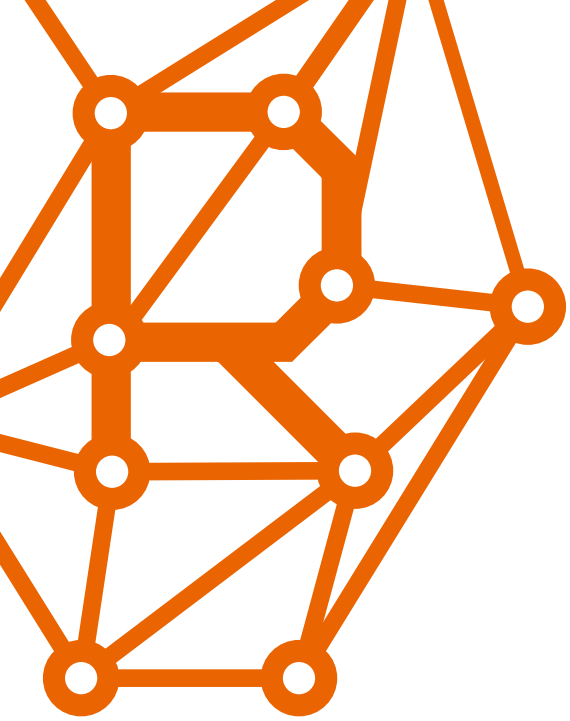


UQAC

UNIVERSITÉ DU QUÉBEC
À CHICOUTIMI

Projet de Big Data : **Analyse à grande échelle de sentiments sur Twitter**

Semestre d'Hiver 2021

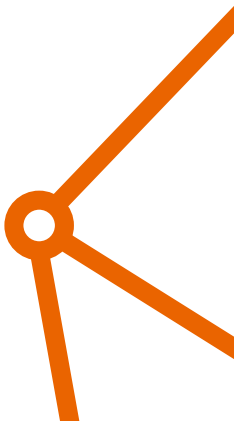
DEHBI Ilyas

POMALEGNI Augustin Primous Prince

Sous la supervision :

Pr A. Bouzouane

28 April, 2021



Abstract

Ce travail présente un ensemble d'expérimentations sur l'analyse de sentiments de données provenant de la plateforme très prisée Twitter en se référant au sujet d'actualité du moment et qui touche le monde entier: la crise du Coronavirus.

Mots clés: Spark. Dataset. PySpark. Tweets. Label. Sentiments. Analyse. NLP. NLTK

Contents

List of Figures

Acronyms

NLP Natural Language Processing

NLTK Natural Language ToolKit

Chapter 1

Introduction

Les mots traduisent généralement la pensée mais plus encore nos sentiments. Ils sont le reflet de nos sentiments en un moment donné si l'on peut dire; sentiments qui peuvent aller du positif au négatif en un temps record. L'internet de nos jours offre en plus de cela des lieux où toute personne peut déferler son ressenti en ligne sans penser aux actions ou aux conséquences que cela peut entraîner. Que ce soit le politicien qui publie des tweets chaque jour pour garder ses fans en haleine (cas de l'ancien président Américain Donald Trump) ou le simple internaute qui donne son avis sur un sujet, la finalité peut être la même. Entraîner du positif (grâce aux commentaires positifs) ou de la colère si les autres internautes n'adhèrent pas aux convictions de l'autre ou encore aller au-delà en entraînant des violences physiques. Ainsi pour contrer cela, plusieurs sites mettent en place des algorithmes permettant de contrer ces appels à la violence que peut causer un texte, un commentaire. C'est le cas de Facebook ou encore du géant de l'expression libre Twitter.

Ce travail vient donc à point nommé afin de nous aider à savoir comment ces algorithmes peuvent être mis en place et que peut être leur précision dans des situations en temps réel. Nous aurons donc à étudier les sentiments des twittos (ou auteurs de tweets) au moyen de leurs tweets en se référant au sujet d'actualité mondiale actuelle qui est **l'épidémie du Coronavirus 2019**. Nous verrons le sentiment que peuvent dégager leurs propos.

Ainsi pour y arriver, nous ferons une synthèse en premier lieu de l'article de recherche qui nous a été proposé afin de voir ou de comprendre la démarche qui a été entreprise dans ce travail, nous parlerons par la suite de notre démarche d'expérimentation en évoquant les différentes technologies que nous avons utilisées pour la mettre en oeuvre et pour finir nous tirerons des conclusions sur les résultats obtenus suite à ce travail.

Chapter 2

Synthèse d'article

Il a été soumis à notre lecture, l'article **"Large Scale Sentiment Analysis on Twitter with Spark"** de Nikolaos Nodarakis, Athanasios Tsakalidis, Spyros Sioutas et de Giannis Tzimas présentant leurs études sur l'analyse de sentiments à grande échelle de données de Twitter en utilisant Spark. Dans ce chapitre, nous en proposons une synthèse en demeurant sur les grands axes sans aller plus en détails et ceci parce que l'article en lui-même détaille de façon claire la méthodologie utilisée au cours de leurs travaux. Cela constitue une source d'inspiration pour la réalisation de travaux similaires dont les expérimentations que nous avons menées dans ce travail.

2.1 Introduction

Avec la grande quantité de données qu'on peut récolter actuellement, il est important d'avoir des systèmes capables de pouvoir les étudier afin d'en tirer les conclusions qu'il faut. L'une de ces sources de données est la plateforme Twitter que les auteurs présentent comme étant celle sur laquelle beaucoup de personnes s'expriment sur des sujets différents au moyen de caractères limités à 140 qu'on appelle tweet. Un tweet peut comprendre des mots d'une langue ou encore des émoticônes ou des hashtags (combinaison du caractère spécial # et d'un terme) et est utilisé pour exprimer son opinion sur un sujet. Etudier les tweets peut donc être intéressant pour comprendre l'appréhension que les personnes ont sur des sujets donnés. Cependant avec des centaines de milions de tweets qui sont postés par jour, les algroithmes d'analyse de sentiments actuellement utilisés ne sont pas adaptés (solutions généralement centralisées avec un temps de calcul énorme). Les auteurs proposent donc un algorithme distribué implémenté dans Spark basé sur le modèle MapReduce et qui exploite les hashtags et les émoticônes présents dans les tweets afin d'en déduire les polarités. Ils y rajoutent (dans leur travail) des filtres Bloom pour améliorer les performances

et utilisent une classification K-NN en MapReduce pour faire de la classification de sentiments. Avec un tel algorithme mis en place, ils étudient par la suite l'impact des facteurs tels que la taille du jeu de données, le nombre de noeuds dans la distrubution en cluster et le coefficient K sur le coût total de classification et les performances de classification.

Pour y arriver, ils présentent leurs travaux en commençant par parler des existants ou travaux connexes, ils nous présentent par la suite le modèle MapReduce et le cadre Spark. Ensuite, ils présentent le fonctionnement de leur algorithme avant de débattre sur les résultats obtenus et de finir avec la conclusion et les améliorations qui peuvent être apportées à leurs travaux.

2.2 Travaux antérieurs ou préliminaires

Le domaine de l'analyse de sentiments n'est pas si nouveau que ça pourrait paraître. Les premiers travaux dans le domaine remontent sur l'analyse de sentiments de documents liés aux critiques de films ou de produits, les articles publiés sur le Web et les blogs. Il a aussi été abordé l'analyse de sentiments sur des phrases en attribuant à chaque phrase une polarité (positive, négative ou neutre). Pour ce qui est des approches utilisées, les précédents travaux dans le domaine se sont servis de l'apprentissage automatique avec les techniques de traitement naturel du langage. Les algorithmes tels que Naive Bayes, les Support Vector Machine et l'entropie maximum ont été appliqués dans les travaux. L'approche utilisant Spark, sans avoir à créer un lexique des sentiments ou de procéder à l'anontation des données, est donc à l'initiative des auteurs de cet article.

Aussi utiliser le modèle MapReduce revient à utiliser deux types de fonctions principales : une fonction Map et une fonction Reduce qui se charge du regroupement en clé-valeur sans se soucier de la distribution des données sur les noeuds.

Apache Spark est quant-à-lui un moteur rapide et général pour le traitement de données à grande échelle. C'est essentiellement l'évolution du framework Hadoop . Hadoop est l'implémentation open source du modèle MapReduce et est largement utilisé pour le traitement distribué entre plusieurs serveurs. Il est idéal pour les processus par lots lorsque nous devons parcourir toutes les données. Cependant, ses performances chutent rapidement pour certains types de problèmes (par exemple lorsque nous devons gérer des algorithmes itératifs ou basés sur des graphes). Spark est une pile unifiée de plusieurs composants étroitement intégrés et surmonte les problèmes d'Hadoop. Il dispose d'un moteur d'exécution DAG (Directed Acyclic Graph) qui prend en charge le flux de données cyclique et le calcul en mémoire. En conséquence, il peut exécuter des programmes jusqu'à

100 fois plus vite que Hadoop en mémoire, ou 10 fois plus vite sur disque. Spark comprend une pile de bibliothèques qui combinent SQL, streaming, apprentissage automatique et traitement de graphes dans un seul moteur.

2.3 Cadre de classification de sentiments

Pour faire de la classification l'ensemble des données est prétraité de telle sorte que chaque mot d'un tweet puisse apppartenir à une des classes définies au cours de cette phase sans faire intervenir un corpus externe. Ainsi les mots sont classés suivant HFW (mots à haute fréquence pour signifier rare), CW (mots de contenu) et RW(pour les mots réguliers). A la suite de ça, on ajoute une fonctionnalité de ponctuation aux tweets en se basant sur les critères comme le nombre de majuscules, le nombre de mots et le nombre de ponctuation (interrogation ou exclamation). On forme grâce à ces méthodes des vecteurs qui portent des points qui seront par la suite compressés au cours de leur stockage grâce aux filtres de Bloom pour les tranformer en vecteurs de bits beaucoup plus stockables sur une machine.

L'étape qui suit est l'usage de l'algorithme K-NN (des k voisins) pour classifier les données en leur attribuant les différents labels par voisinage.

En gros les différentes étapes de l'algorithme mis en place par les auteurs de cet article sont:

- Extraction de caractéristiques:
- Construction de vecteur de caractéristique:
- Calcul de distance
- Classification des sentiments.

2.4 Evaluation

Dans cette section, ils menent une série d'expériences pour évaluer les performances de leur méthode sous de nombreux points de vue. Plus précisément, ils prenent en considération l'effet de k et les filtres Bloom, le taux de compactage de l'espace, la taille du jeu de données et le nombre de nœuds dans les performances de leur solution. L'évaluation de la méthode est faite à l'aide de deux ensembles de données Twitter (un pour les hashtags et un pour les émoticônes) qui ont été collectés via l'API Twitter Search 3 entre novembre 2014 et août 2015. Quatre juges humains

impartiaux pour créer une liste de hashtags et une liste d'émoticônes qui expriment un sentiment fort (par exemple #amazed et :(). Ensuite, ils procèdent à une tâche de nettoyage consistant à exclure de la liste des hashtags et des émoticônes qui ont été abusés par les utilisateurs de Twitter (par exemple #love) ou a renvoyé un très petit nombre de tweets. A la fin, ils obtiennent une liste de 13 hashtags (ie H = #étonné, #génial, #belle, #ennuyé, #excité, #fun, #happy, #lol, #peace, #proud, #win, #wow, #wtf) et une liste de 4 émoticônes (ie E = :), :(, xD, <3). . Le jeu de données final des hashtags contient 942188 tweets (72476 tweets pour chaque classe) et le jeu de données final des émoticônes contient 1337508 tweets (334377 tweets pour chaque classe). La taille de l'ensemble de données hashtags est de 102,78 Mo et la taille de l'ensemble de données d'émoticônes est de 146,4 Mo.

2.4.1 Performance de classification

Dans cette partie, les résultats de deux configurations expérimentales ont été présentés par les auteurs: la classification multi-classes et la classfi cation binaire. Dans le cadre de la classification multi-classes, ils essayent d'attribuer une seule étiquette de sentiment à chacun des vecteurs de l'ensemble de test. Dans l'expérience de classification binaire, ils vérifient si une phrase convient à une étiquette spécifique ou ne porte aucune polarité de sentiment.

La performance de la classification multi-classes n'est pas très bonne, bien qu'elle soit bien au-dessus de la ligne de base aléatoire. De plus, les résultats avec et sans filtres Bloom diffèrent légèrement. La même chose se produit pour la classification binaire, mais avec une meilleure précision. Cela est attendu en raison du nombre plus faible de types de sentiments. Ce comportement peut également s'expliquer par l'ambiguïté des hashtags et un certain chevauchement des sentiments. Néanmoins, il y a une légère augmentation des performances de classification de l'algorithme lors de l'utilisation de filtres Bloom, ce qui est quelque peu inattendu.

2.4.2 Effet de K

Dans cette sous-section, l'effet de k dans les performances de classification de l'algorithme a été mesuré. Quatre configurations différentes ont été testées où k = 50, 100, 150, 200. Pour la classification binaire et multi-classes, l'augmentation de k affecte légèrement (ou pas du tout) la précision de la classification lorsque les filtres Bloom sont incorporés. Au contraire, sans les filtres Bloom, il y a une plus grande amélioration des performances de précision pour les deux configurations de classification (jusqu'à 3% de plus). La conclusion de cette expérience est que des valeurs plus élevées de k peuvent fournir une bonne impulsion dans les performances de l'algorithme

lorsque vous n'utilisez pas de filtres Bloom. Cependant, des valeurs plus élevées de k signifie plus de temps de traitement.

2.4.3 Compression spatiale : filtrage Bloom

Dans la majorité des cas, les filtres Bloom parviennent à diminuer légèrement l'espace de stockage requis pour les vecteurs de caractéristiques (jusqu'à 3%). Dans un cas (hashtags multi-classes), la diminution de l'espace requis est significative (près de 9%). Les raisons de ces petites différences sont deux. Tout d'abord, dans chaque filtre Bloom, on ne met qu'une seule fonction (au lieu de plus) en raison de la nature du problème. Deuxièmement, on garde en mémoire un objet filtre Bloom au lieu d'un objet String. Mais, la taille qu'occupe chaque objet dans la mémoire principale est presque la même (le filtre Bloom est légèrement plus petit). Étant donné que la taille des entrées n'est pas très grande, nous nous attendons à ce que cet écart augmente pour des ensembles de données plus volumineux qui produiront des vecteurs d'entités consommant beaucoup plus d'espace. Par conséquent, on en déduit que les filtres Bloom peuvent être très bénéfiques lorsqu'il s'agit de données d'analyse de sentiment à grande échelle, qui génèrent un nombre excessif de caractéristiques lors de l'étape de construction du vecteur de caractéristiques.

2.5 Conclusion

Dans cet article, les auteurs présentent une nouvelle méthode d'apprentissage des sentiments dans le cadre Spark. Leur algorithme exploite les hashtags et les émoticônes dans un tweet, comme étiquettes de sentiment et procède à une procédure de classification de divers types de sentiments de manière parallèle et distribuée. De plus, ils utilisent des filtres Bloom pour compacter la taille de stockage des données intermédiaires et augmenter les performances de leur algorithme. Grâce à une évaluation expérimentale approfondie, ils ont prouvé que leur système est efficace, robuste et évolutif.

Ils souhaitent également des améliorations à leur travail afin d'en améliorer les performances sur les tâches de classification.

Chapter 3

Présentation des technologies utilisées

Après avoir présenté une synthèse de l'article soumis à lecture, dans ce chapitre, nous présenterons l'ensemble des technologies utilisables pour faire de l'analyse de sentiments. Ces technologies sont nombreuses et peuvent être agencées avec de nombreux algorithmes, notamment les algorithmes de classification ou de prédiction qui peuvent être implémentés en utilisant ces différents environnements. Rappelons que l'article lu aura été inspirant afin de comprendre plus le domaine et de pouvoir s'en inspirer pour monter des algorithmes similaires.

Mais avant tout, nous parlerons des technologies d'acquisition de données indispensables à ce travail, il s'agit bien évidemment des tweets.

3.1 Les différents procédés d'extraction de données sur Twitter

Il existe de nombreuses techniques d'extraction de données sur Twitter allant de formes d'extraction simple à celles nécessitant des librairies construites à ce sujet.

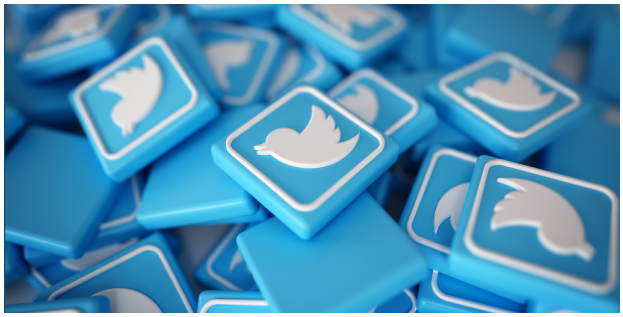


Figure 3.1: Image représentant des tweets

3.1.1 Usage de données disponibles en ligne

Ce n'est pas vraiment une technique d'extraction de données en temps réels mais c'est quand même une technique qui permet de s'assurer du bon fonctionnement de son modèle ou de ses modèles avant de les passer en phase de déploiement: il s'agit de l'usage de dataset déjà extrait et disponible en ligne.

En effet, cette technique permet de ne pas penser à l'extraction en temps réel des données mais plutôt de se focaliser sur son environnement de travail et surtout sur la manière de construire son modèle afin d'améliorer sa précision en phase de déploiement. Aussi, utiliser un dataset déjà disponible permet de ne pas s'encombrer des phases de prétraitement de tweets sachant que dans un tweet il y a une panoplie d'informations qui ne s'avère pas vraiment nécessaire afin de faire fonctionner son modèle. Nous évoquerons dans la suite les différentes phases de prétraitement que peut demander un projet d'analyse de sentiments de tweets.

A présent, abordons les techniques d'extraction de tweets que nous avons pu trouver dans nos recherches.

3.1.2 API de Twitter

Pour faire du machine learning sur des tweets, il faut des tweets. Pour cela, il existe de nombreuses techniques d'extraction de tweets. La première technique que nous vous présentons ici est l'utilisation de l'API de Twitter.

Twitter comme toute grande entreprise de tech a fourni un outil nous permettant de pouvoir accéder aux données dont nous avons besoin en ajoutant certaines conditions sur le type de tweets que nous souhaitons utiliser.

Pour faire usage de cette API, il faut s'inscrire en tant que développeur ce qui permet de pouvoir se servir de cette API car Twitter fournit à cet effet des clés d'accès à son site.

Par la suite, il faut se servir du package Tweepy afin d'extraire les données que nous souhaitons.

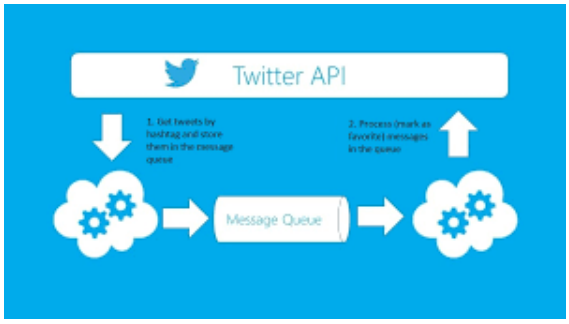


Figure 3.2: Fonctionnement de l'API Twitter

Cette technique représente peut-être le moyen le plus sûr de pouvoir extraire des données contrairement aux autres techniques d'extraction qui peuvent s'avérer défaillantes même après plusieurs succès en phase de test.

3.1.3 Twitterscraper

La seconde option que nous présentons ici est plus simple que la première mais ne fonctionne pas toujours (nous en avons essuyé quelques échecs).

Elle permet(cette option) de pouvoir extraire un grand nombre de tweets en spécifiant des critères comme :

- la date de début d'extraction des tweets;
- la date de fin d'extraction;
- la langue dans laquelle les tweets à extraire seront;
- les mots clés que nous souhaitons avoir dans notre recherche;
- et bien d'autres encore...

Son usage dans un projet est relativement simple car il consiste surtout à importer la librairie query_tweets du package twitterscrapper comme suit:

```
from twitterscraper import query_tweets
```

Mais avant cela, il faut s'assurer d'avoir le package au moyen de la ligne de commande suivante:

```
pip install twitterscrapper
```

3.1.4 Twint

Le package Twint est également un des packages permettant d'extraire des données sur Twitter. Son fonctionnement est similaire au package twitterscrapper car demandant de préciser certaines infos sur les tweets que l'on souhaite extraire. Ces infos vont :

- des mots-clés de notre recherche de tweet;
- en passant par la date;
- sans oublier la langue des tweets.

Son utilisation nécessite aussi l'installation du package twint et son importation dans le projet de Bog Data. Cependant, il présente le désavantage d'être lent au cours de son utilisation. Il faudra donc juger de l'importance que nous accordons ou que l'on peut accorder aux temps d'exécution de notre application avant de l'employer.

3.1.5 GetOldTweets3

La dernière méthode que nous vous présentons ici est celle utilisant le package GetOldTweets3. Comme son nom l'indique, il permet comme ces précédents de faire de l'extraction de données sur Twitter. Demandant aussi de renseigner des critères de recherche comme les mots clés, la date, la langue et même encore plus la source de données que l'on souhaite utiliser (compte twitter d'un particulier ou d'un journal).

Son utilisation dans un projet nécessite son installation et son importation dans le projet de Big Data.

Nous avons pu voir dans cette partie qu'il existe de nombreux moyens de faire de l'extraction de tweets sur le site de l'oiseau bleu comme on aime souvent l'appeler. Cependant, malgré l'abondance de techniques qu'on a, elles sont parfois défaillantes ou encore nécessitent une phase de traitement

sur les tweets récupérés avant de les utiliser dans le projet de Big Data. Nous parlerons de ces phases de traitement sur la partie expérimentation notamment en présentant les données.

3.2 Environnement Pyspark

Dans cette partie, nous allons essayer de prédire les sentiments des tweets à l'aide de l'outil Pyspark. La base de données utilisée dans cette partie vient du site suivant : For Academics - Sentiment140 - A Twitter Sentiment Analysis Tool. Cette base de données est née d'un projet de recherche de Stanford, et chaque enregistrement du dataset est composé de 6 attributs qui sont :

0 - the polarity of the tweet (0 = negative, 4 = positive)

1 - the id of the tweet

2 - the date of the tweet

3 - the query (lyx). If there is no query, then this value is NO_QUERY.

4 - the user that tweeted

5 - the text of the tweet

3.2.1 Importation des données

La première étape de toute programmation Apache est de créer un SparkContext. SparkContext est nécessaire lorsque nous voulons effectuer des opérations dans un cluster. SparkContext indique à Spark comment et où accéder à un cluster. C'est la première étape pour se connecter avec Apache Cluster.

Ensuite, nous allons importer les données sous forme de DataFrame puisqu'elle permet de bien structurer nos données.

```
root
 |-- target: integer (nullable = true)
 |-- ID: integer (nullable = true)
 |-- date : string (nullable = true)
 |-- query: string (nullable = true)
 |-- user: string (nullable = true)
 |-- text: string (nullable = true)
```

Figure 3.3: Schéma du DataFrame

Dans ce qui suit, on a juste besoin de deux attributs : le texte du tweet et son label. Pour cela, nous n'avons gardé que ces deux attributs.

3.2.2 Nettoyage des données

Pour commencer la classification des tweets, nous devons commencer par nettoyer nos données. Premièrement, nous allons remplacer tous les caractères qui ne sont pas des lettres par un espace, ainsi, on va supprimer tous les caractères spéciaux qui ne vont pas nous aider à faire la prédiction. Ensuite, nous allons transformer chaque phrase en une liste de String.

Les mots d'arrêt sont des mots qui peuvent être importants dans la communication humaine, mais qui ont peu de valeur pour les machines. Pyspark nous livre une classe par défaut de mots d'arrêt appelée StopWordsRemover. Les mots d'arrêt sont les mots tels que "the", "in", "out"...

3.2.3 Hashage des données

HashingTF convertit les documents en vecteurs de taille fixe. La dimension de fonctionnalité par défaut est de 262 144. Dans notre cas, la dimension de fonctionnalité est égale à 1048576. Les termes sont cartographiés en indices à l'aide d'une fonction hachage. La fonction de hachage utilisée est MurmurHash 3. Les fréquences de terme sont calculées en ce qui concerne les indices cartographiés.

```
Out[21]: [SparseVector(1048576, {0: 1.0, 170462: 1.0}),
 SparseVector(1048576, {0: 1.0, 799269: 1.0}),
 SparseVector(1048576, {0: 1.0, 178346: 1.0}),
 SparseVector(1048576, {0: 1.0, 158072: 1.0}),
 SparseVector(1048576, {0: 1.0, 1032246: 1.0})]
```

Figure 3.4: Résultat du data après le Hashing

3.2.4 TF-IDF + NaiveBayes

La méthode TF-IDF s'agit d'une méthode pour convertir les documents en vecteur de telle sorte que le vecteur reflète l'importance d'un terme pour un document dans le corpus. Pour quantifier l'importance d'un terme, on se base sur la fréquence que ce terme apparaît dans un document (tweet) et aussi sur la fréquence des documents qui contiennent ce terme.

TF en anglais c'est **Text frequency** et **IDF** c'est **Inverse Document Frequency**

$$TF(M_i, D_j) = \frac{nombre\ de\ fois\ que\ le\ mot\ M_i\ apparait\ dans\ le\ document\ D_j}{nombre\ de\ mots\ dans\ le\ document\ D_j}$$

$$IDF(M_i) = \ln(\frac{Le\ nombre\ total\ des\ documents}{Le\ nombre\ de\ documents\ qui\ contiennent\ M_i})$$

$$TF - IDF(M_i, D_j) = TF(M_i, D_j) \times IDF(M_i)$$

Naive Bayes est un algorithme simple de classification multiclasse avec la prise en charge de l'indépendance entre chaque paire de fonctionnalités. Dans un seul passage aux données de formation, il calcule la distribution conditionnelle des probabilités de chaque entité donnée, puis il applique le théorème de Bayes pour calculer la distribution conditionnelle des probabilités du Label donné par rapport à une observation et l'utiliser pour la prédiction.

3.2.5 Count Vectorizer + Logistic Regression

On peut aussi remplacer la méthode TF-IDF par une autre appelée CountVectorizer. Cette méthode permet de convertir les documents texte en vecteurs qui donnent des informations sur le nombre de jetons. Tout d'abord, CountVectorizer générera un vocabulaire au cas où un vocabulaire de prieuré n'est pas disponible. Au cours du processus d'ajustement, CountVectorizer sélectionne les meilleurs mots VocabSize commandés par fréquence de terme. Le modèle produira un vecteur qui peut être introduit dans d'autres algorithmes. Dans notre cas, on a choisi un vocabulaire de 1500 mots.

La régression logistique est une méthode populaire pour prédire une réponse catégorique. Cette méthode utilise la fonction softmax pour séparer entre les différentes classes:

$$\sigma(Y)_j = P(Y = j/X) = \frac{\exp^{\beta_j X + \beta_{0,j}}}{\sum_{i=1}^{N} \exp^{\beta_i X + \beta_{0,i}}}$$

3.3 Utilisation de la librairie NLTK: seconde approche d'analyse de tweets

Comme dans le cas de l'utilisation de PySpark qui est le socle du travail, nous avons pensé à explorer une autre manière de faire cette analyse de tweets portant sur le sujet du Covid qui nous intéresse.

NLTK pour Natural Language ToolKit est une bibliothèque (librarie) développé dans le langage Python qui fournit un ensemble d'outils permettant du traitement naturel de langage plus communément appelé **NLP**. Cette librairie est très utilisée dans le domaine et est très facile d'utilisation pour des novices dans le domaine du NLP. Elle offre de nombreuses actions sur les données textuelles afin de permettre à la machine de pouvoir comprendre ces données mais surtout

de pouvoir les traiter. Ces actions peuvent être citées comme suit :

- le nettoyage de données textuelles,
- la tokenization des données textuelles (passage de paragraphe en phrase ou en mots...)
- ...

On ne pouvait donc pas faire une implémentation sur l'analyse de sentiments sans parler de cette librairie très utile.

Ainsi on peut diviser l'implémentation en utilisant cette bibliothèque en plusieurs parties comme suit.

3.3.1 Importer des données

La première étape est l'importation des données(c'est-à-dire des tweets et de leur label). Pour cela nous vous invitons à consulter la section Présentation des données qui en dit plus sur le sujet.

3.3.2 Nettoyage des données

Avec NLTK, il existe de nombreuses fonctions permettant de nettoyer les données et de ne retenir que le strict minimum nécessaire pour le travail. Ce nettoyage permettant de gagner notamment en temps et de le retenir que l'essentiel des données textuelles, dans notre travail ici des tweets.

Pour nettoyer les tweets, nous pouvons les passer en trois étapes .

Etape 1: Se séparer des expressions irrégulières

Dans un tweet, on peut se retrouver avec un ensemble de mots ou de termes que l'on ne comprend pas forcément. Prenons en exemple les deux tweets suivants que nous avons imaginés. Le premier provenant d'un utilisateur user1

User1 : Ce Covid-19 a changé radicalement nos habitudes allant jusqu'à nous empêcher de sortir de chez nous pour profiter de la nature, de la famille et des merveilles de la vie. C'est dommage

Et le second d'un utilisateur user2

User2 : covid2Me*de laix ns profiT de la life. 2p8 13 mw u ns empêche 2 vivre.

Grosses différences pour ces deux tweets. Le premier étant lisible, le second nécessitant une certaine patience avant de le décoder pour nous les humains et imaginez ce qui en serait pour une machine. On va donc lui faciliter la tâche en se séparant des expressions irrégulières comme ,#,+,=,?,... et aussi des chiffres en les remplaçant par un espace vide. Nos deux exemples deviendraient donc:

User1 : Ce Covid a changé radicalement nos habitudes allant jusqu à nous empêcher de sortir de chez nous pour profiter de la nature de la famille et des merveilles de la vie C est dommage

Et le second d'un utilisateur user2

User2 : covidMede laix ns profiT de la life p mw u ns empêche vivre.

Bien évidemment on remarque qu'un tweet bien écrit va conserver tout son sens malgré cette première opération.

Etape 2: Se séparer des mots réccurents de la langue

La librairie NLTK possède une structure de données qui contient les stopWords ou les mots communs suivant le langage. En langue française les stopWords sont le, la , les, il, ... On passe l'ensemble des tweets à une fonction permettant de supprimer tous ces mots. Ainsi nos deux exemples de tweets peuvent donner:

User1 : Covid changé radicalement habitudes allant jusqu empêcher sortir chez pour profiter nature famille merveilles vie dommage

Et le second d'un utilisateur user2

User2 : covidMede laix ns profiT life p mw u ns empêche vivre.

Etape 3: Conserver la partie la plus importante de chaque mot

Cette étape de traitement des données consiste à ne conserver que la partie la plus importante de chaque mot. On peut dire que quand nous avons le mot "changement" par exemple dans une phrase en le remplaçant par "change" il est vrai que le sens est modifié mais pour des humains c'est toujours possible d'imaginer dans le texte qu'on lit. C'est un peu le même principe. On conserve que ce qui est important de chaque mot mais si cela consiste à le remplacer par la base du mot.

3.3.3 Hashage des données

Le procédé de Hashage de données avec NLTK est quasiment similaire à celui de PySpark car ceci les mêmes idées développées. Vous pouvez consulter à nouveau la section 3.2.3 pour la relire.

3.3.4 TF-IDF + NaiveBayes

Il en va de même pour cette partie. Vous pouvez également vous reporter à la section 3.2.4 pour en savoir plus sur le sujet.

Chapter 4

Expérimentation

Dans cette partie, nous présenterons les différentes expérimentations que nous avons mises en place en se référant un peu à l'article proposé, aux différents sites que nous avons consultés et surtout aux différentes approches existantes sur le sujet afin de pouvoir faire une meilleure étude comparative et d'explorer un peu plus le domaine de l'analyse des sentiments. Ceci dans le but d'en tirer des conclusions, de comparer les résultats.

Mais avant tout, il est important de présenter les données sur lesquelles nos expérimentations ont été faites. Allons donc à la découverte de ces données.

4.1 Présentation des données d'expérimentation

Comme mentionné plus haut, il existe de nombreux moyens de faire de l'acquisition de données. Qu'il s'agisse d'utiliser les bases de données fournies ou construites par des universitaires ou un particulier, ou plus encore d'une extraction de données en temps réel, la finalité est surtout d'avoir ces données à notre disposition afin de réaliser l'expérimentation que nous souhaitons faire. Bien évidemment ces données doivent répondre au contexte de notre travail qui est ici de faire de l'analyse de sentiments par rapport à la crise mondiale du Covid-19, crise qui ralentit notre quotidien et qui modifie nos habitudes.

Ainsi, les données que nous avons utilisées peu importe leur mode d'acquisition se rapportent au sujet de la crise du Covid. En parlant le mode d'acquisition des données, nous avons optés pour deux modes:

- le mode asynchrone

- le mode synchrone

4.1.1 Le mode asynchrone

Lorsque nous parlons de mode asynchrone, c'est juste pour dire que les données utilisées dans ces situations sont pour la plupart des fichiers csv qui contiennent des tweets récupérés sur le sujet du Covid que nous avons trouvés en accès libres en ligne. Ces deux sources sont les suivantes.

Source 1 : For Academics - Sentiment140 - A Twitter Sentiment Analysis Tool

Cette base de données est née d'un projet de recherche de Stanford, et chaque enregistrement du dataset est composé de 6 attributs qui sont :

1. La polarité du tweet (0 = negative, 2 = neutral, 4 = positive)
2. l'id du tweet
3. la date du tweet
4. the query (lyx). If there is no query, then this value is NO_QUERY.
5. l'utilisateur ayant twitté
6. le texte du tweey

Ce dataset est disponible sur le lien suivant Dataset 1 Bien que ne contenant pas de tweets sur le sujet du covid, cette base de données nous a permis de tester notre algorithme et de pouvoir voir l'efficacité de nos algorithmes. Aussi, l'autre aspect qu'elle présente est le fait que les données soient déjà labellisées en termes de sentiments(voir l'attribut "polarité")

Source 2 : Dataset de tweets sur le Covid 19

Le second dataset utilisé en mode asynchrone que nous allons vous présenter est également disponible en ligne et est le fruit de **Ilyes Talbi** qui a réalisé un travail similaire sur le sujet et notamment sur le Covid 19 pendant la période du premier confinement (période de Janvier 2020 à Avril 2020).Soit un ensemble de 13000 enregistrements(lignes).

En terme de contenu, chaque tuple (ligne) contient un ensemble de 21 attributs dont les informations comme :

- la date du tweet
- son id

- les hashtags utilisés dans le tweet par le twitos
- le nom d'utilisateur du twitos
- le nombre de likes
- le nombre de retweets
- et surtout le text du tweet qui fait partie des attributs qui nous interrese plus dans notre travail.

Comme vous l'aurez constaté la polarité des tweets ne fait pas partie des attributs de ce dataset ce n'est pas un problème car cette polarité peut être construite au moyen d'une fonction toute simple dont nous parlerons au niveau du mode suivant et qui permettra d'avoir un dataset beaucoup plus complet pour notre travail.

Le dataset présenté est disponible sur le lien Dataset 2 et un extrait de son contenu est sur les figures suivantes(un échantillon présentant les dix premières lignes).

	id	conversation_id	created_at	date	timezone	place	tweet	hashtags	cashtags	user_id
0	1226838216705290	1226800492258512	1581335999000	2020-02-10 11:59:59	UTC		Jcrois il repousse le	[]	[]	7149930338849095
1	1226838210346733	1226543995268063	1581335997000	2020-02-10 11:59:57	UTC		Et on nous dit de lim	[]	[]	1210351232331796
2	1226838134744326	1226838134744326	1581335979000	2020-02-10 11:59:39	UTC		□ ALERTE - Le Roya	['#coronavirus', '#con	[]	1214315619031478
3	1226838134144540	1226838134144540	1581335979000	2020-02-10 11:59:39	UTC		Ptdrr ils nettoyent sa	['#contaminesmontjc	[]	362050772
4	1226838094747512	1226838094747512	1581335969000	2020-02-10 11:59:29	UTC		Coronavirus : le bilar	[]	[]	378817170
5	1226838073595613	1226836890407636	1581335964000	2020-02-10 11:59:24	UTC		tout à fait d'accord a le pire c'est que cert	[]	[]	2162947538
6	1226838041999953	1226838041999953	1581335957000	2020-02-10 11:59:17	UTC		On est lundi, la réun on sent l'urgence . #CORONAVIRUSENI #CORONAVIRUS . https://www.lefigaro.	['#coronavirusenfranc	[]	57597804
7	1226838041853145	1226838041853145	1581335957000	2020-02-10 11:59:17	UTC		Football : La mauvai	[]	[]	85862615
8	1226837978896633	1226837978896633	1581335942000	2020-02-10 11:59:02	UTC		Une équipe de l'OM	['#senegal', '#kebetu'	[]	3918185062
9	1226837972131221	1226837972131221	1581335940000	2020-02-10 11:59:00	UTC		Droit aux indemnités	[]	[]	140496388
10	1226837963293777	1226837963293777	1581335938000	2020-02-10 11:58:58	UTC		Les chercheurs de U	['#coronavirus', '#con	[]	4277657452

Figure 4.1: DataSet 2-1 : Dix premières lignes

user_id	user_id_str	username	name	day	hour	link	retweet	nlikes	nreplies	nretweets	quote_url
7149930338849095	7149930338849095	SItcLoucas	ChercheUnTn	1	11	https://twitter.com/SI	FALSE	1	0	0	
1210351232331796	1210351232331796	izaviale	Izaviale	1	11	https://twitter.com/iza	FALSE	0	0	0	
1214315619031478	1214315619031478	Conflits_FR	Conflits	1	11	https://twitter.com/Cc	FALSE	249	11	132	
362050772	362050772	Elblaugrana10	♏ M6SSI □ ** 🅳 2 🅴	1	11	https://twitter.com/El	FALSE	2	0	1	
378817170	378817170	EdithWenke	La Serva Padrona	1	11	https://twitter.com/Ec	FALSE	0	0	0	
2162947538	2162947538	jelogui83	INDEPENDANT	1	11	https://twitter.com/jel	FALSE	0	0	0	
57597804	57597804	bolassiette	alain bolassiette	1	11	https://twitter.com/bo	FALSE	8	1	4	
85862615	85862615	Sport24Team	Sport24 **	1	11	https://twitter.com/Sp	FALSE	0	0	0	
3918185062	3918185062	senegal7com	senegal7	1	11	https://twitter.com/se	FALSE	0	0	0	
140496388	140496388	legisocial	LégiSocial	1	11	https://twitter.com/le	FALSE	0	0	0	

Figure 4.2: DataSet 2-2 : Dix premières lignes - autres attributs

4.1.2 Le mode synchrone

Lorsque nous parlons d'acquisition de données en mode synchrone, nous parlons surtout d'extraction de tweets en temps réel.

En effet, comme nous l'avons évoqué dans le chapitre précédent, il existe de nombreuses manières de faire de l'extraction de tweets en temps réel dans le cadre d'un projet de Big Data comme celui-ci. Nous avons testé l'utilisation des packages mais nous n'avons pas été satisfait. Et l'option finale pour laquelle nous avons opté est l'utilisation de l'API de Twitter pour faire de l'extraction de données. Les différentes étapes que nous avons suivies pour avoir les données en temps réel peuvent être décrites comme suit.

Etape 1 : Inscription sur Twitter et demande de l'API

Pour utiliser l'API Twitter, il est important de disposer d'un compte Twitter et de le faire passer à un compte développeur afin de faire la demande de l'API. Ce que nous avons fait et qui nous a permis de disposer des accès. Cette demande prend généralement une journée après vérification des intentions pour lesquelles nous demandons l'accès.

Vous pouvez voir sur la figure suivante que nos accès nous ont été octroyés et sont utilisables.

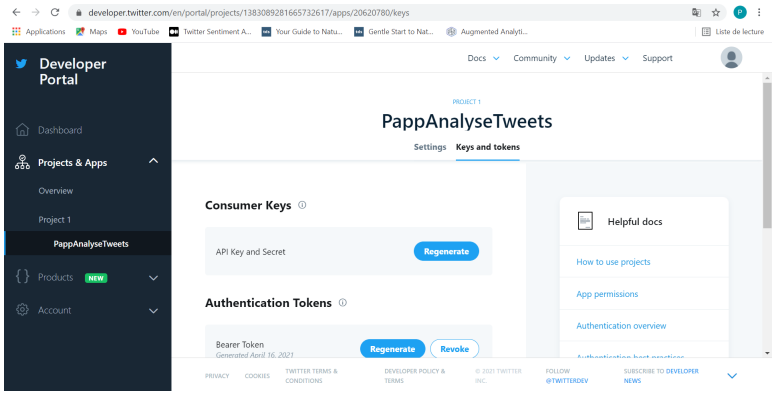


Figure 4.3: Capture du compte développeur créé pour ce projet

Etape 2 : Mise en place d'une fonction pour l'extraction de tweets

Après avoir obtenu les différents accès nous permettant de faire de l'extraction de données sur notre contexte, il faut maintenant passer à l'extraction en elle-même. Et pour cela nous avons mis

en place une fonction qui s'en occupe.

Ainsi pour avoir les données, il faut préciser :

- le sujet sur lequel nous voulons des données, ici il s'agit bien évidemment du Covid
- la langue dans laquelle nous souhaitons avoir les tweets(ici le français)
- la date depuis laquelle nous souhaitons commencer l'extraction,
- et nous avons aussi la possibilité de limiter le nombre de tweets que nous voulons récupérer.

La figure suivante montre le code pour faire de l'extraction, nous avons caché les identifiants API de Twitter sur la photo. Ils sont néanmoins lisibles dans le code fourni.

```
def get_tweets():
    """Récupération des tweets en ligne en utilisant l'api de twitter
        Demande faite à Twitter pour avoir un compte développeur
        Ce qui permet d'avoir les clés d'accès suivantes
        - consumer_key
        - consumer_secret
        - access_key
        - access_secret
    """
    import tweepy as tw
    consumer_key ="
    consumer_secret
    access_key ="1
    access_secret=
    "Gestion de l'authentification"
    auth = tw.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key,access_secret)
    api = tw.API(auth, wait_on_rate_limit=True)

    requete_tweets ="Covid-19 OR Covid OR Corona OR Pandémie OR épidémie OR Coronavirus OR virus"
    tweets = tw.Cursor(api.search,
                   q = requete_tweets,
                   lang = "fr",
                   since='2021-03-01').items(10)

    all_tweets = [tweet.text for tweet in tweets]
    return all_tweets
```

Figure 4.4: Fonction pour récupérer les tweets

Etape 3 : Ajout de l'attribut de polarité

Pour finir avec les données, il faut ajouter un attribut de polarité à ces derniers afin d'avoir un label sur lequel on pourra faire exécuter nos algorithmes. Pour cela, il suffit de faire usage du package **Textblob** qui permet de pouvoir ressortir la polarité d'un texte dans un intervalle compris entre **[-1, 1]** sachant qu'un -1 équivaut à négatif et 1 à positif comme sentiment.

Pour cela, nous avons des fonctions nous permettant de définir la polarité de chaque tweet extrait. Vous pouvez consulter ces fonctions sur les figures suivantes.

```
def calcul_polarite(corpus_clean):
    polarity = []
    for tweet in corpus_clean:
      polarity.append(TextBlob(tweet,pos_tagger=PatternTagger(),analyzer=PatternAnalyzer()).sentiment[0])

    return polarity
```

Figure 4.5: Fonction pour calculer la polarité

```
def polarite_textuelle(polarite_chiffree):
    polarite_data = []
    for polarite in polarite_chiffree:
        if polarite in (0,0.5):
            text ="Neutre"
            polarite_data.append(text)
        if polarite <0:
            text ="Negatif"
            polarite_data.append(text)
        if polarite >0.5:
            text ="Positif"
            polarite_data.append(text)
    print("Définition des polarités textuelles faites")
    return polarite_data
```

Figure 4.6: Fonction pour convertir la polarité en texte

```
def polarite_en_chiffre_Entier(polarite_chiffree):
    polarite_data = []
    for polarite in polarite_chiffree:
        if polarite in (0,0.5):
            text = 0
            polarite_data.append(text)
        if polarite <0:
            text = -1
            polarite_data.append(text)
        if polarite >0.5:
            text = 1
            polarite_data.append(text)
    return polarite_data
```

Figure 4.7: Fonction pour convertir la polarité en entier

La première fonction 4.5 permet de calucler la polarité d'un texte. La deuxième fonction 4.6 permet de convertir cette polarité en champ de texte suivant des intervalles qu'on a défini et la trosième fonction 4.8 permet de convertir cette polarité comme la deuxième mais cette fois en entier soit -1 pour négatif , 0 pour neutre et 1 pour positif.

Etape 4 : Construction finale des données avec les attributs Tweet et Polarité.

Il s'agit de l'étape finale pour les données.Le but est de regrouper les tweets et leur polarité dans un dataframe qui peut être enregistré en fichier CSV dans le besoin. La construction de ce dataframe se fait grâce à la méthode suivante:

```
def create_tweets_df(tweets, polarite):
    tuple_ligne = list(zip(tweets,polarite))
    dataFrame = pd.DataFrame(tuple_ligne, columns= ['Tweets','Label'])
    print("Dataset de travail créé")
    return dataFrame
```

Figure 4.8: Fonction pour créer le dataFrame de travail

Vous pouvez pas la suite voir un échantillon de données extraits de twitter sur le sujet "Covid-19" et dont les tweets ont été postés au début de Mars 2021

Dataset de travail créé

	Tweets	Label
0	rt vaccination obligatoire des soignants gri...	Negatif
1	rt mazagan_ft comme l'a annoncé le directeur d...	Neutre
2	jvais caner laura a le covid jsuis dans sa cha...	Neutre
3	rt patrickademo note éliminatoire parce que ma...	Negatif
4	rt agretdenis le virus est sur toute la planè...	Negatif
5	rt enmodemacaron rt si tu demandes la démissio...	Negatif
6	rt agretdenis non à la vaccination obligatoir...	Negatif
7	tests covid les prélèvements nasopharyngés pe...	Negatif
8	entre culpabilité et reproches de l'entourage ...	Negatif

Figure 4.9: Lecture dataFrame en console

,Tweets,Label
0,rt vaccination obligatoire des soignants grippeâœ... covidâŒ c'est pas pareilâ€¦ nous avons beaucoup plus de recul sur lesâ€¦,Negatif
1,rt mazagan_ft comme l'a annoncÃ© le directeur de cnn piÃ©gÃ© l'urgence climatique va Ãªtre la prochaine psychose aprÃ¨s la planification deâ€¦,Neutre
2,jvais caner laura a le covid jsuis dans sa chambre,Neutre
3,rt patrickademo note Ã©liminatoire parce que malade du covid ah ouais Ã§a se cache mÃªme plus pour pisser sur la gueule des Ã©tudiants câ€™esâ€¦,Negatif
4,rt agretdenis le virus est sur toute la planÃ¨te et trÃ¨s peu dangereux. le vaccin ne protÃ¨ge de rien et n'immunise pas.,Negatif
5,rt enmodemacaron rt si tu demandes la dÃ©mission de vÃ©ran cet individu mÃ©prisant qui prend les franÃ§ais pour des cons et refuse les traitâ€¦,Negatif
6,rt agretdenis non Ã la vaccination obligatoire le vaccin ne passera par nos enfants car il y a un risque de dÃ©cÃ¨s avec ces thÃ©rapieâ€¦,Negatif
7,tests covid les prÃ©lÃ¨vements nasopharyngÃ©s peuventils Ãªtre dangereux httpst.co via le_figaro,Negatif
8,entre culpabilitÃ© et reproches de l'entourage comment vivre avec le poids d'avoir contaminÃ© quelqu'un de cher â€¦ httpst.co,Negatif

Figure 4.10: Lecture en excel

Vous remarquerez que les tweets sont dans un français pas tout le temps lisible et ne sont pas épurés ou nettoyés sur les images montrées. Bien évidemment, ils ont finalement été nettoyés avant d'être passés aux algos. Nous en avons parlé plus haut.

Vous povez voir sur la figure suivante l'ensemble des différentes étapes d'extraction de données afin d'avoir des dataframes prêts à l'emploi pour nos différents algorithmes.

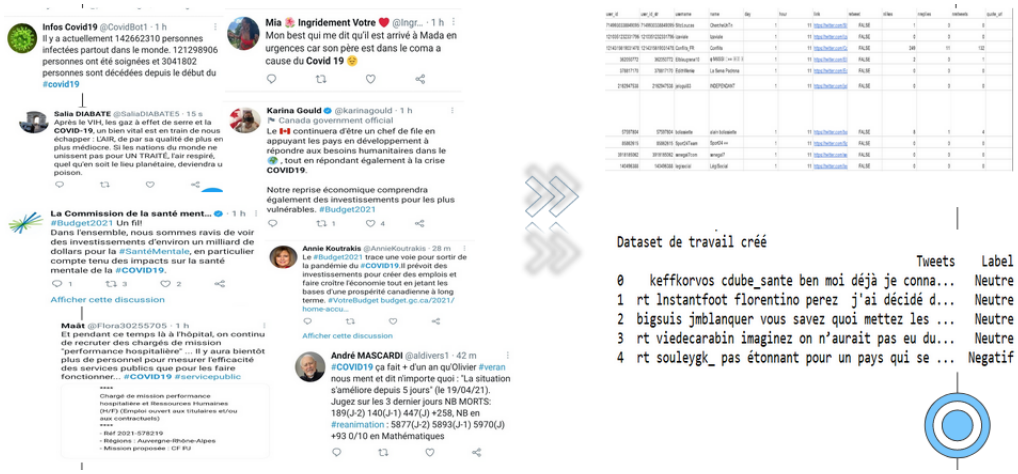


Figure 4.11: Résumé des étapes

4.2 Mise en place de l'expérimentation

Nous parlerons des dispositions matérielles dans lesquelles nous avons exécuté nos différentes implémentations.

4.2.1 La partie PySpark

Dans cette partie, nous avons décidé d'utiliser la plateforme Databricks. Databricks est une société de logiciels d'entreprise fondée par les créateurs originaux d'Apache Spark. Databricks nous a permis d'utiliser un cluster de 8 cœurs.

4.2.2 La partie NLTK

Le code qui a été construit en se basant sur la librairie NLTK a été exécuté sur une machine personnelle qui fonctionne sur Windows 10.

Chapter 5

Résultats des tests effectués

Dans cette partie, nous présenterons les différents résultats de nos expérimentations et nous finirons en faisant une comparaison des deux méthodes que nous avons utilisées pour faire ce travail.

5.1 Projection sur l'ensemble des données

Avant de passer à la présentation des résultats, nous vous présentons quelques tendances qui peuvent être observées sur les données recupérées.

La première tendance que nous pouvons voir est que la taille du tweets (en terme de mots) n'influence pas vraiment sa polarité; un tweet peut être négatif juste avec un seul ou positif également comme il peut également l'être en un paragraphe entier.

La deuxième comme vous pouvez le voir sur la figure suivante est la polorité que peuvent avoir des tweets sur le sujet du Coronavirus. On peut voir que sur les deux échantillons présentés, il n'y a pas vraiment de sentiments positifs qui se dégagent des propos de personnes sur Twitter par rapport à cette crise mondiale et par rapport à sa gestion par les autorités gouvernemantales.

La troisième tendance que nous montrons sur la figure qui suit est le fait que les données ne soient pas correctement réparties entre le trois classes (négatif, neutre, positif). Ce qui peut biaiser l'apprentissage de l'algorithme.

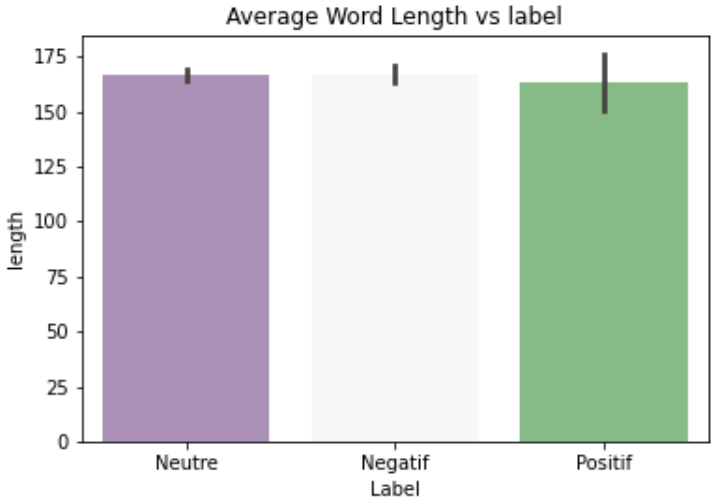


Figure 5.1: Taille des tweets vs polarité

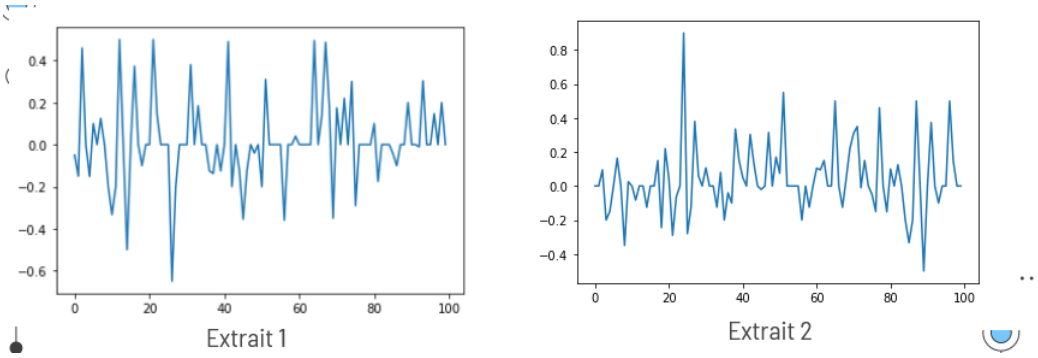


Figure 5.2: Polarité des tweets

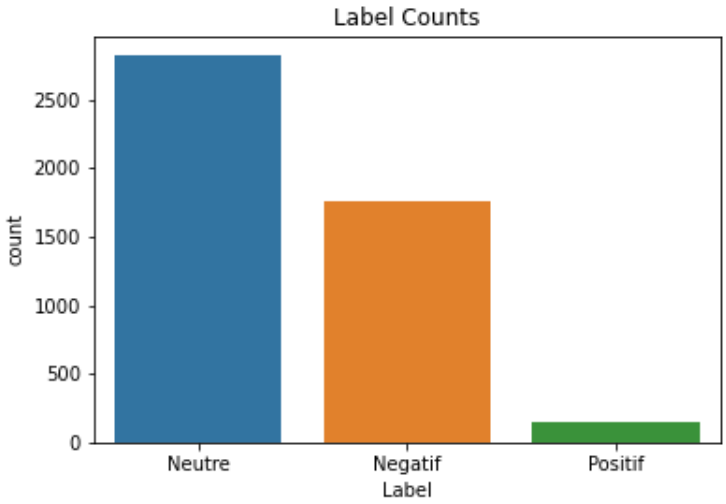


Figure 5.3: Répartition des tweets

5.2 La partie Pyspark

Dans cette partie, nous présenterons les différents résultats de nos expérimentations de l'approche Pyspark. Nous allons détailler les résultats de la méthode TF-IDF + NaiveBayes citée ci-dessus.

- **Sensitivity** = 0.80
- **Specificity** = 0.70
- **PPV** = 0.80
- **NPV** = 0.70
- **Accuracy** = 0.75
- **Temps de classification** 92.938 secondes

Afin d'évaluer notre modèle de classification, nous nous sommes intéressés à plusieurs critères. Premièrement, la sensibilité qui veut dire le pourcentage de vrais positifs. On l'a trouvé égal à 0.8. Deuxièmement, La spécificité qui est le pourcentage de vrais négatifs ,égale à 0.7. Troisièmement, PPV (la valeur prédictive positive) qui mesure la probabilité qu'après un résultat de test positif, cet individu ait vraiment positif. Quatrièmement, NPV (la valeur prédictive négative). Ce critère mesure la probabilité qu'après un résultat de test négatif, cet individu est vraiment négatif. Dernièrement, l'accuracy est à quel point notre modèle est proche de la réalité.

$$Sensitivity = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$specificity = \frac{TrueNegative}{TrueNegative + FalsePositive}$$

$$PPV = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$NPV = \frac{TrueNegative}{TrueNegative + FalseNegative}$$

$$Accuracy = \frac{sensitivity + specificity}{2}$$

On a aussi testé l'algorithme qui utilise la méthode Count Vectorizer + Logistic Regression et on a trouvé qu'il n'y a pas de différence entre les deux méthodes au niveau des résultats et on a trouvé l'accuracy égale à **0.766**.

5.3 La partie NLTK

Les résultats présentés sur cette partie sont ceux obtenus sur la base de récupération en temps réel d'environ 3000 tweets à compter de la période de Mars 2021. Ces résultats sont observables

sur la figure suivante:

```
Temps d'exécution 24.441
              precision    recall  f1-score   support

          -1       0.53      0.81      0.64        90
           0       0.95      0.81      0.88       384
           1       0.00      0.00      0.00         0

    accuracy                           0.81       474
   macro avg       0.49      0.54      0.51       474
weighted avg       0.87      0.81      0.83       474


 Matrice de confusion
[[ 73  16   1]
 [ 64 312   8]
 [  0   0   0]]
```

Figure 5.4: Résultats NLTK

Sur cette figure vous pouvez voir que le temps d'exécution est de 24.44 secondes ce qui peut être dû au fait qu'on avait plusoeurs tâches de lancer au cours son éxecution. Ce qui nous intérresse le plus est bien sûr la précision de l'algorithme sur l'ensemble d'entraînement qu'on lui a fourni. cette précision est de **0.81** sur léchantillon de tests fourni. Cependant, il est loin du standard des 100% pour éviter toute erreur mais en appliquant de nouvelles méthodes de prétraitement sur l'ensemble des données on peut avoir de meilleurs résultats d'après nos recherches.

5.4 Conclusion :Comparaison des deux méthodes utilisées

Dans ce qui précède, nous avons utilisé deux méthodes différentes pour analyser et classifier les sentiments des tweets. La première utilise l'outil Pyspark et la deuxième utilise la bibliothèque NLTK.

La première méthode nous a permis d'avoir une précision de classification égale à 81% alors que la deuxième ne dépasse pas les 76% en utilisant les deux techniques (TF+IDF + NaiveBayes et Count Vectorizer + Logistic Regression). Cependant, la quantité de données traitée dans les deux méthodes est largement différente ce qui explique un peu la différence entre les deux précisions. Pour finir, la première méthode est plus intéressante dans le contexte de Big Data, puisqu' elle utilise la distribution des données et le temps d'apprentissage est beaucoup plus favorable.

Chapter 6

Conclusion générale

Dans ce travail qui a été très intéressant à mettre en place, nous avons pu aborder le domaine de l'analyse de sentiments en utilisant deux librairies : PySpark et NLTK. Cette analyse de sentiments opérée consistait (à partir de données récupérées sur Twitter qui étaient à la crise mondiale actuelle) à étudier les appréhensions des personnes sur le **Covid-19**.

En terme de résultats de performance, nous avons eu des résultats intéressants pour une première approche dans le domaine de l'analyse de sentiment et ceci peu importe la plateforme (PySpark ou NLTK) même si ces résulats ne sont pas identiques car chaque environnement à ses avantages.

Pour ce qui est des sentiments des personnes par rapport à cette crise mondiale qui nous prive de liberté depuis Décembre 2019, les sentiments sont divers et peuvent être regroupés en trois catégories de personnes. Nous avons dans la première catégorie, les personnes avec beaucoup de colère et de fatigue car dépitées par la situation actuelle et par toutes les mesures restrictives : sentiment négatif. Il y a la catégorie des neutres qui regroupe les personnes qui continuent à prendre à la légère la crise ou qui se font une fortune en profitant des restrictions. La troisième catégorie de personnes regroupent majoritairement les autorités gouvernementales qui apportent des ondes positives en promettant une amélioration de la situation. Et nous espérons profondément que tel sera le cas d'ici les semaines prochaines.

6.1 Annexe

6.1.1 Annexe : fichiers joints , documents et code

Vous verrez dans le même répertoire que ce fichier :

- les codes source python que nous avons utilsé dans les deux cas (PySpark, NLTK)
- les échantillons des datasets utilisés
- l'article de synthèse
- la présentation de notre travail (powerpoint)

Autres sources : [11], [10], [12], [1], [2], [4], [9], [3] [5],[8],[7],[6]

References

[1] Mohamed Afham. "Twitter Sentiment Analysis using NLTK, Python". In: *Towardsdata-science*. Consulté du 20 Mars au 27 Avril 2021. URL: https://towardsdatascience.com/twitter-sentiment-analysis-classification-using-nltk-python-fa912578614c.

[2] Chamanthi Aki. "Twitter Sentiment Analysis using NLTK, Python". In: *Medium*. Consulté du 20 Mars au 27 Avril 2021. URL: https://medium.com/@chamschamanthi36/twitter-sentiment-analysis-using-nltk-python-7383e1f2a266.

[3] Harrison. "Twitter Sentiment Analysis with NLTK". In: *Python Programming*. Consulté du 20 Mars au 27 Avril 2021. URL: https://pythonprogramming.net/twitter-sentiment-analysis-nltk-tutorial/.

[4] Nachiketa Hebbar. "Twitter Sentiment Analysis Using Python for Complete Beginners". In: *Medium*. Consulté du 20 Mars au 27 Avril 2021. URL: https://medium.com/swlh/tweet-sentiment-analysis-using-python-for-complete-beginners-4aeb4456040.

[5] Ricky Kim. "Sentiment Analysis with PySpark". In: *towards data science*. 13 Mars 2018. URL: https://towardsdatascience.com/sentiment-analysis-with-pyspark-bc8e83f80c35/.

[6] "MLlib - Feature Extraction and Transformation". In: *Pyspark documentation*. Consulté du 20 Mars au 27 Avril 2021. URL: https://spark.apache.org/docs/1.4.1/mllib-feature-extraction.html/.

[7] Noah Muluberhan-Berhe. "Twitter Sentiment Analysis: Using PySpark to Cluster Members of Congress". In: *Analytics Vidhya*. 6 Mai 2020. URL: https://medium.com/analytics-vidhya/congressional-tweets-using-sentiment-analysis-to-cluster-members-of-congress-in-pyspark-10afa4d1556e/.

[8] "NaiveBayes — PySpark 3.1.1 documentation". In: *Pyspark documentation*. Consulté du 20 Mars au 27 Avril 2021. URL: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.classification.NaiveBayes.html#pyspark.ml.classification.NaiveBayes/.

[9] “NLP avec Python : analyse de sentiments sur Twitter”. In: Consulté du 20 Mars au 27 Avril 2021.

[10] “Spark”. In: *Site de Spark*. Consulté du 20 Mars au 27 Avril 2021. URL: `https://spark.apache.org/`.

[11] “StackOverflow pour les erreurs”. In: *Site de Stackoverflow*. Consulté du 20 Février au 15 Mars 20211. URL: `https://stackoverflow.com/`.

[12] Ewan Klein Steven Bird and Edward Loper. “ Analyzing Text with the Natural Language Toolkit”. In: *Natural Language Processing with Python*. Consulté du 20 Mars au 27 Avril 2021. URL: `https://www.nltk.org/book/`.