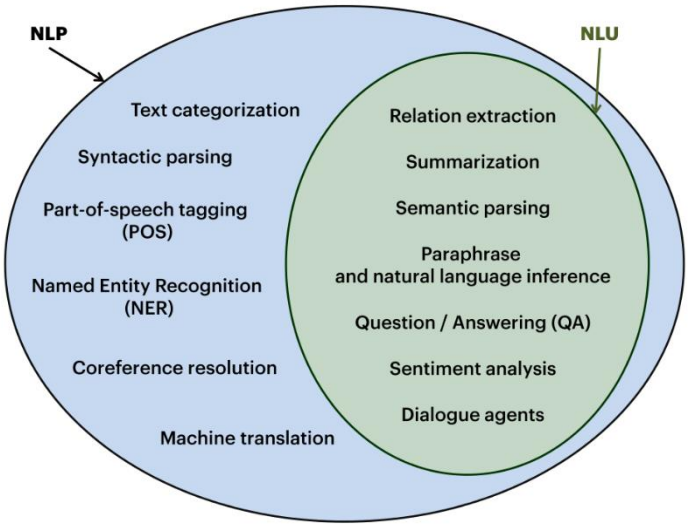
SOMMAIRE

1) Analyse de texte et traitement du langage naturel 3
1.1 - Natural Language Processing (NLP) :.....3
1.2 - Introduction au Text Mining.....3
1.3 - 2. La matrice documents termes.....4
1.4 - La catégorisation de textes (document classification).....6
1.5 - TF-IDF6
1.6 - Word embedding (Word2Vec).....7
2) Explication du jupyter notebook : NLP_critiques_films_inception_sonic2.ipynb 9
2.1 - Web Scraping des données d'avis de spectateurs.....9
2.2 - Préparation des données.....9
2.3 - Préparation des libellés.....9
2.4 - Finalisation de nos jeux de données.....10
3) Entraînement du modèle 10
4) Analyse des résultats 11
Conclusion 11

1) Analyse de texte et traitement du langage naturel

1.1 - *Natural Language Processing (NLP) :*

Le Natural Language Processing (NLP) ou Traitement Automatisé de la Langue (TAL) en Français, vise à extraire de l'information de textes à partir d'une analyse statistique du contenu. Cette définition permet d'inclure de nombreux champs d'applications au sein du NLP : traduction, analyse de sentiment, recommandation, surveillance, etc. ; ainsi que de méthodes.



Cette approche implique de transformer un texte, qui est une information compréhensible par un humain, en un nombre, information appropriée pour un ordinateur et une approche statistique ou algorithmique.

Dans le NLP, il y a 3 niveaux :

- Corpus : l'ensemble des documents
- documents : les différents textes
- tokens

1.2 - *Introduction au Text Mining*

1.2.1. Définition de la fouille de textes. Data Mining vs Text Mining.

Il faut différencier les Data Mining et le Text Mining qui consiste en :

- Le **Data Mining** est le processus qui consiste à trouver des modèles et à extraire des données utiles de grands ensembles de données.
  Il est utilisé pour convertir des données brutes en données utiles.
  Le Data Mining peut être extrêmement utile pour améliorer les stratégies de marketing d'une entreprise, car avec l'aide de données structurées, il est possible d'étudier les données de différentes bases de données et obtenir des idées plus innovantes pour augmenter la productivité d'une organisation.
  Le Text mining n'est qu'une partie du Data Mining.

- Le **Text Mining** est essentiellement une technologie d'intelligence artificielle qui consiste à traiter les données de divers documents textuels.
  De nombreux algorithmes d'apprentissage profond (deep learning) sont utilisés pour une évaluation efficace du texte.
  Dans le text mining, les données sont stockées dans un format non structuré. Il utilise principalement les principes linguistiques pour l'évaluation du texte des documents.

1.2.2. Web Scraping

Le Web Scrapping consiste à récupérer des données du web via le contenu des pages html, en vue d'en réutiliser le contenu.

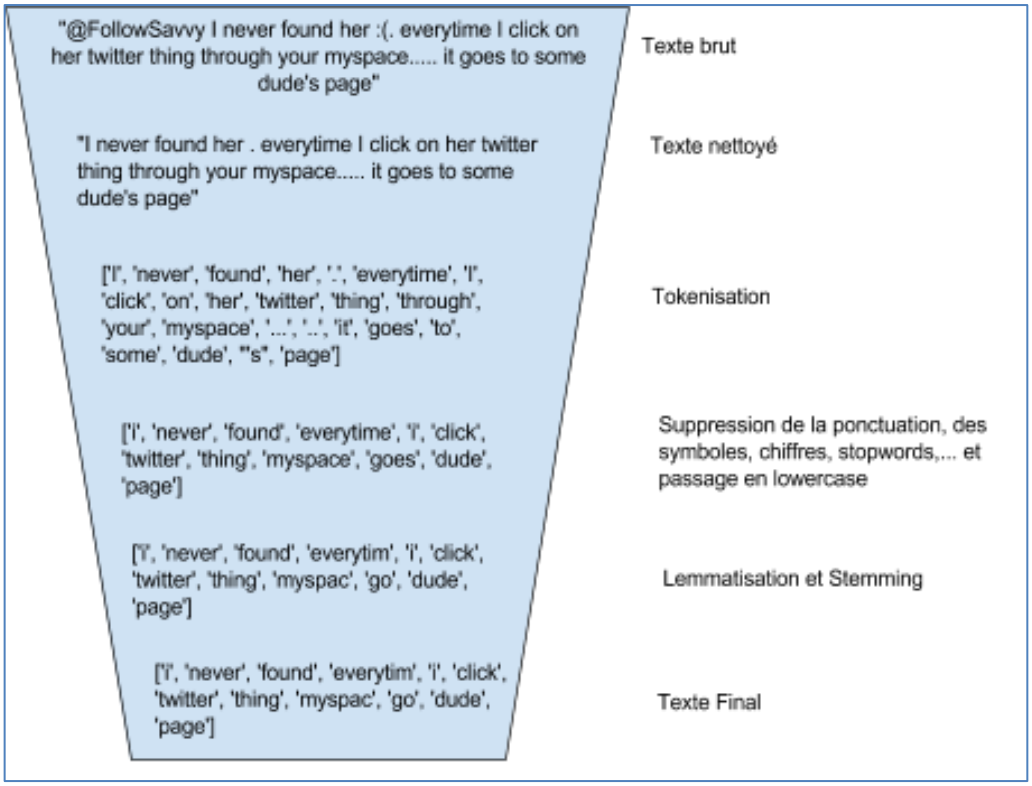
Particulièrement utile quand on a surfé sur internet et trouvé les informations cherchées.

1.2.3. Nuage de mot : WordCloud

Le WordCloud est une représentation graphique en fonction de la fréquence des mots, plus ils sont présents plus le mot sera grand. Par exemple ci-dessous, le mot film est le plus grand, donc le plus fréquent.

1.3 - *2. La matrice documents termes*

C'est l'état du pré-processing des tokens, par la désaccentuation, passer en minuscules, suppressions des stopwords, simplification avec stemmatisation ou Lemmastisation, pour créer le sac de mot.



Les étapes du pré-processing

1.3.1. 1. Tokenisation

La **tokenisation** cherche à transformer un texte en une série de tokens individuels. Il sépare une chaîne de texte, trouve les phrases dans les paragraphes ou les mots dans les phases.

Dans l'idée, chaque token représente un mot, et identifier des mots semble être une tâche relativement simple, avec « split ».

Mais en français, comme l'exemple : "J'ai froid". Il faut que le modèle de tokenisation sépare le "J'" comme étant un premier mot, grâce aux expressions régulières (« **Regex** »). Même au-delà du NLP, les Regex sont très utile pour reconnaître un mail ou un numéro de téléphone, et notamment vérifié si l'utilisateur le renseigne correctement dans un formulaire.

1.3.2. 2. Enlever les mots les plus fréquents

Certains mots se retrouvent très fréquemment dans la langue française. En anglais, on les appelle les « **stop words** ».

Ces mots, bien souvent, n'apportent pas d'information dans les tâches suivantes.

Lorsque l'on effectue par exemple une classification par la méthode CountVectorizer ou Tf-IdF, on souhaite limiter la quantité de mots dans les données d'entraînement.

1.3.3. 4. Stemming ou lemmatisation

Le **stemming** consiste à réduire un mot dans sa forme "racine".

Le but du stemming est de regrouper de nombreuses variantes d'un mot comme un seul et même mot. Par exemple, une fois que l'on applique un stemming sur "Chiens" ou "Chien", le mot résultant est le même.

Le **lemmatisation** consiste à ramener un terme, quels que soient ses accords, déclinaisons, etc. à sa forme la plus simple (pour le français infinitif/masculin-singulier). A ce jour, il n'y a pas de lemmatizer efficace pour la langue française

Cela permet notamment de réduire la taille du vocabulaire dans les approches de type sac de mots ou Tf-IdF.

Nous utiliserons donc un stemmer, dont un des plus connus est le Snowball Stemmer, diponible en français.

1.3.4. sacs de mots (bag of words)

Représentation des corpus (collection de textes) en sac de mots (**bag of words**).

Une représentation classique du bag-of-words est celle dans laquelle on représente chaque document par un vecteur de la taille du vocabulaire. Et on utilise la matrice composée de l'ensemble de ces N documents qui forment le corpus comme entrée de nos algorithmes.

En fin de compte le principe du sac de mots est plutôt simple. On peut dire qu'il ressemble même à celui de l'encodage one-hot.

Son principe se résume en 3 phases :

1. La décomposition des mots. On appelle cela aussi la tokenisation.
2. La constitution d'un dictionnaire global qui sera en fait le vocabulaire.
3. L'encodage des chaînes de caractère par rapport au vocabulaire constitué précédemment.

La vectorisation peut se faire avec CountVectorizer de la bibliothèque de scikit-learn

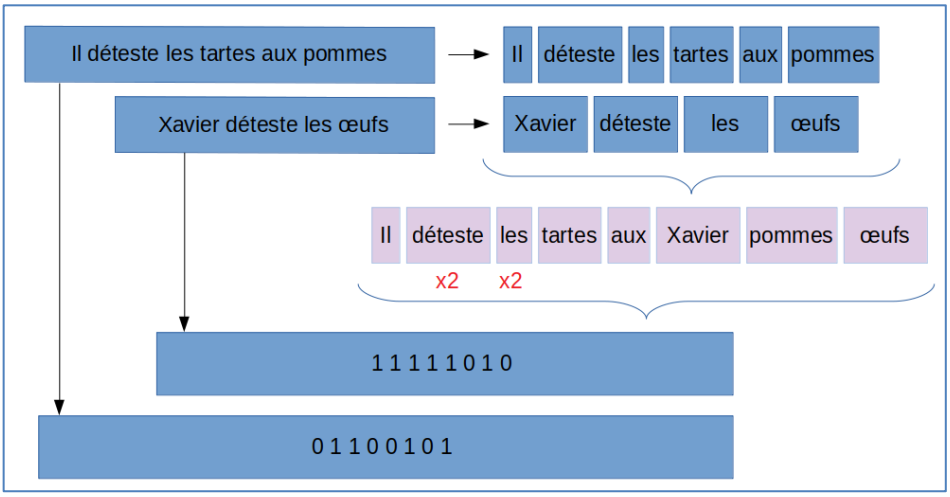


Schéma de principe

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

```
          Rome  Paris                          word V
Rome   = [1,  0,  0,  0,  0,  0,  …,  0]
Paris  = [0,  1,  0,  0,  0,  0,  …,  0]
Italy  = [0,  0,  1,  0,  0,  0,  …,  0]
France = [0,  0,  0,  1,  0,  0,  …,  0]
```

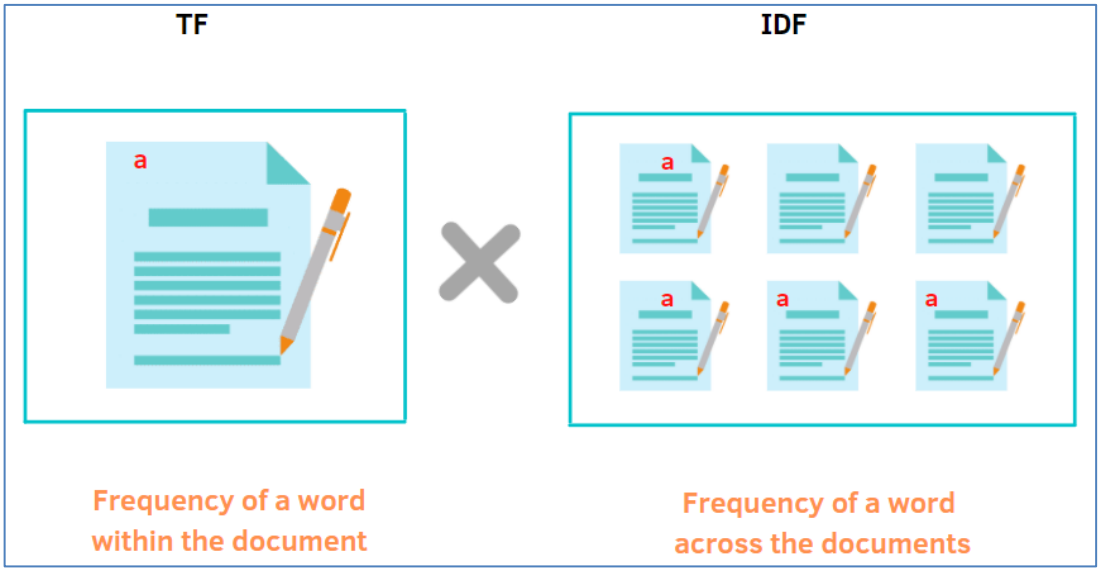
Exemple de résultat de CountVectorizer

1.4 - *La catégorisation de textes (document classification)*

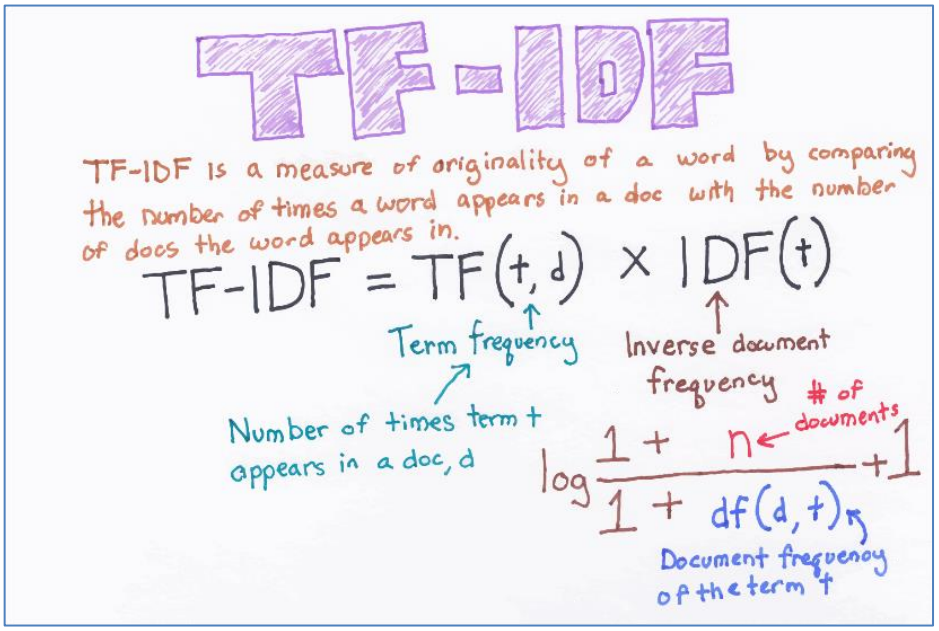
- Analyse prédictive à partir d'une collection de documents étiquetées. Visualisation et mesures spécifiques d'évaluation des performances (Matrice de confusion, accuracy score).

1.5 - *TF-IDF*

A la différence du sac de mot et de CountVectorizer, l'approche **TF-IDF** permet de tenir compte des fréquences relatives des mots.



Ainsi, pour un mot donné, on va multiplier la fréquence d'apparition du mot dans le document (calculé comme dans la méthode précédente) par un terme qui pénalise une fréquence élevée du mot dans le corpus.



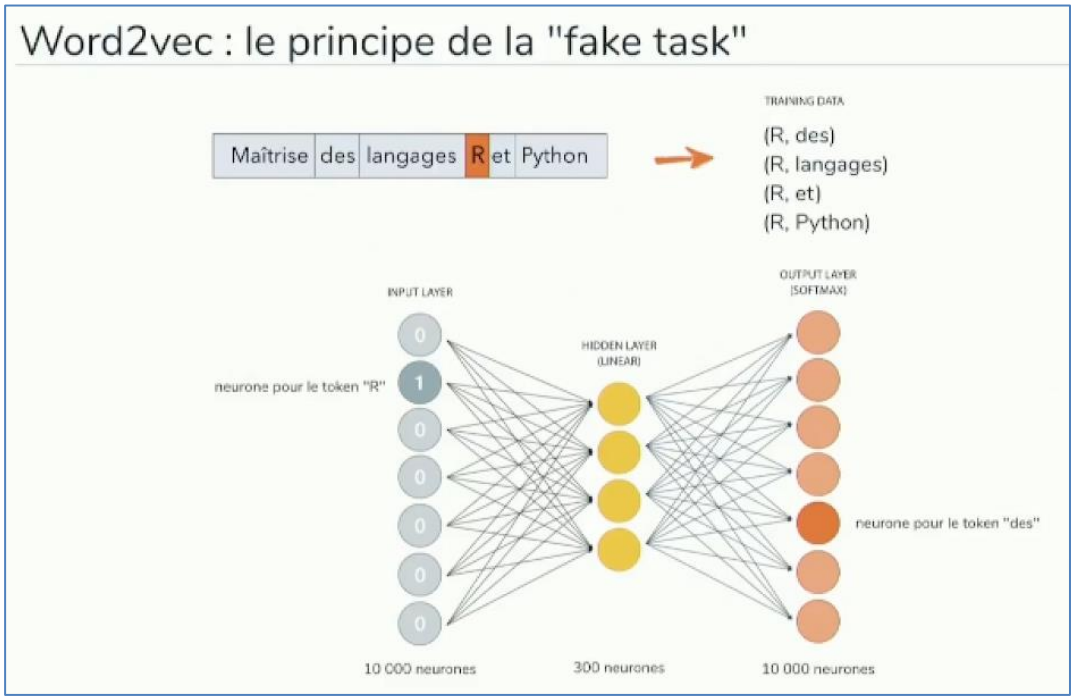
L'image de Chris Albon illustrant cette mesure

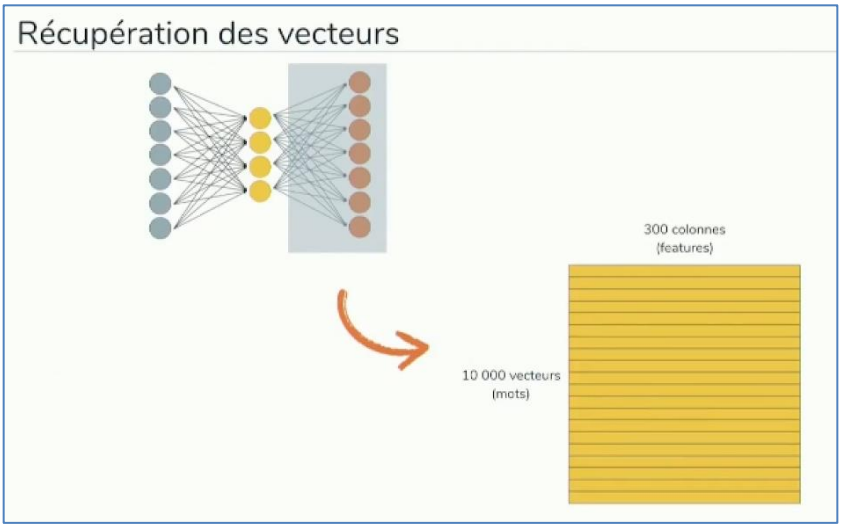
La vectorisation TF-IDF permet donc de limiter l'influence des stop-words et donc de donner plus de poids aux mots les plus saillants d'un document. On observe clairement que la performance de classification est bien supérieure, ce qui montre la pertinence de cette technique.

1.6 - *Word embedding (Word2Vec)*

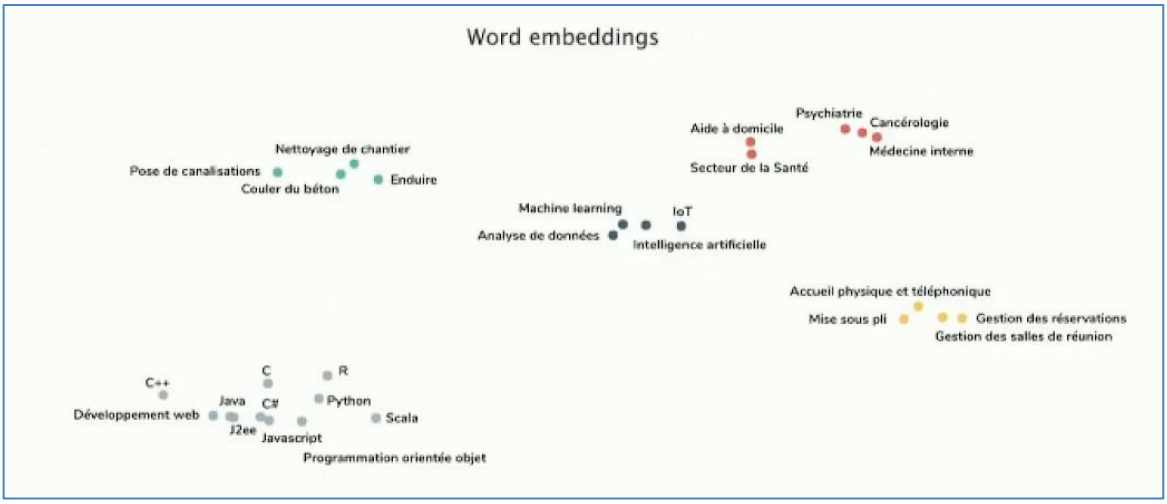
Représentation des termes à l'aide d'un vecteur numérique, contextualisée par le voisinage.

Le **word embedding** désigne un ensemble de méthode d'apprentissage visant à représenter les mots d'un texte par des vecteurs de nombres réels.





Le word embedding est capable en réduisant la dimension de capturer le contexte, la similarité sémantique et syntaxique (genre, synonymes, …) d'un mot.



Exemple de l'aperçu de la vectorisation sur le principe du plus proche voisin

Il existe deux variantes du Word2vec, les deux utilisent un réseau de neurones à 3 couches (1 couche d'entrée, 1 couche cachée, 1 couche de sortie) : Continous Bag Of Words (CBOW) et Skip-gram.

Exemple de code de Word2Vec avec Gensim :

Word embeddings avec Gensim

```
import gensim

stopwords = ["le", "la", "les", "l", "'", "de", "du", "des", "et", "en", "ou", "qui", "un", "une"]

sentences = [["maitrise", "des", "langages", "r", "et", "python"],
             ["connaissance", "avancee", "des", "langages", "python", "ou", "r"],
             ["recherche", "un", "developpeur", "python"],
             ["recherche", "developpeur", "java", "qui", "maitrise", "python"],
             ["comptabilite", "bilan", "bancaire", "declarations", "fiscales"],
             ["comptabilite", "analytique", ",", "bilan", "et", "compte", "de", "resultat"],
             ["vos", "principales", "missions", "comptabilite", "generale", "(", "compte", "de", "resultat",
"bilan", "fiscalite"]]

sentences = [[w for w in sent if w not in stopwords] for sent in sentences]

model = gensim.models.Word2Vec(sentences, window=5, size=300, min_count=1, iter=10)
```

```
>>> model.wv.most_similar("java")[:3]
[('python', 0.0810871571302414),
 ('r', 0.041719600558280945),
 ('recherche', 0.036696240305900574)]

>>> model.wv.most_similar("comptabilite")[:3]
[('avancee', 0.15022701025009155),
 ('bancaire', 0.1351984143257141),
 ('declarations', 0.08660304546356201)]
```

2) Explication du jupyter notebook : NLP_critiques_films_inception_sonic2.ipynb

2.1 - *Web Scraping des données d'avis de spectateurs*

Dans cet étape, on récupère sous forme de dataframe l'ensemble des notes et avis des films Inceptions et Sonic2.

2.2 - *Préparation des données*

2.2.1. Récupération des datas

Vérification des données et fusion des 2 dataframes, et suppression des données nulles (ici 1 note sans commentaire)

2.2.2. Pré-traitement du texte

Suppression des expressions régulière (RegEx) que je remplace par un espace.

Dans notre cas, il s'agit des nombres, des bruits (ponctuation, caractères spéciaux, …) : `("[0-9]")` et `r"(<br\s/><br\s/>)|(-)|(/)|[.'']|[,;:!?$€()|><^*%\"\\\[\]]")`

Puis, je nettoie les espaces en trop avec le Regex : `(r"\s\s+")`

2.2.3. NLP pour tokeniser et réduire le corpus de chaque commentaire

Définition des stopwords (french)

Suppression des stopwords dans les avis en créant une nouvelle colonne Et séparation des mots avec word_tokenize

"Racinisation" ou Stemming des mots.

Nota : pas de lemmatisation parce qu'il ne semble pas exister fonction de lemmatisation de corpus français dans NLTK

Puis suppression des accents et des emoticons

2.2.4. Sacs de mots

Le sac de mots est réalisé après l'étape de la préparation des libellés, et la séparation des donnée d'apprentissage et de test.

2.3 - *Préparation des libellés*

2.3.1. Target « Note » et données train et test

Chaque commentaire est associé une note de 1 à 5 et non une classe binaire.

Chaque note est une string à transformer en float, pour convertir en :

- 1 pour avis positif
- 0 : pour avis négatif

Préparation des données d'apprentissage et de test

2.3.2. Sacs de mots – Suite –

Après séparation des données, création du sac de mot sur le jeu d'apprentissage.

Ici je procède avec 2 méthodes d'approches pour comparer les résultats :

- Les mots les plus utilisés (most_common()) avec .FreqDist() de la bibliothèque nltk
- Les mots les plus utilisés (au moins 20 fois) pour faire un vecteur CountVectorizer

Les résultats sont proches, mais pas identiques.

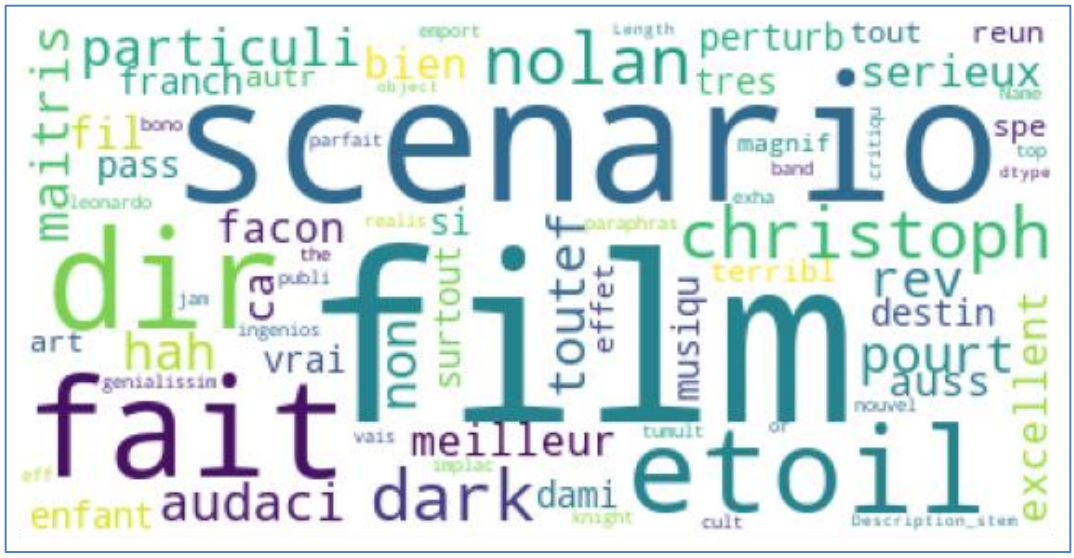
Le brief s'appuyant sur la vectorisation, je poursuis donc avec CountVectorizer

2.4 - *Finalisation de nos jeux de données*

Après l'entrainement de CountVectorizer sur le jeu d'apprentissage, je transforme les 2 jeux (train et test) en vecteurs.

J'affiches les mots les plus utilisés pour les avis positifs et négatifs, avec Word Cloud :

Word Cloud sur les avis négatifs



Word Cloud sur les avis positifs

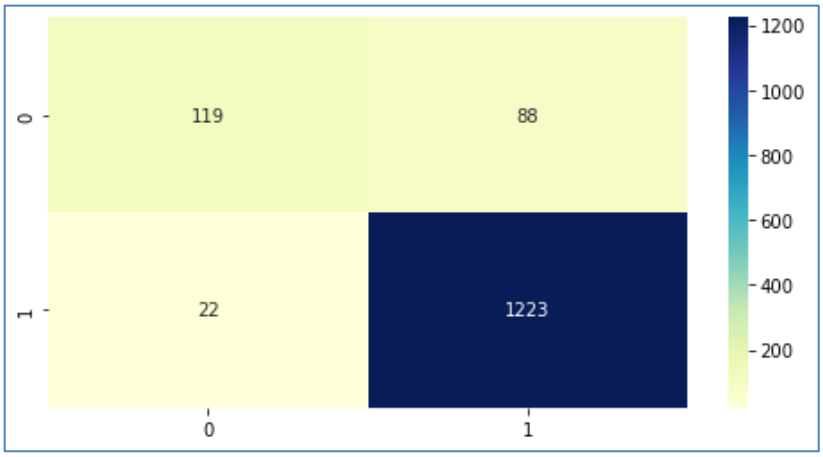
2.5 - *Entraînement du modèle*

Les données sont prêtes, utilisation d'un un algorithme de Régression Logistique comme ici il est question de classification binaire.

2.6 - *Analyse des résultats*

Model accuracy score: 92,42 %

Matrice de confusion :



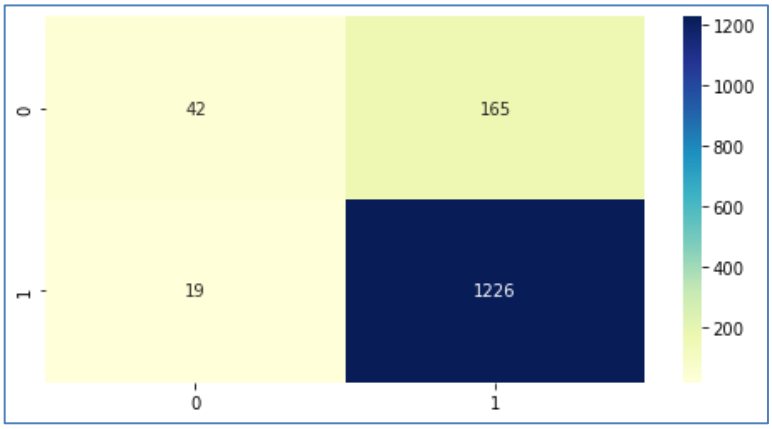
2.7 - *en sus : Essais avec vectorisation de TF-IDF*

Vectorisation des jeux de données avec TfidfVectorizer

Puis entrainement du modèle de régression linéraire

Model accuracy score: 87,33 %

Matrice de confusion :



Conclusion

Le NLP peut être utilisé pour une multitude de données.

Ici on obtient de bon résultat avec l'accuracy, cependant en regardant les matrices de confusion, les résultats ne semblent pas concluants au vu des erreurs, malgré les 2 méthodes de vectorisation.

D'une part les données sont disproportionnées entre les 6171 avis positifs et 1089.

De plus, les avis peuvent être subjectif, et nous pouvons retrouver les mêmes mots en grandes fréquence que l'avis soit positif ou négatif, tel que les mots fils et scénario.

L'approche de word embeddings serait dans ce cas plus appropriées, car le poids des poids se ferait en fonction des autres mots de la phase.