République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abou Bakr Belkaid – Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études pour l'obtention du diplôme de Master en Informatique

Option : Modèles Intelligents Et Décisions (M.I.D)

Thème

Analyse comparative des performances pour la classification de textes « RNN vs Transformers »

Réalisé par :

- **BENABADJI Nazim**

Présenté le 13 Juin 2024 devant le jury composé de :

Mr MEZIANE Abdelfettah	*Président*
Mr SMAHI Mohammed Ismail	*Encadrant*
Mr Meghebbar Ayoub	*Co-Encadrant*
Mr FEKAR Mohammed Riyadh El Mansour	*Examinateur*

Année universitaire : 2023-2024

REMERCIEMENTS

Je commence par exprimer ma gratitude envers Allah le miséricordieux, pour m'avoir donné la force, la patience et le courage nécessaires pour achever cette étape importante de ma vie académique.

Je remercie mon encadrant monsieur SMAHI Ismail pour m'avoir guidé et conseillé. Merci pour votre attention et votre disponibilité.

Mes remerciements vont aussi à mon co-encadrant monsieur MEGHEBBAR Ayoub qui a été à mes côtés chaque fois que j'ai eu besoin de lui.

Je remercie sincèrement Monsieur MEZIANE Abdelfettah de m'avoir honoré par sa présence et d'avoir accepté de présider le jury de ce mémoire.

Mes remerciements vont aussi à Monsieur FEKAR Mohammed Riyadh El Mansour d'avoir accepté d'examiner ce travail et de faire partie de ce jury.

Enfin, je voudrais remercier tout ceux qui ont participé et contribué de prés ou de loin à la réalisation de ce projet.

DÉDICACES

Je dédie ce modeste travail à mes chers parents, piliers inébranlables de mon éducation, dont les prières incessantes, le soutien indéfectible et les sacrifices ont tracé le chemin de ma réussite. Que Dieu leur accorde santé et longue vie, et que leur amour continue à illuminer ma route.

À ma chère sœur, complice de mes joies et soutien dans mes défis, je t'adresse toute ma reconnaissance pour ton appui et tes encouragements constants.

À mes amis de la promotion de 2ème année master, nous avons partagé des moments précieux et construit des souvenirs inoubliables. Votre amitié a enrichi mon parcours académique et personnel.

À mes chers amis de toujours, dont la présence et le soutien ont été des boucliers face aux défis de la vie, je vous remercie du fond du cœur pour votre fidélité et votre amitié sincère.

TABLE DES MATIÈRES

Introduction générale **1**
Contexte 1
Problématique 1
Objectifs 2
Plan du mémoire 2

1 Méthodes de Classification de texte **4**
1.1 Introduction 4
1.2 Modèles mathématique et deep learning 6
1.2.1 Naive Bayes (NB) 7
1.2.2 K-Plus Proche Voisin (KNN) 8
1.2.3 Support Vector Machine (SVM) 9
1.3 Modèles profonds 10
1.3.1 Réseau Neuronal Convolutif 11
1.3.2 Réseau Neuronal Récurrents 13
1.3.3 Transformers 15
1.4 Conclusion 17

2 Mise en pratique et démarche **18**
2.1 Introduction 18
2.2 Chargement du Dataset 21
2.3 Preprocessing 22
2.3.1 Stopwords 22
2.3.2 Suppression de mots et nettoyage de données 23
2.3.3 Stemming et Lemmatization 24
2.3.4 Tokenization 25
2.3.5 Padding 26
2.4 Choix et Création du model 26
2.4.1 RNN : 26
2.4.2 Transformers : 28

2.5 Conclusion 30

3 Discussion et comparaison des résultats **31**
3.1 Introduction 31
3.2 Stopwords avec NLTK 31
3.3 Stemming vs Lemmatization 32
3.4 Tokenization avec NLTK 36
3.5 Résultat Du Padding 36
3.6 RNN vs Transformers 37
3.6.1 RNN 38
3.6.2 Transformers (BERT) 42
3.7 Environnement de développement 46
3.7.1 Langages et Outils utilisés 46
3.7.2 Bibliothèques Utilisées 46
3.8 Description et utilisation du DataSet 47
3.9 Conclusion 48

Conclusion **49**
Rétrospective 49
Perspectives 50

TABLE DES FIGURES

1.1 Processus général de la classification de texte avec des méthodes classiques dans chaque module[5]. 5
1.2 La structure de word2vec, incluant CBOW et Skip-gram[5]. 7
1.3 La structure de Naïve Bayes[5]. 7
1.4 La structure du KNN avec k = 4 (Chaque nœud représente un texte et les nœuds ayant des contours différents représentent des catégories différentes[5].) 8
1.5 Structure du SVM (Chaque nœud représente un texte et les nœuds ayant des contours différents représentent des catégories différentes.)[5] 10
1.6 Représentation des Layers dans les Reseaux de neuronnes convolutifs.[15] . 12
1.7 Type de Reseaux de neuronnes récurrents. 13
1.8 Architecture séquenciel des Reseaux de neuronnes[5]. 14
1.9 architecture Transformers. 16

2.1 Démarche de la classification de textes. 18
2.2 example de représentation pour le word embedding. 27
2.3 Utilisation de BERT Large. 28
2.4 Utilisation de DistilBERT. 29
2.5 Utilisation de Electra. 29
2.6 Utilisation de RoBerta. 30

3.1 Séquences Avant le Stemming 33
3.2 Séquences Après le Stemming 33
3.3 Séquences avant la Lemmatisation 34
3.4 Séquences après la Lemmatisation 34
3.5 Résultat de la Tokenization. 36
3.6 Résultat du Padding. 37
3.7 Représentation du Loss et de l'accuracy. 38
3.8 Représentation du Loss et de l'accuracy. 39
3.9 Accuracy et loss pour la Validation croisée. 40
3.10 Représentation du Loss et de l'accuracy pour Bert Large. 42
3.11 Représentation du Loss et de l'accuracy pour Distill Bert 15%. 42

3.12 Représentation du Loss et de l'accuracy pour Distill Bert 100% 43
3.13 Représentation du Loss et de l'accuracy pour Electra. 43
3.14 Représentation du Loss et de l'accuracy pour Roberta. 44

LISTE DES TABLEAUX

3.1 Liste des stopwords les plus importants. 32
3.2 Comparaison des résultats du Dataset entre le Stemming et la Lemmatisation 35
3.3 Comparaison des métriques pour différentes approches RNN. 41
3.4 Performance des différents modèles de BERT 44
3.5 Tableaux représentatifs des performances pour les RNN et les Transformers. 45

ACRONYMES

BERT Bidirectional Encoder Representations from Transformers. 2, 6, 11, 15, 16, 28, 29, 45

BOW Bag-of-Words. 6

CNN Convolutional Neural Network. 6, 11, 12

CSV Comma-Separated Values. 21

DNN Deep Neural Network. 5, 6, 10

KNN K-Nearest Neighbors. 5, 6, 8

LLM Large Language Model. 15

LSTM Long Short-Term Memory. 2, 14, 26

MGP modèles graphiques probabilistes. 7

NB Naive Bayes. 5–7

NLP Natural Language Processing. 4, 11, 13, 15, 24

PFE Projet De Fin D'Etude. 47

RNN Recurrent Neural Network. 2, 11, 13, 14, 45

SA Analyse de sentiments. 4, 11

SVM Support Vector Machine. 5, 6, 9

TALN Traitement Automatique du Langage Naturel. 1

TF-IDF Term Frequency-Inverse Document Frequency. 6

TL L'Etiquetage Des Sujets. 11

INTRODUCTION GÉNÉRALE

Contexte

Dans un monde où la quantité de données textuelles disponibles explose chaque jour, la classification de texte devient un domaine crucial de l'intelligence artificielle et du traitement automatique du langage naturel (TALN). La capacité à analyser et à catégoriser efficacement de vastes quantités de texte présente un intérêt majeur dans de nombreux domaines, notamment la recherche d'informations, la veille stratégique, la recommandation de contenu et bien d'autres applications.

La classification de texte consiste à attribuer automatiquement des étiquettes ou des catégories à des documents textuels en fonction de leur contenu. Cette tâche peut être réalisée à l'aide de diverses techniques d'apprentissage automatique et de modèles de traitement du langage naturel. L'objectif principal de ce mémoire est d'explorer en profondeur 02 différentes approches, méthodes et algorithmes utilisés dans la classification de texte, en mettant l'accent sur leurs efficacités, ainsi que leurs précisions.

Le mémoire examinera également les défis spécifiques rencontrés dans la classification de texte, tels que la gestion du déséquilibre de classe, la représentation des données textuelles, le prétraitement du texte et l'interprétabilité des modèles. En outre, il abordera les avancées récentes et les tendances émergentes dans ce domaine, telles que l'utilisation de réseaux de neurones profonds, l'exploitation de l'apprentissage par transfert et l'intégration de techniques d'apprentissage actif pour améliorer la qualité des modèles de classification.

Problématique

Dans le domaine de la classification de texte, l'évolution rapide des techniques d'apprentissage automatique et de traitement automatique du langage naturel ouvre de nouvelles perspectives quant à l'amélioration de l'efficacité, de la précision et de la généralisation

des systèmes de classification. Parmi les avancées les plus marquantes, l'émergence des réseaux de neurones récurrents (RNN), en particulier les architectures à mémoire à court et long terme (LSTM), ainsi que des modèles à attention tels que BERT (Bidirectional Encoder Representations from Transformers), suscite un intérêt considérable.

Cette problématique vise donc à examiner comment ces deux approches, les RNN (LSTM) et les Transformers (BERT), peuvent être exploitées pour répondre aux défis complexes de la classification de texte.

Objectifs

Les objectifs de cette étude comprennent l'évaluation comparative de la performance des RNN (LSTM) et des Transformers (BERT) dans divers scénarios de classification de texte, en prenant en compte des facteurs tels que la taille du jeu de données, la complexité de la tâche de classification et la disponibilité des ressources informatiques. Cette analyse approfondie vise à comprendre les forces et les faiblesses de chaque approche, ainsi qu'à fournir des indications précieuses sur les meilleures pratiques et les stratégies d'optimisation pour leur application efficace dans des contextes réels.

Cette étude comparative vise aussi à contribuer au développement de systèmes de classification de texte plus performants, adaptés aux besoins spécifiques des applications réelles, tout en favorisant une compréhension approfondie des mécanismes sous-jacents à ces techniques d'apprentissage automatique avancées.

Plan du mémoire

Dans cette étude, nous abordons le problème de la classification de texte en utilisant des approches d'apprentissage automatique, en mettant particulièrement l'accent sur les réseaux de neurones récurrents (RNN) et les modèles Transformer. Nous commençons par une revue des algorithmes d'apprentissage traditionnels utilisés dans la classification de texte, tels que Naive Bayes et les SVM. Ensuite, nous explorons en détail les RNN, qui sont des architectures de réseaux de neurones capables de traiter des séquences de données, et leur application à la classification de texte. Nous poursuivons avec une présentation des modèles Transformer, qui ont récemment montré des performances impressionnantes dans diverses tâches de traitement du langage naturel.

Dans cette section, nous examinons les principes de fonctionnement des RNN et leur utilisation dans la classification de texte. Nous nous concentrons spécifiquement sur un type de RNN appelé Long Short-Term Memory (LSTM) et décrivons en détail sa conception et son fonctionnement dans le contexte de la classification de texte. En outre, nous présentons une méthodologie d'entraînement et d'évaluation des modèles RNN, suivie d'une analyse approfondie des performances et des résultats obtenus dans nos expériences.

Nous introduisons ensuite les modèles Transformer, une architecture révolutionnaire qui a transformé le domaine du traitement du langage naturel. Nous expliquons les concepts

fondamentaux des Transformers et leur application à la classification de texte. En particulier, nous examinons le modèle BERT (Bidirectional Encoder Representations from Transformers) et détaillons sa conception, son processus d'entraînement et d'évaluation, ainsi que les résultats de nos expériences.

Dans cette section, nous comparons les performances des approches basées sur les RNN et les Transformers pour la classification de texte. Nous discutons des avantages et des inconvénients de chaque méthode, en mettant en lumière leurs capacités respectives et leurs limitations. De plus, nous abordons les défis spécifiques associés à ces modèles et explorons les opportunités de recherche futures dans le domaine de la classification de texte.

En conclusion, cette étude a mis en lumière les principales découvertes concernant la classification de texte en utilisant des approches basées sur les RNN et les Transformers. Nous avons souligné l'importance de ces méthodes dans le domaine du traitement automatique du langage naturel et proposé des perspectives pour des recherches futures visant à améliorer les performances des systèmes de classification de texte.

CHAPITRE

1

MÉTHODES DE CLASSIFICATION DE TEXTE

1.1 Introduction

La classification des textes ou aussi appelée la procédure de désignation d'étiquettes prédéfinies pour les textes - est une tâche essentielle et significative dans de nombreuses applications de **traitement du langage naturel (NLP)**, telles que **l'analyse des sentiments (SA)**[1, 2], **l'étiquetage des sujets (TL)**[3] [4], la réponse aux questions et la classification des actes de dialogue. À l'ère de l'explosion de l'information, le traitement et la classification manuels de grandes quantités de données textuelles prennent du temps et posent problèmes. En outre, la précision de la classification manuelle des textes peut être facilement influencée par des facteurs humains, tels que la fatigue et l'expertise. Il est souhaitable d'utiliser des méthodes d'apprentissage automatique pour automatiser la procédure de classification des textes afin d'obtenir des résultats plus fiables et moins subjectifs. En outre, cela peut également contribuer à améliorer l'efficacité de la recherche d'informations et à atténuer le problème de la surcharge d'informations en localisant les informations requises.

La figure 1.1 illustre un organigramme des procédures impliquées dans la classification des textes, sous l'angle de l'analyse traditionnelle et de l'analyse approfondie. Les données textuelles sont différentes des données numériques, des images ou des signaux. Elles doivent être traitées avec soin à l'aide de techniques **(NLP)**. La première étape importante consiste à prétraiter les données textuelles pour le modèle. Les modèles traditionnels doivent généralement obtenir de bonnes caractéristiques d'échantillon par des méthodes artificielles, puis les classer à l'aide d'algorithmes classiques d'apprentissage automatique.

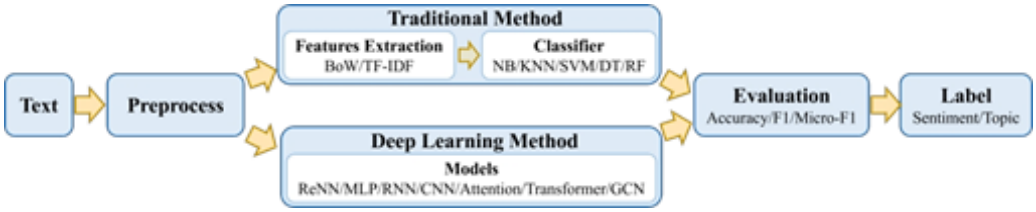


FIGURE 1.1 – Processus général de la classification de texte avec des méthodes classiques dans chaque module[5].

Par conséquent, l'efficacité de la méthode est largement limitée par l'extraction des caractéristiques. Toutefois, contrairement aux modèles traditionnels, l'apprentissage profond intègre l'ingénierie des caractéristiques dans le processus d'adaptation du modèle en apprenant un ensemble de transformations non linéaires qui servent à mettre en correspondance les caractéristiques directement avec les résultats.

Des années 1960 aux années 2010, les modèles traditionnels de classification des textes ont dominé. Les méthodes traditionnelles désignent les modèles basés sur les statistiques, tels que **Naïve Bayes(NB)**[6], **K-Nearest Neighbor (KNN)**[7] et **Support Vector Machine (SVM)** [8]. Par rapport aux méthodes antérieures basées sur des règles, cette méthode présente des avantages évidents en termes de précision et de stabilité. Cependant, ces approches nécessitent toujours une ingénierie des caractéristiques, ce qui prend du temps et est coûteux. En outre, elles ne tiennent généralement pas compte de la structure séquentielle naturelle ou des informations contextuelles des données textuelles, ce qui rend difficile l'apprentissage des informations sémantiques des mots. Depuis les années 2010, la classification des textes est passée progressivement des modèles traditionnels aux modèles d'apprentissage profond. Par rapport aux méthodes traditionnelles, les méthodes d'apprentissage profond évitent la conception de règles et de caractéristiques par les humains et fournissent automatiquement des représentations sémantiquement significatives pour l'exploration de texte. Par conséquent, la plupart des travaux de recherche sur la classification des textes sont basés sur les **réseaux neuronaux profonds (DNN)** [9], qui sont des approches basées sur les données et d'une grande complexité informatique. Peu de travaux se concentrent sur les modèles traditionnels pour résoudre les limitations de calcul et de données.

La classification des textes consiste à extraire des caractéristiques des données textuelles brutes et à prédire les catégories de données textuelles sur la base de ces caractéristiques. De nombreux modèles ont été proposés au cours des dernières décennies pour la classification des textes.

Plusieurs travaux ont récemment examiné la classification des textes et ses sous-problèmes. Deux d'entre eux portent sur la classification des textes. Kowsari et al ont étudié différentes méthodes d'extraction de caractéristiques textuelles et de réduction de la dimensionnalité, la structure du modèle de base pour la classification des textes et les méthodes d'évaluation. Minaee a passé en revue les méthodes récentes de classification de textes basées sur l'apprentissage profond, les ensembles de données de référence et les mesures d'évaluation.

Contrairement aux études de classification de textes existantes, nous concluons l'analyse des modèles traditionnels à l'apprentissage profond avec les travaux de ces dernières an-

nées. Les modèles traditionnels mettent l'accent sur l'extraction des caractéristiques et la conception des classificateurs. Une fois que le texte a des caractéristiques bien conçues, il peut être rapidement convergé en entraînant le classificateur. Les **(DNN)** [9] peuvent effectuer l'extraction des caractéristiques automatiquement et apprendre sans connaissance du domaine. Nous présentons ensuite les ensembles de données et les mesures d'évaluation pour les tâches à étiquette unique et à étiquettes multiples, et nous résumons les défis futurs de la recherche du point de vue des données, des modèles et des performances.
En ce qui concerne les modèles traditionnels, **(NB)** [6] est le premier modèle utilisé pour la tâche de classification des textes. Par la suite, des modèles de classification génériques ont été proposés, tels que **(KNN)**[7] et **(SVM)**, qui sont appelés classificateurs et sont largement utilisés pour la classification des textes. En ce qui concerne les modèles d'apprentissage profond, **TextCNN** a le plus grand nombre de références dans ces modèles, dans lesquels un modèle de réseau neuronal convolutif **(CNN)** a été introduit pour résoudre le problème de classification de texte pour la première fois. Bien qu'il n'ait pas été spécifiquement conçu pour gérer les tâches de classification de textes, le Bidirectional Encoder Representation from Transformers **(BERT)** a été largement utilisé lors de la conception de modèles de classification de textes, compte tenu de son efficacité sur de nombreux ensembles de données de classification de textes.

1.2 Modèles mathématique et deep learning

Les modèles traditionnels accélèrent la classification des textes avec une meilleure précision et élargissent le champ d'application des modèles traditionnels. La première chose à faire est de pré-traiter le texte brut d'entrée pour la formation des modèles traditionnels, ce qui consiste généralement en une segmentation des mots, un nettoyage des données et des statistiques. Ensuite, la représentation du texte vise à exprimer le texte prétraité sous une forme qui est beaucoup plus facile pour les ordinateurs et qui minimise la perte d'informations, comme le **Bag-Of-Words (BOW)** [10], **Term Frequency-Inverse Document Frequency (TF-IDF)** [11], **word2vec**. **(BOW)** signifie que tous les mots du corpus sont regroupés dans un tableau de correspondance.

Selon le tableau de correspondance, une phrase peut être représentée sous la forme d'un vecteur. Le i-ème élément du vecteur représente la fréquence du i-ème mot dans le tableau de correspondance de la phrase. Le vecteur est le **(BOW)** de la phrase. La représentation de chaque texte par un vecteur de la taille d'un dictionnaire est au cœur de la **(BOW)**. La valeur individuelle du vecteur indique la fréquence du mot correspondant à sa position inhérente dans le texte. Par rapport à la **BOW**, le N-gram prend en compte les informations des mots adjacents et construit un dictionnaire en tenant compte des mots adjacents.

Il est utilisé pour calculer le modèle de probabilité d'une phrase. La probabilité d'une phrase est exprimée comme la probabilité conjointe de chaque mot de la phrase. La probabilité d'une phrase peut être calculée en prédisant la probabilité du N -ème mot, étant donné la séquence des (N 1)-èmes mots. **(TF-IDF)**[11] utilise la fréquence des mots et inverse la fréquence des documents pour modéliser le texte. TF est la fréquence d'un mot dans un article spécifique, et IDF est la réciproque de la proportion des articles contenant ce mot par rapport au nombre total d'articles dans le corpus. Le **(TF-IDF)** [11] est la multiplication des deux. Le **(TF-IDF)** [11] évalue l'importance d'un mot pour un docu-

ment dans un ensemble de fichiers ou un corpus.

L'importance d'un mot augmente proportionnellement au nombre de fois qu'il apparaît dans un document. En revanche, elle diminue inversement avec sa fréquence dans l'ensemble du corpus. Le modèle word2vec utilise des informations contextuelles locales pour obtenir des vecteurs de mots, comme le montre la figure 2. Le vecteur de mot fait référence à un vecteur de valeurs réelles de longueur fixe spécifié comme vecteur de mot pour n'importe quel mot du corpus. Le modèle word2vec utilise deux modèles essentiels : **CBOW** et **Skip-gram**. Le premier modèle consiste à prédire le mot actuel en partant du principe que le contexte du mot actuel est connu. Le second permet de prédire le contexte lorsque le mot courant est connu.

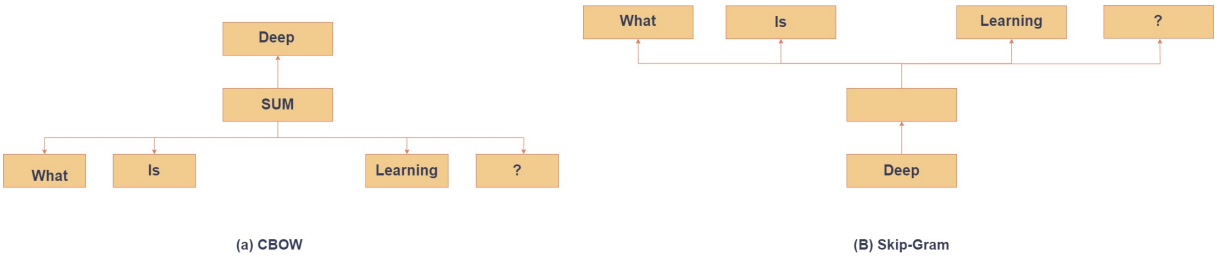


FIGURE 1.2 – La structure de word2vec, incluant CBOW et Skip-gram[5].

1.2.1 Naive Bayes (NB)

Les **modèles graphiques probabilistes (MGP)** expriment les dépendances conditionnelles entre les caractéristiques dans les graphes, tels que le **réseaux bayésien**[6]. Il s'agit d'une combinaison de la théorie des probabilités et de la théorie des graphes.
Naïve Bayes (NB) [6] est le modèle le plus simple et le plus largement utilisé, basé sur l'application du théorème de Bayes.

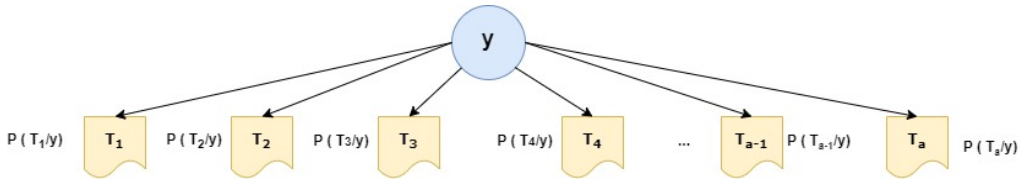


FIGURE 1.3 – La structure de Naïve Bayes[5].

L'article intitulé "Performance Analysis of ANN and Naive Bayes Classification Algorithm for Data Classification" par Mucahid Mustafa Saritas et Ali Yasar, explore l'utilisation de l'algorithme de classification Naive Bayes pour la classification des données[12].

Dans leurs travaux, ils ont utilisé des données anthropométriques et des paramètres d'analyse de sang pour estimer le potentiel d'avoir un cancer du sein[12].

L'étude a été réalisée en utilisant des données de patients qui ont été admis à la clinique avec le soupçon d'un cancer du sein. Les valeurs d'âge, d'IMC, de glucose, d'insuline ont été utilisées[12].

Dans leur étude, le diagnostic de la maladie a été fait en utilisant 9 entrées différentes (paramètres) et une sortie, Cet article met en lumière l'évaluation de la performance basée sur des exemples de classification corrects et incorrects de données en utilisant l'algorithme de Naïve Bayes[12].

1.2.2 K-Plus Proche Voisin (KNN)

L'algorithme des **K-voisins les plus proches (KNN)** [7] consiste essentiellement à classer un échantillon non étiqueté en trouvant la catégorie ayant le plus grand nombre d'échantillons sur les k échantillons les plus proches. Il s'agit d'un classificateur simple qui ne construit pas de modèle et qui peut réduire la complexité grâce au processus rapide d'obtention des k plus proches voisins. La figure 4 illustre la structure du **(KNN)** [7].

Nous pouvons trouver k textes d'entraînement se rapprochant d'un texte spécifique à classer en estimant l a distance intermédiaire. Ainsi, le texte peut être divisé dans les catégories les plus courantes trouvées dans les k textes de l'ensemble d'entraînement. L'amélioration de l'algorithme **(KNN)** [7] comprend principalement la similarité des caractéristiques , la valeur K et l'optimisation de l'index. l'algorithme **(KNN)** [7] prend un temps anormalement long sur les ensembles de données à grande échelle . Pour réduire le nombre de caractéristiques sélectionnées, Soucy et al proposent un algorithme KNN sans pondération des caractéristiques. Il parvient à trouver des caractéristiques pertinentes, en construisant les interdépendances entre les mots à l'aide d'une sélection de caractéristiques. Lorsque les données sont extrêmement inégalement réparties, le **(KNN)** a tendance à classer les échantillons contenant le plus de données. Le K-Nearest Neighbor pondéré **(NWKNN)** est proposé pour améliorer les performances de classification sur les corpus déséquilibrés. Il accorde un poids important aux voisins d'une petite catégorie et un poids faible aux voisins d'une classe plus large.

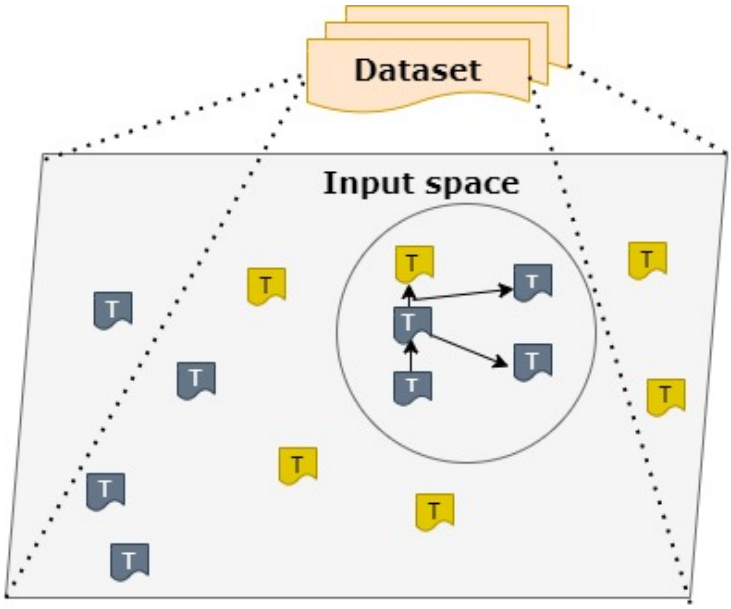


FIGURE 1.4 – La structure du KNN avec k = 4 (Chaque nœud représente un texte et les nœuds ayant des contours différents représentent des catégories différentes[5].)

L'article intitulé "KNN Model-Based Approach in Classification" par Gongde Guo, Hui Wang, David Bell, Yaxin Bi et Kieran Greer, traite de l'utilisation de l'algorithme des k plus proches voisins (KNN) pour la classification[13].

Les auteurs soulignent que le KNN est une méthode simple mais efficace pour la classification. Cependant, il a deux inconvénients majeurs : sa faible efficacité, car c'est une méthode d'apprentissage qui le rend inadapté à de nombreuses applications comme le web mining pour un grand dépôt, et sa dépendance à la sélection d'une "bonne valeur" pour k[13].

1.2.3 Support Vector Machine (SVM)

Vapnik propose une machine à vecteurs de support **(SVM)** [8] pour s'attaquer à la classification binaire de la reconnaissance des formes. Joachims , pour la première fois, utilise la méthode **(SVM)**[8] pour la classification des textes en représentant chaque texte comme un vecteur.

Comme l'illustre la figure 1.5 , les approches basées sur les **(SVM)**[8] transforment les tâches de classification de texte en plusieurs tâches de classification binaire. Dans ce contexte, les **(SVM)** construisent un hyperplan optimal dans l'espace d'entrée unidimensionnel ou l'espace des caractéristiques, en maximisant la distance entre l'hyperplan et les deux catégories d'ensembles d'apprentissage, ce qui permet d'obtenir la meilleure capacité de généralisation.

L'objectif est de faire en sorte que la distance de la limite de la catégorie le long de la direction perpendiculaire à l'hyperplan soit la plus grande. En d'autres termes, le taux d'erreur de classification sera le plus faible possible.

La construction d'un hyperplan optimal peut être transformée en un problème de programmation quadratique afin d'obtenir une solution globalement optimale.

Le choix de la fonction noyau appropriée et la sélection des caractéristiques sont de la plus haute importance pour garantir que le **(SVM)**[8] puisse traiter les problèmes non linéaires et devenir un classificateur non linéaire robuste. En outre, les méthodes d'apprentissage actif et d'apprentissage adaptatif sont utilisées pour la classification des textes afin de réduire l'effort d'étiquetage sur la base de l'algorithme d'apprentissage supervisé **(SVM)**.

Pour analyser ce que les algorithmes **(SVM)**[8] apprennent et quelles tâches sont appropriées, Joachims propose un modèle d'apprentissage théorique combinant les caractéristiques statistiques et les performances de généralisation d'un **(SVM)**[8], en analysant les caractéristiques et les avantages à l'aide d'une approche quantitative.

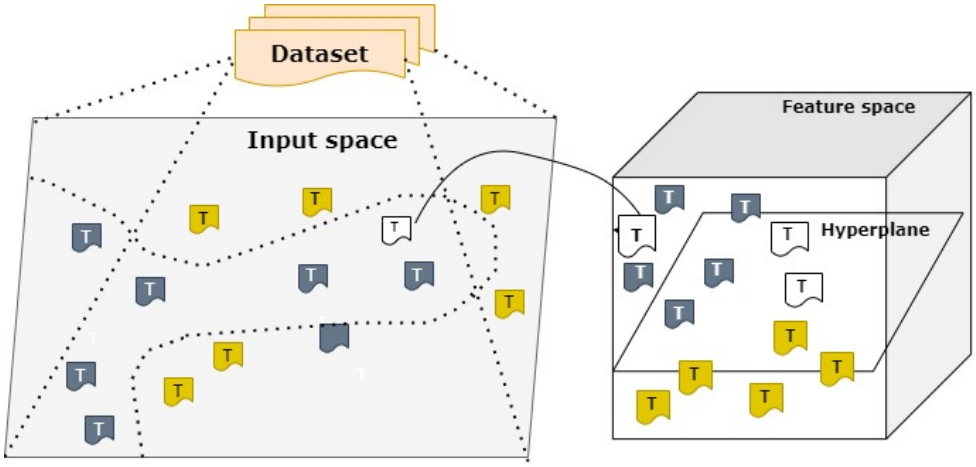


FIGURE 1.5 – Structure du SVM (Chaque nœud représente un texte et les nœuds ayant des contours différents représentent des catégories différentes.)[5]

Parmis les travaux similaires, on peut citer l'article intitulé : "Machine Learning Applications based on SVM Classification : A Review" par Dakhaz Mustafa Abdullah et Adnan Mohsin Abdulazeez qui traite de l'application des techniques de Machine Learning, en particulier la classification SVM dans divers domaines[14].

L'article aborde le traitement des données en utilisant un algorithme de machine à vecteurs de support (SVM) dans différents domaines comme la reconnaissance faciale, le diagnostic des maladies, la reconnaissance de texte et l'analyse des sentiments[14].

1.3 Modèles profonds

Les **(DNN)** sont des réseaux neuronaux artificiels qui simulent le cerveau humain pour apprendre automatiquement des caractéristiques de haut niveau à partir de données, obtenant ainsi de meilleurs résultats que les modèles traditionnels dans la reconnaissance vocale, le traitement d'images et la compréhension de textes. Les ensembles de données d'entrée doivent être analysés pour classer les données, par exemple un ensemble de données à étiquette unique, à étiquettes multiples, non supervisé ou déséquilibré. En fonction des caractéristiques de l'ensemble de données, les vecteurs de mots d'entrée sont envoyés dans le réseau **(DNN)** pour être entraînés jusqu'à ce que la condition d'arrêt soit atteinte. Les performances du modèle de formation sont vérifiées par les tâches en aval, telles que la classification des sentiments, la réponse aux questions et la prédiction des événements. Le tableau 1 présente quelques **(DNN)** au fil des ans, y compris des conceptions différentes des modèles de base correspondants, des mesures d'évaluation et des ensembles de données expérimentales.

De nombreux modèles d'apprentissage profond ont été proposés au cours des dernières décennies pour la classification des textes, comme le montre le tableau 1. Nous présentons sous forme de tableau les informations principales y compris les années de publication, les lieux, les applications, les liens de code, les mesures d'évaluation et les ensembles de données expérimentales - des principaux modèles d'apprentissage profond pour la classification des textes. Les applications de ce tableau comprennent **l'analyse des sentiments**

(SA), l'étiquetage des sujets (TL), la classification des nouvelles (NC), la réponse aux questions (QA). Le perceptron multicouche et le réseau neuronal récursif sont les deux premières approches d'apprentissage profond utilisées pour la tâche de classification de texte, qui améliorent les performances par rapport aux modèles traditionnels. Ensuite, les **(CNN)** , les **réseaux neuronaux récurrents (RNN)** et les mécanismes d'attention sont utilisés pour la classification des textes. De nombreux chercheurs améliorent les performances de la classification des textes pour différentes tâches en améliorant les **(CNN)**, les **(RNN)** et l'attention, ou la fusion de modèles et les méthodes multitâches. L'apparition de **(BERT)** , qui peut générer des vecteurs de mots contextualisés, marque un tournant important dans le développement de la classification des textes et d'autres technologies **(NLP)**. De nombreux chercheurs ont étudié des modèles de classification de texte basés sur **(BERT)**, qui atteignent de meilleures performances que les modèles ci-dessus dans de multiples tâches de **(TAL)**, y compris la classification de texte.

1.3.1 Réseau Neuronal Convolutif

Un réseau neuronal convolutif CNN est une architecture réseau pour le Deep Learning qui apprend directement à partir des données **"1"**.

Ce type de réseau est particulièrement efficace lorsqu'il s'agit de trouver des patterns dans des images afin de reconnaître des objets, des classes et des catégories. Les **(CNN)** peuvent aussi être efficaces pour la classification de données audio et de séries de textes.

Un réseau neuronal convolutif peut disposer de dizaines, voire de centaines de couches qui apprennent chacune à détecter différentes caractéristiques. Des filtres sont appliqués à chaque élément du jeu d'apprentissage, puis la sortie convoluée est utilisée comme entrée de la couche suivante. Au début, ces filtres peuvent concerner des caractéristiques très simples, puis se complexifier jusqu'à représenter des caractéristiques uniques propres à chaques élements.

Un **(CNN)** comporte une couche d'entrée, une couche de sortie et de nombreuses couches intermédiaires cachées. Comme le montre la figure suivante :

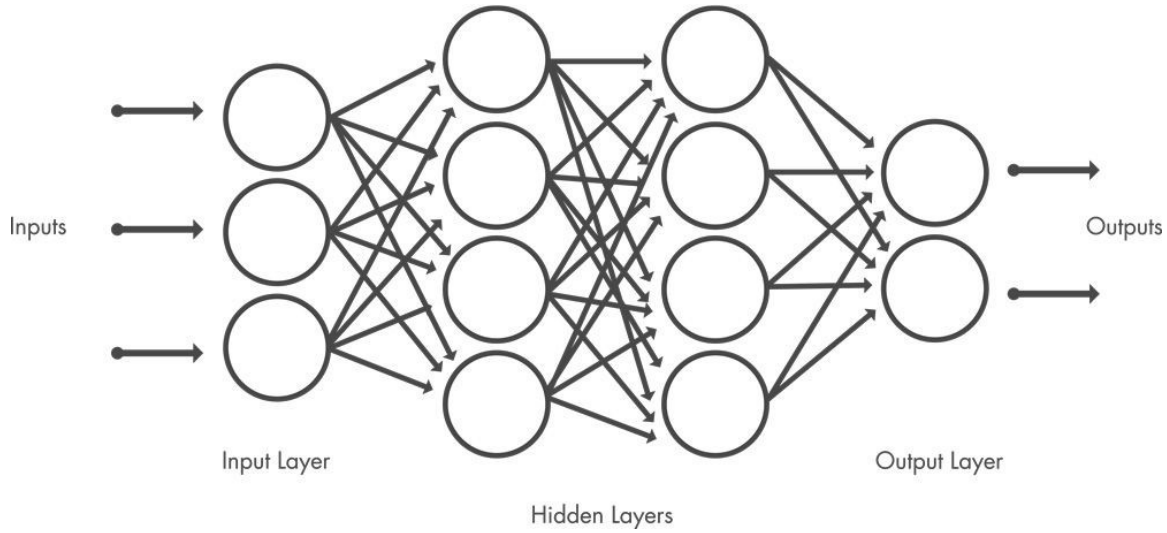


FIGURE 1.6 – Représentation des Layers dans les Reseaux de neuronnes convolutifs.[15]

Les différentes couches d'un réseau neuronal convolutif **(CNN)** réalisent des opérations pour modifier les données et en apprendre les caractéristiques spécifiques. Les trois couches les plus courantes sont la couche de convolution, la couche d'activation (ou ReLU) et la couche de pooling **"1"**.

- La couche de convolution applique un ensemble de filtres convolutifs aux entrées, chaque filtre activant certaines caractéristiques .

- La couche ReLU (Rectified Linear Unit) accélère et améliore l'apprentissage en transformant les valeurs négatives en zéros tout en conservant les valeurs positives. Ce processus est souvent appelé activation car seules les caractéristiques activées sont transmises à la couche suivante.

- La couche de pooling simplifie la sortie en effectuant un sous-échantillonnage non linéaire, réduisant ainsi le nombre de paramètres que le réseau doit apprendre.

Ces opérations sont répétées sur plusieurs couches, souvent des dizaines ou des centaines, chaque couche apprenant à reconnaître différentes caractéristiques.

Dans les travaux similaires nous avons L'article "Deep CNN for Brain Tumor Classification", publié par Neural Processing Letters, est écrit par des chercheurs qui se concentrent sur l'utilisation des réseaux de neurones convolutifs profonds (CNN) pour la classification des tumeurs cérébrales[16].

L'examen des d'imagerie par résonance magnétique (IRM) du cerveau du patient représente une technique efficace pour distinguer les tumeurs cérébrales. Cependant, en raison de la grande quantité de données et des différents types de tumeurs cérébrales, la technique manuelle peut conduire à des erreurs humaines[16].

Par conséquent L'évolution récente des techniques de classification d'images a montré de grands progrès, en particulier les réseaux de neurones convolutifs profonds (CNN) qui ont réussi dans ce domaine[16].

Dans ce contexte, les auteurs ont développé un modèle de classification de tumeurs cérébrales qui a montré une performance significative dans la détection précise des tumeurs cérébrales[16].

1.3.2 Réseau Neuronal Récurrents

Les réseaux neuronaux récurrents **(RNN)** sont principalement utilisés dans la traitement du langage naturel **(NLP)**. L'apprentissage en profondeur et la construction de modèles qui imitent l'activité des neurones dans le cerveau humain utilisent les **RNN**.

Le texte et l'écriture manuscrite sont des exemples de données dans lesquelles les réseaux récurrents sont censés identifier des modèles. Un réseau neuronal récurrent ressemble à un réseau neuronal ordinaire, avec l'ajout d'un état de mémoire aux neurones. Une mémoire simple sera incluse dans le calcul [1]
Les réseaux neuronaux récurrents sont une forme de méthode d'apprentissage profond qui utilise une approche séquentielle. Nous partons toujours du principe que chaque entrée et sortie d'un réseau neuronal dépend de tous les autres niveaux. Les réseaux neuronaux récurrents sont ainsi nommés parce qu'ils effectuent des calculs mathématiques dans un ordre consécutif.

On détecte aussi plusieurs genres de **RNN** comme le montre cette figure :

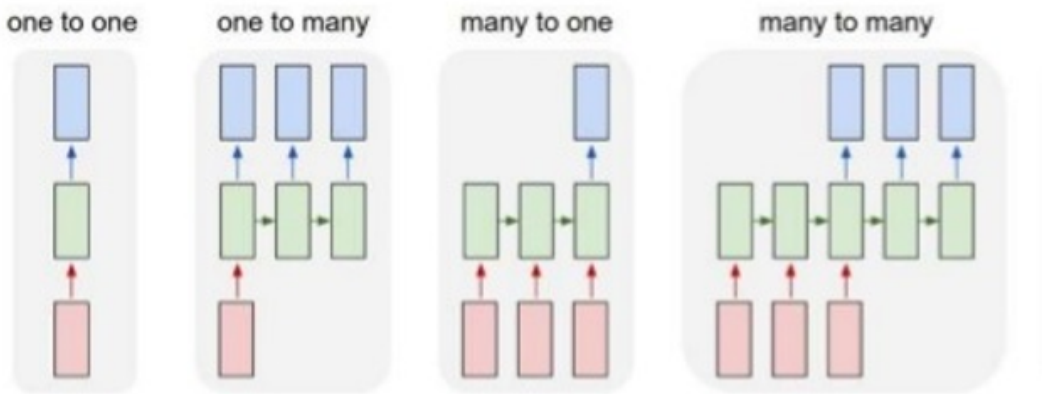


FIGURE 1.7 – Type de Reseaux de neuronnes récurrents.

- **RNN (One to One)** : Il est utilisé pour résoudre les problèmes généraux d'apprentissage automatique qui n'ont qu'une entrée et une sortie.
  Exemple : classification d'images.

- **RNN (One to Many)** : Une seule entrée et plusieurs sorties décrivent un réseau neuronal récurrent un-à-plusieurs.
  Exemple : Description d'image

1. Stanford University Cheatsheet - Recurrent Neural Networks. bit.ly/4aIYklS

- **RNN(Many to One)** :Ce **(RNN)** crée une seule sortie à partir d'une série donnée d'entrées.
  Exemple : L'analyse des sentiments dans lequel un texte est identifié comme exprimant des sentiments positifs ou négatifs.

- **RNN Many-to-Many** : Ce **(RNN)** reçoit un ensemble d'entrées et produit un ensemble de sorties.
  Exemple : Traduction automatique

Ce qu'il faut retenir c'est que les **RNN** sont des modèles séquentiels comme l'indique la figure ci-dessous utilisée pour analyser des données séquentielles telles que du texte, de l'audio ou des séquences temporelles. Ils se distinguent des réseaux de neurones classiques par leur capacité à mémoriser et à prendre en compte les informations précédentes lors du traitement des données actuelles dans la séquence https ://stanford.edu/ shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

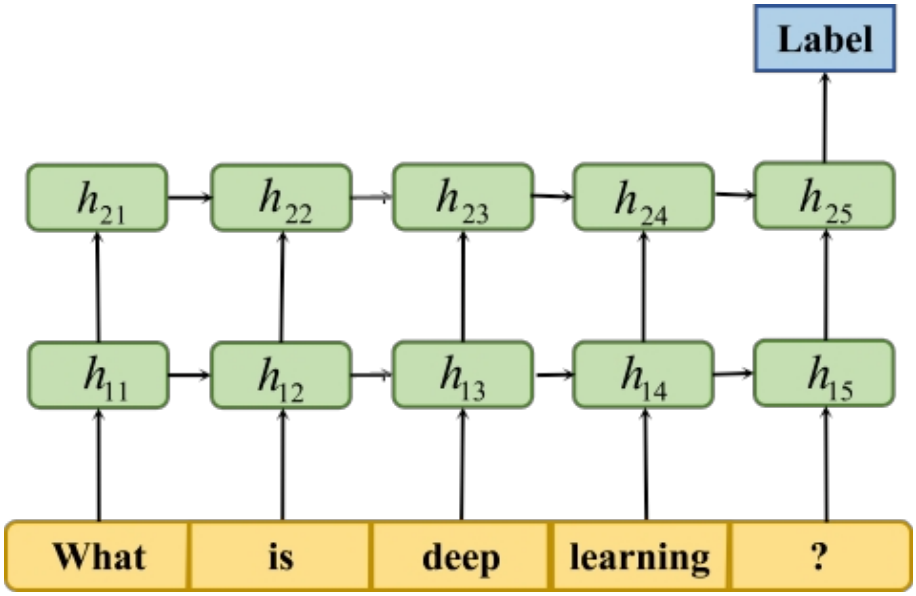


FIGURE 1.8 – Architecture séquenciel des Reseaux de neuronnes[5].

Un problème rencontré avec les réseaux de neurones simples était leur difficulté à conserver les détails importants sur de longues séquences. Par exemple, lors de l'analyse d'un texte long, un réseau de neurones simple pouvait oublier des informations clés au fur et à mesure de la lecture, ce qui limitait sa capacité à comprendre le contexte global.
C'est là qu'interviennent les **LSTM** (Long Short-Term Memory) qui sont une variante des **RNN**.

Les **LSTM** ont été conçus pour résoudre ce problème de mémoire à court terme en permettant au réseau de mémoriser des informations importantes sur de longues périodes. Grâce à leurs mécanismes de porte (portes d'oubli, portes d'entrée et portes de sortie), les **LSTM** peuvent choisir quelles informations garder en mémoire et quelles informations oublier, améliorant ainsi la capacité du réseau à comprendre et à traiter efficacement les séquences complexes.

- **Forget Gate** : indique quel type d'informations d'états peut être oublié.
- **Input Gate** : indique quelles nouvelles informations doivent être gardées ou lesquelles sont à mettre à jour.
- **Output Gate :** indique quels éléments seront utilisés pour calculer la sortie .

parmis les Travaux similaire dans le domaine on peut notemment citer L'article intitulé "Sentiment Analysis by using Recurrent Neural Network" par Alpna Patel et Arvind Kumar Tiwari qui traite de l'analyse des sentiments en utilisant les réseaux de neurones récurrents[17].

Cet article de recherche donne un aperçu détaillé des différentes méthodes de sélection de caractéristiques, des techniques de classification des sentiments et des approches d'apprentissage profond pour l'analyse des sentiments[17].

En résumé, cet article offre une vue d'ensemble complète de l'analyse des sentiments en utilisant les réseaux de neurones récurrents[17].

1.3.3 Transformers

Les Transformers sont des architectures de réseaux neuronaux basées sur des mécanismes d'attention. Ils sont principalement utilisés dans le domaine du traitement automatique du langage naturel **(NLP)** pour des tâches telles que la classification de texte, la génération de texte et la compréhension de texte.
BERT, pour Bidirectional Encoder Representations from Transformers, est un **LLM** (Large Language Model), entraîné sur un grand corpus de données textuelles et conçu pour générer du texte en langage naturel. Google AI a développé ce modèle en 2018, en utilisant des réseaux neuronaux. Cette technique de Machine Learning permet à l'algorithme de mieux comprendre le contexte d'une requête.

Contrairement aux modèles antérieurs qui utilisaient une approche unidirectionnelle pour traiter le texte, **BERT** est capable de comprendre le contexte bidirectionnel du langage. Il prend donc en compte à la fois les mots précédents et les mots suivants dans une phrase. Cela lui permet de mieux comprendre le sens des mots et des phrases.

Diffusé en open source à la communauté scientifique en 2018, Google a annoncé en 2019 avoir appliqué la mise à jour sur son propre moteur de recherche et ses algorithmes.

L'une de ses caractéristiques fondamentales est le mécanisme Self-Attention (auto attention). À chaque étape, chacun des mots de la séquence peut regarder les autres mots de manière à évaluer leur importance dans le contexte. Cela permet de comprendre les relations entre les mots à différentes distances dans une séquence.
[2] C'est le mécanisme clé d'une architecture Transformer. Le modèle va pouvoir tenir compte des relations entre les mots, tout en calculant des pondérations d'attention pour

2. Intelligence Artificielle School (BERT) : t.ly/D64Wj

chaque paire de mots dans une séquence. Cela permet au modèle de déterminer l'importance des mots les uns par rapport aux autres.
Transformer utilise plusieurs couches d'encodeurs. Chacun se compose de deux sous-couches : une sous-couche d'auto attention multitêtes et une sous-couche de réseaux de neurones entièrement connectés (feedforward). Cela permet un apprentissage de représentations de plus en plus abstraites et contextuelles de la séquence.

Transformer n'a pas de notion intrinsèque d'ordre séquentiel. Les positional encoding s'ajoutent aux embeddings de mots pour donner à **BERT** une information sur la position des mots dans la séquence.

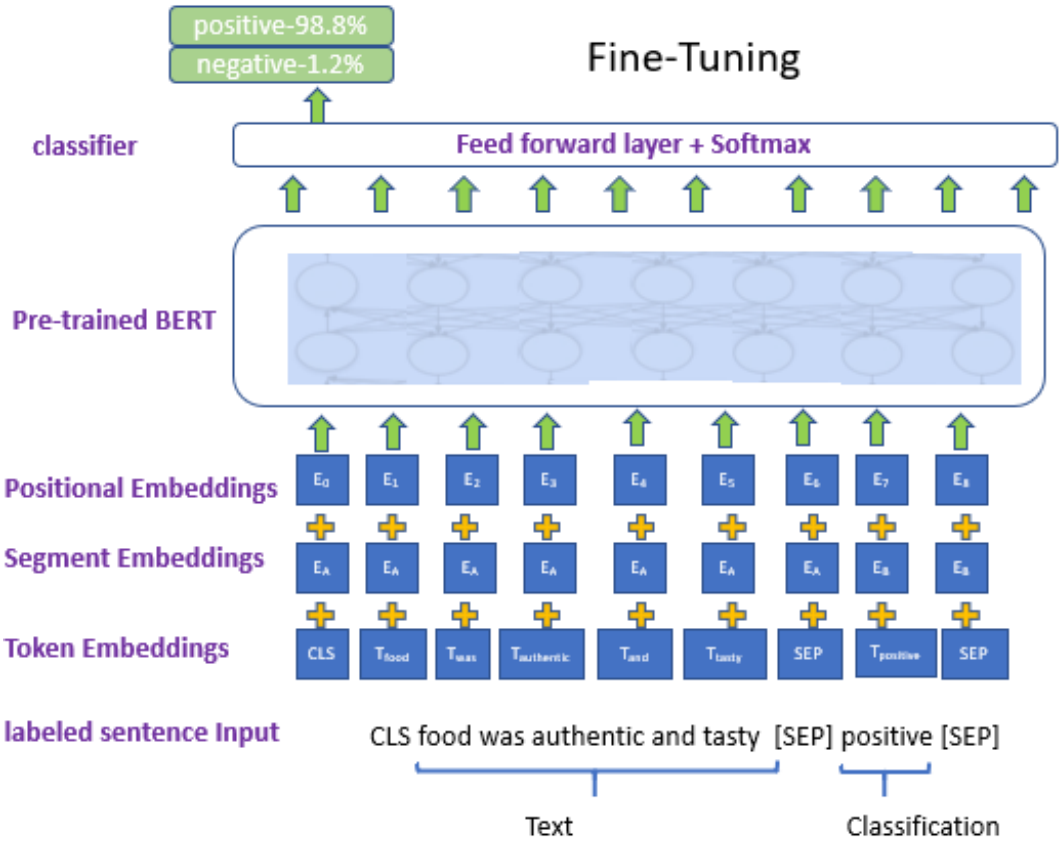


FIGURE 1.9 – architecture Transformers.

On peut citer notemment L'article intitulé : "Fake news detection based on news content and social contexts : a transformer-based approach" par Shaina Raza et Chen Ding qui traite de la détection des fausses nouvelles en utilisant les Transformer[18].

Les résultats expérimentaux sur des données réelles montrent que leur modèle peut détecter les fausses nouvelles avec une précision plus élevée quelques minutes seulement après leur propagation[18].

1.4 Conclusion

En conclusion, la classification des textes est essentielle pour le traitement du langage naturel dans divers domaines. Dans un monde où l'information est en constante expansion, le recours à des méthodes d'apprentissage automatique pour automatiser cette tâche offre des avantages considérables en termes de fiabilité, d'objectivité et d'efficacité.

Alors que les méthodes traditionnelles de classification ont dominé jusqu'aux années 2010, l'avènement des modèles d'apprentissage profond a marqué un tournant significatif. Les approches basées sur les réseaux neuronaux profonds offrent une capacité automatique à extraire des caractéristiques pertinentes des données textuelles, réduisant ainsi la dépendance à l'égard de la conception manuelle de caractéristiques.

Des travaux récents ont mis en lumière les avantages et les inconvénients de chaque approche. Alors que les modèles traditionnels se concentrent sur l'extraction de caractéristiques et la conception de classificateurs, les modèles d'apprentissage profond, se démarquent par leur capacité à automatiser cette extraction de manière efficace.

CHAPITRE

2

MISE EN PRATIQUE ET DÉMARCHE

2.1 Introduction

Dans le domaine en constante évolution de la classification de textes, il existe une séquence d'étapes incontournables qui constituent le socle de tout projet. Cela veut dire que quelque soit le modèle ou la méthode choisie pour effectuer votre entraînement, le processus demeure fondamentalement le même. Comme le montre les figure ci-dessous :

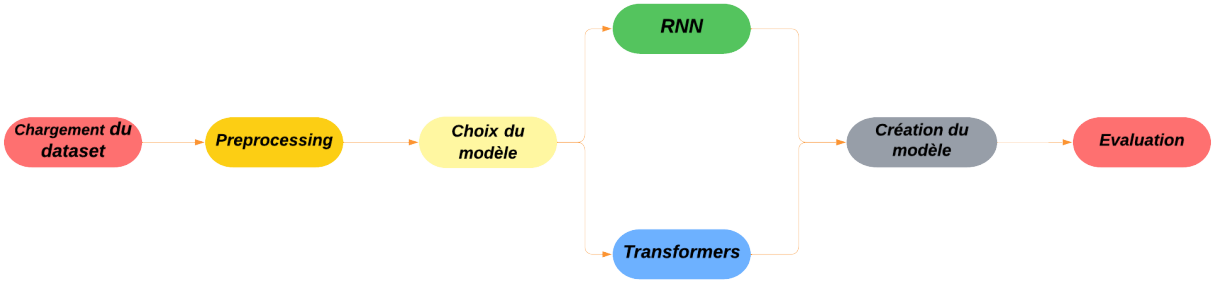


FIGURE 2.1 – Démarche de la classification de textes.

- **chargement du dataset** : c'est l'étape initiale cruciale où l'on rassemble les données nécessaires, souvent hétérogènes et volumineuses, qui serviront de fondement à l'ensemble du processus d'apprentissage (voir 1).

Algorithm 1 Chargement du Dataset

Input : Reviews
Output : newReviews
for each review ∈ Reviews **do**
    review.score ← review.score as float ▷ cast the score data type to float
    if review.score ∉ N **then**
        review.score ← NaN ▷ replace non-numeric values with NaN
    end if
    if review.score ∉ {1, 2, ..., 5} **then**
        review.score ← NaN ▷ replace scores outside 1-5 with NaN
    end if
end for

- **Preprocessing :** c'est une phase de nettoyage et de préparation des données qui implique des techniques de normalisation, de tokenisation, et de suppression des éléments superflus, afin de transformer les données brutes en un format exploitable pour l'entraînement des modèles(voir 2).

Algorithm 2 Preprocessing

Input : newReviews
Stopwords ← load(nltk.corpus) ▷ Import Stopwords from nltk corpus
superfluousWords ← {@, #, URLs, . . .} ▷ User mentions, hashtags, URLs
for each review ∈ newReviews **do**
    cleanedReview ← clean(review,superfluousWords)
end for
CASE 1 :
function LEMMATIZE_TEXT_SPACY(text)
    if text IS null **then**
        RETURN repr(text) ▷ Return the representation of text as a string
    end if
    text ← PROCESS(text) WITH spaCy
    lemmatized_Tokens ← lemmatize(text)
    Output : lemmatized_Tokens
end function
CASE 2 :
function STEM_TEXT_NLTK(text)
    if text IS null **then**
        RETURN repr(text) ▷ Return the representation of text as a string
    end if
    tokens ← SPLIT(text)
    stemmed_tokens ← APPLY(stemming, tokens)
    Output : JOIN(stemmed_tokens, " ") ▷ Return stemmed tokens joined by spaces
end function

- **Choix et Création du Modèle :** Cette étape est cruciale et peut-être l'une des plus stratégiques, car elle définit l'approche algorithmique à utiliser pour analyser

les données. Le choix du modèle est influencé par la nature du problème, la taille et la qualité de l'ensemble de données, ainsi que par les performances souhaitées. Une fois le modèle sélectionné, s'ensuit une phase technique où l'on configure les hyper-paramètres, structure les couches (dans le cas des modèles profonds), et prépare le modèle à apprendre de manière optimale à partir des données prétraitées(voir 3).

Algorithm 3 Choix et création du modèle

Input : max_length ← 160, max_words ← 295053
Output : model
CASE 1 :

BEGIN
model ← CREATE_SEQUENTIAL_MODEL()
EMBEDDING(model, input_dim ← max_words, output_dim ← 128)
LSTM(model, units ← 128, dropout ← 0.2, recurrent_dropout ← 0.2)
DENSE(model, units ← 5, activation ← 'softmax')
END

CASE 2 :

tokenizer ← BertTokenizer.from_pretrained('bert-large-uncased')
model ← BertForSequenceClassification.from_pretrained('bert-large-uncased', num_labels ← 5)

- **Evaluation du modèle :** est l'étape conclusive où l'on mesure l'efficacité du modèle à travers diverses métriques telles que la précision, le rappel, le score F1, ou encore la matrice de confusion. Cette évaluation est essentielle pour juger de la capacité du modèle à généraliser à de nouvelles données et pour identifier les éventuels ajustements nécessaires pour améliorer les performances. Ainsi, peu importe le chemin emprunté dans le vaste univers de l'apprentissage automatique, ces étapes fondamentales demeurent le fil d'Ariane qui guide les scientifiques des données à travers le labyrinthe complexe de la prédiction et de la classification de texte(voir 4).

Algorithm 4 Évaluation

Input : model predictions, data, number of classes (C)
Output : accuracy, average_loss
BEGIN
correct_predictions ← 0
total_predictions ← 0
for each prediction ∈ model predictions **do**
    total_predictions ← total_predictions + 1
    if the prediction is correct **then**
        correct_predictions ← correct_predictions + 1
    end if
end for
accuracy ←

$$\frac{\text{correct\_predictions}}{\text{total\_predictions}}$$

END
BEGIN
total_loss ← 0
for each instance ∈ data **do**
    instance_loss ← 0
    for each class i ∈ {1, ..., C} **do**
        yi ← 1 if class i is the true class for the instance, otherwise 0
        pi ← predicted probability for class i
        instance_loss ← instance_loss -

$$(yi \times \log(pi))$$

    end for
    total_loss ← total_loss + instance_loss
end for
average_loss ←

$$\frac{\text{total\_loss}}{\text{total number of instances}}$$

END

2.2 Chargement du Dataset

- **(CSV)** [1] **(Comma-Separated Values)** : ou en français (Valeurs séparées par une virgule) est un type de fichier texte utilisé pour stocker des données tabulaires, telles que des tableaux ou des bases de données simples. Dans un fichier CSV, chaque ligne correspond à une entrée de données, et les valeurs sont séparées par des virgules (ou parfois d'autres délimiteurs, comme des points-virgules). Les fichiers CSV sont largement utilisés car ils sont faciles à créer, à lire et à manipuler avec des logiciels de traitement de données, comme des tableurs ou des langages de programmation.

1. google-ads-csv :t.ly/KGFGh

Ce qu'il faut savoir c'est que lors de la lecture ou du chargement du dataset, il est judicieux d'intégrer des fonctions permettant de mesurer et de comparer les données avant et après le processus de prétraitement. Par exemple, une fonction pour calculer le nombre de mots dans chaque avis du dataset peut être ajoutée.

Cette approche permettra d'évaluer l'impact du prétraitement sur la longueur des avis, la complexité du langage utilisé, et la qualité globale des données. En mesurant ces métriques avant et après le prétraitement, il devient possible de quantifier les améliorations apportées aux données et de garantir leur pertinence et leur fiabilité pour les analyses ultérieures.

2.3 Preprocessing

Le prétraitement des données, également appelé preprocessing, est une étape essentielle dans le processus d'analyse de données, notamment dans le domaine de l'apprentissage automatique et du traitement du langage naturel.

Cette étape consiste à nettoyer, transformer et normaliser les données brutes pour les rendre plus adaptées à l'analyse et à l'apprentissage des modèles. Le preprocessing peut inclure plusieurs opérations telles que la suppression des doublons, la correction des fautes d'orthographe, la lemmatisation ou la racinisation des mots, la suppression des stopwords (mots très fréquents mais peu informatifs), la normalisation des données numériques, et la conversion des textes en représentations numériques utilisables par les algorithmes d'apprentissage automatique. En effectuant un preprocessing efficace, on améliore la qualité des données, on réduit le bruit, et on facilite la création de modèles plus précis et robustes.

2.3.1 Stopwords

Les experts en sémantique parlent de stopwords pour les mots qui, en théorie, n'ont aucune influence [19].

Selon Wikipédia, un mot vide est un mot non significatif figurant dans un texte. On l'oppose à "mot plein" [19].

La raison en est qu'ils sont trop communs et que la plupart des internautes les omettent lorsqu'ils tapent une requête sur les moteurs de recherche [19].

C'est d'ailleurs pour cela que les algorithmes de Google les ont tout simplement ignorés.Le terme stopword ou mot vide en français désigne tous les mots n'ayant pas de réelle signification. On dit aussi qu'ils ne sont pas porteurs de sens. En effet, ils sont si courants et reviennent de façon tellement régulière qu'ils ne permettent pas de caractériser, au sens lexical, un texte par rapport à un autre texte [19].

Les stopwords sont généralement des mots dits grammaticaux tels que des adverbes, des pronoms ou encore des mots de liaison [19].

Par opposition, les mots pleins sont souvent des mots lexicaux comme les noms, les verbes voire même les adjectifs. Néanmoins, il se peut que certains mots grammaticaux de-

viennent des mots pleins et que des mots lexicaux ne soient plus utilisés dans la différentiation des textes. Il n'existe donc pas de liste de ce type de mots bien définie [19].

En effet, en fonction du groupe de texte ou du sujet que l'on va choisir, les stop words ne vont pas toujours être les mêmes. Par exemple, dans un dataset portant sur des produits médicaux, des termes comme "médicament", "traitement", ou "effets secondaires" peuvent être considérés comme des mots-clés importants, tandis que dans un dataset sur la littérature, des termes tels que "personnages", "intrigue", ou "thèmes" auront une importance primordiale. L'objet du contenu joue donc un rôle important dans la définition de ces stop lists. Évidemment, elles diffèrent également d'une langue à une autre [19].

Dans notre travail, nous utilisons les stopwords du dictionnaire anglais pour le prétraitement des données comme le montre la figure ci-dessous. Étant donné que notre dataset se compose de reviews en anglais, l'utilisation des stopwords anglais est pertinente pour éliminer les mots fréquents mais peu informatifs tels que "the", "is", "and", etc. Ces mots là n'apportent généralement pas de valeur significative à l'analyse et peuvent être éliminés pour améliorer la qualité des données lors de l'entraînement des modèles d'apprentissage automatique [19].

2.3.2 Suppression de mots et nettoyage de données

La suppression d'expressions et de caractères constitue une étape cruciale du prétraitement des données textuelles, même après avoir éliminé les stopwords. En effet, il reste souvent des éléments à nettoyer pour garantir des données de haute qualité et éviter toute perturbation lors de l'analyse.

Parmi ces éléments à nettoyer, on retrouve la suppression des mentions (comme @username), des caractères spéciaux et des symboles de ponctuation, des liens URL, des emojis, des numéros de téléphone, et des adresses e-mail. Par exemple, dans un dataset de reviews en ligne, il peut être nécessaire de supprimer les mentions de marque ou les liens vers des sites externes qui n'apportent pas d'informations pertinentes à l'analyse. Cette étape de nettoyage contribue à améliorer la qualité des données et à réduire le bruit, ce qui permet d'obtenir des résultats plus précis lors de l'entraînement des modèles d'apprentissage automatique.

En nettoyant ces éléments des données textuelles, on obtient des données plus propres et plus adaptées à l'analyse, ce qui améliore la qualité des résultats obtenus à partir de l'apprentissage automatique et du traitement du langage naturel. Ceci dit rien ne vous empêche de faire d'avantage le tri dans vos donnée :

- **Suppression des mentions**
  Les mentions telles que @username peuvent être présentes dans les données textuelles, mais elles ne sont souvent pas pertinentes pour l'analyse. Elles peuvent être supprimées pour réduire le bruit et se concentrer sur le contenu principal.

- **Suppression des caractères spéciaux et de la ponctuation**
  Les caractères spéciaux tels que #, %, etc., ainsi que la ponctuation comme les

points, les virgules et les points d'interrogation, peuvent être éliminés pour simplifier le texte et éviter toute confusion lors de l'analyse.

- **Suppression des liens URL**
  Les liens URL vers des sites web externes peuvent être retirés des données textuelles, car ils ne contribuent généralement pas au contenu analysé et peuvent même perturber les algorithmes de traitement du langage naturel.

- **Suppression des emojis**
  Les emojis, bien que populaires dans les communications en ligne, peuvent être retirés des données textuelles pour se concentrer sur le texte brut et éviter toute interférence avec les modèles d'analyse de texte.

- **Suppression des numéros de téléphone**
  Les numéros de téléphone, s'ils sont présents dans les données textuelles, peuvent être supprimés pour des raisons de confidentialité et pour se concentrer uniquement sur le contenu textuel pertinent.

- **Suppression des adresses e-mail**
  De même, les adresses e-mail peuvent être retirées des données textuelles pour des raisons similaires de confidentialité et de focalisation sur le texte brut.

- **Suppression des mots rares ou peu fréquents**
  Certains mots très peu fréquents dans le corpus peuvent être éliminés pour réduire la dimensionnalité des données et améliorer les performances des modèles.

- **Correction des fautes d'orthographe**
  Les fautes d'orthographe courantes peuvent être corrigées à l'aide de techniques de correction automatique pour garantir la cohérence et la précision des données.

- **Suppression des mots courts**
  Dans certains cas, les mots très courts (par exemple, moins de 3 caractères) peuvent être supprimés car ils ont souvent peu de signification en isolation.

2.3.3 Stemming et Lemmatization

Stemming :

Le stemming [2] est une technique de normalisation de texte utilisée dans le traitement du langage naturel qui transforme les mots en leur radical. Donc Le stemming consiste à réduire un mot dans sa forme "racine". Le but du stemming est de regrouper de nombreuses variantes d'un mot comme un seul et même mot. Par exemple, une fois que l'on applique un stemming sur "Chiens" ou "Chien", le mot résultant est le même traitement de données, comme des tableurs ou des langages de programmation.

Il existe plusieurs algorithmes de Stemming en **NLP** , Parmis ceux-là :

2. mathworks_Stemming :t.ly/qHF9J

- **Porter Stemmer Algorithm [20].**
- **Lancaster Stemming Algorithm [20].**
- **RegexpStemmer [20].**

Dans notre cas nous utilisons l'algorithme de Stemming de Porter dans NLTK , aussi connu sous le nom de Porter Stemmer. Il s'agit d'un algorithme de Stemming couramment utilisé et développé par Martin Porter en 1980.

Le Porter Stemmer est conçu pour réduire les mots à leur racine ou à leur forme de base en supprimant les suffixes communs en anglais, ce qui permet de regrouper les mots ayant une signification similaire [21].

Lemmatization :

La lemmatisation [3] est une technique de normalisation de texte dans le traitement du langage naturel , son but est de réduire le mot à sa forme racine , aussi appelé « Lemma »[22].

La lemmatization utilise le vocabulaire et l'analyse morphologique pour supprimer les affixes des mots.

Pour identifier correctement un lemme, les outils analysent le contexte, le sens et la partie du discours prévue dans une phrase, ainsi que le mot dans le contexte plus large de la phrase [22].

Par exemple, le verbe "walk" peut apparaître comme "walking," "walks" or "walked." Les terminaisons flexionnelles telles que "s", "ed" et "ing" sont supprimées. La lemmatisation regroupe ces mots sous la forme d'un lemme, "marcher".

2.3.4 Tokenization

Les données textuelles, au stade initial, ne sont qu'un ensemble de caractères. Toutes les étapes de l'analyse de texte nécessitent l'identification des mots présents dans le jeu de données. C'est pourquoi un analyseur syntaxique est nécessaire pour effectuer la tokenisation des documents. Bien que ce processus puisse sembler trivial puisque le texte est déjà stocké dans des formats lisibles par machine, certains problèmes subsistent. Par exemple, la suppression des marques de ponctuation et des traits d'union en fin de ligne . Cependant, des caractères tels que les parenthèses, les traits d'union, etc., sont bien pris en charge. Les tokenizers assurent également la fiabilité des documents[23].

La tokenisation, bien qu'apparemment simple, implique des défis comme le traitement des différents types de ponctuation et des symboles spéciaux qui peuvent perturber l'analyse. Les tokenizers modernes sont conçus pour gérer ces problèmes de manière efficace, garantissant que chaque mot est correctement identifié et prêt pour les étapes suivantes de

3. mathworks_Lemmatization :t.ly/2U0i9

l'analyse de texte. De plus, cette étape est cruciale pour assurer que les données textuelles soient nettoyées et formatées de manière cohérente, ce qui est essentiel pour des analyses précises et fiables[23].

De nombreux outils sont disponibles pour la tokenisation d'un document. Ces outils sont énumérés ci-dessous [23] :

- **Mila Tokenizer.**
- **NLTK Word Tokenize.**
- **TextBlob Word Tokenize.**
- **MBSP Word Tokenize.**
- **Pattern Word Tokenize.**
- **Word Tokenization with Python NLTK.**

2.3.5 Padding

Le padding est une technique essentielle dans le prétraitement des données textuelles pour les modèles de deep learning. Lorsque nous transformons des phrases en séquences d'entiers à l'aide de tokenizers, ces séquences peuvent varier en longueur. Les modèles de machine learning, notamment les réseaux de neurones, nécessitent des entrées de taille uniforme pour un traitement efficace. C'est là que le padding intervient.

Pour notre projet, nous avons appliqué le padding sur les séquences tokenisées afin d'assurer une uniformité dans les longueurs des séquences d'entraînement et de validation. Nous avons choisi une longueur maximale de 160 mots pour le padding, car 97% des phrases de notre dataset ne dépassent pas cette longueur.

La fonction de calcul du nombre de mots utilisée révèle les changements significatifs en termes de volume de données textuelles. Ce processus nous permet de voir combien de mots superflus ont été éliminés, contribuant ainsi à la création d'un dataset plus propre et plus pertinent. Les résultats obtenus montrent une réduction substantielle du nombre total de mots, ce qui indique une suppression efficace des éléments indésirables tels que les stopwords, les URLs, les mentions Twitter, et autres bruits textuels.

2.4 Choix et Création du model

2.4.1 RNN :

Pour la création de notre réseau de neurones, nous avons utilisé un modèle séquentiel comportant trois couches distinctes. La première couche est une couche d'embedding qui va transformer les entrées textuelles. Ensuite, nous avons ajouté une couche **LSTM** (Long

Short-Term Memory) pour capturer les dépendances temporelles au sein des séquences de données, avec des mécanismes de dropout pour prévenir le surapprentissage. Enfin, une couche Dense avec une activation softmax est utilisée pour faire les prédictions finales.

- **La commande model = Sequential() :** indique que nous utilisons un modèle séquentiel pour analyser le texte de manière séquentielle. Cela signifie que le modèle va commencer par le premier mot de chaque review et progresser mot par mot jusqu'à atteindre le dernier mot. Cette approche est appliquée à chaque review, permettant au réseau de neurones de traiter les données textuelles dans l'ordre où elles apparaissent.

- **La couche d'embedding :** est couramment utilisée dans les moteurs de recherche pour représenter les mots dans un espace vectoriel. Chaque mot est transformé en un vecteur de dimension fixe, et les mots ayant des significations similaires sont placés à proximité les uns des autres dans cet espace. Par exemple, les mots "chat" et "chien" seront proches dans l'espace vectoriel s'ils sont souvent utilisés dans des contextes similaires. Cette représentation vectorielle permet de mieux comprendre le sens des mots et les relations qu'ils peuvent entretenir entre eux.

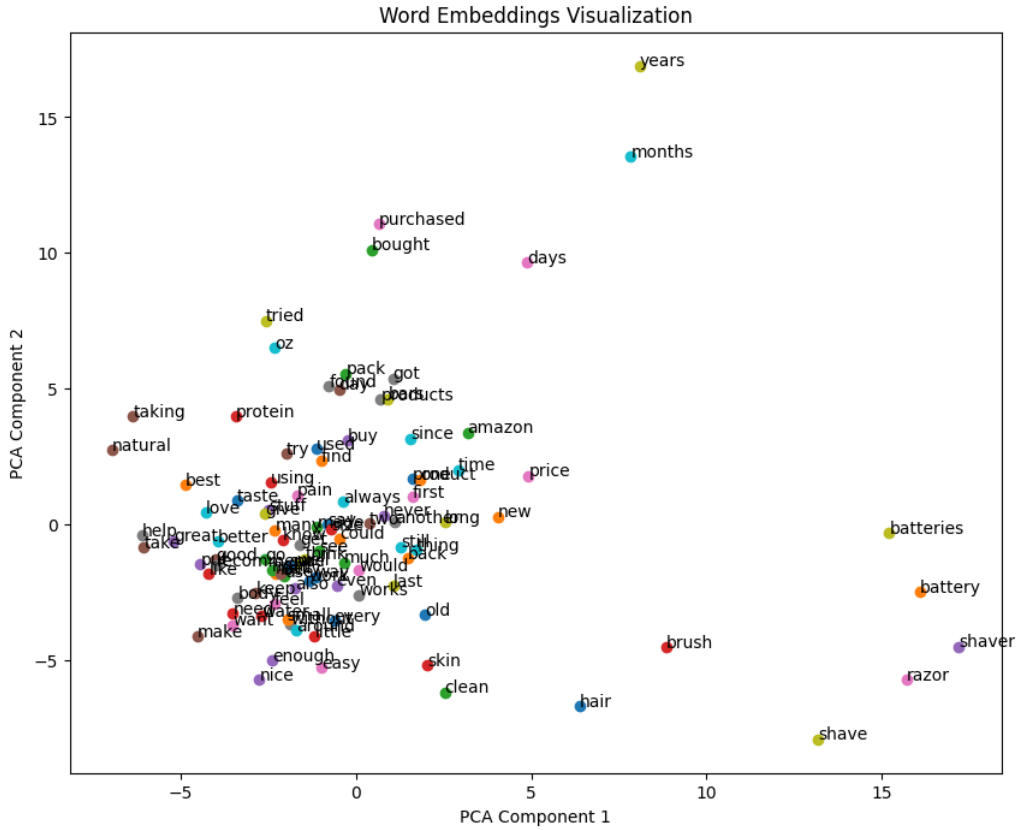


FIGURE 2.2 – example de représentation pour le word embedding.

- **La couche LSTM (Long Short-Term Memory) :** est utilisée pour capturer les dépendances à long terme dans les séquences de données textuelles. En ajoutant model.add(LSTM(units=128, dropout=0.2, recurrent dropout=0.2)), nous intégrons une couche LSTM avec 128 unités cachées. Cette couche est particulièrement efficace pour traiter les séquences, car elle peut se souvenir d'informations pertinentes sur de longues périodes et oublier les informations non pertinentes. Le

paramètre dropout à 0.2 aide à prévenir le surapprentissage en ignorant aléatoirement 20% des unités de sortie pendant l'entraînement. De même, le recurrent dropout à 0.2 ignore aléatoirement 20% des connexions récurrentes, améliorant ainsi la robustesse du modèle.

- **La couche Dense :** est utilisée pour la classification finale dans notre modèle de réseau de neurones.

  En ajoutant **model.add(Dense(units=5, activation='softmax'))**, nous créons une couche dense avec 5 unités de sortie, correspondant aux cinq classes de notes de 1 à 5. La fonction d'activation softmax transforme les scores de sortie en probabilités, permettant ainsi au modèle de prédire la probabilité de chaque classe. Cette configuration est idéale pour les tâches de classification multiclasses, car la somme des probabilités de sortie est toujours égale à 1, facilitant l'interprétation des résultats.

2.4.2 Transformers :

Pour les modèles Transformers, la tokenization est un peu différente de celle des RNN. Au lieu d'analyser le texte de manière séquentielle mot par mot, on utilise la notion d'input IDs et de masks d'attention. Les input IDs sont des identifiants numériques représentant chaque token du texte, tandis que les masks d'attention indiquent au modèle quelles parties du texte doivent être prises en compte lors de l'analyse.

En raison des fortes exigences en ressources de ces modèles, il est courant d'utiliser des modèles préentraînés tels que BERT Large, DistilBERT, RoBERTa, et ELECTRA. Ces modèles ont été formés sur de vastes ensembles de données et peuvent être affinés pour des tâches spécifiques, permettant ainsi de bénéficier de leur puissance sans avoir à les entraîner entièrement depuis zéro.

- **BERT Large : BERT** Large est une version étendue de BERT avec 24 couches de transformateurs et 340 millions de paramètres. Il offre des performances supérieures pour les tâches complexes de traitement du langage naturel, bien qu'il soit plus exigeant en termes de ressources. Pour l'utiliser ainsi que son tokenizer on utilise la commande suivante :

Utilisation de BERT Large

Tokenizer :
tokenizer = BertTokenizer.from_pretrained('bert-large-uncased')
Modèle :
model = BertForSequenceClassification.from_pretrained('bert-large-uncased', num_labels=5)

FIGURE 2.3 – Utilisation de BERT Large.

- **DistilBERT :** est une version allégée et plus rapide de **BERT**, conservant 97% des performances de BERT tout en étant plus rapide et moins gourmand en ressources. Il est formé par distillation de connaissances, où un grand modèle enseigne à un plus petit. Pour l'utiliser ainsi que son tokenizer on utilise la commande suivante :

Utilisation de Distill BERT

Tokenizer :
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
Modèle :

model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased', num_labels=5)

FIGURE 2.4 – Utilisation de DistilBERT.

- **ELECTRA :**ELECTRA entraîne les modèles à distinguer les vrais mots des mots incorrects insérés dans le texte, au lieu de masquer des tokens comme **BERT**. Cette méthode rend l'entraînement plus rapide et efficace, souvent avec de meilleures performances.Pour l'utiliser ainsi que son tokenizer on utilise la commande suivante :

Utilisation de Electra

Tokenizer :
tokenizer = ElectraTokenizer.from_pretrained('google/electra-small-discriminator')
Modèle :

model = ElectraForSequenceClassification.from_pretrained('google/electra-small-discriminator', num_labels=5)

FIGURE 2.5 – Utilisation de Electra.

- **RoBERTa :** améliore **BERT** en optimisant le pré-entraînement avec plus de données et de temps d'entraînement, et sans la tâche NSP (Next Sentence Prediction). Il offre des performances de pointe sur de nombreuses tâches NLP. Pour l'utiliser ainsi que son tokenizer on utilise la commande suivante :

Utilisation de RoBerta

Tokenizer :
tokenizer = RobertaTokenizer.from_pretrained('roberta-base')
Modèle :
model = RobertaForSequenceClassification.from_pretrained('roberta-base', num_labels=5)

FIGURE 2.6 – Utilisation de RoBerta.

2.5 Conclusion

Dans le domaine de la classification de textes, l'identification et la mise en œuvre d'une séquence d'étapes essentielles constituent la base de tout projet. Peu importe les choix du modèle ou de méthode, le processus demeure fondamentalement le même quel que soit le choix du modèle.Ces étapes fondamentales guident les praticiens à travers le labyrinthe complexe de la prédiction et de la classification de texte. En intégrant des techniques telles que le preprocessing, la suppression des stopwords, le stemming, la lemmatisation, la tokenization, et le padding, les praticiens peuvent améliorer la qualité des données et la performance des modèles, contribuant ainsi à l'avancement de ce domaine en constante évolution. En utilisant des modèles variés tels que les réseaux de neurones récurrents (RNN) ou les modèles Transformers, ils peuvent exploiter au mieux les capacités des technologies modernes pour obtenir des résultats précis et fiables dans la classification de textes.

CHAPITRE

3

DISCUSSION ET COMPARAISON DES RÉSULTATS

3.1 Introduction

Dans ce chapitre, nous allons présenter et discuter les résultats obtenus à travers les différentes étapes de prétraitement des données et les performances des modèles d'apprentissage automatique que nous avons utilisés.

Nous examinerons d'abord les résultats des Stopwords, le Stemming ainsi que la lemmatization , et nous finirons par la tokenisation et le padding. Ensuite, nous comparons les performances des Réseaux de Neurones Récurrents (RNN) et des Transformers en utilisant diverses métriques comme la précision (accuracy) et la perte (loss). À chaque étape, nous analyserons les résultats pour déterminer l'efficacité des techniques et des modèles appliqués dans notre projet.

3.2 Stopwords avec NLTK

Après implémentation du processus de suppression des stopwords dans notre dataset en utilisant la bibliothèque NLTK, nous avons obtenu une liste de 179 mots. Cette liste représente la version mise à jour des stopwords du dictionnaire anglais dans la bibliothèque NLTK à ce jour.

Ces stopwords ont été identifiés et sélectionnés spécifiquement pour le traitement des textes en anglais afin d'éliminer les mots courants et peu informatifs qui ne contribuent pas significativement à l'analyse des données textuelles. En utilisant cette liste de stopwords actualisée, nous améliorons la qualité de nos données en éliminant le bruit et en nous concentrant sur les mots-clés essentiels pour nos analyses ultérieures.

Stopwords					
a	about	above	after	again	all
an	and	any	are	as	at
be	because	before	between	both	but
by	can	do	down	during	each
few	for	from	further	have	he
how	i	if	in	into	is
it	its	let	me	more	my
no	not	of	on	only	or
other	our	out	over	so	some
than	that	the	their	them	this
to	up	very	we	were	what
when	where	which	while	who	whom
why	with	you	your	during	against

TABLE 3.1 – Liste des stopwords les plus importants.

3.3 Stemming vs Lemmatization

Dans le cadre de notre travail , nous nous sommes concentrés sur l'analyse et le prétraitement des données textuelles issues des avis clients dans la section santé et soins d'Amazon.

Nous avons comparé deux techniques couramment utilisées en traitement automatique du langage : le stemming et la lemmatisation.

Le stemming, représenté par l'algorithme de Porter, consiste à réduire les mots à leurs racines en supprimant les suffixes, ce qui peut parfois aboutir à des formes non valides mais conserve l'efficacité de traitement.

En revanche, la lemmatisation utilise des dictionnaires et des règles grammaticales pour ramener les mots à leur forme de base ou lemme, offrant une précision accrue en produisant des mots valides. Notre analyse a évalué l'impact de ces deux techniques sur la qualité des données traitées et leur influence sur les performances des modèles de classification de texte que nous avons développés.

En appliquant les 2 algorithmes au dataset, les résultats suivants ont été obtenus :

- **Stemming :**

Séquences Avant le Stemming

Ligne 21 :
The gf loves this and when she lost it when moving it was her first thing she had to get back. Great vibe low noise, alittle on the firm side though.
Ligne 23 :

It is very small compared to what I thought it would be, but even so it didn't work when we got it.
Ligne 26 :

I take 3 at night before I go to bed, it helps me rest better!! TRY IT!! AND THE PRICE IS OK.
Ligne 29 :

I like the Solaray Brand. I am not allergic to it. Vitamin D-3 was recommended by my doctor. The Vitamin D-3 keeps me healthy.

FIGURE 3.1 – Séquences Avant le Stemming

Séquences Après le Stemming

Ligne 21 :
the gf love thi and when she lost it when move it wa her first thing she had to get back. great vibe low nois, alittl on the firm side though.
Ligne 23 :

it is veri small compar to what i thought it would be, but even so it didn't work when we got it.
Ligne 26 :

i take 3 at night be for i got to bed, it help me rest better!! tri it!! and the price is ok.
Ligne 29 :

i like the solaray brand. i am not allerg to it. vitamin d-3 wa recommend by my doctor. the vitamin d-3 keep me healthy.

FIGURE 3.2 – Séquences Après le Stemming

- **Lemmatization :**

Séquences avant la Lemmatisation

Ligne 21 :
The gf loves this and when she lost it when moving it was her first thing she had to get back. Great vibe low noise, alittle on the firm side though.
Ligne 23 :

It is very small compared to what i thought it would be, but even so it didn't work when we got it.
Ligne 26 :

I take 3 at night be for I got to bed, it helps me rest better!! TRY IT!! AND THE PRICE IS OK.
Ligne 29 :

I like the Solaray Brand. I am not allergic to it. Vitamin D-3 was recommended by my doctor. The Vitamin D-3 keeps me healthy.

FIGURE 3.3 – Séquences avant la Lemmatisation

Séquences après la Lemmatisation

Ligne 21 :
the gf love this and when she lose it when move it be her first thing she have to get back . great vibe low noise , alittle on the firm side though .
Ligne 23 :

it be very small compare to what I think it would be , but even so it do not work when we get it
Ligne 26 :

I take 3 at night be for I get to bed , it help I rest well!! try it!! and the price be ok
Ligne 29 :

I like the Solaray Brand . I be not allergic to it . vitamin D-3 be recommend by my doctor . the vitamin D-3 keep I healthy .

FIGURE 3.4 – Séquences après la Lemmatisation

ça nous permet de classer les résultats notables dans le tableau suivant :

Mot (DS)	**Stemming**	**Lemmatization**
had	had	have
always	alway	always
beauty	beauti	beauty
loves	love	love
this	thi	this
wore	wore	wear
does	doe	do
goes	goe	go
lost	lost	lose
relieving	reliev	relieve
PRICE	price	PRICE

TABLE 3.2 – Comparaison des résultats du Dataset entre le Stemming et la Lemmatisation

Dans notre étude comparative entre le Stemming et la Lemmatisation, nous avons constaté des différences notables non seulement en termes de temps d'exécution mais aussi dans la manière dont chaque technique traite les mots. Le Stemming, qui consiste à réduire les mots à leur racine en supprimant les suffixes et les préfixes, est un processus relativement rapide. Dans notre cas, le Stemming a pris 5 minutes et 30 secondes pour traiter l'ensemble du dataset. Cependant, cette rapidité se fait au détriment de la précision et de la signification des mots. Le Stemming ne prend pas en compte le contexte ou la signification des mots ; il se contente de retirer les terminaisons courantes comme "s", "es", et "ing". Cela peut entraîner des erreurs, comme transformer "always" en "alway" alors que c'est un mot invariable, ou "does" en "doe",on comprend que le Stemming ne distingue pas les temps des verbes ni la conjugaison , de plus il convertit systématiquement les majuscules en minuscules.

En revanche, la Lemmatisation est une technique plus sophistiquée et plus lente, prenant 1 heure et 24 minutes pour traiter le même dataset. Contrairement au Stemming, la Lemmatisation considère le contexte et la signification des mots. Elle utilise des dictionnaires et des règles grammaticales pour ramener les mots à leur forme de base ou lemme. Par exemple, elle peut convertir "does" en "do" et "running" en "run", mais elle ne supprimera pas nécessairement les "s" ou "es" à la fin des mots si ceux-ci sont naturellement singuliers. La Lemmatisation est capable de reconnaître les temps des verbes et de les ramener à leur forme infinitive, préservant ainsi la signification correcte des mots dans leur contexte. Elle respecte également la forme originale des mots, évitant les transformations incorrectes des majuscules et des terminaisons en "y".

En résumé, bien que le Stemming soit plus rapide, il présente des inconvénients significatifs en termes de précision et de conservation du sens des mots. La Lemmatisation, bien que plus lente, offre une meilleure qualité de traitement en prenant en compte le contexte

et la signification des mots, ce qui est crucial pour les applications nécessitant une compréhension approfondie du texte. Cette comparaison souligne l'importance de choisir la technique appropriée en fonction des exigences spécifiques du projet et des compromis entre rapidité et précision.

3.4 Tokenization avec NLTK

Nous utilisons la bibliothèque Keras pour la tokenization, ce qui facilite grandement la transformation de nos textes en séquences d'entiers. Cette approche s'intègre parfaitement avec nos modèles de machine learning et de deep learning, améliorant ainsi l'efficacité du prétraitement des données.

Données Encodées en Séquences d'Entiers

Ligne 1 : ensure great product put blender ice cubes almond butter makes delicious shake puts calories needed vitamins minerals diet now sold chocolate would really happy

Encodage :

[17, 1608, 380, 1684, 77, 12, 178, 2, 9679, 167, 5292, 4159, 167, 183, 77, 50, 40, 793, 1296, 380, 215, 831, 2816, 32, 9, 2, 141, 32, 1295, 94698, 394, 122, 148, 83, 20, 440, 707, 860, 1287, 650, 6645, 1828, 543, 329, 149, 740, 35, 973, 42, 427]

FIGURE 3.5 – Résultat de la Tokenization.

Lors de la tokenisation avec notre Tokenizer de TensorFlow Keras, cette phrase est convertie en une séquence d'entiers représentant les mots du texte. Le résultat de cette tokenisation pour notre phrase exemple est la séquence suivante :

[17, 1608, 380, 1684, 77, 12, 178, 2, 9679, 167, 5292, 4159, 167, 183, 77, 50, 40, 793, 1296, 380, 215, 831, 2816, 32, 9, 2, 141, 32, 1295, 94698, 394, 122, 148, 83, 20, 440, 707, 860, 1287, 650, 6645, 1828, 543, 329, 149, 740, 35, 973, 42, 427].

Chaque entier dans cette séquence correspond à un mot unique du texte d'origine, permettant ainsi de traiter les données textuelles de manière structurée et efficace pour nos modèles de machine learning et de deep learning.

3.5 Résultat Du Padding

Pour notre projet, nous avons appliqué le padding sur les séquences tokenisées afin d'assurer une uniformité dans les longueurs des séquences d'entraînement et de validation. Nous avons choisi une longueur maximale de 160 mots pour le padding, car 97% des phrases de notre dataset garantissant ainsi que toutes les séquences d'entrée dans notre modèle ont la même longueur.

Données Encodées en Séquences d'Entiers

Séquence Tokenizé sans Padding :
[17, 1608, 380, 1684, 77, 12, 178, 2, 9679, 167, 5292, 4159, 167, 183, 77, 50, 40, 793, 1296, 380, 215, 831, 2816, 32, 9, 2, 141, 32, 1295, 94698, 394, 122, 148, 83, 20, 440, 707, 860, 1287, 650, 6645, 1828, 543, 329, 149, 740, 35, 973, 42, 427]

Séquence avec Padding :
[17, 1608, 380, 1684, 77, 12, 178, 2, 9679, 167, 5292, 4159, 167, 183, 77, 50, 40, 793, 1296, 380, 215, 831, 2816, 32, 9, 2, 141, 32, 1295, 94698, 394, 122, 148, 83, 20, 440, 707, 860, 1287, 650, 6645, 1828, 543, 329, 149, 740, 35, 973, 42, 427, 0, 0, 0, 0, ..., 0]

FIGURE 3.6 – Résultat du Padding.

Après que nous avons terminé l'étape de prétraitement, nous pouvons affirmer que notre dataset est prêt pour l'analyse. Il faut noter que afin d'évaluer l'impact de cette phase, il est important de réappliquer la fonction de calcul du nombre de mots dans le notre dataset. Cette évaluation nous permettra de quantifier combien de mots ont été supprimés et combien de données ont été nettoyées au cours de ce processus. En effectuant cette analyse, nous obtenons une vision claire des modifications apportées à notre dataset et de la manière dont celles-ci peuvent influencer les performances de nos modèles de machine learning.

3.6 RNN vs Transformers

Pour évaluer les performances de chaque approche, nous avons utilisé plusieurs métriques, notamment l'accuracy (précision) ainsi que le loss (perte). Ces métriques nous ont permis de mesurer la performance du modèle en termes de précision des prédictions et de minimisation des erreurs. En comparant ces métriques à travers les approches, nous avons pu déterminer l'efficacité de chaque méthode et identifier la meilleure stratégie pour notre tâche spécifique.

- **Accuracy (Précision) :** La proportion de prédictions correctes faites par le modèle par rapport au nombre total de prédictions. En d'autres termes, c'est le pourcentage de fois où le modèle a correctement prédit la catégorie de la donnée.

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}}$$

- **Loss (Perte) :** Une mesure de l'erreur du modèle. Elle indique à quel point les prédictions du modèle sont éloignées des valeurs réelles. Un loss plus faible signifie que le modèle fait des prédictions plus proches des valeurs attendues.

$$\text{CrossEntropyLoss} = -\sum_{i=1}^{C} y_i \cdot \log(p_i)$$

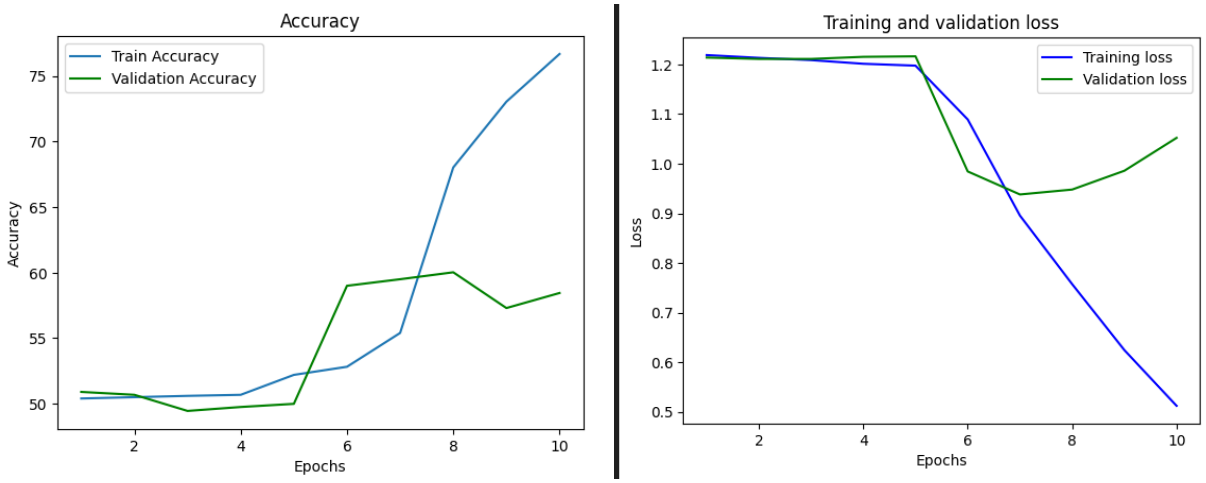
où :

- C : est le nombre de classes.
- y_i : est une variable indicatrice qui est 1 si la classe i est la vraie classe pour l'observation actuelle, et 0 sinon.
- p_i : est la probabilité prédite que l'observation appartienne à la classe i.

3.6.1 RNN

Dans notre étude sur les réseaux de neurones récurrents (RNN), nous avons exploré trois approches distinctes pour évaluer les performances de notre modèle. Premièrement, nous avons formé un modèle RNN en utilisant seulement 15 % de l'ensemble de données. Cette méthode nous a permis de comprendre comment notre modèle se comporterait avec un sous-ensemble limité de données, tout en réduisant le temps de calcul nécessaire pour l'entraînement. Deuxièmement, nous avons formé un RNN en utilisant l'intégralité de l'ensemble de données. Cette approche visait à maximiser l'utilisation des données disponibles pour améliorer la précision et la généralisation du modèle. Enfin, nous avons mis en œuvre une validation croisée (Cross Validation) sur notre RNN. Cette technique consiste à diviser notre ensemble de données en plusieurs sous-ensembles, entraînant le modèle sur une partie de ces sous-ensembles et validant sur les autres. La validation croisée nous a fourni une évaluation plus robuste et fiable des performances du modèle en tenant compte de la variabilité des données et en réduisant le risque de surapprentissage.

- **Réseau Neuronal Récurrents avec utilisation de 15% du Dataset**

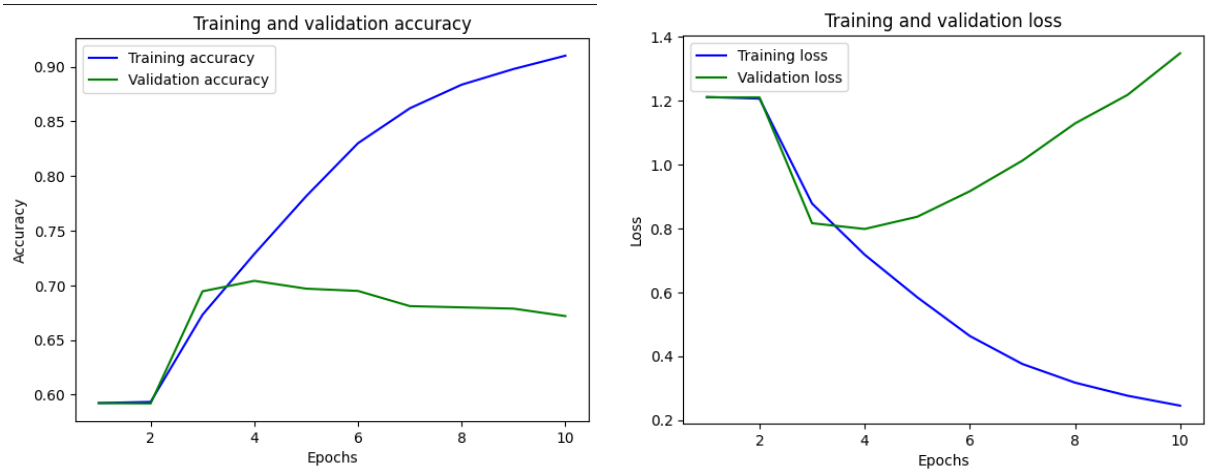


(a) (b)

FIGURE 3.7 – Représentation du Loss et de l'accuracy.

En utilisant seulement 15% du dataset pour votre PFE, on remarque une augmentation significative de la précision d'entraînement à partir de l'époque 5, atteignant environ 77% à l'époque 10, tandis que la précision de validation reste à 59%. Cette divergence indique un surapprentissage. La perte d'entraînement diminue de 1.2 à 0.5, mais la perte de validation ne descend pas en dessous de 1.0, confirmant une capacité de généralisation limitée.

- **Réseau Neuronal Récurrents avec utilisation de 100% du Dataset**



(a) (b)

FIGURE 3.8 – Représentation du Loss et de l'accuracy.

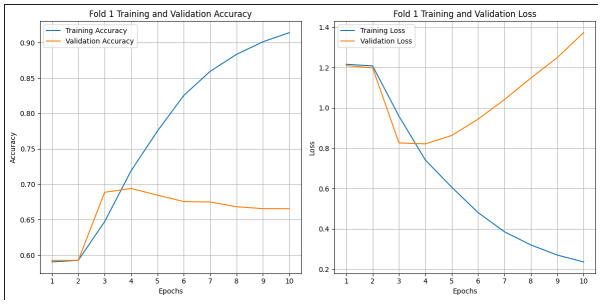
Lors de l'utilisation de 100% du dataset pour votre PFE, on observe une amélioration significative de la précision d'entraînement à partir de la 2ème époque, atteignant près de 90% à la 10ème époque. Cependant, la précision de validation, bien qu'elle augmente initialement, fluctue ensuite autour de 65-67%. Cette divergence entre la précision d'entraînement et de validation suggère un surajustement du modèle.

En ce qui concerne la perte, la perte d'entraînement diminue nettement au début et se stabilise ensuite, tandis que la perte de validation diminue initialement mais commence à augmenter après la 2ème époque. Cette augmentation de la perte de validation indique également un surajustement, car idéalement, la perte de validation devrait diminuer de manière similaire à la perte d'entraînement.

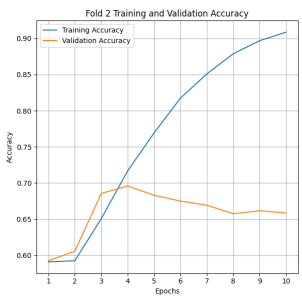
Ces observations soulignent l'importance de surveiller à la fois la perte et la précision pendant l'entraînement et la validation pour évaluer la performance du modèle et éviter le surapprentissage.

- **Réseau Neuronal Récurrents Avec Validation Croisée**

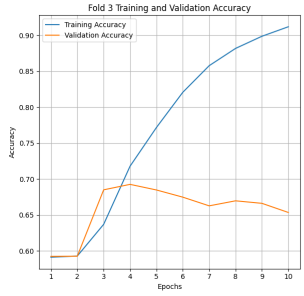
Dans le cadre de notre projet, nous avons adopté une approche de validation croisée de type K-Fold. Cette méthode est ainsi nommée car 'K' représente le nombre de sous-ensembles dans lesquels nous avons divisé notre jeu de données (80% pour l'entraînement et 20% pour le test). À chaque itération, ou 'fold', les tranches de données utilisées pour l'entraînement et le test changent. Cette rotation systématique permet d'obtenir des résultats moins biaisés et plus fiables. En effet, selon les tranches de données sélectionnées, nous pouvons obtenir de meilleurs résultats ou, au contraire, des résultats moins bons. La validation croisée K-Fold nous permet donc d'obtenir une moyenne de ces résultats, offrant ainsi une vision plus précise de la performance de notre modèle. De plus, cette méthode contribue à éviter le surapprentissage dans certains cas, en garantissant que notre modèle est bien généralisable et non spécifique à un sous-ensemble particulier de nos données [24].



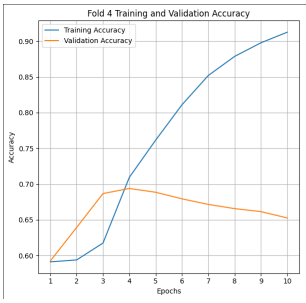
(a) Accuracy et loss pour Fold 1.



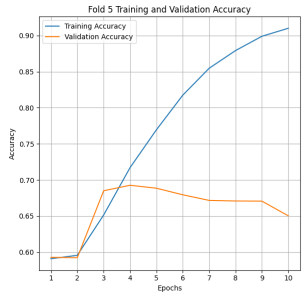
(b) Accuracy et loss pour Fold 2.

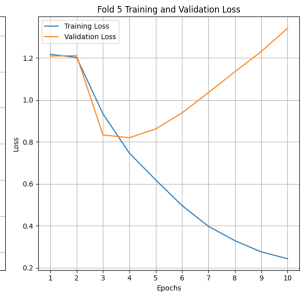


(c) Accuracy et loss pour Fold 3.



(d) Accuracy et loss pour Fold 4.





(e) Accuracy et loss pour Fold 5.

FIGURE 3.9 – Accuracy et loss pour la Validation croisée.

nous avons mis en œuvre une validation croisée sur un Réseau de Neurones Récurrent (RNN). le loss est resté globalement le même, indiquant que le modèle a appris de manière constante à partir des données d'entraînement.

En ce qui concerne l'accuracy, nous avons observé une légère variation entre les différents folds. Par exemple, l'accuracy était de 67% pour Fold 1, tandis que pour le Fold 5, elle était de 65%. Cette variation souligne l'importance de la validation croisée, qui permet d'obtenir une estimation plus précise de la performance du modèle en moyennant les résultats de plusieurs Folds.

La validation croisée change la place des portions de données à chaque fois, ce qui permet d'évaluer la performance du modèle sur différentes parties du dataset. Cela s'est avéré être un outil précieux pour évaluer la généralisation de notre modèle RNN. Ainsi, malgré les défis rencontrés, la validation croisée a permis d'obtenir une moyenne plus représentative de la performance du modèle .

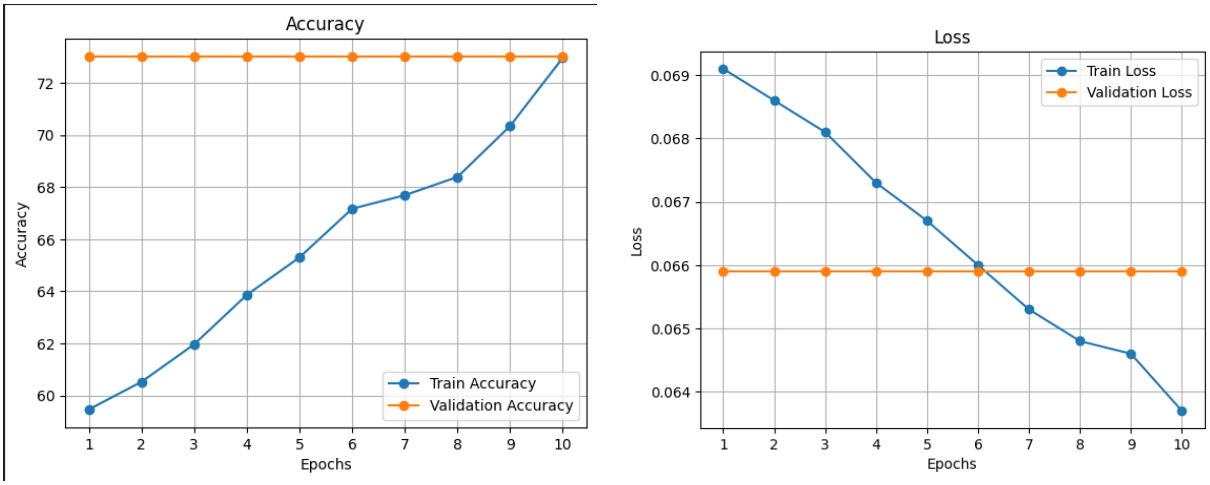
	RNN 15 %	**RNN 100 %**	**RNN Cross Validation**
Accuracy	77	91	91
Loss	0.5	0.23	0.24
Accuracy (Test)	59	**67**	**65**
Loss (Test)	1.05	1.36	1.39

TABLE 3.3 – Comparaison des métriques pour différentes approches RNN.

Ces résultats, comme illustrés dans Table 3.3, soulignent l'importance de l'utilisation de différentes approches pour l'entraînement du modèle. Bien que l'utilisation de 100% du dataset ait permis d'améliorer la précision de l'entraînement, elle a également conduit à une augmentation de la perte sur les données de validation, indiquant un possible surajustement. D'autre part, la validation croisée a permis de maintenir une précision élevée tout en contrôlant la perte, ce qui suggère une meilleure généralisation du modèle. Cela démontre l'importance de la validation croisée dans l'évaluation des performances du modèle.

3.6.2 Transformers (BERT)

- **Bert Large :**

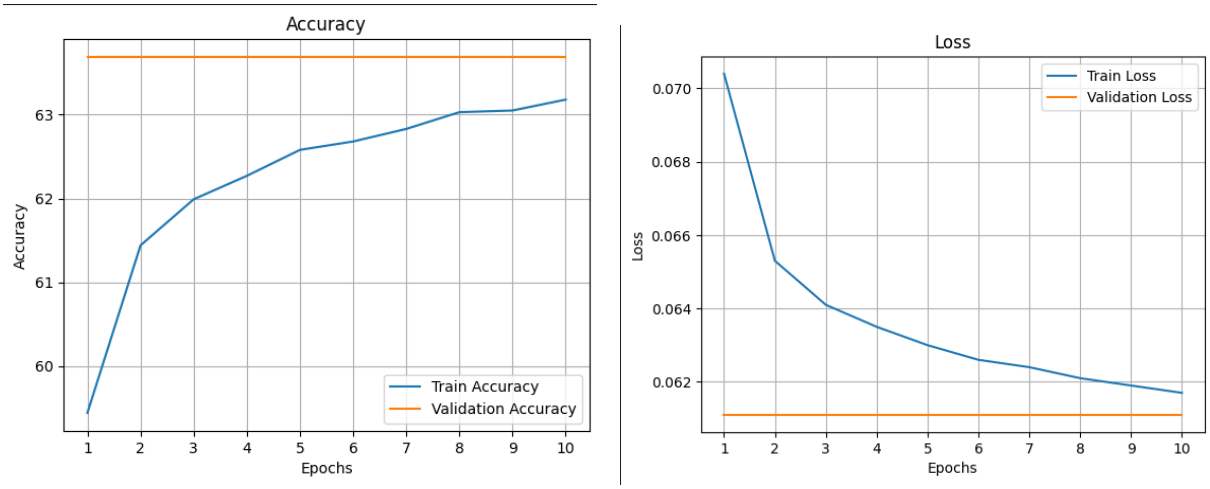


(a) (b)

FIGURE 3.10 – Représentation du Loss et de l'accuracy pour Bert Large.

Nous avons constaté que le modèle Bert Large est extrêmement stable durant son apprentissage. Il a démontré une capacité remarquable à maintenir une progression régulière, nous permettant d'atteindre une accuracy impressionnante de 73% tant pour l'entraînement que pour le test. En ce qui concerne le loss, les valeurs sont restées faibles et constantes, se situant entre 0.06 et 0.07, ce qui indique une convergence efficace du modèle. Ces résultats témoignent de la robustesse et de la fiabilité de Bert Large dans les tâches de classification de textes.

- **Distill Bert en utilisant 15% du Dataset :**



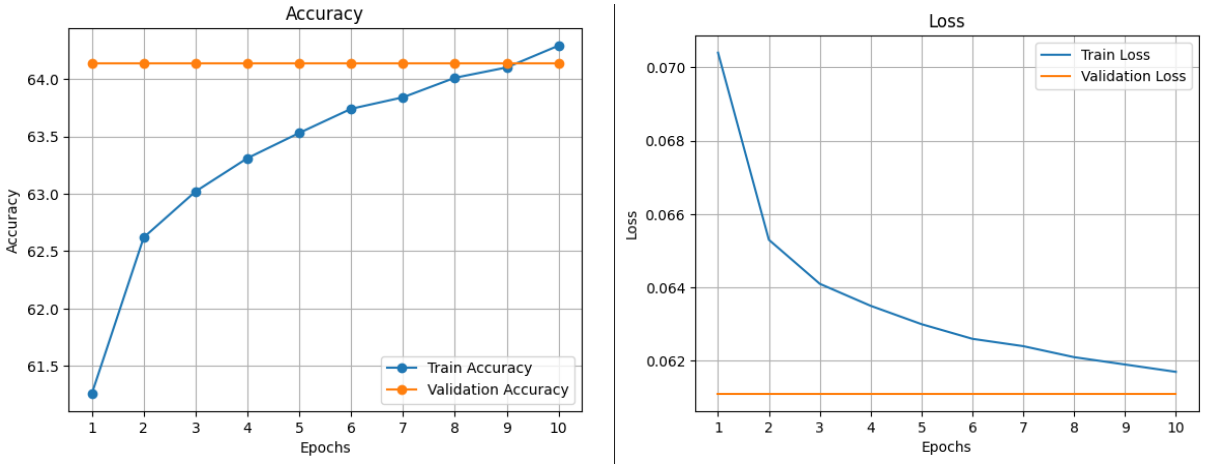
(a) (b)

FIGURE 3.11 – Représentation du Loss et de l'accuracy pour Distill Bert 15%.

nous avons également travaillé avec le modèle Distill Bert, en utilisant seulement 15% du dataset. Nous avons observé que le loss est resté très constant, ce qui indique une convergence stable du modèle. Cependant, en termes de précision, nous

avons atteint seulement 63% pour le test et la validation. Bien que cette précision soit inférieure à celle obtenue avec Bert Large, elle reste néanmoins acceptable.

- **Distill Bert en utilisant 15% du Dataset :**

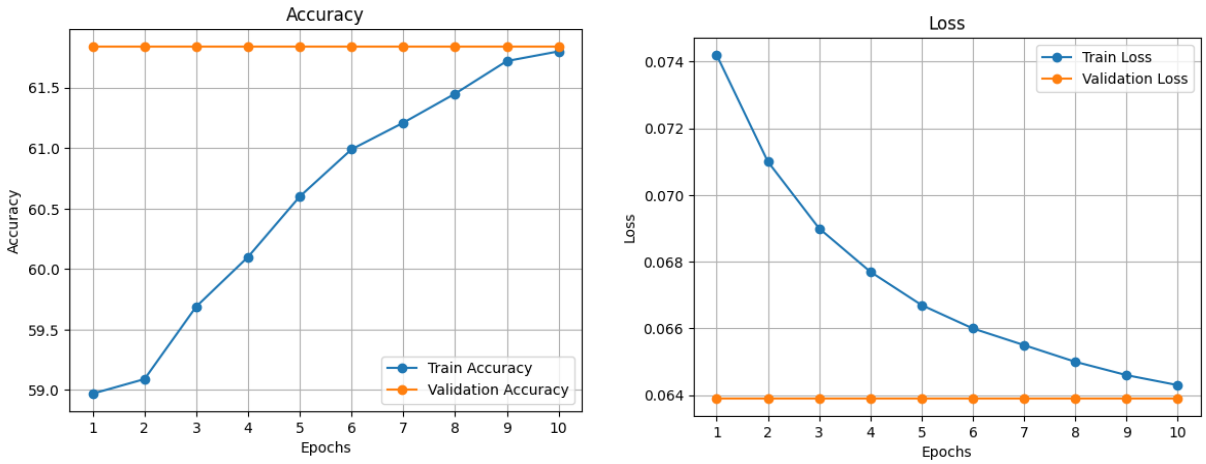


(a) (b)

FIGURE 3.12 – Représentation du Loss et de l'accuracy pour Distill Bert 100%

Nous avons également expérimenté avec le modèle Distill Bert en utilisant 100% du dataset. Nous avons constaté que le modèle reste globalement le même que celui formé précedemment. Le loss est resté aux mêmes valeurs et aussi stable, indiquant une convergence efficace du modèle. En termes de précision, nous avons atteint 64% pour le test et l'entraînement. Ces résultats prouvent que les modèles BERT peuvent être efficaces sans nécessiter une grande taille de données, démontrant leur capacité à apprendre de manière significative à partir d'ensembles de données de différentes tailles.

- **Electra :**



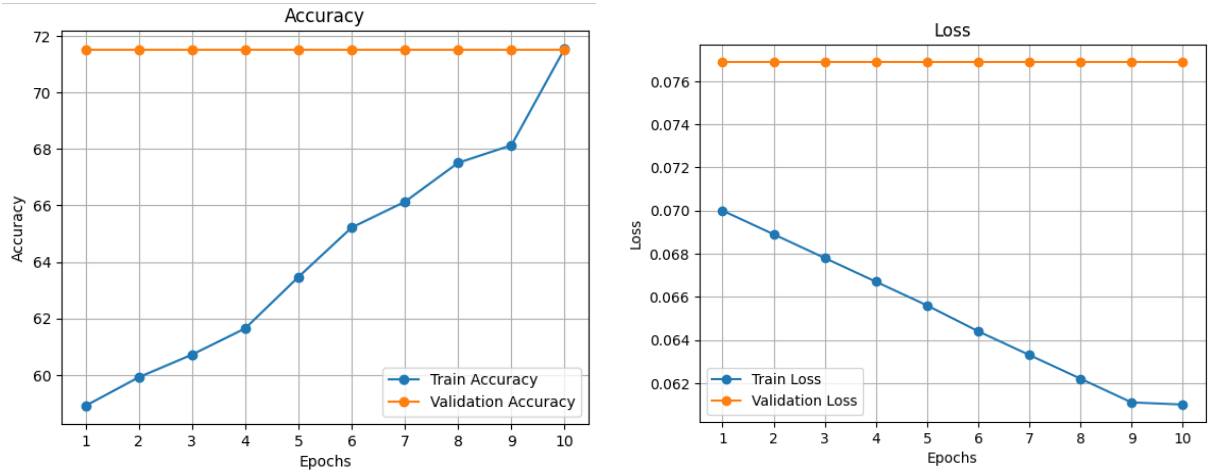
(a) (b)

FIGURE 3.13 – Représentation du Loss et de l'accuracy pour Electra.

Ce modèle s'est avéré être le moins performant jusqu'à présent, avec une accuracy

de seulement 61% pour le test et l'entraînement. Malgré cela, le loss est resté relativement stable,Ces résultats soulignent l'importance de l'exploration de différents modèles et techniques pour optimiser les performances de classification.

- **RoBerta :**
  Nous avons également testé le modèle RoBERTa. Ce modèle s'est avéré être le deuxième plus performant parmi ceux que nous avons testés. RoBERTa a atteint une accuracy impressionnante de 71% pour le test et l'entraînement. En ce qui concerne le loss, il est resté stable et relativement faible, se situant entre 0.07 et 0.06. Ces résultats démontrent la robustesse de RoBERTa et sa capacité à fournir des performances de classification de textes de haute qualité.



(a) (b)

FIGURE 3.14 – Représentation du Loss et de l'accuracy pour Roberta.

	Distill 15%	**Distill 100%**	**Bert Large**	**Roberta**	**Electra**
Accuracy	63.18%	64.29%	72.97%	71.55%	61.80%
Loss	0.0617	0.059	0.063	0.061	0.064
Accuracy(Test)	63.69%	64.14%	**73.02%**	**71.55%**	61.84%
Loss(Test)	0.0611	0.063	0.065	0.07	0.0639

TABLE 3.4 – Performance des différents modèles de BERT

Ces résultats soulignent l'importance de l'exploration de différents modèles et techniques pour optimiser les performances de classification de textes. Même si certains modèles ont montré des performances supérieures à d'autres comme BERT Large et RoBerta, tous les modèles ont démontré une capacité à apprendre de manière stable à partir des données.

- **Comparaison Finale**

Modèle	**Entrainement**		**Test**	
	Accuracy	**Loss**	**Accuracy**	**Loss**
Modèles Transformers				
Distill Bert 15%	63.18%	0.0617	63.69%	0.0611
Distill Bert 100%	64.29%	0.059	64.14%	0.063
Bert Large	72.97%	0.063	73.02%	0.065
Roberta	71.55%	0.061	71.55%	0.07
Electra	61.80%	0.064	61.84%	0.0639
Modèles RNN				
RNN 15%	77%	0.5	59%	1.05
RNN 100%	91%	0.23	67%	1.36
RNN Cross Validation	91%	0.24	65%	1.39

TABLE 3.5 – Tableaux représentatifs des performances pour les RNN et les Transformers.

Dans le cadre de notre travail, nous avons mené une étude comparative entre deux méthodes de classification de texte : les Réseaux de Neurones Récurrents et les Transformers.

Les résultats obtenus montrent que les **RNN** ont atteint une précision d'environ 65%, tandis que le meilleur modèle de Transformers, **BERT**, a atteint une précision de 73%. Malgré le temps d'apprentissage plus long des modèles Transformers, nous avons obtenu de meilleures performances globales avec Bert Large et Roberta.

Cependant, la précision de Distill Bert et Electra était légèrement inférieure à celle des **RNN**. Leur perte (loss) était proche de 0.06, comparativement aux RNN qui avaient une perte entre 1.0 et 1.3.

Nous avons également observé que les Transformers sont un modèle plus stable, avec une perte relativement stable pendant l'entraînement et la validation. Contrairement aux RNN, ils n'ont pas montré de problème de surapprentissage. En effet, alors que les RNN affichaient une précision de 91% pendant l'entraînement, ils n'atteignaient que 67% lors de la validation.

Malgré ces résultats, il est important de noter que la comparaison entre les deux approches ne peut pas être tranchée de manière aussi catégorique. Chaque méthode a ses avantages et ses inconvénients, et le choix entre elles dépend de nombreux facteurs, tels que le temps d'apprentissage et la stabilité du modèle. Cependant, dans le domaine de la classification de texte, les Transformers ont montré des performances relativement supérieures.

3.7 Environnement de développement

Pour la concrétisation de ce projet, nous avons utilisé un ensemble de logiciels spécialisés, d'outils techniques et de bibliothèques informatiques. Chaque élément a joué un rôle crucial pour assurer le développement des différentes étapes de ce travail.

3.7.1 Langages et Outils utilisés

- **Python 3 :** Python est un langage de programmation puissant et facile à apprendre. Il permet une approche simple mais efficace de la programmation. Sa syntaxe élégante, son typage dynamique et son interprétation en font un langage idéal pour l'écriture et le développement rapide dans de nombreux domaines et sur la plupart des plateformes.

- **Visual Studio code :** est un environnement de développement intégré (IDE) complet développé par Microsoft. Il permet de développer l'aide du langage de votre choix. Il est livré avec un ensemble d'outils et de fonctionnalités qui améliorent chaque étape du développement de votre projet.

- **Google Colab :** Google Colab, ou Colaboratory, est un service cloud gratuit offert par Google. Il est basé sur Jupyter Notebook et est destiné à la formation et à la recherche en apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud, sans avoir besoin d'installer quoi que ce soit sur votre ordinateur à part un navigateur.

- **LaTeX :** est un système de composition de documents qui permet de créer des documents scientifiques et techniques de haute qualité. Contrairement aux traitements de texte courants Avec LaTeX, on prend du texte simple et on l'enrichit avec des balises.

- **Overleaf :** est un éditeur LaTeX en ligne, collaboratif en temps réel. Il permet aux utilisateurs d'écrire, de modifier et de publier des documents universitaires et scientifiques.

3.7.2 Bibliothèques Utilisées

- **Pandas :** C'est une bibliothèque Python open source qui fournit des structures de données performantes et faciles à utiliser, ainsi que des outils d'analyse de données. Elle est particulièrement utile pour la manipulation et l'analyse des données comme les datasets.

- **spaCy :** C'est une bibliothèque Python open source pour le traitement avancé du langage naturel (NLP). Elle est conçue spécifiquement construire des applications qui traitent et "comprennent" de grands volumes de texte.

- **NLTK (Natural Language Toolkit) :** C'est une suite de bibliothèques et de programmes pour le traitement symbolique et statistique du langage naturel. Elle fournit des interfaces faciles à utiliser , ainsi qu'une suite de bibliothèques de traitement de texte pour la classification, la tokenisation, le stemming, le marquage, l'analyse syntaxique et le raisonnement sémantique.

- **Scikit-learn :** C'est une bibliothèque Python pour l'apprentissage machine. Elle comprend des algorithmes de classification, de régression et de clustering, ainsi que des outils pour la sélection et l'évaluation des modèles, le prétraitement des données et bien d'autres.

- **Transformers :** C'est une bibliothèque Python pour le traitement du langage naturel (NLP) qui fournit des milliers de modèles pré-entraînés pour effectuer des tâches sur différentes modalités telles que le texte, la vision et l'audio.

- **TensorFlow :** C'est un framework open source pour l'apprentissage machine développé par Google. Il permet de développer et de former des modèles d'apprentissage profond avec une grande facilité et une grande flexibilité.

- **PyTorch :** C'est une bibliothèque d'apprentissage machine open source pour Python, développée principalement par l'équipe de recherche en intelligence artificielle de Facebook. Elle fournit principalement le calcul tensoriel (comme NumPy) avec une forte accélération GPU.

- **Tqdm :** C'est un module Python qui permet de générer des barres de progression. Il est conçu pour avoir un faible surcoût, en utilisant des algorithmes pour prédire le temps restant et pour sauter les affichages d'itération inutiles.

- **tf-keras :** C'est une interface de haut niveau pour TensorFlow qui permet de construire et d'entraîner des modèles en utilisant l'API Keras, tout en profitant des avantages de performance de TensorFlow.

3.8 Description et utilisation du DataSet

Le dataset que nous utilisons pour notre **(PFE)** provient de la rubrique health and care d'Amazon, une section dédiée à la vente de produits de beauté et de soin. Ce dataset est une mine d'informations précieuses pour comprendre les tendances, les préférences des clients et évaluer la qualité des produits dans ce domaine spécifique.

Il se compose d'environ 428 000 avis clients, chacun étant associé à un produit spécifique. Chaque avis est structuré en trois éléments principaux :

- **Titre :** Ce champ représente le nom du produit évalué. Il permet d'identifier rapidement le produit concerné par l'avis.

- **Avis :** Ce champ contient le retour d'expérience détaillé du client sur le produit. Les avis peuvent inclure des informations sur l'efficacité du produit, son rapport qualité-prix, son emballage, son odeur, sa texture, etc. Les clients peuvent égale-

ment partager leurs impressions subjectives, leurs anecdotes, et leurs recommandations.

- **Note :** La note attribuée par le client est un indicateur essentiel de l'appréciation globale du produit. Cette note varie de 1 à 5, où 1 représente une évaluation très négative et 5 une évaluation très positive.

Chaque produit du dataset est associé à un ensemble d'avis, avec un minimum de 5 avis par produit. Cette diversité d'opinions permet d'avoir une vision plus complète de la satisfaction des clients vis-à-vis de chaque produit.

Après un processus de prétraitement des données, tel que la suppression des doublons, la correction des fautes d'orthographe, et la lemmatisation des mots, environ 97.7% des avis ont été réduits à moins de 160 mots. Cela garantit que les avis analysés sont concis tout en conservant l'essentiel de l'opinion du client.

3.9 Conclusion

Ces observations soulignent l'importance de choisir la technique et le modèle appropriés en fonction des besoins spécifiques du projet, en tenant compte des compromis entre rapidité, précision, et capacité de généralisation. Les résultats de notre analyse des stopwords, lemmatisation et stemming, ainsi que la comparaison des modèles, nous guident dans la sélection des meilleures approches pour optimiser les performances de notre projet de classification de textes.

CONCLUSION

Rétrospective

Au cours des derniers mois, nous avons mené une étude de recherche approfondie sur la classification de texte en utilisant des méthodes de traitement automatique du langage naturel (NLP). Après avoir détaillé nos travaux dans les différents chapitres de ce rapport, il est essentiel de les comparer aux objectifs initialement fixés et d'analyser les résultats obtenus.

- Sélection des méthodes de classification de texte à comparer : Après une revue approfondie des méthodes existantes, nous avons choisi de nous concentrer sur deux approches principales : les Réseaux de Neurones Récurrents (RNN) et les Transformers.

- Collecte et préparation des données : Nous avons rassemblé un ensemble de données représentatives pour entraîner et évaluer nos modèles de classification de texte. Les données ont été soigneusement prétraitées pour garantir leur qualité pour une analyse précise.

- Entraînement et évaluation des modèles : Nous avons mis en œuvre des modèles RNN et des modèles Transformers, en utilisant des architectures populaires. Les modèles ont été entraînés sur les données préparées et évalués en utilisant des métriques de performance pertinentes.

- Comparaison des performances : Nos résultats ont montré que les modèles Transformers, en particulier certaines architectures spécifiques, ont généralement surpassé les RNN en termes de précision. Nous avons également observé que les modèles Transformers étaient plus stables et moins sujets au surapprentissage par rapport aux RNN.

- Conclusion et recommandations : En conclusion, notre étude comparative a démontré que les modèles Transformers étaient généralement plus performants dans

le domaine de la classification de texte par rapport aux RNN. Cependant, il est important de prendre en compte d'autres facteurs tels que le temps d'apprentissage et la stabilité du modèle lors du choix de l'approche la plus adaptée à un cas spécifique. Nous recommandons d'explorer davantage les modèles Transformers pour des tâches de classification de texte plus complexes.

Perspectives

En termes de perspectives, plusieurs directions peuvent être envisagées pour améliorer et étendre le travail réalisé dans le cadre de cette étude :

- Amélioration des modèles existants : Il serait intéressant de continuer à optimiser les modèles RNN et Transformers en ajustant les hyperparamètres, en utilisant différentes techniques de régularisation pour éviter le surapprentissage, ou en explorant d'autres architectures de modèles.

- Exploration de nouvelles méthodes : Au-delà des RNN et des Transformers, d'autres méthodes de NLP pourraient être explorées, comme les Convolutional Neural Networks (CNN) ou les modèles basés sur l'attention.

- Utilisation de données plus diversifiées : L'utilisation d'un ensemble de données plus large et plus diversifié pourrait aider à améliorer la robustesse et la généralisation des modèles.

- Application à d'autres tâches de NLP : Les méthodes utilisées dans cette étude pourraient être appliquées à d'autres tâches de NLP, comme la génération de texte, l'analyse de sentiments, la traduction automatique, etc.

- Déploiement en production : Une fois les modèles optimisés et validés, une étape importante serait leur déploiement en production pour permettre leur utilisation dans des applications réelles.

BIBLIOGRAPHIE

[1] Kai Sheng Tai, Richard Socher et Christopher D. Manning. « Improved semantic representations from tree structured long short-term memory networks ». In : *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*. 2015, p. 1556-1566. URL : http://dx.doi.org/10.3115/v1/p15-1.

[2] Xiao-Dan Zhu, Parinaz Sobhani et Hongyu Guo. « Long short-term memory over recursive structures ». In : *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. 2015, p. 1604-1612. URL : http://proceedings.mlr.press/v37/zhub15.html.

[3] Junyang Chen, Zhiguo Gong et Weiwen Liu. « A dirichlet process biterm-based mixture model for short text stream clustering ». In : *Applied Intelligence* 50.5 (2020), p. 1609-1619. URL : http://dx.doi.org/10.1007/s10489-019-01606-1.

[4] Junyang Chen, Zhiguo Gong et Weiwen Liu. « A nonparametric model for online topic discovery with word embeddings ». In : *Information Sciences* 504 (2019), p. 32-47. URL : http://dx.doi.org/10.1016/j.ins.2019.07.048.

[5] Qian Li et al. « A Survey on Text Classification : From Traditional to Deep Learning ». In : *ACM Transactions on Intelligent Systems and Technology* 13.2 (2022). URL : https://arxiv.org/abs/2008.00364.

[6] M. E. Maron. « Automatic indexing : An experimental inquiry ». In : *Journal of the ACM (JACM)* 8.3 (1961), p. 404-417. URL : http://dx.doi.org/10.1145/321075.321084.

[7] Thomas M. Cover et Peter E. Hart. « Nearest neighbor pattern classification ». In : *IEEE Transactions on Information Theory* 13.1 (1967), p. 21-27. URL : http://dx.doi.org/10.1109/TIT.1967.1053964.

[8] Thorsten Joachims. « Text categorization with support vector machines : Learning with many relevant features ». In : *Proceedings of the 10th European Conference on Machine Learning (ECML)*. 1998, p. 137-142. URL : http://dx.doi.org/10.1007/BFb0026683.

[9] Rami Aly, Steffen Remus et Chris Biemann. « Hierarchical multi-label classification of text with capsule networks ». In : *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 2 : Student Research Workshop*. 2019, p. 323-330. URL : http://dx.doi.org/10.18653/v1/p19-2045.

[10] Yin Zhang, Rong Jin et Zhi-Hua Zhou. « Understanding bag-of-words model : A statistical framework ». In : *International Journal of Machine Learning and Cybernetics* 1.1-4 (2010), p. 43-52.

[11] Ricardo Baeza-Yates et Berthier Ribeiro-Neto. *Modern Information Retrieval*. T. 463. ACM Press, 1999.

[12] Mucahid Mustafa Saritas et Ali Yasar. « Performance Analysis of ANN and Naive Bayes Classification Algorithm for Data Classification ». In : *International Journal of Intelligent Systems and Applications in Engineering* 7.2 (2019), p. 88-91. URL : https://ijisae.org/index.php/IJISAE/article/view/934.

[13] Gongde Guo et al. « KNN Model-Based Approach in Classification ». In : *On The Move to Meaningful Internet Systems 2003 : CoopIS, DOA, and ODBASE*. T. 2888. Lecture Notes in Computer Science. Springer, 2003, p. 986-996. URL : https://link.springer.com/chapter/10.1007/978-3-540-39964-3_62.

[14] Dakhaz Mustafa Abdullah et Adnan Mohsin Abdulazeez. « Machine Learning Applications based on SVM Classification : A Review ». In : *Quarterly Advances in Mathematics* 1.2 (2019). DOI : 10.48161/qaj.v1n2a50. URL : https://www.researchgate.net/publication/367214904_Machine_Learning_Applications_based_on_SVM_Classification_A_Review.

[15] MathWorks. *Convolutional Neural Network - MathWorks*. [Online ; accessed 05-June-2024]. 2024. URL : https://www.mathworks.com/discovery/convolutional-neural-network.html.

[16] Xinyu Lei, Hongguang Pan et Xiangdong Huang. « A Dilated CNN Model for Image Classification ». In : *IEEE Access* 7 (2019), p. 124087-124095. DOI : 10.1109/ACCESS.2019.2927169. URL : https://ieeexplore.ieee.org/document/8756165/references.

[17] Alpna Patel et Arvind Kumar Tiwari. « Sentiment Analysis by using Recurrent Neural Network ». In : *Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE)*. 2019, p. 4. URL : https://ssrn.com/abstract=3349572.

[18] Shaina Raza et Chen Ding. « Fake news detection based on news content and social contexts : a transformer-based approach ». In : *International Journal of Data Science and Analytics* 13 (2022), p. 335-362. DOI : 10.1007/s41060-021-00302-z. URL : https://link.springer.com/article/10.1007/s41060-021-00302-z.

[19] Audrey Tips. *Définition Marketing : qu'est-ce qu'un stopword ou mot vide*. 2021. URL : %5E1%5E.

[20] Divya Khyani et al. « An interpretation of lemmatization and stemming in natural language processing ». In : *Journal of University of Shanghai for Science and Technology* 22.10 (2021), p. 350-357.

[21] Mael Fabien. *NLP en Français - Mael Fabien*. URL : https://maelfabien.github.io/machinelearning/NLPfr/#3-tokenisation-par-phrases.

[22] Toms Bergmanis et Sharon Goldwater. « Context Sensitive Neural Lemmatization with Lematus ». In : *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana : Association for Computational Linguistics, 2018, p. 1391-1400. DOI : 10.18653/v1/N18-1126. URL : https://aclanthology.org/N18-1126.

[23] S. Vijayarani et R. Janani. « TEXT MINING : OPEN SOURCE TOKENIZATION TOOLS – AN ANALYSIS ». In : *Advanced Computational Intelligence : An International Journal* 3.1 (2016), p. 37-47. URL : https://www.researchgate.net/publication/301764643_TEXT_MINING_OPEN_SOURCE_TOKENIZATION_TOOLS_AN_ANALYSIS.

[24] Raphael Kassel. *Cross-Validation ou Validation croisée : définition et importance*. 2021.

Résumé

Ce travail se concentre sur l'application de deux approches différentes pour la classification de texte ; à savoir les Réseaux de Neurones Récurrents (RNN), en particulier les architectures LSTM, ainsi que les modèles à attention tels que BERT. L'objectif est d'évaluer la performance de ces approches dans divers scénarios de classification de texte, en tenant compte de plusieurs facteurs tels que la taille du jeu de données, la complexité de la tâche, etc. Pour cela, nous avons commencé par une revue des algorithmes d'apprentissage traditionnels, en explorant en détail, les RNN et les modèles Transformer. Par la suite, nous avons effectué une analyse approfondie des performances et des résultats obtenus, ainsi qu'une comparaison des performances des deux approches. A la fin, nous avons souligné l'importance de ces méthodes dans le domaine du traitement automatique du langage naturel où nous avons proposé des orientations et recommandations pour des travaux futurs visant à améliorer les performances des systèmes de classification de texte.

Abstract

This work focuses on the application of two different approaches for text classification ; namely Recurrent Neural Networks (RNN), in particular LSTM architectures, as well as attention-based models such as BERT. The aim is to evaluate the performance of these approaches in various text classification scenarios, taking into account several factors such as dataset size, task complexity, etc. To this end, we began with a review of traditional learning algorithms, exploring in detail, RNNs and Transformer models. Subsequently, we carried out an in-depth analysis of the performance and results obtained, as well as a comparison of the performance of the two approaches. At the end, we highlighted the importance of these methods in the field of automatic natural language processing, and proposed guidelines and recommendations for future work aimed at improving the performance of text classification systems.

ملخص

يركز هذا العمل على تطبيق نهجين مختلفين لتصنيف النصوص؛ وهما الشبكات العصبية المتكررة(RNN) ، ولا سيما خلاياLSTM، والنماذج القائمة على الانتباه مثل BERT .والهدف من ذلك هو تقييم أداء هذه الأساليب في سياقات مختلفة لتصنيف النصوص، مع مراعاة عدد من العوامل مثل حجم مجموعة البيانات. وللقيام بذلك، بدأنا بمراجعة خوارزميات التعلّم التقليدية، واستكشفنا بالتفصيل الشبكات العصبيةRNN . ثم أجرينا بعد ذلك تحليلًا متعمقًا للأداء والنتائج التي تم الحصول عليها، وقارنا أداء النهجين. في النهاية، سلطنا الضوء على أهمية هذه الأساليب في مجال المعالجة الآلية للغة الطبيعية، واقترحنا مبادئ توجيهية وتوصيات للعمل المستقبلي الذي يهدف إلى تحسين أداء أنظمة تصنيف النصوص.