Sciences mathématique et informatique

Projet Tutoré

Semestre S6

Mémoire

Analyse de sentiments des tweets : Cas de Covid-19

Présenté par :
Aouatif ES SAMERY
Imane EL AMMARI

Encadré par :
Pr : Fatima EL HAOUSSI
Pr : Hanane CHERGUI HALI

Soutenu le 05/07/2021 devant le jury constitué de :

Pr.	Abdellatif EL ABDERRAHMANI	FSDM	Fès
Pr.	Hassan SATORI	FSDM	Fès
Pr.	Fatima EL HAOUSSI	FSDM	Fès
Pr.	Hanane CHERGUI HALI	Ministère de l'éducation national	Fès

Année Universitaire : 2020/2021

Dédicaces

A mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études, A ma chère soeur, pour ses encouragements permanents, et leur soutien moral.

A mes chers frères, pour leur appui et leur encouragement, A toute ma famille pour leur soutien tout au long de mon parcours universitaire.

Que ce travail soit l'accomplissement de vos voeux tant allégués, et le fuit de votre soutien infaillible.

Sans oublier mon binôme IMANE EL AMMARI pour son soutien moral, sa patience et sa compréhension tout au long de ce projet.

Merci d'être toujours là pour moi.

AOUATIF ES-SAMERY

Je dédie ce projet :
A ma chère mère, A mon cher père, Qui n'ont jamais cessé, de formuler des prières à mon égard, de me soutenir et de m'épauler pour que je puisse atteindre mes objectifs.

A mes frères, Pour ses soutiens moral et leurs conseils précieux tout au long de mes études.

A mes chers grand-pères, Qui je souhaite une bonne santé. A ma chère binôme , AOUATIF ES-SAMERY Pour sa entente et sa sympathie. Pour leurs indéfectibles soutiens et leurs patiences infinies. A mes chères ami(e)s Pour leurs aides et supports dans les moments difficiles.

A toute ma famille, A tous ceux que j'aime et ceux qui m'aiment.

IMANE EL AMMARI

Remerciement

Au premier temps, je remercie le bon Dieu, tout puissant, de m'avoir donné la force et l'audace pour dépasser les difficultés.
Je tiens, avant tout, à adresser mes plus vifs remerciements à Madame EL HAOUSSI fatima mon encadrante à l'université FSDM pour l'encadrement sérieux dont elle a fait preuve et sur les meilleures conseils qu'elle nous a donner,Nos vifs remerciements s'adressent aussi à Madame CHERGUI HALI Hanane pour sa patience, sa disponibilité qui ont contribué à alimenter ma réflexion.

Nous souhaitons aussi, présenter toute notre reconnaissance à nos familles et amis respectifs, pour leur soutien de tous les instants, et pour la patience dont ils ont fait preuve tout au long de cette période.
Enfin, j'adresse mes remerciements aux membres du jury pour m'avoir honoré en acceptant d'évaluer ce travail et d'être présents dans de telles conditions.

Resumé

L'analyse des sentiments est une technologie d'analyse automatique des discours, écrits ou parlés dans le but à extraire des informations subjectives comme des jugements, des évaluations ou des émotions pour détecter la polarité d'une opinion.
L'importance de l'analyse des sentiments est présente dans plusieurs domaines, à savoir politique, marketing, gestion de la réputation, etc...
Dans ce mémoire, notre objectif consiste à dévoiler les secrets d'un tel domaine en adoptant une approche pour l'apprentissage automatique. Pour ce faire nous avons implémenté les méthodes de machine Learning et deep learning.
Dans ce travail, nous allons utiliser cinq algorithmes qui permettent d'analyser et classifier de tweets pendant la pandémie de Covid-19. Les classes que nous avons définies sont : positive, négative et neutre.

Table des matières

Dédicaces **1**

Remerciement **3**

Resumé **4**

Introduction Générale **10**

1 Etat d'art **12**

1.1 Introduction 12
1.2 Réseaux sociaux 12
1.3 Source de données 13
1.3.1 Site d'avis 13
1.3.2 Blogs 13
1.3.3 Microblogs 13
1.4 Twitter 13
1.4.1 Définitions 14
1.4.2 Caractéristiques d'un tweet 15
1.5 Historique Covid 19 15
1.6 Traitement Automatique du Langage Naturel (NLP) 16
1.7 Analyse de sentiment 17
1.7.1 Définition 17
1.7.2 Analyse de sentiment des tweets 17
1.8 Les approches d'analyse de sentiment basées sur Machine Learning (ML) . 17
1.8.1 Apprentissage supervisé 18
1.8.2 Apprentissage non supervisé 20
1.8.3 Approche lexicale 21
1.8.4 Approches hybride 22
1.9 Les approches d'analyse de sentiment basées sur Deep Learning (DL) . . . 22
1.9.1 Définition de DL 22
1.9.2 Applications de DL 22

1.9.3 Le réseau neuronal 24
1.9.4 Algorithmes du Deep Learning 25
1.9.4.1 Les réseaux de neurones convolutifs (CNN) 25
1.9.4.2 Les réseaux de neurones récurrents (RNN) 25
1.10 Conclusion 28

2 Extraction et prétraitement des données 29
2.1 Introduction 29
2.2 Environnement de Travail 29
2.2.1 Environnement matèriel 29
2.2.2 Environnement logiciel 30
2.3 Extraction des données de Twitter 30
2.3.1 Interface de programmation d'applications (API) de Twitter 31
2.3.2 Tweepy 32
2.3.3 Twint 32
2.4 Nettoyage des tweets 34
2.4.1 Le principe de fonctionnement 34
2.4.2 Regex et NeatText 34
2.4.3 Statistiques et visualisation des résultats 36
2.5 L'analyse de sentiments avec TextBlob 37
2.5.1 Qu'est-ce que TextBlob ? 37
2.5.2 Statistiques et visualisation des résultats 39
2.6 Tokenisation des tweets 40
2.7 Nuage des mots (WordCloud) 45
2.8 Conclusion 46

3 Classification des données 49
3.1 Introduction 49
3.2 La classification avec Machine Learning 49
3.2.1 Les algorithmes d'apprentissage supervisé avec SVM et NB 49
3.2.1.1 Préliminaires 50
3.2.1.2 Algorithme de classification naive bayes (NB) 52
3.2.1.3 Algorithme de classification SVM 53
3.2.2 Les algorithmes d'apprentissage non supervisé avec K-means . . . 54
3.2.2.1 PCA et TSNE 55
3.2.2.2 Conversion des données 56
3.2.2.3 Visualisation des résultats 57
3.3 La classification avec Deep Learning 61
3.3.1 Présentation des bibliothèques utilisées en DL 61

3.3.1.1 TensorFlow 61
3.3.1.2 Keras 61
3.3.1.3 Word Embeddings 62
3.3.2 Le classificateur RNN 62
3.3.3 Le classificateur CNN 65
3.4 Conclusion 68

Conclusion Générale et Perspictives **69**

Bibliographies **70**

Table des figures

1.1 Compte Twitter. 14
1.2 Techniques d'apprentissage. 18
1.3 Modele de Naive Bayes. 19
1.4 Le principe de fonctionnement de SVM. 20
1.5 Modele de Machine à vecteurs de support. 21
1.6 Relation entre les trois concepts [27]. 23
1.7 Un neurone réel. 24
1.8 Un neurone artificiel. 24
1.9 Traitement de l'image par CNN. 26
1.10 Traitement de l'image par CNN. 26
1.11 Un exemple d'un réseau récurrent. 27
1.12 Exemple de l'activation et la sortie dans RNN. 28

2.1 Distribution des tweets par mois. 37
2.2 Fréquence des tweets par mois. 38
2.3 Polarité et subjectivité en fontion des sentiments. 39
2.4 Densité de polarité et de subjectivité. 40
2.5 Frequence de polarité des tweets. 41
2.6 Pourcentage des sentiments. 41
2.7 Tokenisation des tweets. 43
2.8 Score des tokens négatifs 44
2.9 Score des tokens positifs 44
2.10 Score des tokens neutres 45
2.11 Icône de twitter avec nuage des mots 46
2.12 Nuage des mots positifs 47
2.13 Nuage des mots négatifs 47
2.14 Nuage des mots neutres 48

3.1 Résulat de vectorisation. 52
3.2 Résulat d'exécution de NB. 54

3.3 Résulat d'exécution de SVM. 55
3.4 Exemple explicatif de CountVectorizer. 57
3.5 Matrice obtenu par CountVectorizer. 57
3.6 Matrice obtenu par CountVectorizer. 59
3.7 Nuage des points du PCA et TSNE. 60
3.8 Le nombre des clusters par Elbow. 60
3.9 Réseau de neurones récurrent simple. 63
3.10 L'architecture du RNN. 64
3.11 Précision et erreur du modèle RNN. 65
3.12 L'architecture du CNN. 67
3.13 Précision et erreur du modèle CNN. 68

Introduction Générale

De nos jours, Internet est devenu plus que jamais un outil essentiel et puissant pour l'échange d'informations dans le monde entier. Elle nous offre une quantité considérable de données, et ses services s'adaptent de plus en plus aux besoins des internautes. Ils peuvent utiliser Internet pour acheter des produits, trouver des lieux et des informations, partager leurs sentiments et leurs opinions, envoyer des e-mails, lire des journaux en ligne, etc...
Au cours des dernières années, un grand nombre de personnes ont été attirées par les plateformes de réseau social comme Facebook, Twitter et Instagram. La plupart utilisent des réseaux sociaux pour exprimer leurs émotions, leurs croyances ou leurs opinions sur des choses, des lieux ou des personnalités.

La popularité des médias sociaux est liée à la demande d'information qui est devenue plus importante dans notre société, car nous savons que ce genre de sites rendent l'information circule très rapidement et disponible partout dans le monde, donc l'accès à tout type de données et l'absence d'information ne sont plus un problème, et c'est pourquoi les gens utilisent de plus en plus les réseaux sociaux chaque jour. En général, nous constatons que les gens aiment consulter les interactions et les réactions des autres, comme les opinions par exemple, avant de prendre une décision ou d'exprimer leur propre opinion, même s'ils connaissent ces personnes ou non. De ce fait, l'information est devenue plus précieuse que jamais, de nombreux domaines sont de plus en plus dominés par la circulation de l'information sur Internet et plus particulièrement sur les réseaux sociaux comme l'économie, la politique, le sport, etc...
Actuellement, l'un des meilleurs exemples de réseaux sociaux permettant d'observer l'évolution de ces opinions est Twitter[35], du fait de la concision et de la structure des messages publics partagés sur cette plate-forme.
Twitter qui nous intéresse particulièrement dans ce travail, constitue une source continue et illimitée de données en langage naturel qui est particulièrement difficile à traiter avec les approches classiques de traitement automatique du langage naturel (TALN). Ce type de langage est très éloigné des normes du langage traditionnel, avec ses conventions (telles que hashtags, les mentions, les retweet, etc...). Son lexique particulier est souvent grossier et contient des abréviations, des émoticons, des acronymes. Sa syntaxe est parcellaire dans le meilleur des

cas. Pour cette raison, il est utile d'analyser son contenu car il a acquis une grande popularité dans le monde entier.
L'analyse des sentiments [32] (également appelée analyse des opinions) fait référence à l'utilisation du traitement du langage naturel (NLP) et de l'apprentissage automatique(Machine Learning) pour identifier et caractériser les états affectifs et les opinions à partir de textes ou de données vocales.
Ce procédé apparait au début des années 2000 et connait un succès grandissant dû à l'abondance de données provenant de réseaux sociaux, notamment celles fournies par Twitter. Lorsqu'elle est basée sur la fouille textuelle, l'analyse des sentiments peut porter sur des verbatims issus des réseaux sociaux, des avis, forums, de données d'études qualitatives ou quantitatives, etc...
Les sentiments sont généralement classés en trois types : négatifs, neutres et positifs.
Dans l'apprentissage automatique(Machine Learning), un défi fondamental est l'analyse des données pour identifier les sentiments à l'aide d'algorithmes . Twitter est une source d'informations précieux, principalement utilisée pour exprimer des points de vue et des pensées personnels. Sur la base de ces connaissances, nous proposons une analyse des sentiments des tweets anglais pendant la pandémie COVID-19 en 2020.

Dans ce projet, notre objectif est d'explorer le domaine de l'analyse du sentiment sur Twitter, nous sommes particulièrement concentrés sur les deux phases de prétraitement et de classification des tweets de données . Nous avons également l'intention de présenter l'approche Machine Learning et Deep Learning, et leurs applications en tant qu'une contribution à la problématique de l'analyse du sentiment.
On va s'intéresser par la classification du contenu textuel qui possède des informations sentimentales, donc la problématique de notre travail consiste à :

- Extraction les tweets durant une année.
- Nettoyage des tweets.
- Calcul la polarité et la subjectivité des tweets.
- Classification à l'aide des différentes algorithmes d'apprentissage automatique (machine Learning) et d'apprentissage profond (Deep Learning).

Ce mémoire est structuré en trois chapitres comme suit :
Le premier chapitre introduit des concepts de base pour l'analyse des sentiments puis décrit les différentes définitions des algorithmes utilisés.
Le seconde chapitre sera une description des étapes de prétraitement des tweets, et quelque statistiques de comparaison.
Le troisième chapitre donnera une représentation des outils utilisés et les résultats obtenus dans nos expérimentations par l'utilisation de Machine Learning et Deep Learning.

Chapitre 1

Etat d'art

1.1 Introduction

L'analyse des sentiments c'est le domaine d'étude qui analyse les opinions et les sentiments, les attitudes et les émotions des gens. C'est l'un des domaines de recherche les plus actifs dans le traitement du langage naturel (NLP). Dans ce chapitre, on va définir quelques concepts de base importants utilisés.

1.2 Réseaux sociaux

Dans le domaine des technologies, un réseau social consiste en un service permettant de regrouper diverses personnes afin de créer un échange sur un sujet particulier ou non. En quelque sorte, le réseau social trouve ses origines dans les forums, groupes de discussion et salons de chat introduits dès les premières heures d'Internet. Depuis le début des années 2000, la présence des réseaux sociaux, également appelés réseaux communautaires, devient de plus en plus importante et tend à se multiplier selon diverses caractéristiques. Les réseaux sociaux font désormais partie intégrante du quotidien des gens. Ils peuvent être utilisés à diverses fins, notamment la publicité, la diffusion d'opinions politiques,l'obtention des commentaires des utilisateurs sur les produits et la diffusion d'informations. Ils créent des liens virtuels entre les utilisateurs, dans lesquels les personnes s'expriment et développent des relations à travers des publications, des commentaires, des messages, etc...
Les médias sociaux permettent aux gens de partager leurs pensées, leurs sentiments et leurs opinions avec d'autres personnes instantanément et facilement. Selon cette vision, l'expression médias sociaux regroupe sous la même enseigne une grande variété de dispositifs, tels les blogs, les wiki, les sites réseaux socio-numérique (Facebook,LinkedIn), les microblogues (Twitter, Jaiku), le bookmarking collectif, les partage de contenus médiatique comme la musique, les photos et les vidéos [24].

1.3 Source de données

Les opinions des utilisateurs présentent le critère principal pour l'amélioration de la qualité des services fournis et la mise en valeur des produits livrés. Ces opinions se présentent sous différentes sources de données, à savoir, sites d'avis, blog et micro-blogs.

1.3.1 Site d'avis

Un avis client ou consommateur est un ensemble d'éléments d'appréciation (commentaires, notes, étoiles,..) porté sur un produit, service ou vendeur sur un site Internet ou dans d'autres contextes. Les avis clients existaient et existent évidemment en dehors de leur affichage sur Internet, mais ils sont alors moins structurés et visibles. Même si les consommateurs savent qu'il existe de faux avis clients, l'impact des avis consommateur sur le comportement d'achat et le taux de transformation est indéniable et se vérifie dans de nombreuses études. Il est par exemple particulièrement fort dans le domaine de l'hôtellerie et de la restauration.

1.3.2 Blogs

Un blog est un endroit où les personnes peuvent s'exprimer et partager avec d'autres personnes leurs avis et leurs expériences sur le même site. La simplicité de la création des postes blogs ainsi que leurs formes libres a donné accès libre à tout le monde. La blogosphère est un nom associé à l'univers de tous les blogs. Sur la blogosphère, nous trouvons un nombre important de messages relatif à une panoplie de sujets d'intérêt. Les blogs sont utilisés comme sources d'opinions dans la plupart des études relatives à l'analyse des sentiments [5, 25].

1.3.3 Microblogs

Le microblogging représente toutes les actions en ligne liées à des contenus courts, et ce, surtout sur les réseaux sociaux. La plateforme de microblogging la plus connue est Twitter puisque ses messages ne doivent pas dépasser 140 caractères. La majorit e des travaux, qui s'intéressent à l'analyse de contenu dans le web, utilisent les outils des réseaux sociaux qui offrent le service de microblogage, les données trouvées dans ces outils étant par défaut publiques.

1.4 Twitter

En Mars 2006, Twitter a vu le jour par le développeur Jack Dorsey, comme outil de communication, pour rester en contact avec les amis. Twitter est un service Web qui permet aux

FIGURE 1.1 – Compte Twitter.

utilisateurs d'envoyer et de lire un message court [16].

1.4.1 Définitions

Twitter est un réseau social et un microblog qui permet aux utilisateurs de publier des messages en temps réel, appelés tweets. Les tweets sont des messages courts, limités à 280 caractères. En raison de la nature de ce service de microblogging (messages rapides et courts), les gens utilisent des acronymes, commettent des erreurs d'orthographe, utilisent des émoticônes et d'autres symboles qui expriment des significations particulières [4].
Twitter est actuellement l'une des plates-formes de micro-blogage les plus populaires. Son premier slogan était : "Que faites-vous ?" néanmoins, son utilisation a pris une autre piste où les utilisateurs s'échangent des avis et des informations, le slogan devient "Quoi de neuf ?". Plusieurs célébrités utilisent Twitter, on y trouve même des chefs d'Etat. Selon les derniers chiffres :

- Twitter a plus que 645 millions d'utilisateurs inscrits.
- 58 millions de tweets envoyés chaque jour.

Dans le cadre de l'analyse des sentiments, la taille minimale du message (280 caractères) formule l'hypothèse que ce dernier ne renferme, à priori, plus d'une seule idée, ce qui facilite l'identification de l'opinion exprimée. Certains tweets apparaissent comme des messages codés à cause de l'usage des hashtags, abréviations en tout genre, argot, et émoticons (voir la figure 1.1).
Les termes à connaître pour bien utiliser Twitter [15] :

1. Followers : les personnes qui vous suivent.
2. Followings : les personnes que vous suivez.
3. Friends : les personnes que vous suivez et qui vous suivent.

4. Twittos : les utilisateurs de Twitter.
5. Tweet : court message.
6. Tweeter : envoyer/poster un message.

1.4.2 Caractéristiques d'un tweet

On peut se sentir un peu perdu dans le vocabulaire utilisé dans les tweets, notamment, à cause du langage et des symboles spécifiques à l'utilisation de Twitter. A quoi sert le @ et ? C'est quoi RT ? Toutes ces abréviations peuvent paraître un peu floues. Dans une perspective de classement, un petit lexique des principaux mots et signes Twitter est présenté [10] :

- Mention@ : se présente sous la forme @NomUtilisateur. Il cible un utilisateur de Twitter dans le tweet posté. Exemple : salut à vous de la part de @Imane et @Aouatif.
- Hashtag : se présente sous la forme mot-clé. Il identifie le mot-clé en question comme important et peut en faire un sujet populaire. Exemple : gouvernement.
- RT (ReTweet) : se présente sous la forme RT Nom Utilisatuer. Il permet de partager le tweet d'un utilisateur. Exemple : RT Hanae Excellente.
- URL (Lien) : se présente sous la forme https :// ou http ://www. Twitter `https://twitter.com/` , permet à l'utilisateur de rejoindre les liens dans son tweet. Exemple : `http://www.fsdmfes.ac.ma/`
- VIA : s'utilise pour mentionner votre source dŠinformation, dans votre tweet. Exemple : Via YouTube, Via Facebook.

1.5 Historique Covid 19

Parmi les infections virales les plus courantes qui affectent les humains figurent les infections respiratoires, qui sont causées par des virus respiratoires humains (RV).
Le type d'infection virale respiratoire le plus connu est la grippe ou "influenza", et chaque année provoque entre 250 000 et 500 000 décès dans le monde, le virus H_1N_1 étant la variante la plus connue [17].
cellules épithéliales des voies respiratoires, parfois imperceptibles, mais dans certains cas mortelles, et peuvent même affecter d'autres mammifères et oiseaux. Il existe plusieurs types de coronavirus, les plus connus sont : le syndrome respiratoire du Moyen-Orient (MERS), le syndrome respiratoire aigu sévère (SRAS) et de nos jours la maladie à coronavirus (COVID-19) [19].
La pandémie COVID-19, également connue sous le nom de pandémie de coronavirus, est un problème mondial grave et permanent dans le monde entier. L'épidémie est apparue pour la première fois en décembre 2019 à Wuhan, en Chine. Cela a été déclaré pandémie par l'Organisation mondiale de la santé le 11 mars 2020.

Le virus COVID-19 a infecté des personnes et a tué des centaines de milliers de personnes aux États-Unis, au Brésil, en Russie, en Inde et dans plusieurs autres pays. Depuis que cette pandémie continue d'affecter des millions de vies, un certain nombre de pays ont recouru à un verrouillage partiel ou complet. Les gens ont utilisé les plateformes de médias sociaux pour partager leurs émotions et leurs opinions pendant ce verrouillage afin de trouver un moyen de se détendre et de se calmer.
Dans ce travail de recherche, une analyse des sentiments sur les tweets de personnes . Sur la base des tweets publiés en anglais, une analyse des sentiments a été effectuée. Cette recherche a été menée pour comprendre comment les personnes infectés font face à la situation. Les tweets ont été collectés, prétraités, puis des algorithmes d'extraction de texte utilisés et enfin une analyse des sentiments a été effectuée et présentée avec les résultats [11].

1.6 Traitement Automatique du Langage Naturel (NLP)

Le Natural Language Processing (NLP) ou Traitement Automatique du Langage Naturel (TALN) est une discipline complexe. Le NLP se situe ainsi à l'intersection de la linguistique, du code, de la statistique et des sciences cognitives. C'est également l'un des champs de l'intelligence artificielle. le NLP commence à se développer dans les années 1950. D'abord sous l'impulsion de chercheurs russes, puis aux États-Unis. À la fin des années 1950, le rapport Bar-Hillel fait état d'obstacles scientifiques. Le NLP est trop ambitieux, on doute que les recherches aboutissent. Les financements dès lors se tarissent, les travaux cessent. Ils reprennent in extremis.
Dans les années 1980, , les computationnels Linguistiques, approche symbolique et statistique du langage naturel, se développent. Ensuite, les algorithmes d'apprentissage, le machine learning et le deep learning, se perfectionnent. Désormais, des réseaux de neurones profonds secondent l'approche procédurale (ensemble d'instructions exécutées par la machine). Les apprentissages supervisés et non supervisés permettent de générer des modèles avec des niveaux d'abstraction élevés.
Parmi les domaines d'application les plus utilisés sont :

- La traduction automatique telle que Google Translator qui est capable de traduire des textes entiers sans aucune intervention humaine.
- l'analyse des sentiments consiste à identifier les informations subjectives d'un texte pour extraire l'opinion de l'auteur.
- Marketing.
- Chatbots comme la plateforme Facebook et Messenger aux chatbots en 2016.
- etc ...

1.7 Analyse de sentiment

1.7.1 Définition

L'analyse du sentiment (ou de la tonalité), aussi appelé opinion mining. Il s'agit de mesurer les sentiments des gens[11], c'est-à-dire les pensées sur un contexte spécifique tel que les critiques de produits, etc...
L'analyse des sentiments est le processus par lequel une partie des lettres est positive, négative ou neutre. Un système d'analyse des sentiments pour l'analyse de texte incorpore des techniques de traitement du langage naturel (NLP) et d'apprentissage automatique pour attribuer des scores de sentiment pondérés dans une phrase ou une phrase à des entités, des sujets, des thèmes et des catégories. On pense que l'analyse automatique des sentiments doit également implémenter des algorithmes finement réglés pour détailler les émotions humaines.
Récemment, l'analyse de sentiments a reçu beaucoup d'attention non seulement de la part de la recherche scientifique mais aussi par les domaines de la publicité et du marketing. Cela peut être attribué aux récentes avancées dans les réseaux sociaux et à la rapidité du relais de l'information.

1.7.2 Analyse de sentiment des tweets

Twitter est une plate-forme de communication basée sur le Web qui permet à ses abonnés de diffuser des messages appelés "tweets" d'une taille maximale "280 caractères", dans le but de partager des pensées, des liens ou des images. Par conséquent, Twitter est une source riche de données pour l'exploration d'opinion et l'analyse de sentiment. La simplicité d'utilisation et les services offerts par la plate-forme Twitter lui permettent d'être largement utilisée dans le monde. Cette popularité nous donne accès à des informations précieuses qui peuvent servir comme une base de données pour l'analyse des tweets.

1.8 Les approches d'analyse de sentiment basées sur Machine Learning (ML)

Le thème de l'opinion mining a beaucoup d'importance dans le domaine de la recherche scientifique qui est traité avec de nombreux travaux récents. Nous résumons certaines de ces méthodes dans la figure 1.2.
L'apprentissage machine (ML : Machine Learning) est un sous-domaine de l'intelligence artificielle, qui donne à un système une capacité de compréhension grâce à des algorithmes. Il est basé sur l'idée de faire apprendre des algorithmes à partir de données et de faire des prédictions avec ces données et par cela les ordinateurs apprennent à résoudre des tâches

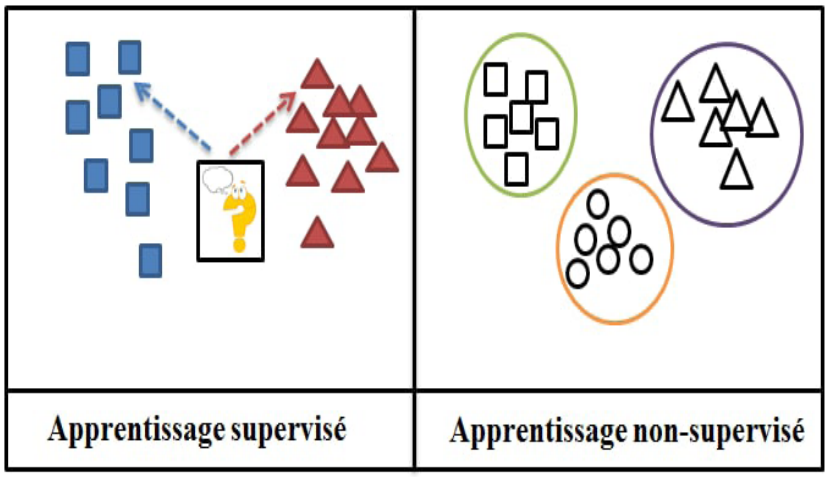


FIGURE 1.2 – Techniques d'apprentissage.

spécifiques, sans avoir besoin de les programmer. Il existe quatre types ou catégories de l'apprentissage machine. Le premier s'appelle l'apprentissage supervisé ,Le second est l'apprentissage non supervisé, Le troisième est l'apprentissage par renforcement et le quatrième est l'apprentissage semi-supervisé. Dans notre travail on s'interesse aux deux premiers :

1.8.1 Apprentissage supervisé

Il est basé sur les données libellées et par conséquent, les étiquettes sont fournies au modèle au cours du processus d'apprentissage. Ces données libellées sont utilisées par l'algorithme d'apprentissage pour donner un modèle qui sera utilisé lors de la prise de décision. Certains modèles d'apprentissage automatique ont été formulés pour classer le texte. Les techniques d'apprentissage automatique comme Naïve Bayes (NB), les machines à vecteurs de support (SVM) ont donné un grand succès à l'analyse des sentiments.

Classification Naive Bayes

L'algorithme de la classification Bayésienne est considéré parmi les classifications les plus simples et les plus puissants basé sur le théorème de Bayes, avec une hypothèse forte et naïve de l'indépendance. Naïve Bayes suppose l'indépendance entre tous les attributs. C'est-à-dire que la présence d'une fonctionnalité dans une classe n'est pas liée à une autre fonctionnalité. Même si ces caractéristiques dépendent les unes des autres ou de l'existence d'autres caractéristiques, toutes ces propriétés contribuent indépendamment. Malgré l'hypo-

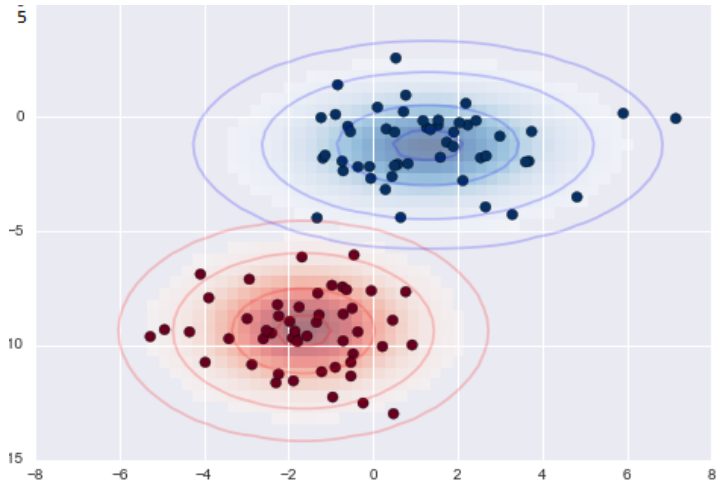


FIGURE 1.3 – Modele de Naive Bayes.

thèse de Naïve Bayes, cette technique s'est avérée très efficace pour la classification de texte et le modèle Naïve Bayes représenté dans la figure 1.3 est facile à construire et particulièrement utile pour les très grands ensembles de données.
Dans la théorie des probabilités, le théorème de Bayes (1.1) décrit la probabilité d'un événement, sur la base de la connaissance préalable de conditions pouvant être liées à l'événement. C'est un moyen de déterminer la probabilité conditionnelle.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{1.1}$$

$P(c|x)$ est la probabilité à postériori de la classe (c, cible) en fonction du prédicteur (x, attributs).
$P(c)$ indique la probabilité à priori pour une classe donnée.
$P(x|c)$ c'est la probabilité d'un prédicteur à une classe donnée.
$P(x)$ représente la probabilité d'un prédicteur donné.
Le calcul du terme $P(x|c)$ va dépendre de la distribution que suivent nos entités.
Dans le contexte de la classification de texte, où les entités peuvent représenter un nombre de mots, les entités peuvent suivre une distribution multinomiale. Dans d'autres cas, où les entités sont continuées, elles peuvent suivre une distribution gaussienne.

Classification Machine à vecteurs de support (SVM : Support Vector Machine)

Les SVMs sont une famille d'algorithmes d'apprentissage automatique qui permettent de résoudre des problèmes de classification telle que la régression ou la détection d'anomalie

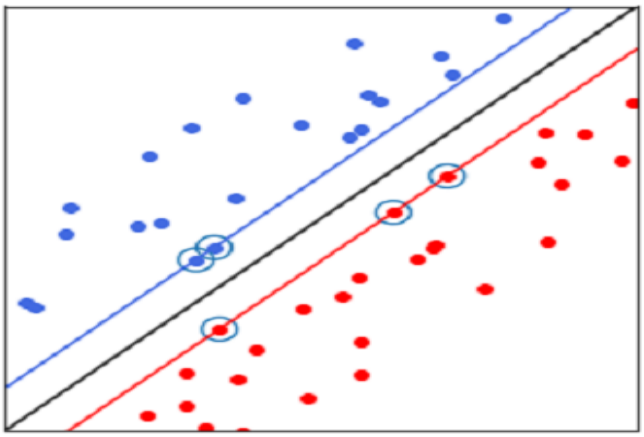


FIGURE 1.4 – Le principe de fonctionnement de SVM.

[26].
Les SVMs ont été développés dans les années 1990. Comme le montre la figure 1.4, il ont pour but de séparer les données en classes à l'aide d'une frontière, de telle façon que la distance entre les différents groupes de données et la frontière qui les sépare soit maximale. Les SVM "vecteurs de support" étant les données les plus proches de la frontière. Cette notion de frontière suppose que les données soient linéairement séparables, ce qui est rarement le cas. Les SVM reposent souvent sur l'utilisation d'une fonction noyau pour les séparer linéairement qui permet de projeter les données dans une espace de plus grande dimension Les SVMs permettent de projeter les données dans une espace de plus grande dimension via une fonction noyau pour les séparer linéairement (voir la figure 1.5). Les SVMs sont utilisés dans une variété d'applications (bioinformatique, recherche d'informations, vision par ordinateur, finance, etc...). On peut les utiliser sans comprendre leur fonctionnement, il existe des jeux d'hyperparamètres par défaut pour la classification, la régression ou la détection d'anomalie qui fonctionnent dans l'immense majorité des cas.

1.8.2 Apprentissage non supervisé

La classification, ou clustering, est un thème de recherche majeur en apprentissage automatique, en analyse et en fouille de données où l'objectif est, la répartition en classes d'un ensemble de données non étiquettées. On se contente de donner seulement l'algorithme k-means qu'aura inclus dans ce travail.

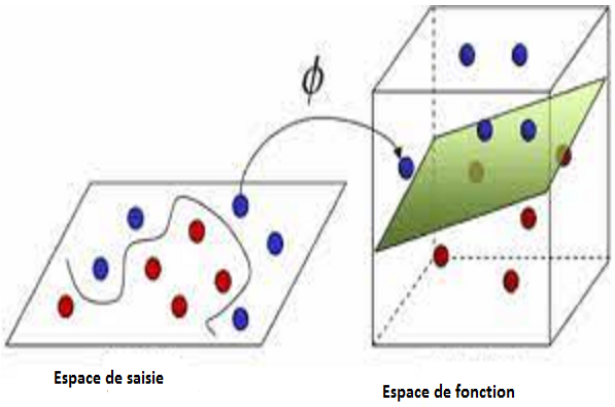


FIGURE 1.5 – Modele de Machine à vecteurs de support.

K-means (ou K-moyennes)

C'est l'un des algorithmes de clustering les plus répandus [28] qui permet d'analyser un jeu de données caractérisées par un ensemble de descripteurs, afin de regrouper les données "similaires" en groupes (ou clusters).
La similarité entre deux données peut être inférée grâce à la "distance" séparant leurs descripteurs, ainsi deux données très similaires sont deux données dont les descripteurs sont très proches. Cette définition permet de formuler le problème de partitionnement des données comme la recherche de K "données prototypes", autour desquelles peuvent être regroupées les autres données. Ces données prototypes sont appelés centroïdes, en pratique l'algorithme associe chaque donnée à son centroïde le plus proche, afin de créer des clusters. D'autre part, les moyennes des descripteurs des données d'un cluster, définissent la position de leur centroïde dans l'espace des descripteurs : ceci est à l'origine du nom de cet algorithme (Kmoyennes ou K-means en anglais).

1.8.3 Approche lexicale

Méthode basée sur le lexique qui utilise un dictionnaire des sentiments avec des mots d'opinion et les faire correspondre avec les données pour déterminer la polarité. Elle attribue les scores de sentiment aux mots d'opinion décrivant si les mots sont positifs, négatif ou neutre. Les approches fondées sur le lexique reposent principalement sur un lexique de sentiment, à savoir, une collection de termes de sentiment connue et précompilée, des phrases et même des expressions idiomatiques, développés pour les genres traditionnels de communication, tels que le lexique Opinion Finder.

1.8.4 Approches hybride

Cette approche est appelée aussi classification semi-supervisées qui combinent les points forts des deux approches précédentes. Elles prennent en compte tout le traitement linguistique des approches symboliques avant de lancer le processus d'apprentissage comme dans les approches statistiques.

1.9 Les approches d'analyse de sentiment basées sur Deep Learning (DL)

Deep Learning est l'une des raisons qui ont conduit les récents mouvements et progrès de l'intelligence artificielle, et c'est la principale cause qui fait penser que finalement, il existe une possibilité pour l'intelligence artificielle de devenir plus réaliste.

1.9.1 Définition de DL

Selon les fondateurs Yann LeCun, Yoshua Bengio Geoffrey Hinton dans [27] :
"L'apprentissage profond permet aux modèles informatiques composés de plusieurs couches de traitement d'apprendre des représentations de données avec plusieurs niveaux d'abstraction."
Autre définition par les auteurs dans [13] :
"L'apprentissage profond permet aux modèles informatiques composés de plusieurs couches de traitement d'apprendre des représentations de données avec plusieurs niveaux d'abstraction."
En d'autres termes, DL est un sous-ensemble des méthodologies et techniques de ML qui utilisent le réseau neuronal artificiel (ANN). C'est l'adaptation des réseaux neuronaux qui imite la structure du cerveau humain. La force de DL réside dans le fait que la machine peut extraire des caractéristiques et apprendre toute seule, indépendamment de l'intervention d'un expert. Il a été appliqué dans de nombreux domaines différents (traitement des images, textes, paroles et vidéos). Le succès de DL appartient à la disponibilité de plus de données d'entrainement. Google, Facebook et Amazon a déjà commencé à l'utiliser pour faire l'analyse de leurs énormes quantités de données [34]. La relation entre les trois concepts IA, ML et DL est résumée par les auteurs dans la figure 1.6 :

1.9.2 Applications de DL

L'importance de DL est présente dans plusieurs domaines, nous citons brièvement quelques domaines d'applications de cette approche ci-dessous [6] :

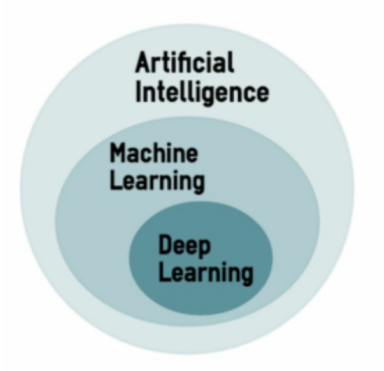


FIGURE 1.6 – Relation entre les trois concepts [27].

- Le traitement automatique de langage naturel : Son but étant d'extraire le sens des mots, voire des phrases pour faire de l'analyse de sentiments.
- La reconnaissance faciale : Les yeux, le nez, la bouche, tout autant de caractéristiques qu'un algorithme de DL va apprendre à détecter sur une photo.
- Voitures autonomes : Les entreprises qui construisent de tels types de services d'aide à la conduite, ainsi que des voitures autonomes telles que Google.
- Recherche vocale et assistants à commande vocale.
- traduction automatique : Il s'agit d'une tâche dans laquelle des mots, expressions ou phrases donnés dans une langue sont automatiquement traduits dans une autre langue.
- Reconnaissance d'image : Son objectif est de reconnaître et d'identifier les personnes et les objets dans les images, ainsi que de comprendre le contenu et le contexte.
- la description automatique d'image : Le sous titrage automatique des images est la tâche pour laquelle le système doit générer une légende décrivant le contenu de l'image.
- Colorisation automatique : La colorisation de l'image pose le problème de l'ajout de couleurs aux photographies noir et blanc. Le DL peut être utilisé pour traiter les objets et leur contexte dans la photographie pour colorer l'image.
- Analyse des sentiments du texte : La recherche sur le traitement du langage naturel et les réseaux de neurones récurrents ont parcouru un long chemin et il est maintenant tout à fait possible de déployer ces modèles sur le texte de votre application pour extraire des informations de niveau supérieur. Cela peut être très utile pour évaluer la polarité sentimentale dans les sections de commentaires ou pour extraire des sujets significatifs à l'aide de modèles de reconnaissance d'entités nommées.
- Recherche en marketing : La segmentation du marché, l'analyse des campagnes marketing et bien d'autres peuvent être améliorés à l'aide de modèles de régression et de classification DL.

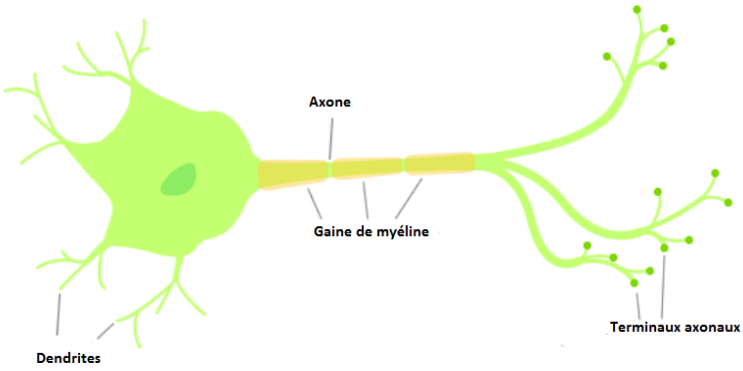


FIGURE 1.7 – Un neurone réel.

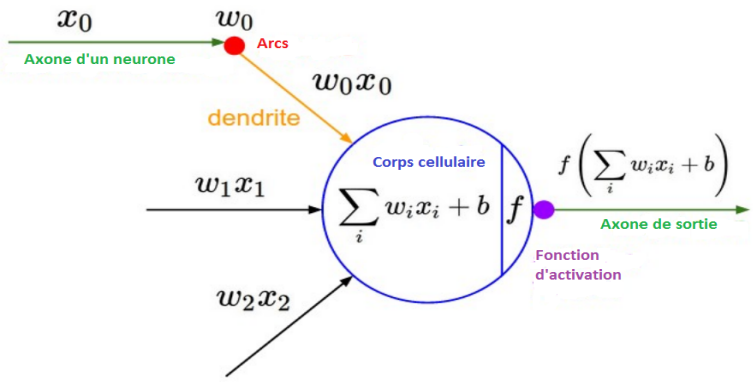


FIGURE 1.8 – Un neurone artificiel.

1.9.3 Le réseau neuronal

Avant de rentrer dans le détail des différents algorithmes utilisés en apprentissage profond pour différents cas d'usage, commencons par définir simplement le modèle au coeur de l'apprentissage profond : le "réseau de neurones".
Les réseaux neuronaux, aussi appelés ANN, sont des modèles de traitement de l'information qui simulent le fonctionnement d'un système nerveux biologique. C'est similaire à la façon dont le cerveau manipule l'information au niveau du fonctionnement. Tous les réseaux neuronaux sont constitués de neurones inter-connectés qui sont organisés en couches [27].

Les neurones artificiels sont inspirés du vrai neurone qui existe dans notre cerveau. Les deux figures 1.7 et 1.8 montrent une représentation d'un neurone réel et d'un neurone artificiel.
Les x_i sont des valeurs numériques qui représentent soit les données d'entrée, soit les valeurs sorties d'autres neurones. Les poids w_i sont des valeurs numériques qui représentent

soit la valeur de puissance des entrées, soit la valeur de puissance des connexions entre les neurones. Il existe des opérations qui se passent au niveau du neurone artificiel. Le neurone artificiel fera un produit entre le poids w et la valeur d'entrée x, puis ajouter un biais b, le résultat est transmis à une fonction d'activation f qui ajoutera une certaine non-linéarité.

Après que le neurone a effectué le produit entre ses entrées et ses poids, il applique également une non-linéarité sur ce résultat. Cette fonction non linéaire s'appelle la fonction d'activation.

1.9.4 Algorithmes du Deep Learning

Voyons maintenant deux types d'architectures différents de réseaux de neurone.

1.9.4.1 Les réseaux de neurones convolutifs (CNN)

Ces réseaux [30] sont utilisés pour tout usage autour de l'image ou de la vidéo dont fait partie la reconnaissance faciale ou encore la classification d'image.
L'entreprise Bai Du (l'équivalent de Google en Chine) a par exemple mis en place des portiques actionnés par la reconnaissance visuelle qui laissent passer uniquement leurs employés.
Snapchat et de nombreuses applications mobiles ont utilisé la percée de l'apprentissage et des CNN pour augmenter leurs fonctionnalités de "filtres".
Le nom réseau convolutif renvoit à un terme mathématique : le produit de convolution.
En termes simples, l'idée est qu'on applique un filtre à l'image d'entrée, les paramètres du filtre seront appris au fur et à mesure de l'apprentissage. Un filtre appris permettra par exemple de détecter les angles dans une image si les angles servent à classifier au mieux l'image (voir la figure 1.9).
L'image est d'abord décomposé dans les 3 cannaux (R,G,B) pixels par pixels, on obtient donc 3 matrices de taille $n \times n$ (où n est le nombre de pixels). Voici un exemple de convolution avec une matrice de taille 6×6 est donné dans la figure 1.10 :
Il est important de noter deux avantages importants inhérents aux réseaux convolutifs :
le réseau peut apprendre par étape à reconnaitre les éléments caractéristiques d'une image. Pour reconnaître un visage par exemple : il apprendra à reconnaître d'abord des paupières, des pupilles, pour arriver à identifier des yeux, une fois un élément appris à un endroit de l'image le réseau sera capable de le reconnaître n'importe où d'autre dans l'image.

1.9.4.2 Les réseaux de neurones récurrents (RNN)

Les réseaux de neurones récurrents (en anglais recurrent neural networks) [30], aussi appelés RNNs, sont une classe de réseaux de neurones qui permettent aux prédictions anté-

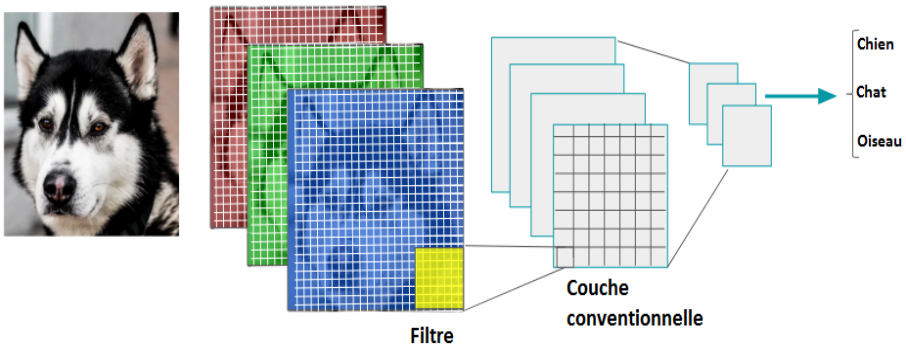


FIGURE 1.9 – Traitement de l'image par CNN.

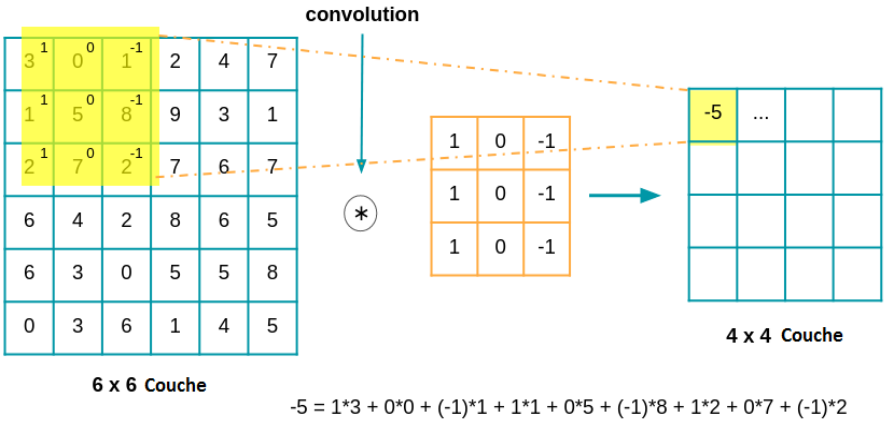


FIGURE 1.10 – Traitement de l'image par CNN.

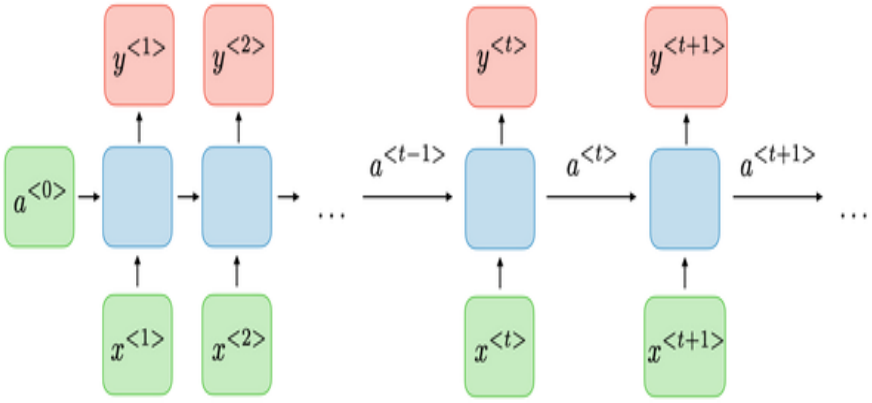


FIGURE 1.11 – Un exemple d'un réseau récurrent.

rieures d'être utilisées comme entrées, par le biais d'états cachés (en anglais hidden states). Ils sont présentée dans la figure 1.11 :
Les calculs se déroule comme dans la figure 1.12. À l'instant t, l'activation $a^{<t>}$ et la sortie $y^{<t>}$ sont donnée par 1.2 :

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \ et \ y^{<t>} = g_2(W_{ya}x^{<t>} + b_y) \tag{1.2}$$

Où W_{ax}, W_{aa}, W_{ya}, b_a, b_y sont des coefficients indépendants du temps et où g_1, g_2 sont des fonctions d'activation. Les réseaux de neurones récurrents sont adaptés pour des données d'entrée de taille variable. Ils conviennent en particulier pour l'analyse de séries temporelles. Ils sont utilisés en reconnaissance automatique de la parole ou de l'écriture manuscrite, plus généralement en reconnaissance de formes ou encore en traduction automatique, ils sont comparables à des réseaux de neurones classiques avec des contraintes d'égalité entre les poids du réseau.
Les techniques d'entraînement du réseau sont les mêmes que pour les réseaux classiques (rétropropagation du gradient), néanmoins les réseaux de neurones récurrents se heurtent au problème de disparition du gradient pour apprendre à mémoriser des évènements passés.

Des architectures particulières répondent à ce dernier problème, on peut citer en particulier les réseaux LSTM (Long short term memory) qui ont été inventées par les informaticiens Sepp Hochreiter et Jurgen Schmidhuber en 1997. Les RNN construites avec des unités LSTM classent les données en cellules de mémoire à court terme et à long terme. Les RNN peuvent ainsi déterminer quelles données sont importantes et doivent être mémorisées et réintégrées dans le réseau, et quelles données peuvent être oubliées.

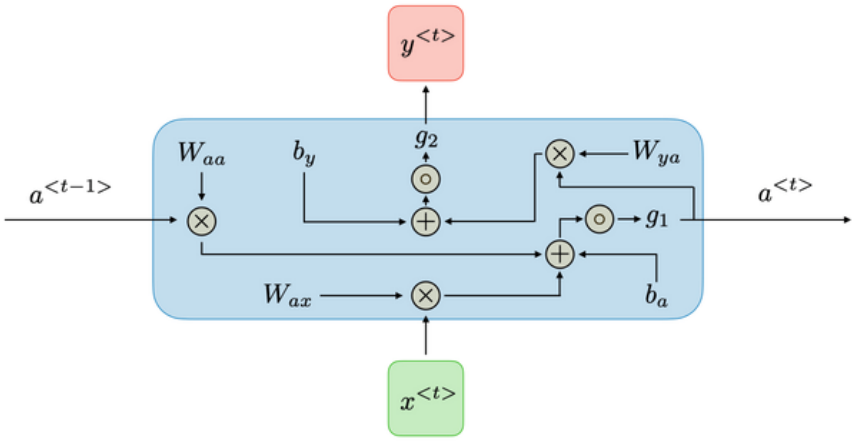


FIGURE 1.12 – Exemple de l'activation et la sortie dans RNN.

1.10 Conclusion

Dans ce chapitre nous avons présenté les caractéristiques et les particularités des microblogs, notamment Twitter. De même, nous avons donné les différents algorithmes qui s'intéressaient à l'analyse des sentiments que ce soient en machine learning ou deep learning.

Chapitre 2

Extraction et prétraitement des données

2.1 Introduction

Dans ce chapitre, nous présentons les corpus que nous avons collectés, les expérimentations que nous avons faites par les différentes méthodologie du prétraitement des données (Tokenization, Stemming, stop words removal, Vectorisation et Transformation).
Dans nos expérimentations, nous avons traité les tweets pour le cas covid-19 qui portent sur le Royaume-Uni et sont écrits en anglais standard.

2.2 Environnement de Travail

Dans cette section nous reprenons les descriptions de l'environnement de notre expérimentation.

2.2.1 Environnement matèriel

Afin de mener notre expérimentation et évaluation, nous avons utilisé des machines avec quelques caractéristiques spécifiées dans le tableau suivant :

PC	HP EliteBook 820	HP EliteBook 840
OS	Ubuntu 18.04	Windows 10
RAM de taille :	8 GO	10 GO
Precesseur	INTEL CORE I5 5300U 2,3GHZ	INTEL CORE I7 4300U 2,3GHZ
Carte Graphique :	Intel®HD Graphics 5500 (BDW GT2)	Intel®HD Graphics 4500

2.2.2 Environnement logiciel

D'abord on va commencé par l'environnement de travail linux :
Ubuntu[37] :est un système d'exploitation GNU/Linux basé sur Debian Linux qui est livrés avec Python 3 et Python 2 préinstallés. Les environnements virtuels nous permettent d'avoir un espace isolé sur notre serveur pour les projets Python garantissant que chaque projet peut avoir son propre ensemble de dépendances.
Nous avons aussi utilisé le langage de programmation Python qui est un langage de programmation portable, dynamique, extensible, gratuit, etc..., et il prend la programmation orientée objet. Python est développé depuis 1989 par Guido van Rossum et de nombreux contributeurs bénévoles.
Pour celà nous avons utilisé l'environnement de développement :
Anaconda[8] : est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique qui vise à simplifier la gestion des paquets et de déploiement. La distribution Anaconda est utilisée par plus de 6 millions d'utilisateurs et comprend plus de 250 paquets populaires en science des données adaptés pour Windows, Linux et MacOS.
Jupyter : est une application web utilisée pour programmer dans plus de 40 langages de programmation, dont Python, Julia, Ruby, R, ou encore Scala. C'est un projet communautaire dont l'objectif est de développer des logiciels libres, des formats ouverts et des services pour l'informatique interactive.
Jupyter permet de réaliser des calepins ou notebooks qui sont utilisés en science des données pour explorer et analyser des données. Pour plus d'information consulter `https://jupyter.org/about`.
Google Colab : Colaboratory, souvent raccourci en "Colab", est un produit de Google Research. Colab permet à n'importe qui d'écrire et d'exécuter le code Python de son choix par le biais du navigateur. C'est un environnement particulièrement adapté au machine learning, à l'analyse de données et à l'éducation. En termes plus techniques, Colab est un service hébergé de notebooks Jupyter qui ne nécessite aucune configuration et permet d'accéder gratuitement à des ressources informatiques (pour plus de détail voir la source `https://research.google.com/colaboratory/faq.html?hl=fr`).

2.3 Extraction des données de Twitter

Twitter est une mine d'or de données, contrairement aux autres plateformes sociales, les tweets sont presque tous les utilisateurs entièrement publics et extractibles. C'est un énorme avantage si nous essayons d'obtenir une grande quantité de données sur lesquelles exécuter des analyses.

2.3.1 Interface de programmation d'applications (API) de Twitter

L'interface de programmation d'applications (API) de Twitter permet aux développeurs d'accéder à la plupart des fonctionnalités de Twitter. Nous pouvons utiliser l'API pour lire et écrire des informations relatives aux entités Twitter telles que les tweets, les utilisateurs et les tendances.
L'API se décompose en quatre classes :

- SEARCH : permet d'interroger twitter pour récupérer des données simples, essentiellement des tweets.
- REST : extension avancée de REST qui permet d'accéder à des fonctionnalités avancées de twitter : chercher des utilisateurs, des followers, voir les status, éditer des informations sur son compte, etc...
- STREAMING : permet de communiquer avec twitter en mode streaming. La particularité de cette API avancée est de permettre l'accès à de gros volume de données twitter et d'être moins contrainte par les limites d'accès et d'interrogations de twitter. Cependant, cette API requiert la mise en place de mécanisme plus complexe pour l'accès aux données.
- WEBSITES : permet d'intégrer des fonctions de base twitter dans des sites webs.

Création des informations d'authentification de l'API Twitter

Comme nous l'avons vu précédemment, l'API Twitter exige que toutes les requêtes utilisent $OAuth$ pour s'authentifier. Nous devons donc créer les informations d'authentification requises pour pouvoir utiliser l'API. Ces informations d'identification sont quatre chaînes de texte :

- La clé du consommateur.
- Secret du consommateur.
- Jeton d'accès.
- Secret d'accès.

Étape 1 : demander un compte de développeur Twitter
Étape 2 : créer une application.
Twitter accorde des informations d'authentification aux applications, pas aux comptes. Une application peut être n'importe quel outil ou bot qui utilise l'API Twitter. Nous devons donc enregistrer notre application pour pouvoir passer des appels API.
Pour enregistrer notre application, accéder à la page de nos applications Twitter `https://apps.twitter.com/` et sélectionner l'option "Créer une application". Si on dispose déjà d'un compte d'utilisateur Twitter, on suit la procédure de création le clé, du jeton et des

secrets. Sinon, on doit s'inscrire en tant qu'utilisateur Twitter avant de continuer.

2.3.2 Tweepy

Tweepy est un package Python open source qui vous offre un moyen très pratique d'accéder à l'API Twitter avec Python. Tweepy comprend un ensemble de classes et de méthodes qui représentent les modèles de Twitter et les points de terminaison d'API, et il gère de manière transparente divers détails d'implémentation, tels que [36] :

- Encodage et décodage des données
- Requêtes HTTP
- Pagination des résultats
- Authentification OAuth
- Limites de taux
- Ruisseaux

Presque toutes les fonctionnalités fournies par l'API Twitter peuvent être utilisées via Tweepy. La seule limitation actuelle, à partir de la version $3.7.0$, est que les messages directs ne fonctionnent pas correctement en raison de certains changements récents dans l'API Twitter.
Pour voir plus d'information sur la bibliothèque Tweepy(installtion, les méthodes ...), on aura besoin du lien suivant `https://docs.tweepy.org/en/stable/`.

2.3.3 Twint

Twint est une bibliothèque python open source qui est utilisée pour le scraping Twitter, c'est-à-dire que nous pouvons utiliser twint afin d'extraire des données de Twitter et cela aussi sans utiliser l'API Twitter [39]. Il existe certaines fonctionnalités de twint qui le rendent plus utilisable et unique par rapport aux autres API de scraping Twitter, à savoir :

- Facile à utiliser et très rapide
- Aucune connexion ou inscription initiale requise pour récupérer des données.

Twint peut être utilisé pour gratter les tweets en utilisant différents paramètres comme les hashtags, les noms d'utilisateur, les sujets, etc...
Il peut même extraire des informations comme le numéro de téléphone et les identifiants d'email à partir des tweets.

L'API Twitter a une limite de récupération de seulement 3200 (derniers) tweets tandis que twint n'a pas de limite de téléchargement de tweets, elle peut télécharger presque tous les tweets.
Importation des bibliothèques requises
Nous allons récupérer les données de Twitter en utilisant **twint**, donc nous importerons twint à part cela, nous devons importer **Pandas** qui est une bibliothèque écrite pour le langage de

programmation Python permettant la manipulation et l'analyse des données.
Package CSV : CSV (Comma Separated Values) module pour lire et écrire des données au format CSV.
Il y a aussi Le module **datetime** qui fournit des classes permettant de manipuler les dates et les heures.
Pour installer et utiliser **twint** pour l'extraction des tweets on peut suivre ce lien : `https://medium.com/analytics-vidhya/how-to-scrape-tweets-from-twitter`
Consulter aussi la vidéo pour enricher nos informations :
`https://www.youtube.com/watch?v=_SqgSh3aR1g`
Le code que nous avons utilisé est donc le suivant :

```
import twint
import datetime
import pandas
# Configure
list_=["Coronavirus","Covid-19","Covid","Corona" ]
count=3
for i in list_:
    print(i)
    c=twint.Config()
    c.Search=i
    c.Lang="en"
    c.Pandas=True
    c.Since="2020-04-01"
    c-Until="2021-04-01"
    #c.Localisation=True
    #c.Limit=5
    c.Near="London"
    c.Custom_csv=["id","user_id","username","date","tweet"]
    c.output=f"hh{count}.csv"
    twint.run.Search(c);
    count+=1
```

Cela mérite quelques explications :
On crée un objet c qui permet de paramétrer ce que nous allons demander à Twint, puis on définit ces paramètres :

- on définit la requête que l'on recherche, à savoir les mots clés qui nous intéressent.
- on peut spécifier la langue et la localisation des tweets.
- on veut indiquer la date à partir du premier avril 2020 jusqu'à le premier avril 2021.
- on stocke ces informations dans un fichier qui porte l'éxtention ".csv".

Ce qui est pratique avec Twint, c'est qu'il renvoie pas mal d'informations sur chaque tweet, comme par exemple :

- l'identifiant du tweet
- la date du post

- le nom de la personne l'ayant posté
- le contenu du tweet
- le nombre de likes, retweets et échanges du tweet
- les hashtags inclus dans le tweet

Cette liste n'est pas exhaustive, mais elle contient les éléments qui nous intéresseront par la suite.

2.4 Nettoyage des tweets

Une rapide inspection de la base nous permet de voir que la compréhension de certains tweets est difficile même pour des êtres humains. Le nettoyage sera d'autant plus important. C'est pour ça, on commence toujours par la construction d'un pipeline de nettoyage des données NLP (Natural Language Processing) qui est une branche de l'intelligence artificielle déstinée au traitement du langage écrit.

2.4.1 Le principe de fonctionnement

Le nettoyage des tweets est une étape essentielle car il prépare le texte brut pour l'extraction, il comprendra plusieurs choses :

- Retirer les ponctuations $(., ?, ", etc...)$, les caractères spéciaux $(@, +, §, etc...)$ et les nombres $(1, 2, 3, etc...)$.
- Changer les lettres majuscules en minuscules.
- Enlever le discripteur de twitter, les liens (HTTPS : / HTTP :) et les termes qui n'ont pas beaucoup de poids dans le contexte du texte.
- Supprimer les mots vides des tweets tels que ("are", "one", "the", etc...) existant dans la liste des mots vides fournie par le package **Nltk**.
  Nltk (Natural Language Toolkit) est une plate-forme pour la création de programmes Python pour travailler avec des données de langage humain.
- Ensuite, nous devons normaliser les tweets en utilisant **Stemming**.
  Stemming est un processus basé sur des règles de suppression des suffixes ("ing", "ly", "es", "ed", "s", etc...) d'un mot.
  Par exemple "play", "played", "playing" sont les différentes variantes du mot "play".

2.4.2 Regex et NeatText

Nous utilisons les **regex** avec le module Python **re** et **NeatText** qui permettent de faire facilement le nettoyage des données.

RegEx : Un RegEx ou expression régulière, est une séquence de caractères qui forme un

modèle de recherche. C'est pourqoui Python a un package intégré appelé **re**, qui peut être utilisé pour travailler avec des expressions régulières.
Le module **re** propose un ensemble de fonctions qui nous permet de rechercher une chaîne pour une correspondance :

- split : Renvoie une liste dans laquelle la chaîne a été divisée à chaque correspondance.
- sub : Remplace une ou plusieurs correspondances par une chaîne.

RegEx peut être utilisé pour vérifier si une chaîne contient le modèle de recherche spécifié.
Voila le premier code que nous avons utilisé :

```
def tweets(tweet):
    tweet = tweet.lower()
    tweet.replace('[^s\n\n]', '')
    tweet = re.sub('https?://\S+|www\.\S+', '', tweet)
    tweet = re.sub(r"[A-Za-z\.]*[0-9]+[A-Za-z%ř\.]*", "", tweet)
    tweet = re.sub('\[.*?\]', '', tweet)
    tweet = re.sub('<.*?>+', '', tweet)
    tweet = re.sub(r"[,\!\?\%\(\)\/\"]", "", tweet)
    tweet = re.sub(r"\&\S*\s", "", tweet)
    tweet = re.sub(r"\&", "", tweet)
    tweet = re.sub(r"\+", "", tweet)
    tweet = re.sub(r"\#", "", tweet)
    tweet = re.sub(r"\$", "", tweet)
    tweet = re.sub(r"\č", "", tweet)
    tweet = re.sub(r"\%", "", tweet)
    tweet = re.sub(r"\:", "", tweet)
    tweet = re.sub(r"\@", "", tweet)
    tweet = re.sub(r"\-", "", tweet)
    tweet = re.sub('[%s]' % re.escape(string.punctuation), '', tweet)
    tweet = re.sub('\n', '', tweet)
    tweet = re.sub('\w*\d\w*', '', tweet)
    tweet = " ".join(word for word in tweet.split() if word not in
   stop_words)
    tweet = " ".join(Word(word).lemmatize() for word in tweet.split())
    return(tweet)
df['processed_tweet'] = df['Tweet'].astype(str).apply(lambda row: tweets(
   row))
#df['processed tweet'] = df['Tweet'].apply(lambda x: tweets(x,
   custom_stopwords))
df.head(10)
```

Et le deuxième code que nous pouvons donner :

```
def remove_emoji(tweet):
  EMOJI_PATTERN = re.compile(
    "(["
    "\U0001F1E0-\U0001F1FF"  # flags (iOS)
    "\U0001F300-\U0001F5FF"  # symbols & pictographs
```

```
    "\U0001F600-\U0001F64F"  # emoticons
    "\U0001F680-\U0001F6FF"  # transport & map symbols
    "\U0001F700-\U0001F77F"  # alchemical symbols
    "\U0001F780-\U0001F7FF"  # Geometric Shapes Extended
    "\U0001F800-\U0001F8FF"  # Supplemental Arrows-C
    "\U0001F900-\U0001F9FF"  # Supplemental Symbols and Pictographs
    "\U0001FA00-\U0001FA6F"  # Chess Symbols
    "\U0001FA70-\U0001FAFF"  # Symbols and Pictographs Extended-A
    "\U00002702-\U000027B0"  # Dingbats
    "])"
  )
  tweet = re.sub(EMOJI_PATTERN, r' \1 ', tweet)
  return tweet
```

NeatText : C'est un package NLP simple pour nettoyer les données textuelles et le prétraitement de texte.

Problème :

- Nettoyage des données textuelles non structurées.
- Réduire le bruit (caractères spéciaux, mots vides)
- Réduire la répétition de l'utilisation du même code pour le prétraitement de texte.

Solution : Convertir la solution déjà connue de nettoyage de texte en un package réutilisable Voici un exemple de deux tweets avant et aprés le nettoyage :

Avant Nettoyage	Aprés Nettoyage
<yungdaveGH> ITS OUT!!! YOUR NEW SUMMER NIGHTS FAVOURITE....Go stream on YouTube https ://t.co/4zUuWH16T5 unsignedhour ASMR BritishSummerTime clocksgoforward AWeekAway trending summer sun coronavirus https ://t.co/bFjzSMisq7	new summer night favouritego stream youtube unsignedhour asmr britishsummertime clocksgoforward aweekaway trending summer sun coronavirus
<BlueChipIT> So blame the government rather than the parent who committed a crime BBC News - Coronavirus : Prisoners' children 'forgotten' during pandemic https ://t.co/FvEGJaEfny	blame government rather parent committed crime bbc news coronavirus prisoner child forgotten pandemic

2.4.3 Statistiques et visualisation des résultats

D'après avoir nettoyer les tweets extraits, on va les visualiser dans un histograme qui nous affiche le nombre des tweets classés dans un ordre chronologique par rapport au mois

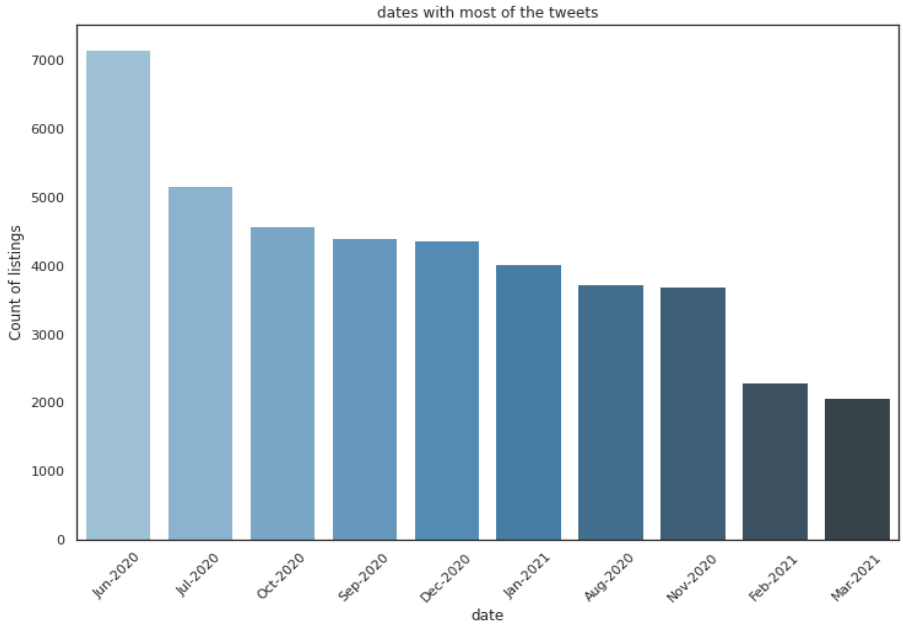


FIGURE 2.1 – Distribution des tweets par mois.

comme nous voyons dans la figure 2.1. On vois que le mois juin de l'année 2020 est le mois le plus actif qui contient le maximum des tweets.
Voici un graphe représenté dans la figure 2.2 avec le même concept que l'histogramme précédent mais sous forme d'une fréquence des tweets par mois.

2.5 L'analyse de sentiments avec TextBlob

Dans cette partie, nous utiliserons l'un des excellents packages Python **TextBlob**, pour construire un analyseur sentimental simple. Nous savons tous que les tweets sont l'un des exemples de jeux de données préférés en matière d'analyse de texte en science des données et en apprentissage automatique.

2.5.1 Qu'est-ce que TextBlob ?

TextBlob est une bibliothèque Python construite sur nltk. Il est plus facile à utiliser et fournit des fonctionnalités supplémentaires, telles que des scores de sentiment basés sur des règles. **TextBlob** trouve tous les mots et expressions auxquels il peut attribuer une polarité et une subjectivité, et fait la moyenne de tous ensemble. **TextBlob** est un module NLP sur Python utilisé pour l'analyse de sentiment. La fonction de TextBlob qui nous intéresse

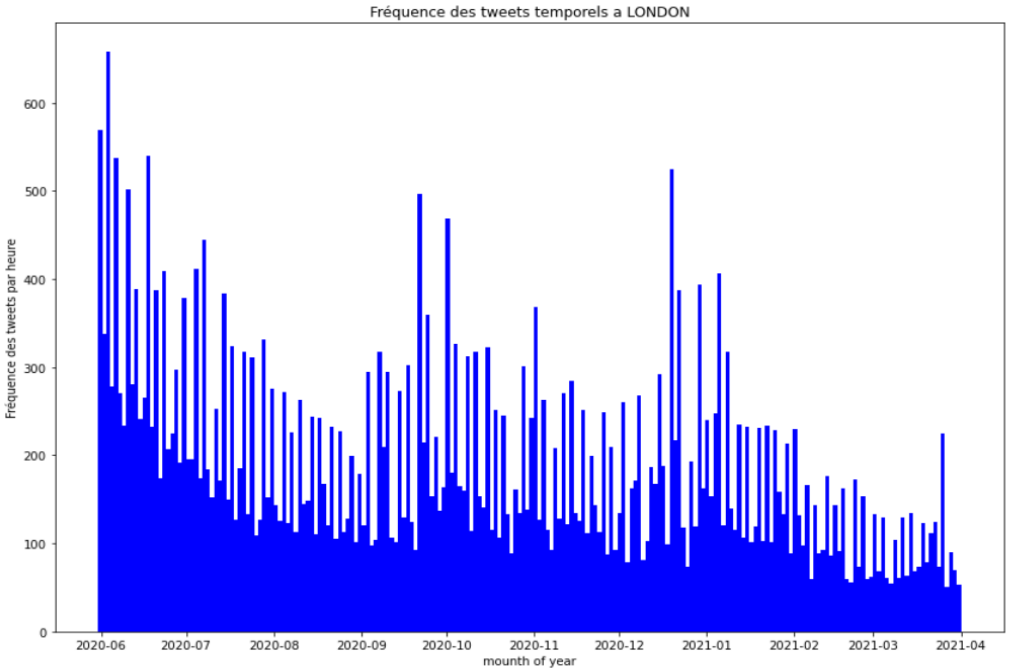


FIGURE 2.2 – Fréquence des tweets par mois.

permet pour un texte donné de déterminer le ton du texte et le sentiment de la personne qui l'a écrit. Pour plus d'informations, il vaut mieux consulter `https://textblob.readthedocs.io/en/dev/`.

Pour chaque mot, nous obtiendrons un score de sentiment (à quel point ils sont positifs / négatifs/neutres) et un score de subjectivité (à quel point sont-ils avisés ?)

Polarité : à quel point un mot est positif, négatif ou neutre .

- -1 est très négatif.
- +1 est très positif.
- 0 est neutre.

Subjectivité : Quel est le degré de subjectivité ou d'opinion d'un mot.

- 0 est un fait.
- +1 est une très grande opinion.

On peut utiliser le code suivant pour trouver la polarité et la subjectivité des sentiments avec **TextBlob** :

```
############################ Calculer la polarité:
def get_sentiment(Tweet):
    blob = TextBlob(Tweet)
    sentiment_polarity = blob.sentiment.polarity
    sentiment_subjectivity = blob.sentiment.subjectivity
```

	polarity	subjectivity	sentiment
0	0.136364	0.454545	Positive
1	0.000000	0.000000	Neutral
2	-0.020833	0.313889	Negative
3	0.068182	0.227273	Positive
4	-0.088889	0.255556	Negative

FIGURE 2.3 – Polarité et subjectivité en fontion des sentiments.

```
    if sentiment_polarity > 0:
        sentiment_label = 'Positive'
    elif sentiment_polarity < 0:
        sentiment_label = 'Negative'
    else:
        sentiment_label = 'Neutral'
    result = {'polarity':sentiment_polarity,
              'subjectivity':sentiment_subjectivity,
              'sentiment':sentiment_label}
    return result
##############################
import neattext.functions as nfx
dir(nfx)
################## supprimer Stopwords et convertir pour tokeniser:
positive_tweet_list = positive_tweet.apply(nfx.remove_stopwords).tolist()
negative_tweet_list = negative_tweet.apply(nfx.remove_stopwords).tolist()
neutral_tweet_list = neutral_tweet.apply(nfx.remove_stopwords).tolist()
positive_tweet_list
```

2.5.2 Statistiques et visualisation des résultats

On peut maintenant faire une analyse des sentiments à l'aide de TextBlob et trouver à la fois la polarité et la subjectivité, et voila le dataFrame représenté par la figure 2.3 qui affiche les résultats obtenus :
Maintenant que nous avons une liste avec toutes les polarités des tweets du corpus, nous pouvons tracer une première courbe pour voir à quoi l'évolution ressemble. Pour cela on utilise matplotlib de Python, c'est la librairie de référence pour le tracé de courbes sur Python :
D'après le DataFrame précedent, on a obtenu la figure 2.4 qui montre la variation de la densité de polarité, et de la subjectivité.
Après on va faire un graphe qui affiche la variance de fréquence de polarité en fonction des

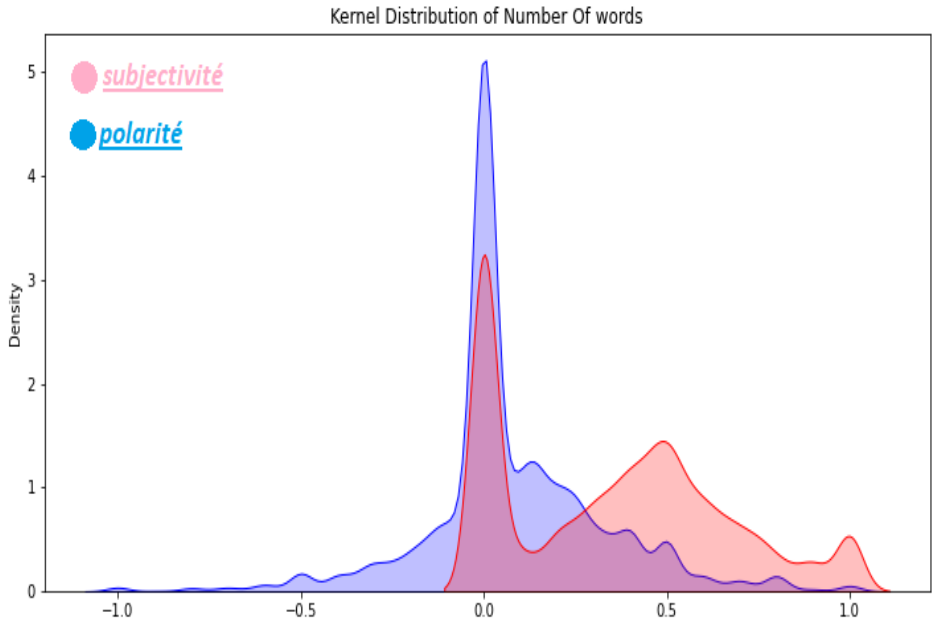


FIGURE 2.4 – Densité de polarité et de subjectivité.

nombres des tweets (voir la figure 2.5).
D'abord, tout au long de l'épidémie, les gens étaient assez partagés sur le danger du virus. De plus, les réactions des gens face au danger ne sont pas toutes identiques. Certains préfèrent plutôt prendre les choses à la rigolade. Ces réactions contribuent à augmenter la polarité. C'est pourquoi, on voit que rien à conclure du graphe de la figure 2.5 qui est très condensé.
Maintenant on va résumer ce qu'on a obtenu dans le calcul des sentiments dans un diagramme circulaire montré dans la figure 2.6. Ce diagramme nous montre que le sentiment positif possède le plus grand pourcentage 46% par rapport aux autres (négatif 20%, neutre 34%).

2.6 Tokenisation des tweets

Nous pouvons inclure aussi le processus de tokenisation qui consiste à convertir un groupe de phrases en jetons. La tokenisation est également appelée segmentation de texte ou analyse lexicale. Il s'agit essentiellement de diviser les données en un petit morceau de mots.
La tokenisation en python peut être effectuée par la fonction **word_tokenize()** de la bibliothèque python **NLTK**, aussi on peut utiliser la méthode **split()**.
On donne le code suivant de tokinisation :

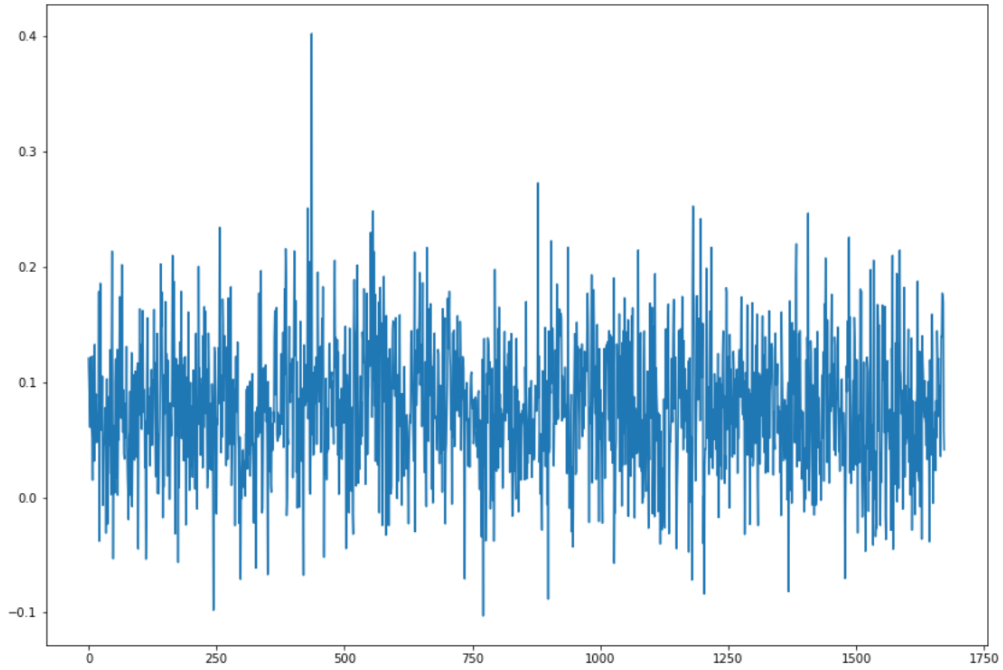


FIGURE 2.5 – Frequence de polarité des tweets.

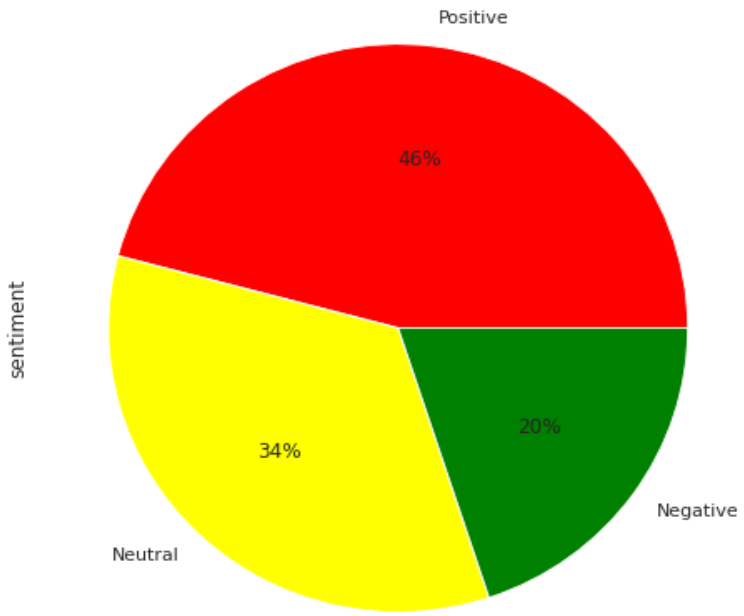


FIGURE 2.6 – Pourcentage des sentiments.

```
################### Tokenisation
for line in positive_tweet_list:
#      print(line)
    for token in line.split():
        print(token)
###########################
pos_tokens = [token for line in positive_tweet_list  for token in line.
    split()]
neg_tokens = [token for line in negative_tweet_list  for token in line.
    split()]
neut_tokens = [token for line in neutral_tweet_list  for token in line.
    split()]
```

Voici un exemple qui montre la tokenisation d'un tweet positif, négatif et neutre :

Tweet	Polarite	Tokens
'excellent observeruk sweden socialsolidarity'	Positif	'excellent', 'observeruk', 'sweden', 'socialsolidarity'
'spain france uk suffer covid surge italy ð9ƒ87®ð9ƒ87[1] adapted brutal phase rome milan conte italian europe pandemic coronavirus coronavirusoutbreak coronavirusupdate covid uklockdown madrid france'	Négatif	'spain', 'france', 'uk', 'suffer', 'covid', 'surge', 'italy', 'ð9ƒ87®ð9ƒ87[1]', 'adapted', 'brutal', 'phase', 'rome', 'milan', 'conte', 'italian', 'europe', 'pandemic', 'coronavirus', 'coronavirusoutbreak', 'coronavirusupdate', 'covid', 'uklockdown', 'madrid', 'france',
'poland report daily record coronavirus case	Neutre	'poland', 'report', 'daily', 'record', 'coronavirus' 'case'

Maintenant on va compter chaque mot obtenu pour voir les mots ou les words les plus propagés dans les trois cas des tweets(positif, négatif, neutre).
Pour faire cette étape on a besoin de la bibliothèque **collection** , dans laquelle on peut importer le module **counter**, dont il contient les données dans une collection non ordonnée, tout comme les objets de table de hachage. Les éléments ici représentent les clés et le décompte en tant que valeurs. Il vous permet de compter les éléments dans une liste itérable.

On peut résumer les tokens obtenus au niveau de notre DataFrame (Voir la figure 2.7) dans les histogrammes representés par les figures 2.8, 2.9 et 2.10 :
Nous remarquons que les mots "covid" et "coronavirus" sont répétés dans les trois histogrammes, ils occupent le plus grand score d'utilisation dans les tweets.

	tweetid	Timestamp	time	plus	Tweet	month_year	year	processed_tweet
0	1377305967138836482	2021-03-31	18:05:06	+0000	‹yungdaveGH› ITS OUT!!! YOUR NEW SUMMER NIGHTS...	Mar-2021	2021.0	new summer night favouritego stream youtube un...
1	1377218238992445443	2021-03-31	12:16:30	+0000	‹BlueChipIT› So blame the government rather th...	Mar-2021	2021.0	blame government rather parent committed crime...
2	1377202645975195648	2021-03-31	11:14:32	+0000	‹BritishHeraldUK› After a major plunge due to ...	Mar-2021	2021.0	major plunge due coronavirus britainâs econo...
3	1377194242393657346	2021-03-31	10:41:09	+0000	‹FelixxMoritz› Well, I guess itâs time to ex...	Mar-2021	2021.0	well guess itâs time explore uk ban leaving ...
4	1377186411929948162	2021-03-31	10:10:02	+0000	‹E7Drew› @SBurke10 @BBCNews Try reading the St...	Mar-2021	2021.0	bbcnews try reading statutory instrument coron...
5	1377172069079642112	2021-03-31	09:13:02	+0000	‹dawnvhardy› @BBCNews All Doctors and Nurses w...	Mar-2021	2021.0	bbcnews doctor nurse exposed coronavirus sayin...
6	1377166255208030210	2021-03-31	08:49:56	+0000	‹beecee› BIG DAY! Second vaccination for me an...	Mar-2021	2021.0	big day second vaccination mother morningcommu...
7	1377106144422887424	2021-03-31	04:51:04	+0000	‹Pairsonnalites› Coronavirus vaccine priority ...	Mar-2021	2021.0	coronavirus vaccine priority list changed bloo...
8	1377045099998277638	2021-03-31	00:48:30	+0000	‹FatimaDuerden› @DrewLawDesign Coronavirus	Mar-2021	2021.0	drewlawdesign coronavirus
9	1377016806154543105	2021-03-30	22:56:04	+0000	‹Pairsonnalites› Lloyds Pharmacy to launch UK'...	Mar-2021	2021.0	lloyd pharmacy launch uk first ever coronaviru...

FIGURE 2.7 – Tokenisation des tweets.

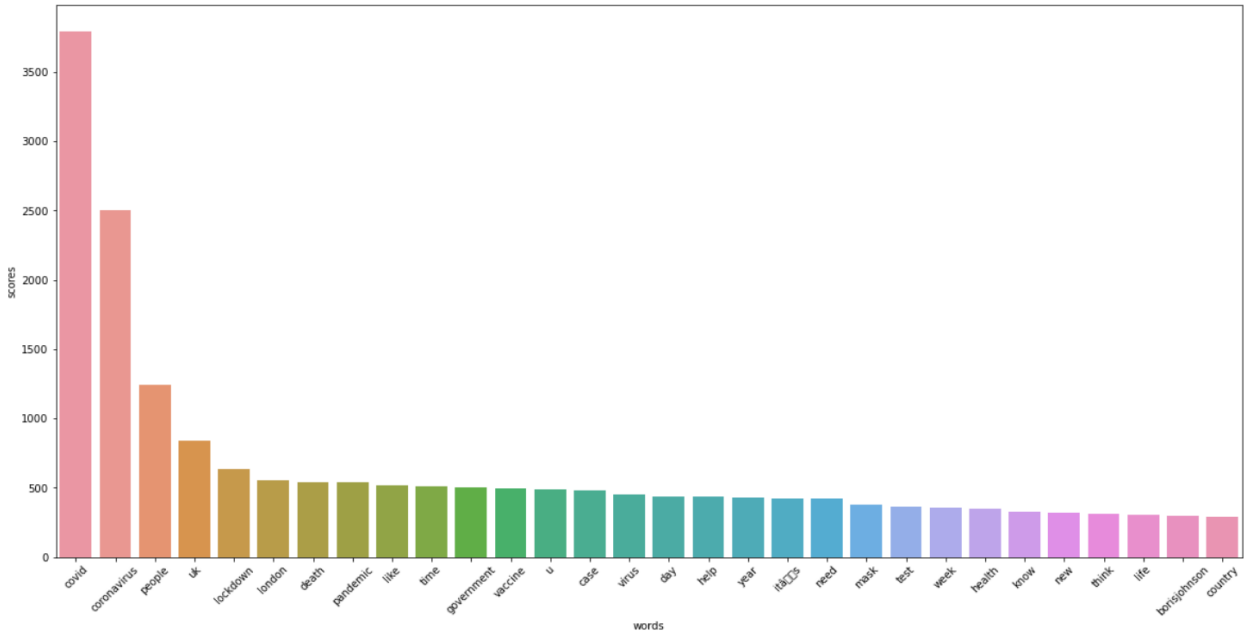


FIGURE 2.8 – Score des tokens négatifs

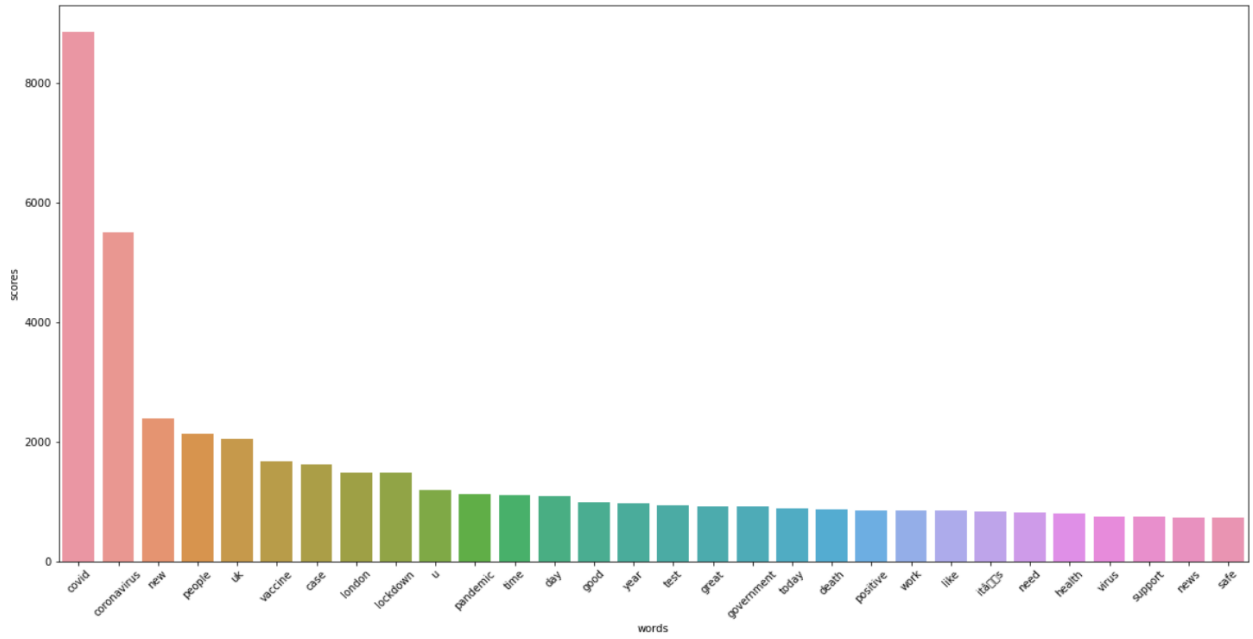


FIGURE 2.9 – Score des tokens positifs

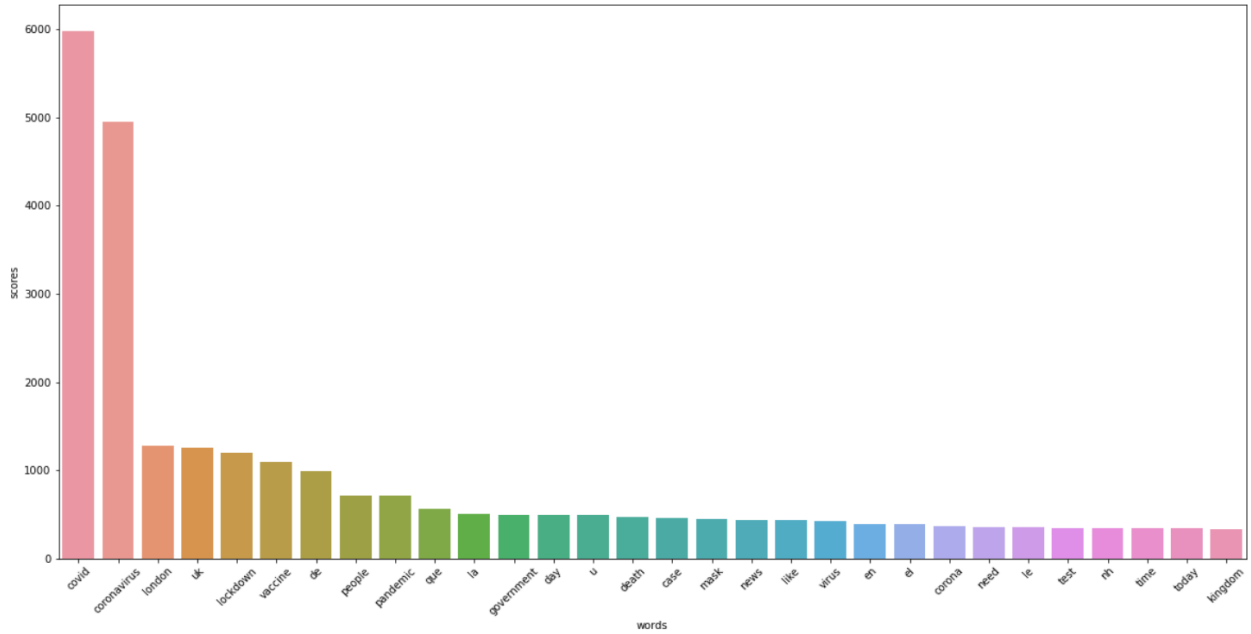


FIGURE 2.10 – Score des tokens neutres

2.7 Nuage des mots (WordCloud)

Le principe du nuage de mots est basé sur une méthode d'analyse de textes qui nous permet de mettre en évidence les mots clés les plus fréquemment utilisés dans un paragraphe de textes. Le nuage de mots est également appelé **word cloud** ou **tag cloud** en anglais.
Python propose plusieurs librairies pour réaliser des nuages de mots telles que **stylecloud** et **wordcloud**.
la librairie **wordcloud** permet de générer un nuage de mots clés pour nous aider à analyser des textes et de visualiser rapidement les mots clés en nuage de mots.
Stylecloud est une librairie python qui offre de nombreuse personnalisations pour rendre ses nuages de mots stylés et surtout les adapter à la représentation qu'on veut mettre en évidence. Notamment comme nous allons faire dans la figure 2.11, nous pouvons présenter nos nuages de mots sous forme d'icônes de réseaux sociaux, d'émoji et symboles etc...
De même, il est facile de supprimer les mots non pertinents en les stockant dans une liste avec la librairie **stopwords**, on peut donner maintenant le code correspondant :

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
def word_cloud(wd_list):
    stopwords = set(STOPWORDS)
    logomask = imread("twitter_mask.png")
    all_words = ' '.join([tweet for tweet in wd_list])
    wordcloud = WordCloud(
```

FIGURE 2.11 – Icône de twitter avec nuage des mots

```
        background_color='white',
        stopwords=stopwords,
        width=1600,
        height=800,
        random_state=1,
        colormap='jet',
        max_words=80,
        mask=logomask
        max_font_size=200).generate(all_words)
    plt.figure(figsize=(12, 10))
    plt.axis('off')
    plt.imshow(wordcloud, interpolation="bilinear");
word_cloud(df['tweet'])
```

Les figures 2.12, 2.13 et 2.14 montrent les représentations des nuages de mots des trois classes.

2.8 Conclusion

Dans ce chapitre, nous avons présenté les méthodes que nous avons utilisées pour extraire les données à partir de Twitter, et les expériences effectuées sur les données collectées. Nous avons vu aussi les différentes méthodes pour faire la représentation des tweets (nettoyage, polarité, Tokenization, worcloud...), et nous avons donner quelques interpretations des résultats obtenus et les visualisations avec des courbes qui représentent leurs évolutions.

FIGURE 2.12 – Nuage des mots positifs

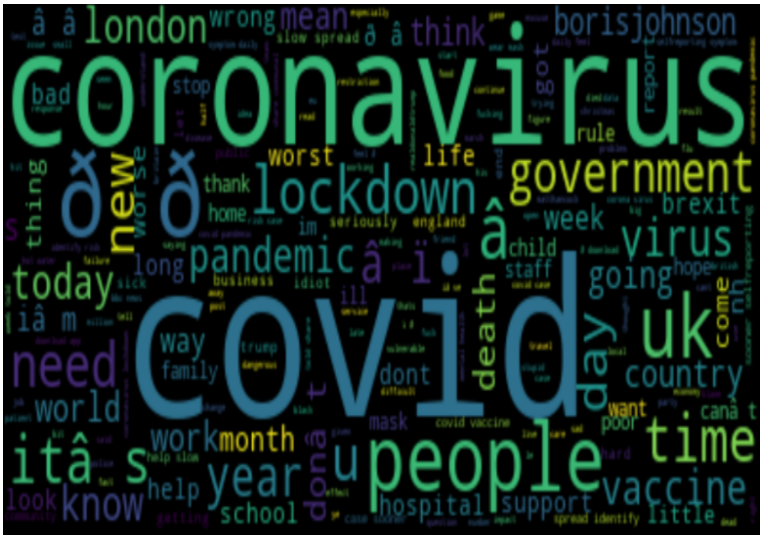
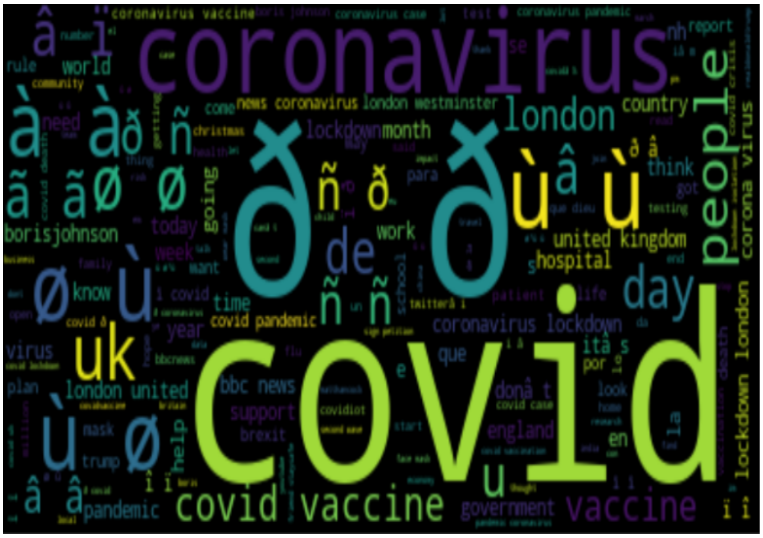


FIGURE 2.13 – Nuage des mots négatifs


coronavirus
covid
people
day
uk
vaccine
vaccine
covid
london
country
united kingdom
hospital
borisjohnson
world
lockdown
month
coronavirus
lockdown
covid pandemic
bbc news
government
pandemic
london united
coronavirus
westminster
virus
corona virus
lockdown
london
de
work
going
today
week
time
year
know
think
support
brexit
trump
mask
help
plan
report
community
rule
need
christmas
news
para
que
england
en
look
donà t
ità s
covidiot
covid case
coronavirus vaccine
coronavirus case
coronavirus pandemic
test
nh
se
number
case
thank
ià n
crisis
covid death
covid lockdown
vaccination
business
family
open
covid à
covid lockdown
bbcnews
flu
sign petition
patient
life
twitterà i
school
mant
i covid
mum
dad
travel
testing
got
dieu
end
research
home
face mask
vaccination
thought
mummy
pandemic coronavirus
local
national
million
covid vaccination
brexit
start
covid vaccine
sound save
brilliant
covid case
covid death
vaccine
ia
ia
fear
ii
ii
ii

Chapitre 3

Classification des données

3.1 Introduction

Le machine learning est un concept stipulant qu'il existe des algorithmes génériques pouvant nous révéler des informations intéressantes sur des données, sans avoir besoin de construire ou de développer un code spécifique.
Il y a aussi le deep learning qui est une forme d'intelligence artificielle, dérivée du machine learning. Pour bien comprendre, le deep dearning est basé sur les réseaux de neuronnes (Neural Network).
Ce chapitre comprend deux parties essencielles de classification :

- Le machine learning avec les deux principales catégories : l'apprentissage supervisé et l'apprentissage non supervisé.
- Le deep learning qui traite les réseaux de neuronnes, et plus précisemment les réseaux de neuronnes convolutionnelles (CNN), et les réseaux de neuronnes récurrents (RNN) et avec leurs applications.

3.2 La classification avec Machine Learning

On commence par la classification avec machine learning par les deux approches supervisé et non supervisé.

3.2.1 Les algorithmes d'apprentissage supervisé avec SVM et NB

On va présenter les points communs entre les deux algorithmes SVM et NB avant l'étape d'éxécution.

3.2.1.1 Préliminaires

On va présener les principales bibliothèques utilisées :

Librairies

Scikit-learn : est une bibliothèque libre Python destinée à l'apprentissage automatique qui est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria. Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support. Elle est conçue pour s'harmoniser avec d'autres bibliothèques libres Python, notamment NumPy et SciPy.
accuracy_score permet d'évaluer le taux de bonne classification, en comparant les classe prédites et les classes réelles.
LabelEncoder : est une librairie utilisée dans le prétraitement des données pour l'encodage des niveaux de classe. Il peut également être utilisé pour transformer des étiquettes non numériques en étiquettes numériques.
StandardScaler : implémente l'API Transformer pour calculer la moyenne et l'écart-type sur un ensemble d'apprentissage afin de pouvoir réappliquer ultérieurement la même transformation sur l'ensemble de test.
Tout d'abord, faisons quelques importations de base :

```
from sklearn.feature_extraction.text
import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, GridSearchCV,
    cross_val_score
from sklearn.metrics import f1_score, recall_score, precision_score,
    confusion_matrix
from sklearn.metrics import r2_score, roc_auc_score, roc_curve,
    classification_report
import sklearn
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
```

Train et Test

Le Corpus est divisé en deux ensembles de données : Train et Test. L'ensemble de données d'apprentissage est utilisé pour s'adapter au modèle et les prédictions sont effectuées sur l'ensemble de données de test.

Le **SVC** est un modèle de classification dont l'implémentation est basée sur **libsvm**.
On a tout d'abord importé la classe **SVC** puis on a fractionné notre dataset à l'aide de **train_test_split** de la bibliothèque **sklearn** . Les données d'entraînement auront 70% du corpus et les données de test auront les 30% restants car nous avons défini le paramètre $test_size = 0.3$.
Voyons l'ensemble de données de base avec deux types de données train et test :

```
Train_X, Test_X, Train_Y, Test_Y = model_selection.train_test_split(df['processed_tweet'],
                                                                    df['sentiment'],test_size=0.3)
```

Encodage

L'étiquette code la variable cible pour transformer les données catégorielles de type chaîne de l'ensemble de données en valeurs numériques, créant ainsi des données que le modèle peut comprendre.
Nous utilisons la classe **LabelEncoder**. Donc, tout ce que nous avons à faire, pour étiqueter encoder la variable, est d'importer la classe LabelEncoder de la bibliothèque sklearn, ajuster et transformer cette variable, puis remplacer les données texte existantes par les nouvelles données encodées.

```
Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y)
Test_Y = Encoder.fit_transform(Test_Y)
```

Vectorisation des mots

La vectorisation de mots est un processus général de transformation d'une collection de documents texte en vecteurs de caractéristiques numériques. Il existe de nombreuses méthodes pour convertir des données textuelles en vecteurs que le modèle peut comprendre. Mais la méthode la plus répandue est **TF-IDF** un acronyme qui signifie "Term Frequency - Inverse Document Frequency". Ce sont les composants des scores obtenus attribués à chaque mot.

- Fréquence des termes : cela résume la fréquence à laquelle un mot donné apparaît dans un document.
- Fréquence inversée des documents : cela réduit les mots qui apparaissent beaucoup dans les documents.

Sans entrer dans les mathématiques, les **TF-IDF** sont des scores de fréquence de mots qui tentent de mettre en évidence les mots les plus intéressants, par exemple fréquents dans un document mais pas entre les documents.
De plus, cela aidera à construire un vocabulaire de mots appris à partir des données du corpus

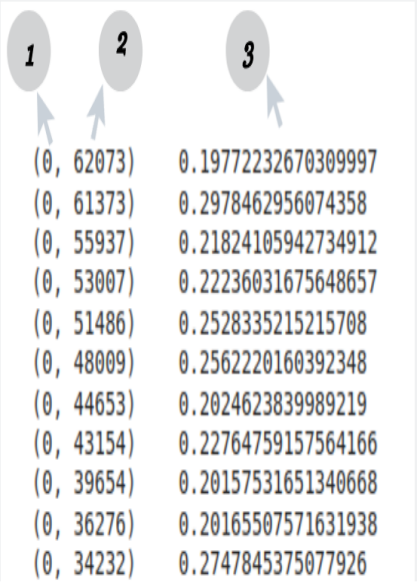


FIGURE 3.1 – Résulat de vectorisation.

et attribuera un nombre entier unique à chacun de ces mots. Il y aura un maximum de 4000 mots par des caractéristiques uniques car nous avons défini le paramètre $max_features = 4000$.

```
Tfidf_vect = TfidfVectorizer(max_features=40000)
Tfidf_vect.fit(df['processed_tweet'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X)
Test_X_Tfidf = Tfidf_vect.transform(Test_X)
```

Enfin, nous transformerons $Train_X$ et $Test_X$ en $Train_X_Tfidf$ et $Test_X_Tfidf$ vectorisés. En conséquence, chaque ligne contiendra désormais une liste de nombres entiers uniques comme indiquée dans la figure 3.1 et son importance associée calculée par **TF-IDF**.

1. Numéro de ligne de $Train_X_Tfidf$
2. Numéro entier unique de chaque mot de la première ligne
3. Score calculé par **TF-IDF Vectorizer**

Par conséquent, nos ensembles de données sont prêts à être introduits dans différents algorithmes de classification.

3.2.1.2 Algorithme de classification naive bayes (NB)

Puisque nous avons des valeurs binaires, nous utiliserons le modèle binomial Naive Bayes en python. Gardons également à l'esprit qu'il s'agit d'une stratégie longue unique-

ment, mais nous pouvons la modifier pour utiliser également des signaux courts. Le classificateur multinomial Naive Bayes convient à la classification avec des caractéristiques discrètes (par exemple, le nombre de mots pour la classification de texte). Cependant, en pratique, les comptages fractionnaires tels que **TF-IDF** peuvent également fonctionner. Voila le

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_NB = Naive.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score -> ",accuracy_score(predictions_NB,
    Test_Y)*100)
```

Le classifieur Naive Bayes a obtenu des résultats de précision de 55.15%.

```
# basic method
print('Accuracy Score: {:.4f}'.format(accuracy_score(predictions_NB,
    Test_Y)))
print('SVC f1-score  : {:.4f}'.format(f1_score(predictions_NB, Test_Y,
    average='micro')))
print('SVC precision : {:.4f}'.format(precision_score(predictions_NB,
    Test_Y, average='micro')))
print('SVC recall    : {:.4f}'.format(recall_score(predictions_NB, Test_Y
    , average='micro')))
print("\n",classification_report(predictions_NB, Test_Y))
```

Le résultat d'exécution de NB montré dans la figure 3.2. Le modèle Naive Bayes, bien qu'il soit naïf, est assez simple et efficace dans un grand nombre de cas d'utilisation de la vie réelle. Bien qu'il soit principalement distiné pour l'analyse de texte, il a été considéré comme outil de vérification dans le domaine du trading. Le modèle Naive Bayes peut également être utilisé comme tremplin vers des modèles d'apprentissage automatique basés sur une classification plus précis et plus complexes.

3.2.1.3 Algorithme de classification SVM

Tout d'abord, nous essayons l'algorithme de classification Support Vector Machine(SVM). Nous pouvons dès à présent créer notre SVM dans **sklearn** cela consiste à créer un objet **SVC** (support vector classificator). L'un des paramètres importants est le noyau **kernel**, c'est une fonction qui sert à transformer les données dans une représentation spécifique. On va donner le code suivant pour mieux comprendre ce que nous avons fait.

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
SVM.fit(Train_X_Tfidf,Train_Y)
```

```
Accuracy Score: 0.5515
SVC f1-score  : 0.5515
SVC precision : 0.5515
SVC recall    : 0.5515

              precision    recall  f1-score   support

           0       0.02      1.00      0.04         6
           1       0.40      0.83      0.54       211
           2       0.97      0.49      0.65       996

    accuracy                           0.55      1213
   macro avg       0.46      0.77      0.41      1213
weighted avg       0.86      0.55      0.63      1213
```

FIGURE 3.2 – Résulat d'exécution de NB.

```
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y)
    *100)
```

Le classifieur SVM a obtenu des résultats de précision de 72.54%.

```
# basic method
print('Accuracy Score: {:.4f}'.format(accuracy_score(predictions_SVM,
    Test_Y)))
print('SVC f1-score  : {:.4f}'.format(f1_score(predictions_SVM, Test_Y,
    average='micro')))
print('SVC precision : {:.4f}'.format(precision_score(predictions_SVM,
    Test_Y, average='micro')))
print('SVC recall    : {:.4f}'.format(recall_score(predictions_SVM,
    Test_Y, average='micro')))
print("\n",classification_report(predictions_SVM, Test_Y))
```

La fonction imprime les résultats d'exécution de SVM présenté dans la figure 3.3.
Nous concluons donc que le SVM peut non seulement effectuer des predictions fiable, mais aussi minimiser le nombre d'informations redondantes. Le SVM a également performer des résultats comparables à ceux obtenus par rapport NB.

3.2.2 Les algorithmes d'apprentissage non supervisé avec K-means

A l'inverse de l'apprentissage supervisé qui tente de trouver un modèle depuis des données labellisées, l'apprentissage non supervisé prend uniquement des données sans label.

```
Accuracy Score: 0.7255
SVC f1-score  : 0.7255
SVC precision : 0.7255
SVC recall    : 0.7255

              precision    recall  f1-score   support

           0       0.33      0.77      0.46       107
           1       0.85      0.71      0.77       547
           2       0.81      0.74      0.77       559

    accuracy                           0.73      1213
   macro avg       0.66      0.74      0.67      1213
weighted avg       0.79      0.73      0.74      1213
```

FIGURE 3.3 – Résulat d'exécution de SVM.

Ils permettent de regrouper en des ensembles, les données qui sont similaires. L'algorithme K-Means (K-moyennes) est le plus connu dans l'apprentissage non supervisé. Il s'agit d'un algorithme de Clustering.

3.2.2.1 PCA et TSNE

D'abord, on va donner les définitions de deux méthodes PCA et TSNE que l'on utilisera par la suite.

Définitions

Analyse en composantes principales (PCA) : PCA est une technique de réduction de dimensionnalité linéaire et de visualisation de données non supervisée pour des données de très grande dimension. Comme il est très difficile d'obtenir des informations en ajoutant des données de grande dimension, cela demande beaucoup de calcul. L'idée principale derrière de cette technique est de réduire la dimensionnalité des données qui est fortement corrélée en transformant l'ensemble original de vecteurs en un nouvel ensemble appelé composant principal.
PCA essaie de préserver la structure globale des données, puis il essaie de cartographier tous les clusters dans leur ensemble en raison des structures locales qui pourraient se perdre. L'application de cette technique comprend le filtrage du bruit, les extractions de caractéristiques, les prévisions boursières et l'analyse des données génétiques.
Incorporation de voisinage stochastique distribué en (t-SNE) : t-SNE est également une technique de réduction de dimensionnalité non linéaire et de visualisation de données non

supervisée. Le calcul derrière t-SNE est assez complexe mais l'idée est simple. Il incorpore les points d'une dimension supérieure à une dimension inférieure en essayant de préserver le voisinage de ce point.
Contrairement à PCA, il tente de préserver la structure locale des données en minimisant la divergence Kullback-Leibler (divergence KL) entre les deux distributions en ce qui concerne les emplacements des points sur la carte. Cette technique trouve une application dans la recherche sur la sécurité informatique, l'analyse musicale, la recherche sur le cancer, la bio-informatique et le traitement du signal biomédical.

Comparaison entre PCA et TSNE

PCA	T-SNE
C'est une technique de réduction de dimensionnalité linéaire.	C'est une technique de réduction de dimensionnalité non linéaire.
Il essaie de préserver la structure globale des données.	Il essaie de conserver la structure locale des données.
Cela ne fonctionne pas bien par rapport au t-SNE.	C'est l'une des meilleures techniques de réduction de dimensionnalité.
Il n'implique pas d'hyperparamètres.	Cela implique des hyperparamètres tels que la perplexité, le taux d'apprentissage et le nombre d'étapes.
Il est fortement affecté par les valeurs aberrantes.	Il peut gérer les valeurs aberrantes.
L'ACP est un algorithme déterministe.	Il s'agit d'un algorithme non déterministe ou aléatoire.
Cela fonctionne en faisant tourner les vecteurs pour préserver la variance.	Cela fonctionne en minimisant la distance entre le point dans un gaussien.
Nous pouvons déterminer la quantité de variance à préserver en utilisant les valeurs propres.	Nous ne pouvons pas préserver la variance à la place, nous pouvons préserver la distance en utilisant des hyperparamètres.

3.2.2.2 Conversion des données

Afin d'utiliser des données textuelles pour la modélisation prédictive, le texte doit être analysé pour supprimer certains mots, ce processus est appelé tokenisation. Ces mots doivent ensuite être codés sous forme d'entiers, ou de valeurs à virgule flottante, pour être utilisés comme entrées dans les algorithmes d'apprentissage automatique. Ce processus est appelé extraction de caractéristiques (ou vectorisation).
Le **CountVectorizer** de **Scikit-learn** est utilisé pour convertir une collection de documents

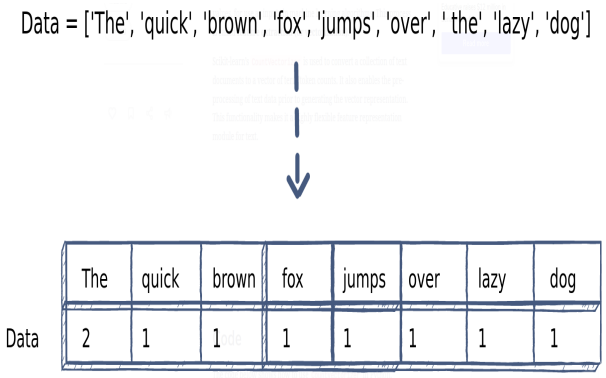


FIGURE 3.4 – Exemple explicatif de CountVectorizer.

```
(0, 38998)    1
(1, 58051)    1
(1, 13790)    1
(1, 26080)    1
(1, 57859)    1
(1, 46234)    1
(1, 53871)    1
(1, 18221)    1
(1, 33906)    1
(1, 55299)    1
```

FIGURE 3.5 – Matrice obtenu par CountVectorizer.

texte en un vecteur de nombre de termes par jetons. Il permet également le prétraitement des données textuelles avant de générer la représentation vectorielle (voir la figure 3.4).
Cette fonctionnalité en fait un module de représentation d'entités très flexible pour le texte.
Par notre algorithme, on a obtenu la matrice suivante représenté dans la figure 3.5.

3.2.2.3 Visualisation des résultats

Après avoir importé des K-means, nous devons décider le nombre de clusters qu'on veut à partir de nos données. La meilleure façon de connaître le nombre idéal de clusters, nous utiliserons le graphe de méthode du coude. Mais, pour tracer cela, nous devons calculer la somme des carrés au sein du cluster . Donc, nous allons créer une variable WCSS avec des crochets et exécuter une boucle. Voila le code qui nous permet d'évaluer notre résultat.

```
############ LE CODE DANS LE QUEL ON A CALCULER LES ITERATIONS POUR TRACER LE ELBOW GRAPH
from sklearn.cluster import KMeans
wcss=[]
for i in range(3,30):
    Kmeans=KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0,verbose=True)
    Kmeans.fit(X)
    wcss.append(Kmeans.inertia_)
#################### LE CODE DE LE GRAPH
import matplotlib.pyplot as plt
plt.plot(range(3,30),wcss)
plt.xlabel('number of clusters')
plt.ylabel('word per cluster')
plt.show()
plt.savefig("cluster.png")
######################### LE CODE PCA ET TSN-E
clusters = KMeans(n_clusters=4,init='k-means++',max_iter=300,n_init=10,random_state=0,verbose=True).fit_predict(X)

from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import matplotlib.cm as cm
def plot_tsne_pca(data, labels):
    max_label = max(labels)
    max_items = np.random.choice(range(data.shape[0]), size=1000, replace=False)
    pca = PCA(n_components=2).fit_transform(data[max_items,:].todense())
    tsne = TSNE().fit_transform(PCA(n_components=50).fit_transform(data[max_items,:].todense()))
    idx = np.random.choice(range(pca.shape[0]), size=300, replace=False)
    label_subset = labels[max_items]
    label_subset = [cm.hsv(i/max_label) for i in label_subset[idx]]
    f, ax = plt.subplots(1, 2, figsize=(14, 6))
    ax[0].scatter(pca[idx, 0], pca[idx, 1], c=label_subset)
    ax[0].set_title('PCA Cluster Plot')

    ax[1].scatter(tsne[idx, 0], tsne[idx, 1], c=label_subset)
    ax[1].set_title('TSNE Cluster Plot')
plot_tsne_pca(X, clusters)
plt.savefig("plot_tsne_pca.png")
```

Donc, dans le code ci-dessus on a déterminé ce qui suit :

- $n_clusters =$ nombre de clusters
- $init =$ méthode d'initialisation aléatoire (pour éviter tout piège d'initialisation aléatoire, nous utiliserons $k - means + +$)
- $max_iter =$ nombre maximum d'itérations (300 est la valeur par défaut)

```
Initialization complete
Iteration 0, inertia 1039546.0
Iteration 1, inertia 706796.1998390098
Iteration 2, inertia 705174.2308788303
Iteration 3, inertia 705056.3533539904
Iteration 4, inertia 704807.2503555577
Iteration 5, inertia 704700.7285178691
Iteration 6, inertia 704398.0843997772
Iteration 7, inertia 703743.4931575437
Iteration 8, inertia 703542.1442088046
Iteration 9, inertia 703535.6356318361
Iteration 10, inertia 703535.3680952885
Iteration 11, inertia 703535.3477591865
Iteration 12, inertia 703535.3301090588
Converged at iteration 12: strict convergence.
```

FIGURE 3.6 – Matrice obtenu par CountVectorizer.

- $n_init =$ nombre de fois que l'initialisation sera exécutée (10 est la valeur par défaut)
- $random_state =$ corrige toutes les valeurs aléatoires de KMeans
- $kmeans.fit$ ajustera toutes les valeurs de la variable kmeans dans notre variable X.

Puis, on a obtenus des itérations montrées dans la figure 3.6.

Maintenant, nous allons tracer notre **Elbow Graph** à travers lequel nous apprendrons à savoir quel sera un bon nombre de clusters pour nos données.

Une fois que cela est fait, nous savons que maintenant la forme du **Elbow** nous a donné le nombre idéal de grappes pour nos données qui est de 3. Donc, on répétons maintenant l'étape precedente, mais avec le nombre de clusters à 4.

Pour faire cette étape on a besoin de deux méthode PCA et TSNE que nous avons déjà défini dans la section précédente. Maintenant, on va visualiser les résultats à l'aide de deux méthodes PCA et TSNE de K-means de notre avec les figures 3.7 et 3.8.

Sur la première courbe de la figure 3.7, nous avons le résultat obtenu avec une analyse en composantes principales PCA, on vois que la séparation des données dans l'espace en deux dimensions est beaucoup moins nette. Nous pouvons voir, que pour certains couleurs comme le bleu les points correspondants sont bien regroupés entre eux. Cependant, pour d'autres couleurs comme les points associés a la couleur vers ils sont répartis de manière plus diffuse.

Sur la deuxième courbe de la figure 3.7, nous avons le résultat obtenu par réduction de dimension avec la méthode TSNE, il est clair que ce dernier a réussi de regrouper les données

FIGURE 3.7 – Nuage des points du PCA et TSNE.

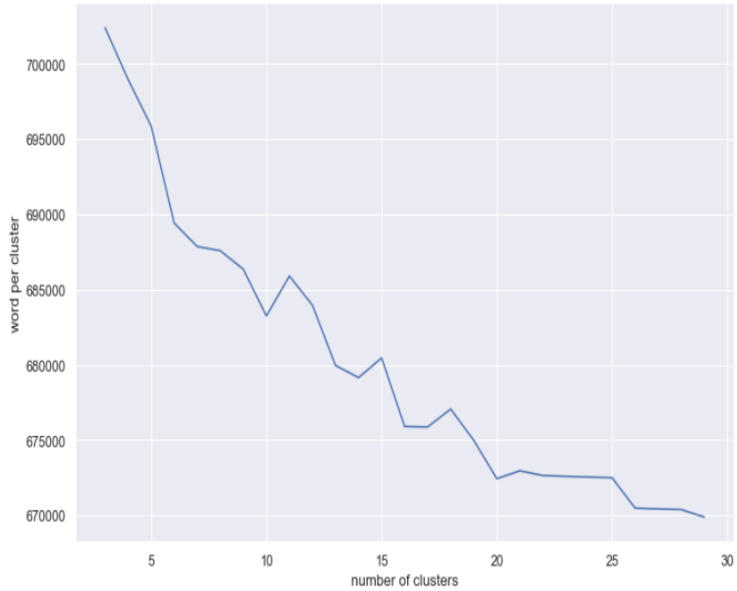


FIGURE 3.8 – Le nombre des clusters par Elbow.

proches et éloigner les données dissemblables. Les points sont représentés en grappe, chacune des grappes correspondant à une couleur different, il y a le vers, le rouge, et le bleu.
Dans la figure 3.8, nous utilisons une approche connue sous le nom de "Elbow method". Ce graphe est marqué les nombres aleatoire de clusters (entre 3 et 30) et on vois que La meilleure valeur pour le nombre de clusters est presque 10 parce que c'est dans laquelle l'endroit de la courbe se brise, aussi on vois clairement que puisque le nombre de clusters augmente la courbe de Elbow se diminue.

3.3 La classification avec Deep Learning

Avant de commencer par le processus du classement du texte par des modèles de Deep learning. On va présenter les principles bibliothèques de Deep Learning.

3.3.1 Présentation des bibliothèques utilisées en DL

3.3.1.1 TensorFlow

Il s'agit d'une bibliothèque Open Source de calcul numérique et de Machine Learning compatible avec le langage Python. Elle simplifie le processus d'acquisition de données, d'entraînement des modèles de Machine Learning, de génération de prédictions et de raffinement des résultats futurs. TensorFlow regroupe des modèles et des algorithmes de Machine Learning et de Deep Learning.
Le langage Python offre une API front-end pratique et confortable pour créer des applications à l'aide de ce framework. Ces applications peuvent ensuite être exécutées en C++.
Il est possible de se servir de ce framework pour entraîner et exécuter des réseaux de neurones profonds pour la classification de chiffres manuscrits, pour la reconnaissance d'image, pour le plongement lexical, pour les réseaux de neurones récurrents, pour les modèles "sequence-to-sequence" de traduction automatique, pour le traitement naturel du langage, et pour les simulations basées sur les équations aux dérivées partielles.
Son nom s'inspire notamment du fait que les opérations actuelles des réseaux de neurones s'effectuent principalement via une table multidimensionnelle de données, appelée Tenseurs (Tensor) qui est l'équivalent d'une matrice. Aujourd'hui, ce framework est utilisé par les grandes entreprises [38] comme Intel, Twitter, Lenovo, Nvidia, etc...

3.3.1.2 Keras

Keras [14] est une bibliothèque open source écrite en Python (sous licence MIT) basée principalement sur les travaux du développeur de Google François Chollet dans le cadre du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System). Une première version du logiciel multiplateforme a été publiée le 28 mars 2015. Le but de cette

bibliothèque est de permettre la constitution rapide de réseaux neuronaux. Dans ce cadre, Keras ne fonctionne pas comme un framework propre mais comme une interface de programmation applicative (API) pour l'accès et la programmation de différents frameworks d'apprentissage automatique. Theano, Microsoft Cognitive Toolkit (anciennement CNTK) et TensorFlow font notamment partie des frameworks pris en charge par Keras.

Un autre outil permettrait de donner plus de sens à la transformation vectorielle d'un texte et permettrait aussi une meilleure classification.

3.3.1.3 Word Embeddings

Word embeddings est le mappage de mots à des vecteurs de nombres réels dans un espace dimensionnel réduit. Avec cette méthode, il est capable de capturer le contexte d'un mot dans un document, la similarité sémantique, et la relation entre les mots, etc...
il y a trois méthodes pour obtenir Embeddings.

- Modèles descendants des "Modèles du langage neuronal" : Word2vec, GloVE
- Décomposition de valeur singulière
- Le clustering Brown

Au cours de nos expériences, nous avons créé deux modèles avec des architectures différentes, où le premier était un modèle RNN (Recurrent Neural Networks) et le second était un modèle CNN (Convolutional Neural Networks).

3.3.2 Le classificateur RNN

Définition : Un réseau de neurones récurrent (Récurrent Neural Network) est un réseau de neurones dont le graphe de connexion contient au moins un cycle. Au cours des dernières années, un type de RNN est devenu la norme grâce à ses excellentes performances sur des tâches aussi nombreuses que variées : les réseaux de neurones à base de cellules Long Short-Term Memory (LSTM). Tout comme les systèmes de contrôle, les réseaux neuronaux récurrents fonctionnent selon le principe du bouclage et du chaînage, qui peut être élaboré à l'aide de la figure 3.9.
Importons d'abord les bibliothèques et les données requises.

```
import re
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from keras.models import Sequential, load_model
from keras.layers import Dense, LSTM, Embedding, Dropout
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
```

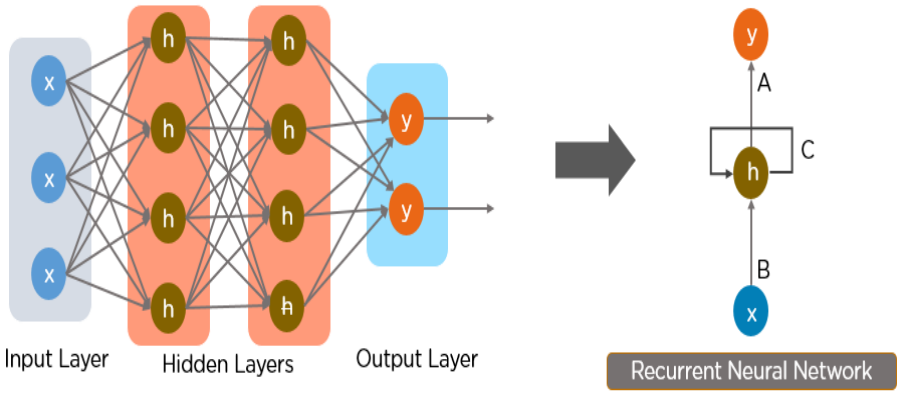


FIGURE 3.9 – Réseau de neurones récurrent simple.

Et pour réaliser notre architecture de RNN, nous avons besoin du code suivant :

```
import tensorflow
#Creating model
input_data = tensorflow.keras.layers.Input(shape=(500))
model = Sequential()
model.add(Embedding(5000, 256, input_length=X.shape[1]))
model.add(Dropout(0.3))
model.add(LSTM(256, return_sequences=True, dropout=0.3, recurrent_dropout
    =0.2))
model.add(LSTM(256, dropout=0.3, recurrent_dropout=0.2))
model.add(Dense(3, activation='softmax'))
output_data = tensorflow.keras.layers.Activation('sigmoid')
#model = tensorflow.keras.models.Model(inputs=input_data, outputs=
    output_data)
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics
    =['accuracy'])
model.summary()
```

Dans ce qui suit, nous présentons l'architecture de RNN où la figure 3.10 représente les différentes couches que contient notre modèle.

- La couche Embedding : nous avons appliqué le modèle GloVe pour Twitter avec une dimension égale à 200.
- La couche Dropout : nous avons appliqué une seule couche Dropout pour la régularisation
- La couche entièrement connectée : nous avons utilisé trois couches entièrement connectées avec 256 neurones cachés et exploitation de la fonction **softmax** pour prédire la sortie.
- La couche LSTM : nous ajoutons deux couches de LSTM avec 256 unités cachées pour chaque couche.

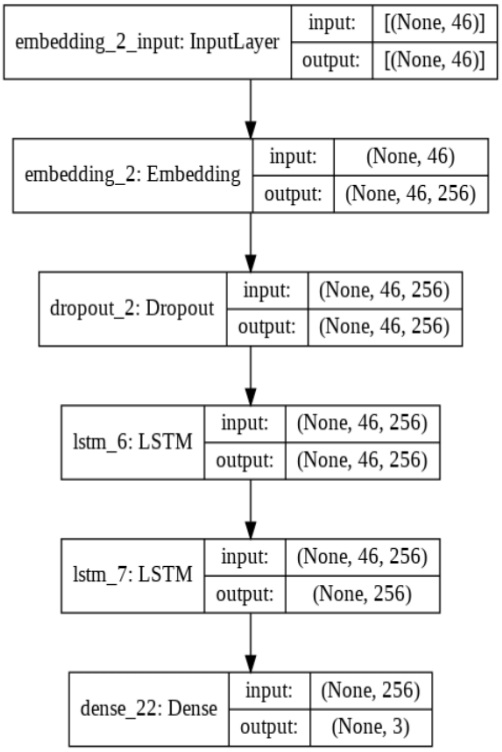


FIGURE 3.10 – L'architecture du RNN.

Nous allons maintenant évaluer les performances du modèle avec l'utilisation du code ci-dessous.

```
#Training model
batch_size = 128
epochs = 10
history1 = model.fit(X_train, y_train, epochs=epochs, batch_size=
    batch_size, validation_split=0.6, verbose=1)
# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(X_train,y_train)
print("test loss, test acc:", results)
######### Plot 1
plt.plot(history1.history['loss'], label='training data')
plt.plot(history1.history['val_loss'], label='testing data')
plt.title('Loss for Text Classification')
plt.ylabel('Loss value')
plt.xlabel('No. epoch')
plt.legend(loc="upper left")
plt.show()
########## Plot 2
```

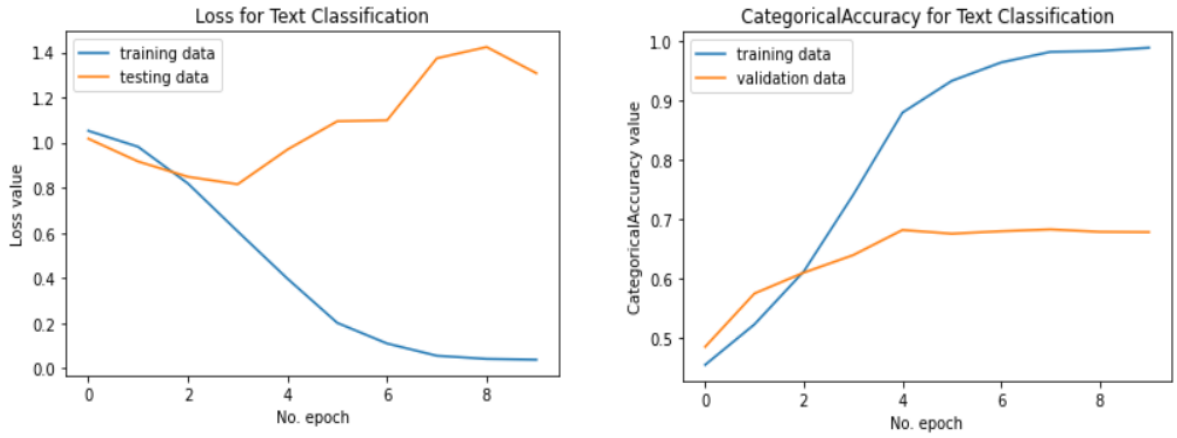


FIGURE 3.11 – Précision et erreur du modèle RNN.

```
plt.plot(history1.history['accuracy'], label='training data')
#plt.plot(history1.history['acc'], label='training')
plt.plot(history1.history['val_accuracy'], label='validation data')
#plt.plot(history1.history['val_acc'], label='testing')
plt.title('CategoricalAccuracy for Text Classification')
plt.ylabel('CategoricalAccuracy value')
plt.xlabel('No. epoch')
plt.legend(loc="upper left")
plt.show()
```

Maintenant, on va visualiser les résultats de notre modèle RNN avec la figure 3.11. D'après la figure 3.11, la précision de l'apprentissage augmente avec le nombre d'époque, jusqu'à elle est presque stable, ceci reflète qu'à chaque époque le modèle apprenne plus d'informations.
Par contre validation elle est presque stable entre $[0.1; 0.5]$, avec un overfitting. De même, l'erreur d'apprentissage et diminue, d'autre part la validation augment avec le nombre d'époque.

3.3.3 Le classificateur CNN

Definition : Un réseau convolutionnel (CNN : Convolutional Neural Networks) suppose une structure spatiale particulière dans son entrée. En particulier, il suppose que les entrées qui sont proches les unes des autres dans l'entrée originale sont sémantiquement liées [**?**].
CNN est une séquence de couches, et chaque couche transforme un volume d'activations en un autre par une fonction différentiable. Les trois principaux types de couches pour construire ce type de réseau sont : couche convolutive, couche de pooling et couche entièrement connectée [33].

- La couche convolutive : c'est la couche la plus importante et le coeur des éléments constitutifs du réseau convolutif, et c'est aussi elle qui effectue le plus de calculs lourd.

- La couche de pooling : il est courant d'insérer périodiquement une couche Pooling dans ce type d'architecture. Sa fonction est de réduire progressivement la taille spatiale de la représentation pour réduire le nombre de paramètres et de calculs dans le réseau, et donc de contrôler également le overfitting.
- La couche entièrement connectée : les neurones d'une couche entièrement connectée ont des connexions complètes à toutes les activations de la couche précédente.

Maintenant, on va donner le code ci-dessous de CNN.

```
######  Les librairies ###########
from keras.layers import Dense, Input
from keras.layers import GlobalAveragePooling1D, Embedding
from keras.models import Model

EMBEDDING_DIM = 50
N_CLASSES = 2
######CNN avec la couche de poling
# input: a sequence of MAX_SEQUENCE_LENGTH integers
sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
embedding_layer = Embedding(MAX_NB_WORDS, EMBEDDING_DIM,
                            input_length=MAX_SEQUENCE_LENGTH,
                            trainable=True)
embedded_sequences = embedding_layer(sequence_input)
#L'utilisation du bloc de mise en commun 1D Global average peut remplacer
    les blocs entièrement connectés de votre CNN.
average = GlobalAveragePooling1D()(embedded_sequences)
predictions = Dense(N_CLASSES, activation='softmax')(average)
model_1 = Model(sequence_input, predictions)
model_1.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['acc'])
model_1.summary()
############# le code des 2 graph de loss and accuracy ################
plt.plot(history1.history['training'], label='training')
plt.plot(history1.history['validation'], label='validation')
plt.title('Loss for Text Classification')
plt.ylabel('Loss value')
plt.xlabel('No. epoch')
plt.legend(loc="upper left")
plt.show()
```

La figure 3.12 représente les différentes couches que contient notre modèle CNN.

- La couche Embedding : nous avons appliqué le modèle GloVe pour Twitter et choisi une dimension égale à 200.
- Les Couches Conv1D et MaxPooling1D : nous utilisons trois couches convolutives d'une dimension suivie de trois couches de max pooling.
- La couche Dropout : nous avons appliqué une seule couche Dropout pour la régularisation.

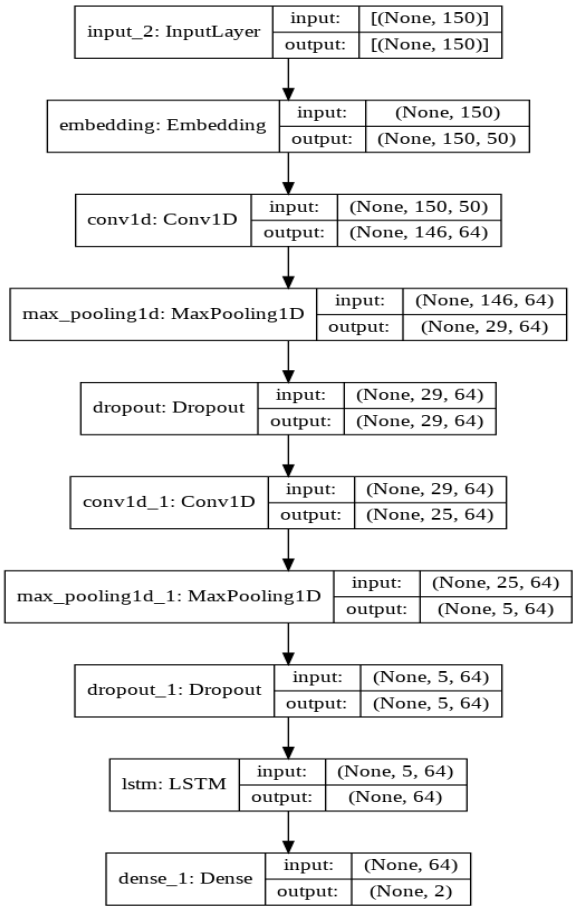


FIGURE 3.12 – L'architecture du CNN.

– La couche entièrement connectée : nous avons utilisé trois couches entièrement connectées avec 200 neurones cachés et utilisé la fonction softmax pour prédire la sortie.

On va représenter les résultats de notre modèle CNN avec la figure 3.13.

D'après la figure 3.13, la précision de l'apprentissage augmente avec le nombre d'époque, jusqu'à elle est presque stable, ceci reflète qu'à chaque époque le modèle apprenne plus d'informations.

Par contre "validation", elle est presque stable entre $[0.992; 0.993]$, avec un overfitting. De même, l'erreur d'apprentissage et diminue, d'autre part la validation augment avec le nombre d'époque.

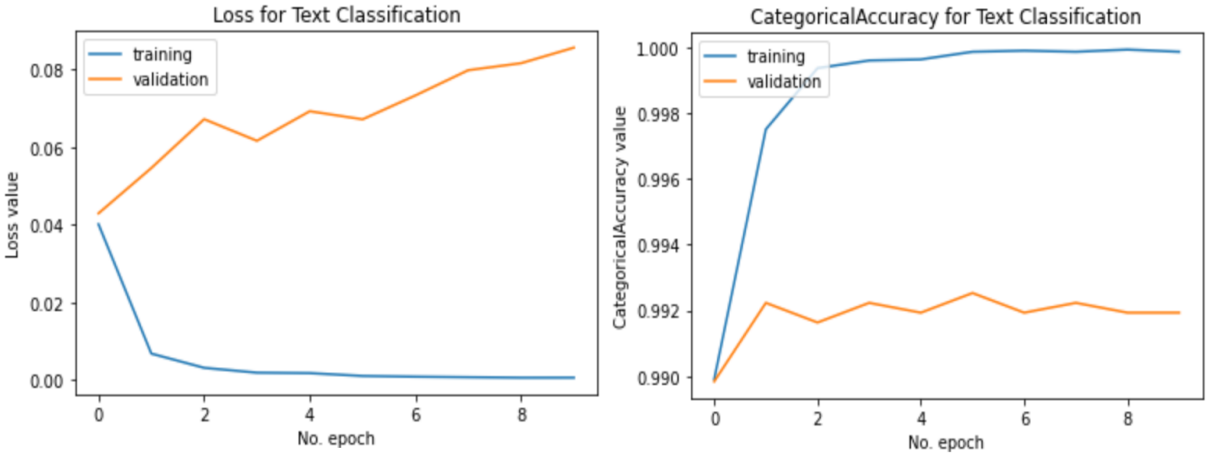


FIGURE 3.13 – Précision et erreur du modèle CNN.

3.4 Conclusion

Dans ce chapitre, nous avons traité deux parties essencielles de classification ML et DL avec ces différentes techniques telles que (Stopwords, TF-IDF et Word Embedding) pour réaliser nos expérimentations :

- Le machine learning avec les deux principales catégories : l'apprentissage supervisé par deux algorithme NB et SVM et l'apprentissage non supervisé avec K-means.
- Le deep learning qui traite les réseaux de neurones, et plus précisément les réseaux de neurones convolutionnelles (CNN), et les réseaux de neurones récurrents (RNN) et avec leurs applications.

Conclusion Générale et Perspictives

Dans ce mémoire, nous avons présenté : les principaux travaux qui s'intéressaient à l'analyse des sentiments, les différences entre les blogs et les Microblogs, la plateforme Twitter et les principaux travaux qui analysent les données de Twitter.

Nous avons présenté, également, les différentes techniques et études soit pour l'extraction des données ou pour la classification concernant Machine learning et Deep Learning que nous avons utilisées dans nos expérimentations.

Nous avons implémenté des méthodes liées à l'analyse de sentiments avec des données provenant de la plate-forme Twitter qui portent sur le Royaume-Uni et sont écrits en anglais standard pendant Covid-19.
L'objective de ce mémoire est la détection des polarités des tweets selon trois voies, positive, négative et neutre. De même, nous avons fait la classification des tweets en utilisant une source de données par l'extraction à l'aide de twint (Dataset, format .csv).
Ensuite,les classificateurs que nous avons appliqués sont (SVM, NB et clustring avec k-means) pour Machine Learninig et (RNN et CNN) pour Deep Learning.
En vue de vérifier l'efficacité des différents algorithmes, nous avons effectué plusieurs scénarios en utilisant des méthodes telles que les Stopwords removal, Term Frequency Inverse Document Frequency TF-IDF et Word Embedding.
Les résultats que nous avons obtenus sont très encourageant et très prometteurs qui montrent son éfficacité et sa performance.

Pour les travaux futurs, nous pouvons citer :
- L'application d'autres classificateurs et l'utilisation d'autres fonctionnalités.
- L'analyse de sentiments à partir des commentaires par d'autres médias sociaux comme (Facebook, instagram,...).
- L'analyse de sentiments de textes extraits de vidéos.

Bibliographie

[1] Alexander Pak and Patrick Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining", Université de Paris-Sud, Laboratoire LIMSI- CNRS,France 2010.

[2] Ansen, B.J., Zhang, M., Sobel, K., Chowdury, A.,"Twitter power : Tweets as electronic word of mouth", Journal of the American societyfor information science and technology 60, pp. 2169-2188, 2009.

[3] Antonio Gulli, Sujit Pal, "Deep Learning with Keras" 2017.

[4] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow and Rebecca Passonneau, "Sentiment analysis of Twitter data", LSM 11 Proceedings of the Workshop on Languages in Social Media, 2011.

[5] Arti Buche, Dr. M. B. Chandak and Akshay Zadgaonkar, "Opinion mining and analysis :a survey", International Journal on Natural Language Computing, India 2013.

[6] Beyond and Vartul Mittal,"Le DL est essentiellemnt basé sur les réseaux de neuronnes" Top 15 Deep Learning applications that will rule the world in 2018, 3 Oct 2017.

[7] Carvalho J.P. ,Rosa H. , Brogueira G. and Batista F. "MISNIS : An intelligent platform for twitter topic mining", Expert Syst. Appl., vol. 89, pp. 374-388, 2017.

[8] Christine Doig, "Anaconda for R users : SparkR and rBokeh", sur Developer Blog, Continuum Analytics, 1er février 2016.

[9] De Choudhury, M. ; Counts, S. ; Horvitz, E. "Predicting Postpartum Changes in Emotion and Behavior via Social Media", In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Paris, France, pp. 3267-3276, 2013.

[10] Fred Colantonio, Communication professionnelle en ligne : comprendre et exploiter les médias et réeseaux sociaux,Edipro, 2011.

[11] (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 12, No. 2, 2021.

[12] Python Deep Learning. Ivan Vasilev, Daniel Slater, Gianmario Spacagna, Peter Roelants, Valentino Zocca. 2019

[13] Ivan Vasilev, Daniel Slater, Gianmario Spacagna, Peter Roelants,Valentino Zocca, "Python Deep Learning", 2019.

[14] "Keras Documentation", sur keras.io,consulté le 25 septembre 2019.

[15] Laurent Dijoux, Boostez votre business avec Twitter,Almabic, 2009.

[16] Matthew Eric Glassman, Jacob R. Straus and Colleen J. Shogan, "Social Networking and Constituent Communications : Members Use of Twitter and Facebook During a Two-Month Period in the 112th Congress",Congressional Research Service, 2009.

[17] McCallum L and Partridge J. "Epidemiological characteristics of the influenza A(H1N1) 2009 pandemic in the Western Pacific Region", Western Pac Surveill Response J., 1(1)pp., 5-11, 2010.

[18] Mokri M.Z, "Classification des images avec les réseaux de neurones convolutionnels", Université Abou Bakr Belkaid Tlemcen Faculté Des Sciences Département D'informatique mémoire, 2017.

[19] Pascal Marichalar, "Savoir et prévoir : Première chronologie de l'émergence du Covid-19", La Vie des idées, 25 mars 2020.

[20] M. Ra, B. Ab, and S. Kc, "COVID 19 Outbreak : Tweet based Analysis and Visualization towards the Influence of Coronavirus in the World".

[21] Reza Bosagh Zadeh, Bharath Ramsundar, "TensorFlow for Deep Learning", 2017.

[22] Samuel, J. ; Holowczak, R. ; Benbunan-Fich, R. ; Levine, I. "Automating Discovery of Dominance in Synchronous Computer-Mediated Communication", In Proceedings of the 2014 47th IEEE Hawaii International Conference on System Sciences, Waikoloa, HI, USA, pp. 1804-1812, 6-9 January 2014.

[23] Samuel, J., Ali, G. G., Rahman, M. M., Esawi, E., and Samuel, Y."COVID-19 Public Sentiment Insights and Machine Learning for Tweets Classification", Information, 11(6), 314. doi :10.3390/info11060314, 2020.

[24] Serge Proulx, Mélanie Millette et Lorna Heaton Médias sociaux : enjeux pour la communication. Presse de l'Université du Québec, 2011.

[25] Vivek Kumar Singh and Debanjan Mahata, "A clustering and opinion mining approach to socio-political analysis of the blogosphere",İ Computational Intelligence and Computing Research (ICCIC), IEEE International Conference on 2010.

[26] Wang, J., Chen, Q., and Chen, Y. "RBF kernel based support vector machine with universal approximation and its application", In International Symposium on Neural Networks, pp. 512-517, Springer, Berlin, Heidelberg, 2004.

[27] Yann LeCun, Yoshua Bengio Geoffrey Hinton "Deep Learning", 2015.

[28] https ://dataanalyticspost.com/Lexique/k-means-ou-k-moyennes/

[29] https ://www.datacamp.com/community/tutorials/

[30] https ://actualiteinformatique.fr

[31] https ://www.journaldunet.com/solutions/dsi/1493139-intelligence-artificielle/

[32] https ://datafranca.org/wiki/Analyse-des-sentiments

[33] http ://cs231n.github.io/convolutional-networks/overview, 25 Mars 2019.

[34] https ://ai.google, 20 Avril 2019.

[35] https ://twitter.com/

[36] https ://docs.tweepy.org/en/stable/

[37] https ://www.digitalocean.com/community/tutorials/how-to-install-python-3-and-set-up-a-programming-environment-on-an-ubuntu-18-04-server

[38] https ://www.tensorflow.org/

[39] https ://towardsdatascience.com/analysis-on-tweets-using-python-and-twint-c7e6ebce8805

[40] https ://www.cs.toronto.edu/tijmen/csc321/slides/lecture slides lec2.pdf, 2014.