



Université de Paris Cité

MASTER 1 APPRENTISSAGE MACHINE
POUR LA SCIENCE DES DONNÉES

PORJET TER

ANALYSE

AVANCÉE DE

TWEETS

RÉALISÉ PAR :

Wessal HAMZA
Amira TLATI

ENDADRÉ PAR:

Séverine Affeldt
Lazhar Labiod

Année universitaire 2021-2022

Table des matières

1	Contexte général	3
1.1	Motivations	3
1.2	Objectifs	3
2	État de l’art et veille technologique	4
2.1	Twitter	4
2.1.1	Twitter API	4
2.2	Traitement du langage naturel	4
2.2.1	Pré-traitement du texte :	5
2.2.2	Vectorisation des données	5
2.2.3	Modélisation de sujets	6
2.3	Veille technologique	7
2.3.1	Python	7
2.3.2	Jupyter Notebook	8
2.3.3	Flask	9
3	Présentation des réalisations	10
3.1	Pré-traitement et traitement des données	10
3.1.1	Pré-traitement des données	10
3.1.2	Traitement des données	12
3.1.3	Modélisation de thématiques	14
3.2	Intégration à l’interface	16
4	Principaux résultats et étude de cas	18
4.1	Fonctionnement de l’interface	18
4.2	Étude de cas	21

Chapitre 1

Contexte général

1.1 Motivations

Le traitement du langage naturel (NLP) est un domaine de l'intelligence artificielle qui peut être défini comme étant le processus de lecture, d'analyse et de compréhension d'une grande quantité de données textuelles. L'objectif du NLP est de comprendre le langage humain à travers la lecture de textes et d'en tirer de précieux aperçus de celui-ci.

Les réseaux sociaux génèrent une quantité très importante de données comme par exemple le réseau social Twitter. En effet, Twitter est la deuxième plus grande plate-forme de réseau social après Facebook, qui génère 347 222 tweets par minute et 21 millions de tweets par heure. Cela crée une opportunité pour l'exploration de données, l'analyse de sentiments des utilisateurs et l'apprentissage automatique de thématiques. L'utilisation de NLP est un atout important pour l'avenir de la compréhension de texte non structuré.

Ainsi, nous étudierons l'utilisation du langage naturel sur les données provenant de Twitter pour extraire les avis qui se dégagent de ces données dans le cadre d'un sujet défini au préalable.

1.2 Objectifs

La modélisation thématique est une technique d'apprentissage automatique non supervisée capable d'analyser un ensemble de documents, de détecter les modèles de mots et d'expressions qu'ils contiennent et de regrouper automatiquement les groupes de mots et les expressions similaires qui caractérisent le mieux un ensemble de documents.

Twitter contient une quantité importante de documents "texte court" qui peuvent fournir des informations potentiellement utiles s'ils sont exploités correctement.

Ainsi, notre objectif est de créer une application Web qui sera utilisée pour analyser des tweets. Elle effectuera en premier temps la normalisation de texte puis une analyse des sentiments sur les tweets provenant de Twitter et déterminera ainsi leur polarité (négatif, positif ou neutre), enfin mettre en place des méthodes basées sur des modèles probabilistes ou sur la factorisation matricielle dans l'optique de modéliser des thématiques sur des textes courts "tweets". Dans ce projet nous examinons 2 approches différentes de la modélisation de thématiques : LDA (Latent Dirichlet Allocation), NMF (Non-Negative Matrix Factorization)

L'objectif principal de notre application Web est de pouvoir permettre de déterminer l'opinion d'une population sur un sujet.

Chapitre 2

État de l'art et veille technologique

2.1 Twitter

Twitter est une plateforme de médias sociaux en ligne où les utilisateurs peuvent communiquer par le biais de messages courts appelés tweets. Un tweet est un message publié publiquement sur Twitter qui ne peut contenir plus de 140 caractères et peut être composé de texte, d'images, de vidéos ou de liens. Plus de 500 millions tweets sont publiés chaque jour et Twitter compte plus de 340 utilisateurs actifs mensuels. Les utilisateurs de twitter peuvent ainsi communiquer et partager leur avis sur divers sujets.

2.1.1 Twitter API

Tweepy est un package Python open source qui nous offre un moyen très pratique d'accéder à l'API Twitter avec Python. Tweepy inclut un ensemble de classes et de méthodes qui représentent les modèles et les points de terminaison de l'API de Twitter, et il gère de manière transparente divers détails d'implémentation, tels que :

- Encodage et décodage des données
- Requêtes HTTP
- Pagination des résultats
- Authentification OAuth

Tout accès à l'API Twitter nécessite un compte de développeur, qui peut être créé rapidement en s'inscrivant. L'accès essentiel sera disponible immédiatement, et l'accès élevé peut être demandé.

- Essentiel : Accès gratuit et instantané à l'API Twitter. Comprends 500 000 Tweets/mois et un environnement d'application unique. 1 application, 1 projet.
- Élevé : Accès gratuit jusqu'à 2M Tweets/mois, et 3 environnements d'applications. 3 applications, 1 projet. Nécessite une demande de compte de développeur approuvée.

Dans ce projet, l'API de recherche Élevé a été utilisée pour obtenir des données de Twitter.

2.2 Traitement du langage naturel

Natural Language Processing (NLP), ou traitement du langage naturel, est une branche de l'intelligence artificielle qui s'attache à comprendre le langage humain tel qu'il est écrit et/ou parlé. Pour ce faire, des programmes informatiques spécifiques sont développés. Le traitement du texte est généralement composé de deux à trois grandes étapes :

2.2.1 Pré-traitement du texte :

Une étape qui cherche à standardiser du texte afin de rendre son usage plus facile. Cela nécessite beaucoup de nettoyage et de traitement avant que les données puissent être utilisées pour l'analyse.

Voici quelques-unes des méthodes de traitement des données dans le traitement du langage naturel :

- Normalisation : Toutes les ponctuations du texte sont supprimées et le texte est converti en minuscule.
- Tokenization : La tokenisation décompose le texte brut en mots, phrases appelées tokens. Ces tokens aident à comprendre le contexte ou à développer le modèle pour le NLP.
- Stop words removal : Les mots les plus courants et qui n'apportent pas beaucoup d'informations au texte. En supprimant ces mots, nous supprimons les informations de bas niveau de notre texte afin de mettre davantage l'accent sur les informations importantes.
- Lemmatization : La transformation utilise un dictionnaire pour mapper différentes variantes d'un mot vers son format racine. Ainsi, avec cette approche, nous sommes en mesure de réduire les flexions non triviales

2.2.2 Vectorisation des données

La vectorisation des mots est le processus d'encodage des mots individuels en vecteurs afin que le texte puisse être facilement analysé ou consommé par l'algorithme d'apprentissage automatique. Il est difficile d'analyser le corpus brut, d'où la nécessité de le convertir en nombres entiers (le meilleur format est celui des vecteurs) où nous pouvons appliquer des opérations mathématiques et obtenir des informations à partir des données. Les différentes techniques de vectorisation des mots incluent :

Bag of Words :

Un modèle de Bag of Words est un moyen d'extraire des caractéristiques du texte pour les utiliser dans la modélisation, par exemple avec des algorithmes d'apprentissage automatique.

Un Bag of Words est une représentation du texte qui décrit l'occurrence des mots dans un document. Il implique deux éléments :

- Un vocabulaire de mots connus.
- Une mesure de la présence de mots connus.

	the	red	dog	cat	eats	food
1. the red dog →	1	1	1	0	0	0
2. cat eats dog →	0	0	1	1	1	0
3. dog eats food →	0	0	1	0	1	1
4. red cat eats →	0	1	0	1	1	0

FIGURE 2.1 – Matrice documents-termes

On l'appelle Bag of Words ("Sac de Mots" en français), car toute information sur l'ordre ou la structure des mots dans le document est écartée. Le modèle se préoccupe uniquement de la présence de mots connus dans le document, et non de leur emplacement dans le document.

TF-IDF :

TF-IDF (terme frequency-inverse document frequency) est une mesure statistique qui évalue la pertinence d'un mot par rapport à un document dans une collection de documents. TF-IDF pour un mot dans un document est calculé en multipliant deux métriques différentes :



FIGURE 2.2 – Les dates Clés

- La fréquence d'un mot dans un document : Il existe plusieurs façons de calculer cette fréquence, la plus simple étant le comptage brut des occurrences d'un mot dans un document. Ensuite, il existe des moyens d'ajuster la fréquence, par la longueur d'un document, ou par la fréquence brute du mot le plus fréquent dans un document.
- La fréquence inverse du mot dans un ensemble de documents. Cela signifie la fréquence ou la rareté d'un mot dans l'ensemble des documents. Plus elle est proche de 0, plus le mot est fréquent. Cette métrique peut être calculée en prenant le nombre total de documents, en le divisant par le nombre de documents qui contiennent un mot, et en calculant le logarithme.

Ainsi, si le mot est très courant et apparaît dans de nombreux documents, ce nombre s'approchera de 0. Sinon, il s'approchera de 1.

2.2.3 Modélisation de sujets

La modélisation thématique est une approche d'apprentissage automatique non supervisée qui permet d'analyser une série de documents, d'y trouver des modèles de mots et d'expressions, et de regrouper automatiquement les groupes de mots et les expressions connexes qui représentent le mieux l'ensemble. Dans notre projet nous avons donc utilisé deux algorithmes de modélisation de sujets qui sont le modèle Latent Dirichlet Allocation (LDA) et le modèle Non Negative Matrix Factorisation (NMF).

LDA :

Le modèle LDA est un modèle probabiliste génératif nommé "topic models" permettant de trouver des structures de thématiques cachées dans un corpus. Ce modèle utilise deux valeurs de probabilité : $P(\text{mot} | \text{sujet})$ (probabilité que le sujet dans le corpus soit assigné au mot) et $P(\text{sujet} | \text{documents})$ (probabilité que le document appartienne au sujet). Ces valeurs sont calculées sur la base d'une affectation aléatoire initiale, après elles sont répétées pour chaque mot de chaque document, afin de décider de leur affectation à un sujet. Dans une procédure itérative, ces probabilités sont calculées plusieurs fois, jusqu'à la convergence de l'algorithme.

NMF :

La factorisation de matrices non négatives est un groupe d'algorithmes en analyse multivariée et en algèbre linéaire où une matrice V est factorisée en deux matrices W et H , avec la propriété que les trois matrices n'ont pas d'éléments négatifs et sans ligne ou colonne ne comportant que des 0. Cette non-négativité rend les matrices résultantes plus faciles à inspecter.

Étant donné la matrice originale A , nous pouvons obtenir deux matrices W et H , telles que $A = WH$. La NMF possède une propriété de regroupement, de sorte que W et H représentent les informations suivantes sur A :

- A (matrice mots-documents) - entrée qui contient quels mots apparaissent dans quels documents.
- W (vecteurs de base) - les sujets (clusters) découverts à partir des documents.
- H (matrice de coefficients) - les poids d'appartenance des sujets dans chaque document.

2.3 Veille technologique

2.3.1 Python

Python est un langage de programmation interprété, multi-paradigme et multiplate-forme. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions. Il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.



FIGURE 2.3 – Python

Pourquoi utilisé python :

Les projets d'IA diffèrent des projets logiciels traditionnels. Les différences résident dans la pile de technologies, les compétences requises pour un projet basé sur l'IA et la nécessité d'une recherche approfondie. Du développement au déploiement et à la maintenance, python aide les développeurs à être productifs et confiants quant aux logiciels qu'ils construisent. Les avantages qui font de python le meilleur choix pour l'apprentissage automatique et les projets basés sur l'IA incluent :

- La simplicité et la cohérence.
- L'accès à d'excellentes bibliothèques.
- Flexibilité.
- Indépendance de la plateforme.
- Une large communauté.

Bibliothèques utilisées :

- *NumPy* :

Un package de traitement de tableau à usage général bien connu. Une vaste collection de fonctions mathématiques de haute complexité rend NumPy puissant pour traiter de grands tableaux et matrices multidimensionnelles.

- *Scikit-learn* :

Un large éventail d'algorithmes d'apprentissage supervisés et non supervisés qui fonctionnent sur une interface cohérente en Python. La bibliothèque peut également être utilisée pour l'exploration de données et l'analyse de données. Les principales fonctions d'apprentissage automatique que la bibliothèque Scikit-learn peut gérer

sont la classification, la régression, le clustering, la réduction de dimensionnalité, la sélection de modèle et le prétraitement.

- *Pandas* :

La bibliothèque Python la plus populaire utilisée pour l'analyse de données avec prise en charge de structures de données rapides, flexibles et expressives conçues pour fonctionner à la fois sur des données «relationnelles» ou «étiquetées». Pandas est aujourd'hui une bibliothèque incontournable pour résoudre des analyses de données pratiques et réelles en Python. Pandas est très stable, offrant des performances hautement optimisées.

- *Matplotlib* :

Une bibliothèque de visualisation de données qui est utilisée pour le traçage 2D afin de produire des graphiques et des figures de qualité affichée dans une variété de formats. La bibliothèque permet de générer des histogrammes, des graphiques, des graphiques d'erreur, des nuages de points, des graphiques à barres avec seulement quelques lignes de code.

- *spaCy* :

SpaCy est une bibliothèque Python à code source ouvert qui analyse et "comprend" de grands volumes de texte. Des modèles distincts sont disponibles pour des langues spécifiques (anglais, français, allemand, etc.).

- *NLTK* :

NLTK - Natural Language Toolkit est un logiciel libre très populaire. Initialement publié en 2001, il est beaucoup plus ancien que Spacy (publié en 2015). Il fournit également de nombreuses fonctionnalités, mais comprend des implémentations moins efficaces.

- *Gensim* :

Gensim est une bibliothèque Python pour la modélisation de sujets, l'indexation de documents et la recherche de similarités avec de grands corpus. Le public cible est la communauté du traitement du langage naturel (NLP) et de la recherche d'information (IR).

- *TextBlob* : TextBlob est une bibliothèque Python pour le traitement des données textuelles. Elle fournit une API simple permettant de se plonger dans les tâches courantes de traitement du langage naturel (NLP), telles que le marquage des parties du discours, l'extraction des phrases nominales, l'analyse des sentiments, la classification, la traduction, etc.

- *wordcloud* :

wordcloud est une bibliothèque Python pour la représentation visuelle de données textuelles. Les mots sont généralement des mots isolés, et l'importance de chacun est indiquée par la taille de la police ou la couleur.

2.3.2 Jupyter Notebook

Jupyter Notebook est une application client-serveur créée par l'organisation à but non lucratif Project Jupyter. Elle a été publiée en 2015. Elle permet la création et le partage de documents Web au format JSON constitués d'une liste ordonnée de cellules d'entrées et de sorties et organisés en fonction des versions successives du document. Les cellules peuvent contenir, entre autres, du code, du texte au format Markdown, des formules mathématiques ou des contenus médias. Le traitement se fait avec une application client fonctionnant par Internet, à laquelle on accède par les navigateurs habituels. Il est nécessaire pour cela que soit installé et activé dans le système le serveur Jupyter Notebook. Les documents Jupyter créés peuvent s'exporter aux formats HTML, PDF, Markdown ou Python par exemple, ou bien se partager par email, avec Dropbox, GitHub ou un lecteur Jupyter Notebook.

Les avantages de jupyter Notebook :

Jupyter Notebook met à disposition un environnement qui répond parfaitement aux exigences et au flux de travail dans le domaine de la science des données et de la simulation. Une seule et unique instance permet

à l'utilisateur d'écrire du code, de l'accompagner d'explications et de l'exécuter, mais aussi de visualiser des données, d'exécuter des calculs et de vérifier les résultats obtenus. Autres avantages de Jupyter Notebook :

- Open source.
- Usage gratuit.
- S'utilise à partir d'un navigateur.
- Live-coding.
- Variété d'options d'export et de partage des résultats.
- Gestion des versions.
- Possibilité de travail collaboratif.
- Permet d'utiliser plus de 50 langages de programmation.

2.3.3 Flask

Flask est un micro framework open-source de développement web en Python. Il est classé comme micro-framework car il est très léger. Flask a pour objectif de garder un noyau simple mais extensible. Il n'intègre pas de système d'authentification, pas de couche d'abstraction de base de données, ni d'outil de validation de formulaires. Cependant, de nombreuses extensions permettent d'ajouter facilement des fonctionnalités. Il est distribué sous licence BSD.

Flask se base sur deux modules `werkzeug` et `jinja2` pour proposer plusieurs des fonctionnalités suivantes :

- Serveur de développement et debugger
- Simplifie l'écriture de tests unitaires
- Moteur de template pour le rendu HTML
- Supporte les cookies sécurisés (session)
- Entièrement compatible avec WSGI 1.0
- Se base sur l'Unicode
- Documentation complète
- Déploiement aisé sur plusieurs hébergeurs
- Ajout de fonctionnalités via les extensions

Chapitre 3

Présentation des réalisations

3.1 Pré-traitement et traitement des données

3.1.1 Pré-traitement des données

Construction du dataset

L'API Twitter utilise OAuthHandler, un protocole d'autorisation ouvert largement utilisé, pour authentifier toutes les demandes. Avant d'effectuer tout appel à l'API Twitter, nous avons créé et configuré nos informations d'authentification. Nous avons donc :

- Importé le package tweepy.
- Défini les identifiants d'authentification.
- Créé un nouvel objet tweepy.API

Les objets appartenant à la classe tweepy.API offrent un vaste ensemble de méthodes qu'on peut utiliser pour accéder à presque toutes les fonctionnalités de Twitter.

```
import tweepy
import datetime

auth = tweepy.OAuthHandler(TWITTER_CONSUMER_KEY, TWITTER_CONSUMER_SECRET)
auth.set_access_token(TWITTER_ACCESS_TOKEN, TWITTER_ACCESS_TOKEN_SECRET)

api = tweepy.API(auth, wait_on_rate_limit=True)
```

FIGURE 3.1 – Authentification Tweepy

Avant de construire un jeu de données contenant nos Tweets, nous avons besoin de les récupérer. Pour cela nous devons nous connecter à l'API à l'aide de code d'accès. Nous utilisons la fonction Cursor de Tweepy. La fonction nécessite un mot-clé permettant la récupération de tous les Tweets contenant le mot d'intérêt. Nous avons également spécifié l'intervalle de temps dans lequel les Tweets ont été créés. Nous avons opté pour un intervalle de temps d'hier à aujourd'hui et un nombre de 1000 Tweets à récupérer car Tweepy impose une limite. Cet intervalle a été choisi car Tweepy ne nous permettait pas de récupérer des Tweets à d'autres intervalles de temps.

```
today = datetime.date.today()
yesterday = today - datetime.timedelta(days=1)
search_words = "macron since:" + str(yesterday) + " until:" + str(today)
new_search = search_words + " -filter:retweets"
tweets_list = tweepy.Cursor(api.search, q=new_search, tweet_mode='extended', lang='fr').items(1000)
```

FIGURE 3.2 – Extraction des Tweets

Une fois les tweets récupérées nous construisons un dataframe contenant quelques informations importantes relatif aux Tweets tel que le contenu du Tweet, le nombre à retweet et le nombre de favoris, la date de création du Tweet et le nom de l'utilisateur. Ce dataframe constituera une base importante pour la manipulation de nos Tweets.

```
output = []
for tweet in tweets_list:
    text = tweet._json["full_text"]
    favourite_count = tweet.favorite_count
    retweet_count = tweet.retweet_count
    created_at = tweet.created_at
    location = tweet.user.location
    name = tweet.user.screen_name
    line = {'text': text, 'favourite_count': favourite_count, 'retweet_count': retweet_count, 'created_at': created_at, 'location': location, 'name': name}
    output.append(line)
```

FIGURE 3.3 – Dataframe des tweets récupérés

Normalisation du texte

Pour le nettoyage de nos Tweets nous avons tout d'abord commencé par la suppression des stop-words. Les stops words ou mot vide sont des mots communs qui ne contiennent aucune informations utile à notre analyse. Nous avons récupéré une première liste de stop words en français de la bibliothèque *spacy* et une liste de stop words récupéré à partir de la fonction *stopwords* de *nltk*. Nous avons ajouté des stop words non présents dans les listes évoquées précédemment à l'aide de la visualisation une représentation des modèles de topic modelling (voir section suivante). En effet, divers mots familiers ou abréviations sont utilisés sur twitter dû notamment à la limitation de la longueur des Tweets. Nous avons par exemple ajouté dans la liste le terme "mdr" signifiant "mort de rire" qui est spécifique au langage français familier. Il peut être utilisé dans des Tweets français (principalement par des jeunes) et n'est pas présent dans la liste de stop words évoqués précédemment.

```
from nltk.tokenize import TweetTokenizer
from spacy.lang.fr.stop_words import STOP_WORDS as fr_stop
from nltk.corpus import stopwords

list_of_stop_word_88 = list(fr_stop)
l = ["pr", "p-e", "j'ai", "jai", "ya", "y'a", "c'est", "cest", "qu'il", "quil", "faut", "sil", "s'il", "n'est", "nest", "jsuis", "j'suis",
    "d'un", "dun", "ans", "macron", "faire", "c'était", "cétait", "c'etait", "cetait", "naura", "n'aura", "mettre", "deja", "déjà", "déja",
    "dejà", "ds", "ça", "ca", "wsh", "wesh", "tt", "jsp", "quon", "qu'on", "sil", "s'ils", "jen", "j'en"]

for i in l:
    list_of_stop_word_88.append(i)

punct = list(string.punctuation)
stopword_list = stopwords.words('french') + punct + ['rt', 'via', '...'] + list_of_stop_word_88 #stopword list
stopword_list = list(set(stopword_list))
```

FIGURE 3.4 – Création de la liste des stop-words

Après élimination des stops Words, nous avons converti le texte dans la même casse (en minuscules). Les Tweets présentes également plusieurs types de ponctuation tels que les hashtags qui se placent en début de mot ou de phrase sans espace et permet de lier tous les Tweets contenant ce même terme (exemple : présidentielle2022). Il y a également les mentions : ils se créent en mettant un @ devant le nom de l'utilisateur que l'on souhaite mentionné. Les liens sont également utilisés dans les tweets pour faire référence par exemple à un article. Les ponctuations et les numéros sont également courants dans les Tweets. Dans certains cas, on souhaite les supprimer afin d'obtenir uniquement le contenu propre d'un tweet. Nous supprimons donc ces hashtags et mentions à l'aide de Regex (bibliothèque *re*).

Un emoji permet d'exprimer un sentiment et se présente comme une petite image numérique. Ils ne sont pas des données textuelles que l'on peut analyser, donc nous avons également décider de les supprimer à l'aide de la bibliothèque *emoji*.

Après l'application de l'ensemble de processus de nettoyage nous avons noté la présence des lignes vides (ou liste vide) dans la colonne qui contient les tweets nettoyés. En effet, certains Tweets ne contiennent par exemple que des stops words ou des liens. Puisqu'ils ne contiennent aucune information nous avons décidé de

les éliminer.

```
import re
import emoji
from nltk.tokenize import TweetTokenizer

#Process the text of a tweet: Lowercase, Tokenize, Stopword removal, Digits removal, Return: list of strings
def process(text, tokenizer=TweetTokenizer(), stopwords=[]):
    text = text.lower() #transformer en minuscule
    text = text.replace('\n', ' ').replace('\r', '')
    text = ' '.join(text.split())
    text = re.sub(r"[A-Za-z\.\,]*[0-9]+[A-Za-z%\.]*", "", text)
    text = re.sub(r"@[A-Za-z0-9_]+", "", text) #Removing mentions
    text = re.sub(r"#[A-Za-z0-9_]+", "", text) #Removing hashtags
    text = re.sub(r"http(S+)", "", text) #Removing links
    text = re.sub(r"[\^a-z0-9 \_\.\u00C0-\u00D6\u00D8-\u00F6\u00F8-\u00FF]", " ", text) #supprimer les termes contenant une lettre avec accent, ch
    text = re.sub(r'&mdr[r]*', '', text) #Removing mdr[r]*
    text = re.sub(r'+', ' ', text)
    text = emoji.get_emoji_regexp().sub(u'', text)
    tokens = tokenizer.tokenize(text)
    return [tok for tok in tokens if tok not in stopwords and not tok.isdigit()]
```

FIGURE 3.5 – Fonction pour standardiser le texte

La *lemmatisation* consiste à donner à un mot sa forme canonique neutre que l'on trouve dans un dictionnaire. La bibliothèque NLTK ne fournissant pas de *lemmatisation* en français nous avons opté pour la bibliothèque spaCy.

Nous avons tout d'abord commencé par charger `fr_core_news_md` à l'aide `spacy.load()`. En effet, `fr_core_news_md` est un pipeline français entraîné sur du texte contenu sur divers site Web (par exemple des blogs) contenant de la syntaxe et du vocabulaire en français.

Lorsqu'on appelle la fonction `nlp` (voir l'image ci-dessous) sur un texte, spaCy commence par *tokeniser* le texte pour produire un objet Doc (que l'on stocke dans la variable `doc`). Puis nous avons appliqué la fonction `lemma_` sur chaque élément de la variable `doc` pour les transformer en leur forme canonique neutre .

```
nlp = spacy.load('fr_core_news_md')
def lemmatisation(row):
    phrase = row['tokens_test']
    doc = nlp(phrase)
    txt = [token.lemma_ for token in doc ]
    return txt
```

FIGURE 3.6 – Lemmatisation du texte

Pour vérifier si le pré-traitement a été effectué, nous allons réaliser un nuage de mots à l'aide du package wordcloud pour obtenir une représentation visuelle des mots les plus courants. C'est essentiel pour comprendre les données et s'assurer que nous sommes sur la bonne voie, et si un pré-traitement supplémentaire est nécessaire avant l'entraînement du modèle.

```
# Create and generate a word cloud image for all data :
wordcloud = WordCloud(max_words=100, background_color="white").generate(text)
# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

FIGURE 3.7 – Lemmatization du texte

3.1.2 Traitement des données

Analyse de sentiment

Nous avons effectué une analyse de sentiment en utilisant TextBlob. La fonction `sentiment` de TextBlob renvoie un tuple de la forme `Sentiment(polarity, subjectivity)` où `polarity` correspond à la polarité et `subjectivity` à la subjectivité. Le score de polarité est un nombre décimal compris entre -1 et 1 et le score de subjectivité

est un nombre compris entre 0 et 1 (0 correspond à une très forte objectivité et 1 correspond à une très forte subjectivité). Cette fonction a été appliquée à chaque contenu de Tweet puis stocker dans une liste nous permettant une utilisation plus simple des résultats obtenus de l'ensemble des tweets.

```
from textblob import TextBlob
from textblob_fr import PatternTagger, PatternAnalyzer
import pandas as pd

polarity = []
subjectivity = []

for my_tweet in df['tokens_concat']:
    polarity.append(TextBlob(str(my_tweet),
                             pos_tagger=PatternTagger(),
                             analyzer=PatternAnalyzer()).sentiment[0])
    subjectivity.append(TextBlob(str(my_tweet),
                                  pos_tagger=PatternTagger(),
                                  analyzer=PatternAnalyzer()).sentiment[1])
```

FIGURE 3.8 – Analyse des sentiments

Pour l'analyse de sentiment nous nous intéressons plus particulièrement à la polarité. En effet si la polarité est négative cela correspond au fait que le Tweet est de sentiment négatif (c'est-à-dire qu'il contient beaucoup de terme à connotation négatif) si la polarité est positive cela correspond au fait que le Tweet est positif (c'est-à-dire contient beaucoup de terme à connotation positif) et sinon s'il est égale à 0 cela correspond à un Tweet de sentiment neutre (c'est-à-dire qu'il ne contient pas de mot négatif ou positif ou qu'il n'a pas été possible de juger de sa polarité). Grâce à cela nous avons pu récupérer la liste des Tweets négatifs et positifs.

```
corpus_pola['polarity']
positif=0
negatif=0
neutre=0
for i in corpus_pola['polarity']:
    if i == 0:
        neutre=neutre+1
    if i>0:
        positif=positif+1
    if i<0:
        negatif=negatif+1
total=negatif+positif+neutre
dic_polarite={"negatif":(negatif/total)*100,
              "positif":(positif/total)*100,
              "neutre":(neutre/total)*100}
```

FIGURE 3.9

Nous avons ainsi pu calculer la proportion de Tweets positif, négatif et neutre et en affichant cela à l'aide d'un diagramme en camembert. En utilisant le paquet Wordcloud, nous serons en mesure de générer une image qui nous donne les mots les plus représentatifs dans chaque ensemble de Tweets (positif, négatif, neutre). Dans l'exemple ci-dessous, on a pris les Tweets qui ont une polarité supérieure à 0.25 pour sélectionner juste les Tweets positifs.

```

tweet_en_pos = corpus_pola[corpus_pola['polarity'] > 0.25]
text = ' '.join(tweet_en_pos.tokens_concat_test)
print ("There are {} words in the combination of all review.".format(len(text)))
# Create and generate a word cloud image:
wordcloud = WordCloud(max_words=100, background_color="white").generate(text)
# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()

```

FIGURE 3.10 – Nuage de mot pour les Tweets positifs

Top Words

Afin de trouver les mots les plus utilisés dans les Tweets, nous avons utilisé la fonction Counter qui stocke les éléments comme des clés de dictionnaire, et leurs comptes sont stockés comme des valeurs de dictionnaire. A l'aide de la fonction `most_common(8)`, nous avons pu choisir le nombre des mots les plus utilisés.

```

from collections import Counter
dic = Counter(tokensList)
dic_freq = dic.most_common(8)
dictt=dict(dic_freq)
dic_freq

```

FIGURE 3.11 – Nuage de mot pour les Tweets positifs

3.1.3 Modélisation de thématiques

LDA

Après avoir transformé les données textuelles dans un format qui servira d'entrée pour l'entraînement du modèle LDA, nous convertissons l'objet *tokénisé* en un corpus et un dictionnaire. Il est important de connaître le nombre optimal de sujets à analyser. Pour ce faire, il est possible de calculer la cohérence pour différents nombres de sujets afin de choisir celui qui convient le mieux. Généralement, le nombre de sujets optimaux selon cette mesure correspond au moins grand nombre de sujets avant que le score de cohérence ne se mette à baisser.

```

# Créer le dictionnaire
id2word = corpora.Dictionary(data_lemmatized)
# Créer le corpus
texts = data_lemmatized
corpus = [id2word.doc2bow(text) for text in texts]
#CALCUL du nombre de topic coherent
def compute_coherence_values(dictionary, corpus, texts, limit, start=2, step=3):
    coherence_values = []
    model_list = []
    for num_topics in range(start, limit, step):
        model=gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=id2word, num_topics=num_topics,
                                                random_state=1, update_every=1, chunksize=100, passes=10,
                                                alpha='auto', per_word_topics=True)

        model_list.append(model)
        coherencemodel = CoherenceModel(model=model, texts=data_lemmatized, dictionary=id2word, coherence='c_v')
        coherence_values.append(coherencemodel.get_coherence())
    return model_list, coherence_values

```

FIGURE 3.12 – Fonction pour déterminer le nombre de sujet optimal

Maintenant que nous disposons d'un modèle entraîné, nous visualisons les sujets pour faciliter l'interprétation. Pour ce faire, nous utiliserons un package de visualisation populaire, `pyLDavis` qui est conçu pour aider de manière interactive à :

- Mieux comprendre et interpréter les sujets individuels.
- Mieux comprendre les relations entre les sujets.

Le package pyLDavis nous donne la possibilité de sélectionner manuellement chaque sujet pour visualiser ses termes les plus fréquents et aussi de remarquer la différence entre chaque sujet à l'aide du positionnement de chaque bulle puisque plus deux sujets sont loin plus ils sont différents.

```
# Construire le modèle LDA
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=id2word,
                                             num_topics=topic_optimal, random_state=1,
                                             update_every=1, chunksize=100, passes=10,
                                             alpha='auto', per_word_topics=True)

#Visualier le modèle avec pyLDavis
pyLDavis.enable_notebook()
vis = pyLDavis.gensim_models.prepare(lda_model, corpus, id2word)
vis
```

FIGURE 3.13 – Fonction pour déterminer le nombre de sujet optimal

NMF

Comme dans LDA, il est essentiel de disposer d'un moyen pour sélectionner automatiquement le meilleur nombre de sujets. En utilisant le score de cohérence, nous pouvons exécuter le modèle pour différents nombres de sujets et ensuite utiliser celui qui a le score de cohérence le plus élevé.

```
texts = df['tokens']
dictionary = Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]
topic_nums = list(np.arange(3, 40, 1))

coherence_scores = []

for num in topic_nums:
    nmf = gensim.models.nmf.Nmf(corpus=corpus, num_topics=num, id2word=dictionary,
                                chunksize=2000, passes=5, kappa=.1, minimum_probability=0.01, w_max_iter=300,
                                w_stop_condition=0.0001, h_max_iter=100, h_stop_condition=0.001, eval_every=10,
                                normalize=True, random_state=42 )

    # Run the coherence model to get the score
    cm = CoherenceModel( model=nmf, texts=texts, dictionary=dictionary,
                        coherence='c_v')
    coherence_scores.append(round(cm.get_coherence(), 5))

# Get the number of topics with the highest coherence score
scores = list(zip(topic_nums, coherence_scores))
best_num_topics = sorted(scores, key=itemgetter(1), reverse=True)[0][0]
```

FIGURE 3.14 – Déterminer le nombre de sujet optimal

Après que nous avons déterminé le nombre de sujets, la première étape consiste à créer le *TF-IDF* à l'aide de la fonction *TfidfVectorizer* puis nous avons construit le modèle NMF qui générera les matrices des caractéristiques et des composantes.

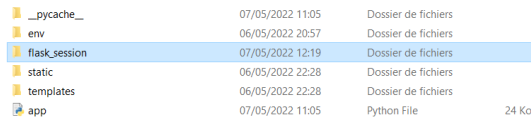
```
vect = TfidfVectorizer(min_df=10)
# Fit and transform
X = vect.fit_transform(df.tokens_concat_test)
model = NMF(n_components= best_num_topics)
# Fit the model to TF-IDF
model.fit(X)
# Transform the TF-IDF: nmf_features
nmf_features = model.transform(X)
```

FIGURE 3.15 – Déterminer le nombre de sujet optimal

3.2 Intégration à l'interface

Pour pouvoir intégrer toutes les fonctionnalités évoquées dans la section précédente à une interface, nous avons opté pour l'utilisation de flask. Ce framework nous a permis de créer une interface en utilisant également le langage HTML et le framework Bootstrap pour le front-end.

Il se compose d'un répertoire templates contenant la partie front-end de notre application, d'un répertoire static contenant les fichiers css et images générées par notre programme. Il se compose également d'un fichier app.py situé à la racine de notre projet.



__pycache__	07/05/2022 11:05	Dossier de fichiers	
env	06/05/2022 20:57	Dossier de fichiers	
flask_session	07/05/2022 12:19	Dossier de fichiers	
static	06/05/2022 22:28	Dossier de fichiers	
templates	06/05/2022 22:28	Dossier de fichiers	
app	07/05/2022 11:05	Python File	24 Ko

FIGURE 3.16 – Répertoire de l'interface

La première étape consiste à activer l'environnement Python et à installer Flask en utilisant pip. Une fois que l'environnement de programmation est activé nous remarquons "env" dans l'invite de commande et nous pouvons commencer à utiliser Flask.

- Fichier app.py

Dans le fichier app.py, nous avons tout d'abord importé l'objet Flask du paquet flask pour créer notre instance d'application Flask.

Après avoir créé l'instance de l'application, nous l'avons utilisé pour gérer les requêtes Web entrantes et envoyer les réponses à l'utilisateur. À l'aide d'app.route() nous avons transformé une fonction Python ordinaire en une fonction de vue Flask c'est-à-dire une fonction qui convertit la valeur de retour de la fonction en une réponse HTTP à afficher par un client HTTP, tel qu'un navigateur web.

Les applications Web utilisent principalement le langage HTML pour afficher des informations pour les utilisateurs. Nous avons donc intégré différents fichiers HTML à notre application (ces fichiers sont placés dans le répertoire templates). En effet, Flask fournit une fonction render_template() à laquelle nous passons en paramètres le nom d'un fichier situé dans le répertoire templates (par exemple main.html pour afficher notre page d'accueil).

```
from flask import Flask
import ...
app = Flask(__name__)
*****

//On définit ici les fonctions de pré-traitement,
traitement, calculs des modèles de thématique
def fonctions():
    return...
*****

@app.route("/", methods = ['GET', 'POST'])
def index():
    return...

@app.page("/page")
def page():
    return render_template('page.html')

if __name__ == "__main__":
    app.run()
```

FIGURE 3.17 – Structure du fichier app.py

- Repertoire Templates

Ce répertoire contient les différents fichiers HTML permettant la visualisation de nos résultats.

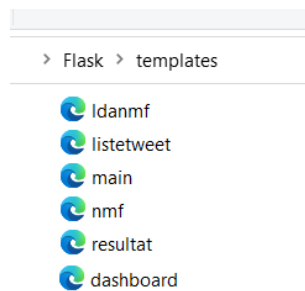


FIGURE 3.18 – Contenu du repertoire templates

Le fichier main.html permet d'afficher la page d'accueil de notre interface.

Le fichier resultat.html permet, une fois que les tweets sont chargés d'accéder aux différentes fonctionnalités offertes par l'application.

Le fichier dashboard.html nous offre la possibilité de visualiser des diagrammes d'analyse descriptive et sentimentale.

Le fichier listetweet.html nous permet de visualiser dans un tableau les tweets avec quelques informations en plus telles que le nombre de retweet.

Les fichiers ldanmf.html contiennent les résultats de la visualisation du modèle LDA et le fichier nmf.html contient le résultat pour le modèle NMF.

-Repertoire Static

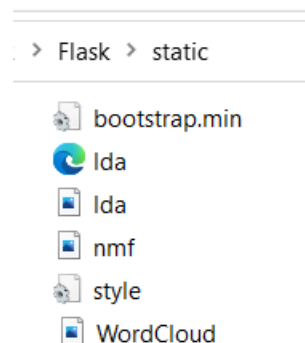


FIGURE 3.19 – Contenu du repertoire static

Le fichier bootstrap.min nous permet de faire appel au framework bootstrap (version 4) utile à la création du design de notre application.

Le fichier lda.html est le résultat de la bibliothèque pyLDAvis permettant une visualisation dynamique des résultats du modèle LDA.

lda.png et nmf.png sont les résultats de la visualisation de diagramme en bâton utilisant la bibliothèque matplotlib. Le fichier WordCloud est le résultat de la création de nuage de mot à l'aide de la bibliothèque wordcloud. Ces images générées dans notre fichier app.py sont appelées au niveau des différents fichiers du template et permettant ainsi leur visualisation l'utilisateur.

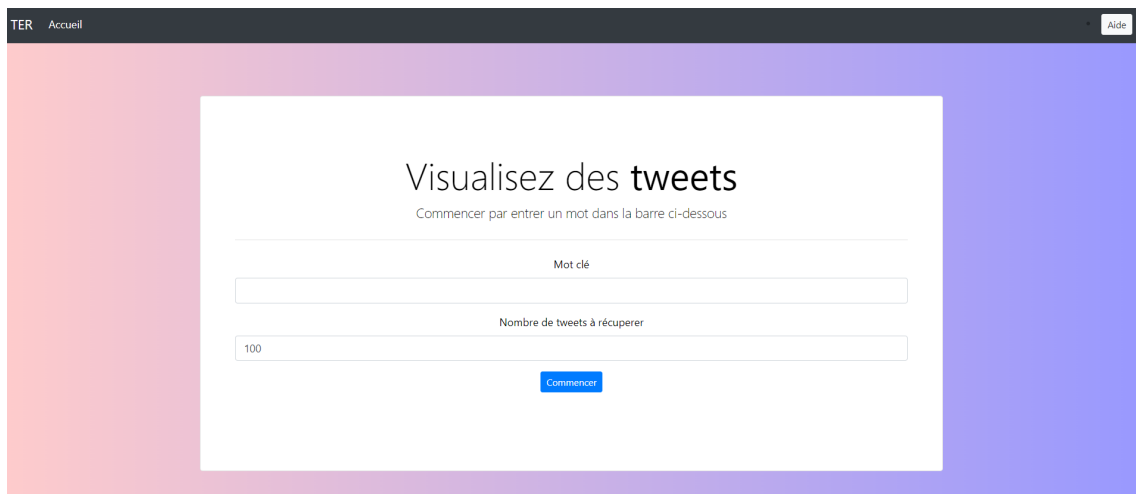
Chapitre 4

Principaux résultats et étude de cas

Dans ce chapitre nous nous attacherons premièrement à présenter le fonctionnement global de l'interface et ses fonctionnalités puis dans un second à étudier un cas précis permettant l'utilisation de l'application.

4.1 Fonctionnement de l'interface

Lorsque l'on lance l'application, une barre de texte permettant d'entrer le mot que l'on souhaite utiliser pour récupérer les tweets contenant uniquement ce mot et le nombre de tweets à récupérer (minimum=100 et maximum=1000) s'affiche (voir l'image ci-dessous) :



TER Accueil Aide

Visualisez des tweets

Commencer par entrer un mot dans la barre ci-dessous

Mot clé

Nombre de tweets à récupérer

100

Commencer

FIGURE 4.1 – Page d'accueil

Lorsque l'on entre les informations demandées, une autre page s'affiche contenant différents boutons.

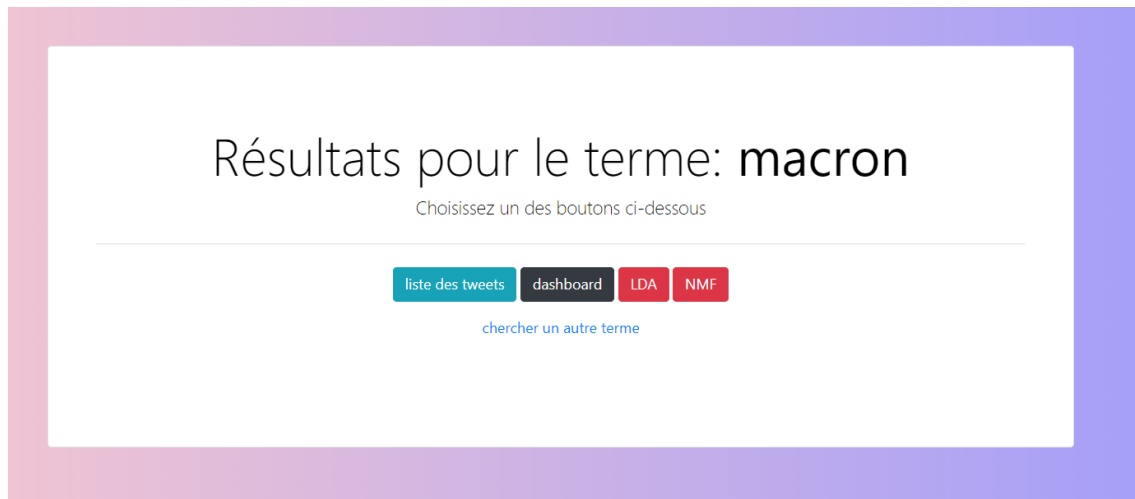


FIGURE 4.2 – Page résultat

Cette page contient 4 boutons que nous allons présenter.

- Le bouton "liste des tweets" permet d'afficher un tableau contenant les tweets récupérés avec le contenu du tweet, le nombre de retweet et le nombre de favoris. Les tweets n'ont pas subi de traitement.

Liste des tweets		
Tweets	Nombre de j'aime	Nombre de retweet
@AngiePercic @Tinemar4 @lecurci @nicolasputsch 1) on favorise @MLP_officiel la machine à perdre en face de macron 2) on radiabolise @MLP_officiel au second tour; la nullité du programme #rn fait le reste au cours du débat. 3) maçon réélu en dépit des 72% opposants râleurs mais peureux Les français sont si prévisibles.	1	1
@jeanneysh @nicolasberrod Si Macron est arrogant il n'est pas tout seul !!! 😡😡😡	0	0
@GabJune02 @OhZelidee @JLMelenchon Pardon ? Là vous venez de décrire les 5 dernières années de Macron c'est ça ? Insulter, mépriser une partie de la population ne pas écouter les revendications sociales et écologiques Et vous venez faire la morale sur les gens de gauche qui ignorent les critiques de ceux de droite?	0	0
@yoshispower @guilltes @besse_pascale La salope collabo ! Elle a voté macron et a des actions pfizer je suis sur !	0	0
@Pricel1411 @franceinfo "groupe de gens" Ca inclut EELV, le PS, et autre mouvement de gauche qui ont rejoint NUPES Il a pas appelé à voter Macron, mais c'est tout comme, le refus d'un éventuel CNR afin de dégager Manu direct, et avoir une place garantie de PM. Trop dur d'être un personnage historique.	0	0
@Willem_X Le fléau du macronisme et hop là une nouvelle entreprise crée sous le quinquennat Macron.	2	0

FIGURE 4.3 – Liste des tweets récupérés

- Le bouton "dashboard" contient les résultats de l'analyse de sentiment et de l'analyse descriptive. Les images ci-dessous contiennent les différents graphes présents dans la page "dashboard" :

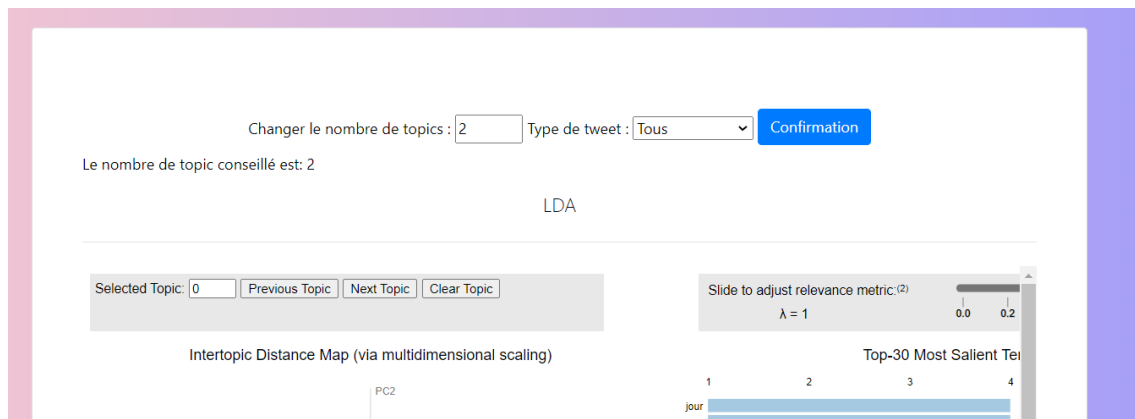


FIGURE 4.6 – LDA

Il est en est de même pour la page contenant les résultats du modèle NMF.

4.2 Étude de cas

Aux élections présidentielle 2022, Emmanuel Macron a récolté 27.84% des suffrages le plaçant en tête des résultats. Néanmoins cela signifie que 71.26% des électeurs ne souhaitent pas avoir Emmanuel Macron comme président de la République pour le prochain quinquennat. De plus, au second tour e. Macron récolte face à Marine Le Pen 58,55 % des suffrages contrairement aux élections présidentielles de 2017 dans lesquelles il récolte 66.10% des suffrages. A noter, qu'il y a eu 28% de taux d'abstention pour ces élections qui est un donc un record historique depuis 1969. Ces résultats nous amènent à réfléchir sur la popularité du président élu notamment à travers les tweets écrit par la population. Cette réflexion pose la problématique suivante : Quelle serait la proportion d'avis négatif et positif sur un certain nombre de tweets et ainsi les thématiques abordées lorsque le nom Macron est évoqué.

Pour répondre à la problématique, nous commençons premièrement par entrer dans notre application le terme "macron". Cela nous permettra de récupérer tous les tweets relatifs au président de la république. Nous fixons une limite de 800 tweets.

Mot clé

macron

Nombre de tweets à récupérer

800

[Commencer](#)

Dans un second temps, nous accédons à la page "dashboard .html" Celle-ci nous permettra de visualiser plus en détail les avis négatifs ou positifs de l'ensemble des tweets récupérés. En analysant les tweets, voici les tops 8 des termes qui ont été le plus abordé quand on cherche le mot macron pour les tweets négatif et positif :

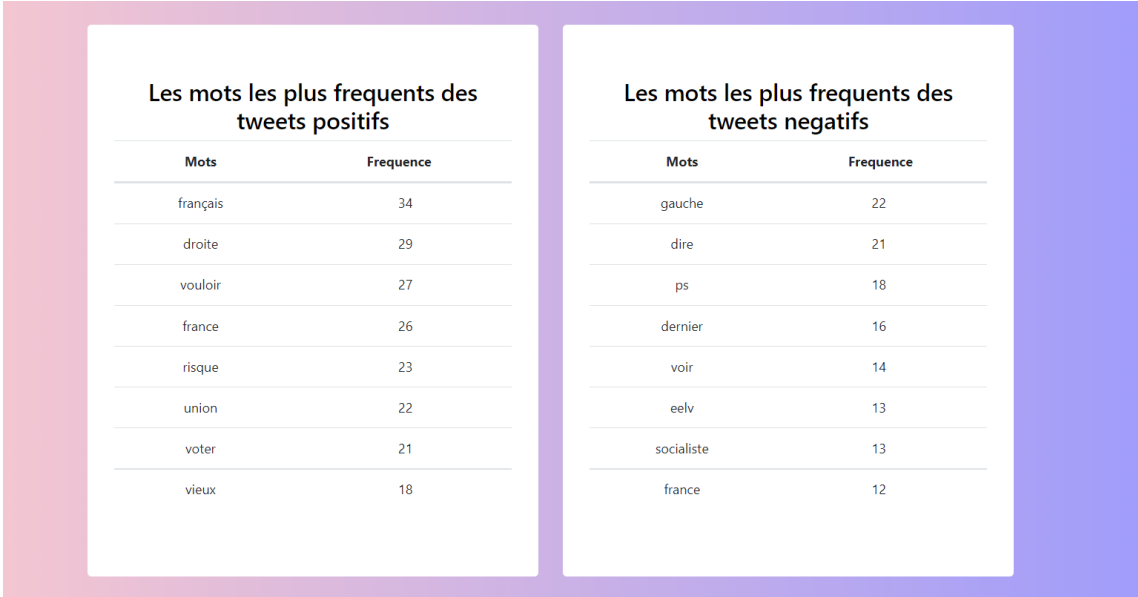


FIGURE 4.7
Polarité des tweets positifs et négatifs

Pour mieux avoir une idée sur les termes les plus populaires du moment, nous avons décidé de représenter ces termes avec un nuage des mots. Les mots fréquents sont "voter", "france", "gauche", "droite" et "vouloir" ce qui est normal puisque nous sommes en période des élections législatives. Donc la plupart des tweets vont contenir le terme "voter". On note aussi que le mot droite est plus fréquent que le mot gauche donc on peut dire que Macron a une popularité au sein de la droite plus que la gauche. De plus, on remarque que le mot "guerre" est présent, cela est en relation avec la guerre en Ukraine.

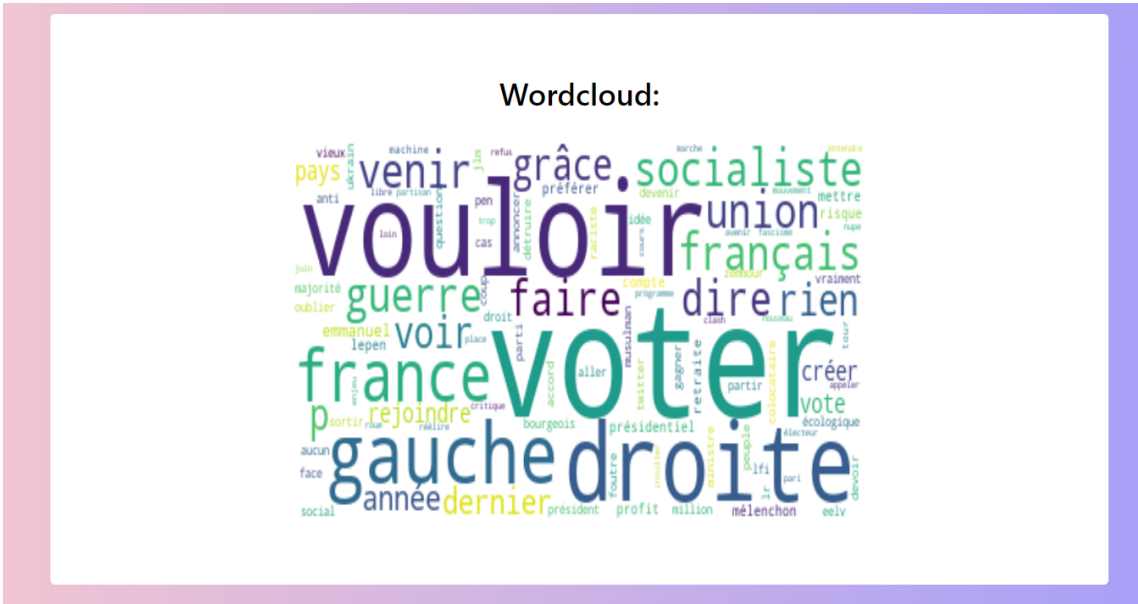


FIGURE 4.8 – Nuage de mot de l’ensemble des tweets récupérés

Puisqu'un politicien doit avoir une bonne image auprès du peuple, une analyse de sentiment est primordiale pour mesurer le ressenti des internautes et leurs impressions sur le président. Nous observons d'après l'image ci-dessous que 36% des gens ont eu des avis positifs contrairement à 20% ont eu des avis négatifs et 44% qui représentent presque la moitié des internautes ont eu des avis neutres.

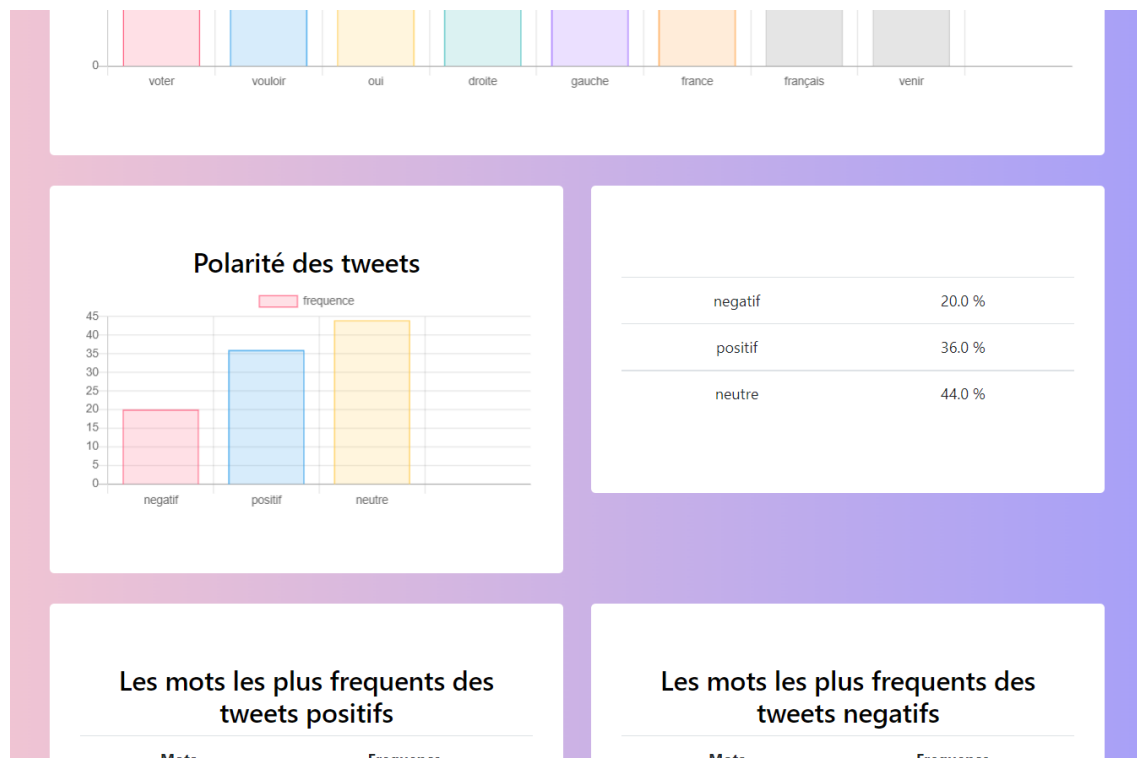


FIGURE 4.9 – Pourcentage des polarités

À l'aide des modèles LDA et NMF nous souhaitons maintenant connaître les principales thématiques abordées sur l'ensemble de tweets récupérés. Pour la LDA notre application nous suggère d'utiliser 2 topics. Nous remarquons à l'aide de la figure 4.12 que pour le premier topic on a plusieurs de terme en rapport avec le vote tel que voter ou présidentielle. Le topic 2 présentes beaucoup de mots en rapport avec la gauche (telle que les mots "gauches", "socialiste") et la droite.

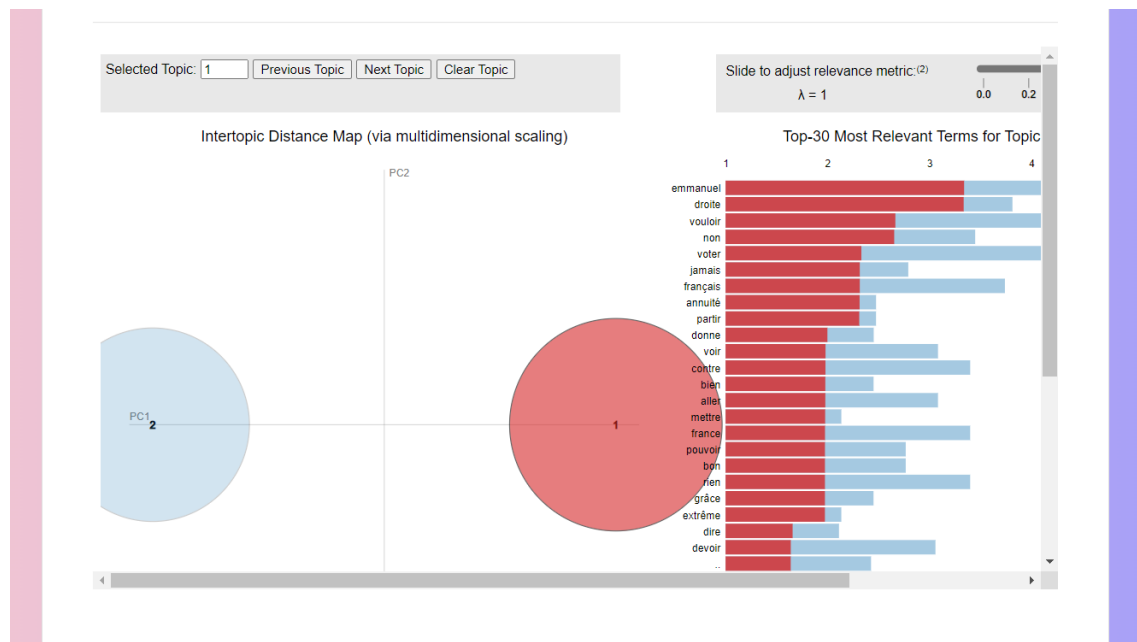


FIGURE 4.10 – Résultat de la LDA avec la bibliothèque pyLDAvis (affichage du topic 1)

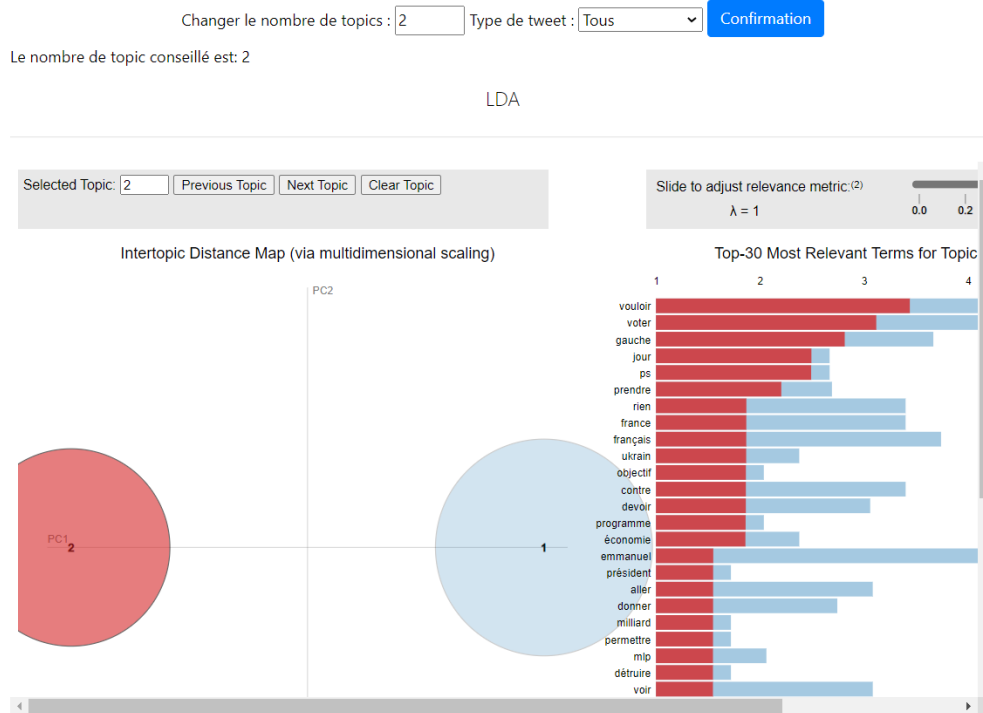


FIGURE 4.11 – Résultat de la LDA avec la bibliothèque pyLDAvis (affichage du topic 2)



FIGURE 4.12 – LDA avec le nombre de topics conseillé (2 topics)

Nous effectuons une variation nombre de Topic pour la LDA pour voir s'il serait possible d'obtenir de meilleur résultat plus simple à interpréter. Nous faisons varier le nombre de Topics à 3. Voici les résultats obtenus :



FIGURE 4.13 – LDA avec 3 Topics

On peut observer dans la figure ci-dessus que pour le premier Topic (Topic 0) nous avons des termes tels que "investiture", "cérémonie" ou "quinquennat" sont présents et nous permettent de dire que ce topic est lié à l'investiture d'Emmanuel Macron qui a eu lieu le jour où les Tweets ont été récupéré. Le troisième Topic (Topic 2) contient des mots tels que "melenchon", "vote" ou "candidat" nous amenant à penser que ce topic est en relation avec les élections législative. L'interprétation n'est pas simple à établir de par le fait que certains termes contenus dans chaque thématique sont très proches et peuvent être présent dans les autres thématiques.

Le modèle NMF nous permet de visualiser 3 thématiques. D'après l'image ci-dessous, le premier topic (Topic 0) évoque des mots tels que "voter", "majorité", "juin", "colocataire" et "melenchon". Ces termes nous évoquent les élections législatives notamment par le fait que Jean-Luc Mélenchon et son parti souhaitent avoir une majorité de sièges à l'assemblée et ainsi imposer une cohabitation. Le deuxième topic (Topic 2) contient les termes "LR", "écologique", "union", "gauche" ou "socialiste". Cela pourrait évoquer le fait que la gauche veuille s'unir. Le 3ème topic (Topic 2) contient les mots "guerre", "ukrain", "million, et "détruire" par exemple. Ce topic peut évoquer la guerre en Ukraine.



FIGURE 4.14 – NMF avec le nombre de topics conseillé (3 topics)

Pour conclure nous pouvons dire d’après les résultats obtenus que Macron obtient un pourcentage de tweets positifs, ce qui est un peu plus élevé comparé aux tweets négatifs. Néanmoins le pourcentage de tweet neutre est très élevé. Cela ne nous permet pas de conclure avec certitude que les tweets récupérés dégagent un sentiment positif à l’égard d’Emmanuel Macron. Nous avons déduit que les sujets principaux évoqués sont la guerre en Ukraine et les élections législatives. Nous avons également observé que le modèle NMF nous a permis plus facilement d’interpréter les topics que le modèle LDA. Notamment à cause du fait que les clusters de thématique étaient plus homogènes pour le modèle NMF que LDA.

Conclusion

Notre interface permet d'analyser différents tweets contenant un terme précis et permettant ainsi l'extraction des tweets relatant de ce mot. Dans un premier temps nous nous sommes attaché à effectuer un prétraitement des tweets, nous permettant d'obtenir un corpus de Tweets ne contenant que des termes utiles à notre analyse. Cela nous a permis dans un second temps de pouvoir effectuer une analyse de sentiment sur notre corpus de Tweets et une petite analyse descriptive. Enfin, à l'aide de cet ensemble de Tweets traité nous avons pu effectuer une modélisation de thématique à l'aide de deux modèles : LDA et NMF. À l'aide d'étude de cas, nous avons pu visualiser les résultats provenant des deux modèles. Ainsi, nos observations nous ont permis de voir que le modèle NMF donnait de meilleurs résultats en matière de topic interprétable sémantiquement que le modèle LDA. Cette interface implémentée nous permet donc d'observer les termes et thématiques qui reviennent le plus souvent pour un mot d'intérêt sur un corpus de Tweets.

Référence

- [1] <https://realpython.com/twitter-bot-python-tweepy/>
- [2] <https://gardeso.com/python-verification-de-la-coherence-de-modeles-lda>
- [3] <http://https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3step-1-installing-flask>
- [4] <https://machinelearningmastery.com/clean-text-machine-learning-python/>
- [5] <https://towardsdatascience.com/tokenization-for-natural-language-processing-a179a891bad4>
- [6] <https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a>
- [7] <https://towardsdatascience.com/lemmatization-in-natural-language-processing-nlp-and-machine-learning-a4416f69a7b6>
- [8] <https://towardsdatascience.com/a-simple-explanation-of-the-bag-of-words-model-b88fc4f4971>
- [9] <https://www.analyticsvidhya.com/blog/2021/11/how-sklearn-tfidfvectorizer-calculates-tf-idf-values/>
- [10] <https://neptune.ai/blog/vectorization-techniques-in-nlp-guide>
- [11] <https://monkeylearn.com/blog/introduction-to-topic-modeling/>
- [12] <https://towardsdatascience.com/twitter-topic-modeling-e0e3315b12e2>