UNIVERSITE SAAD DAHLEB DE BLIDA

Faculté des sciences

Département d'informatique

MEMOIRE DE MASTER

En Informatique

Option : Traitement Automatique de la Langue

THÈME :

Détection des discours haineux en langue arabe

Réalisé par

SIDI-MOUSSA Abdellah

BRAHIMI Ahmed

Encadré par

Mourad Abbas

KAMECHE Abdallah

. . Juillet 2023

Remerciements

Nous tenons à exprimer nos sincères remerciements à toutes les personnes qui ont contribué de près ou de loin à la réalisation de notre projet de fin d'études. Tout d'abord, nous souhaitons exprimer notre gratitude à Dieu le Tout-Puissant pour nous avoir accordé la santé et la persévérance nécessaires pour mener à bien ce travail.

Nous sommes également extrêmement reconnaissants envers notre encadrant, le Dr. Mourad Abbas, pour sa précieuse guidance, ses conseils éclairés et son soutien constant tout au long de notre projet. Son expertise et son engagement ont été d'une valeur inestimable pour notre réussite.

Nous tenons à remercier chaleureusement notre promoteur, M. Kammeche Hicham, pour son suivi attentif et son soutien indéfectible. Ses conseils pertinents et sa bienveillance ont grandement contribué à l'avancement de notre travail.

Nos sincères remerciements vont également aux membres du jury qui ont accepté d'évaluer notre travail et nous ont fait part de leurs précieuses remarques et suggestions pour son amélioration.

Enfin, nous n'oublions pas de mentionner tous les membres de notre famille, nos chers amis et camarades de promotion, ainsi que toutes les personnes qui, de près ou de loin, ont contribué à la réussite de notre projet. Votre soutien et votre confiance ont été d'une valeur inestimable pour nous.

Résumé

Les réseaux sociaux, tels que Twitter, sont devenus des espaces propices à la propagation du discours de haine, y compris en langue arabe. Dans cette étude, nous avons développé un système de détection du discours offensant et haineux sur Twitter . Pour ce faire, nous avons constitué un ensemble de données en collectant des tweets en arabe standard et dialectal apartir deux ensemble de donnes open source, en nous concentrant sur deux tâches : l'offensivité et la haine. Nous avons exploré plusieurs approches de représentation vectorielle, notamment TF-IDF, Word2Vec et Sentence Transformers (SBERT), et nous avons comparé divers modèles d'apprentissage automatique tels que les machines à vecteurs de support (SVM), la régression logistique et les arbres de décision. En parallèle, nous avons également utilisé les modèles d'apprentissage profond, notamment AraBERT avec réseau neuronal entièrement connecté (FFNN) et à un réseau neuronal convolutif à une dimension (CNN-1D). Les résultats ont montré que lorsque AraBERT était fine-tuné avec FFNN, il a obtenu des scores F1 de 88,18 % et 83,22 % pour les tâches "Offensive" et "Hate" respectivement. De même, l'utilisation d'AraBERT avec CNN1D a permis d'atteindre des scores F1 de 87,60 % et 84,83 % pour les mêmes tâches.

Mots clés :
Détection des discours haineux, Apprentissage automatique, Apprentissage profonde, la classification des textes, AraBERT, SBERT, CNN-1D

ملخص

Abstract

Social media platforms, such as Twitter, have become spaces conducive to the spread of hate speech, including in the Arabic language. In this study, we developed a system for detecting offensive and hateful speech on Twitter in Arabic. To do so, we collected a dataset of Arabic tweets from standard and dialectal Arabic using two open-source datasets, focusing on two tasks : offensiveness and hate. We explored several vector representations approaches, including TF-IDF, Word2Vec, and Sentence Transformers (SBERT), and compared various machine learning models, such as Support Vector Machines (SVM), Logistic Regression, and Decision Trees. We also used deep learning models, including AraBERT with a fully connected neural network (FFNN) and a one-dimensional convolutional neural network (CNN1D). The results showed that fine-tuning AraBERT with FFNN achieved F1 scores of 88.18% and 83.22% for the "Offensive" and "Hate" tasks, respectively. Similarly, using AraBERT with CNN1D achieved F1 scores of 87.60% and 84.83% for the same tasks

Keywords :
detection of hate speech, machine learning, deep learning, text classification, AraBERT, SBERT, CNN-1D

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

ANN Artificial Neural Network

BERT Bidirectional Encoder Representations from Transformers

CART Classification and Regression Tree

CBOW Continuous Bag Of Words

CNN Convolutional Neural Network

CNN-1D One-Dimensional Convolutional Neural Network

DL Deep Learning

DT Decision Tree

FCNN Fully Connected Neural Network

FFNN Feed-Forward Neural Network

GPT Generative Pre-trained Transformer

HS Hate Speech

LR Logestic Regression

ML Machine Learning

NLP Natural Language Processing

OFF Offensive

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

SBERT Sentence-BERT

Seq2Seq Sequence-to-sequence

SG Skip-Gram

SVM Support Vector Machine

TALN Traitement Automatique du Langage Naturel

TF-IDF Term Frequency-Inverse Document Frequency

Introduction Générale

Les réseaux sociaux occupent aujourd'hui une place prépondérante dans les moyens de communication modernes, étant le canal le plus utilisé sur Internet. Cependant, cette popularité a également conduit à l'émergence de comportements malveillants, voire interdits, où de nombreux utilisateurs profitent de l'anonymat offert par ces plateformes pour publier des commentaires offensants et haineux qui peuvent nuire aux autres. Le monde de l'informatique et les entreprises technologiques s'efforcent de lutter contre ce phénomène en mettant en place des mesures appropriées. Dans ce contexte, nous avons entrepris un travail axé sur la classification du discours de haine dans les commentaires sur Twitter, en se concentrant spécifiquement sur le contenu en langue arabe. Notre objectif principal est de développer un système capable de classer les commentaires selon deux catégories : offensants et haineux. Afin d'atteindre cet objectif, nous avons proposé une approche reposant sur différentes représentations vectorielles et différents algorithmes d'apprentissage automatique et profond. Nous visons à identifier les meilleurs modèles d'apprentissage automatique et profond, ainsi que les représentations vectorielles les plus performantes.

Organisation du mémoire

Pour réaliser ce travail, nous avons structuré notre étude de la manière suivante :

— Dans le premier chapitre, nous introduisons les notions générales liées au discours de haine, à la classification des textes, ainsi qu'à la manipulation des textes en langue arabe. Nous définissons également les algorithmes d'apprentissage automatique et profond, et discutons des travaux connexes.

— Le deuxième chapitre se concentre sur l'architecture de notre système de détection du discours de haine dans le contenu arabe. Nous présentons les méthodes utilisées pour la sélection des caractéristiques et les algorithmes d'apprentissage mis en œuvre.

— Dans le dernier chapitre, nous réalisons des expérimentations sur notre ensemble de données. Après avoir évalué et comparé différents algorithmes, nous mettons en œuvre les modèles choisis qui présente les meilleures performances. Nous présentons également les résultats obtenus avec leurs interprétations.

— Enfin, nous concluons notre travail avec une synthèse générale

L'objectif global de notre projet est de détecter le discours de haine dans le contenu textuel arabe, en particulier sur les réseaux sociaux, afin de fournir une solution efficace pour lutter contre ces comportements préjudiciables.

Chapitre 1

Discours de haine et sa détection automatique

1.1 Introduction

De nos jours, la propagation du discours de haine est devenue un problème incontrôlable en raison de sa présence en ligne, qui a des effets nuisibles sur les individus et les communautés et peut également conduire à des activités illégales. Dans ce chapitre, nous définirons le discours de haine et ses différentes catégories, ainsi que ses effets.

1.2 Définition et Analyse du discours de haine et concepts connexes

Comprendre et détecter le discours de haine est un défi complexe, tant pour les humains que pour les systèmes automatiques. Dans cette partie, nous explorons différentes définitions du discours de haine de sources, ainsi que les concepts connexes tels que le cyberharcèlement, le langage abusif et le langage offensant. Une compréhension précise de ces notions facilite la détection et la prévention du discours de haine.

1.2.1 Discours de haine

Les définitions officielles du discours de haine fournies par les Nations Unies, Facebook et Twitter offrent des perspectives essentielles pour comprendre la nature et les manifestations de ce phénomène.

— Nations Unies définissent le discours de haine comme : « tout type de communication, orale ou écrite, ou de comportement, constituant une atteinte ou utilisant un langage péjoratif ou discriminatoire à l'égard d'une personne ou d'un groupe en raison de leur identité, en d'autres termes, de l'appartenance religieuse, de l'origine ethnique, de la nationalité, de la race, de la couleur de peau, de l'ascendance, du genre ou d'autres facteurs constitutifs de l'identité » [UN].

— Facebook : "Nous n'autorisons pas les discours de haine sur Facebook et Instagram. Nous définissons le discours de haine comme un discours violent ou déshumanisant, des déclarations d'infériorité, des appels à l'exclusion ou à la ségrégation basés sur des caractéristiques protégées ou des insultes. Ces caractéristiques comprennent la race, l'origine ethnique, l'origine nationale, l'appartenance religieuse, la caste, le sexe, le genre, l'identité de genre et un handicap ou une maladie grave."[Fb]

— Twitter : " Vous ne pouvez pas attaquer directement d'autres personnes sur la base de la race, de l'ethnie, de l'origine nationale, de la caste, de l'orientation sexuelle, du sexe, de l'identité de genre, de l'appartenance religieuse, de l'âge, du handicap ou d'une maladie grave."[Tw]

Plusieurs travaux antérieurs ont présenté des branches significatives du discours de haine. Ici, nous discuterons de certaines des catégories essentielles qui sont considérées comme pertinentes dans la plupart des études sur le discours de haine.

1.2.2 Cyberharcèlement

Les chercheurs de [10] Ont défini la forme électronique du harcèlement traditionnel, également appelé cyberharcèlement, comme l'agression et le harcèlement ciblés sur une personne qui est incapable de se défendre. Selon [11] Le harcèlement est connu pour son acte répétitif envers la même personne, contrairement au discours de haine qui est plus général et n'est pas nécessairement destiné à blesser une personne spécifique.

1.2.3 Le langage abusif

Dans la vie quotidienne, surtout sur les réseaux sociaux, la propagation de discours de haine est souvent accompagnée d'un langage abusif [12]. Le langage abusif est une expression qui contient des mots ou des phrases abusifs qui sont transmis à l'interlocuteur (personnes ou groupes), à la fois verbalement et par écrit.[13] Et [14] caractérise le vocabulaire abusif comme suit : "tout langage fortement impoli, grossier ou blessant utilisant des jurons, qui peut montrer une dégradation de quelqu'un ou de quelque chose, ou montrer une émotion intense".

1.2.4 Le langage offensant

Le langage offensant est un langage profane, grossièrement impoli, ou vulgaire, exprimé sous la forme de mots agressifs ou blessants, destiné à insulter l'individu ou le groupe ciblé.[15] Le langage offensant est "tout mot ou chaîne de mots qui a ou peut avoir un impact négatif sur le sens de soi et / ou le bien-être de ceux qui le rencontrent - c'est-à-dire, cela les fait ou peut les faire se sentir, légèrement ou extrêmement, mal à l'aise et / ou insulté et / ou blessé et / ou effrayé" [16]. Ainsi, nous pouvons définir le discours de haine comme toute langue offensante, abusive ou insultante contre une personne ou un groupe [17].

1.3 Les effets des discours de haine

Le discours de haine a un impact significatif sur la société et les individus, et il est considéré comme une infraction dans de nombreux pays à travers le monde. Ce type de discours a la capacité de se propager rapidement et d'inciter les individus enclins à la haine, ce qui peut entraîner des actes de violence et des crimes graves dans la vie réelle. Par exemple, le génocide des Tutsis au Rwanda en 1994 a été l'un des génocides les plus terribles causés par la propagation de discours de haine. Dans cette tragédie, certains groupes ont propagé un discours de haine en affirmant que l'ethnie Tutsie était responsable des pressions croissantes en politique, en économie et en société. [Rw] Des événements plus récents, tels que le génocide de la communauté Rohingya au Myanmar, la violence antimusulmane au Sri Lanka et la fusillade de Pittsburgh, ont également été attribués à la propagation de discours de haine [18]. En plus des conséquences sociales et physiques, le discours de haine a également des effets psychologiques néfastes, tels que la dépression, l'anxiété, la baisse de l'estime de soi et, dans certains cas, le suicide. En conséquence, cela est devenu un domaine de recherche et d'étude pour prévenir ces pratiques.

1.4 La détection automatique de discours de haine

La détection de discours haineux dans le texte est abordée dans toute la littérature comme un problème de classification d'apprentissage automatique. La branche de l'apprentissage automatique ou Machine Learning en anglais (ML) se divise en plusieurs sous-catégories de problèmes. Si l'on considère les données, elles peuvent être divisées en deux catégories principales : l'apprentissage supervisé et l'apprentissage non supervisé, notre attention se portant sur le premier. L'apprentissage supervisé est caractérisé par l'approximation d'une fonction à l'aide de données étiquetées, tandis que l'objectif de l'apprentissage non supervisé est de modéliser des données non étiquetées. Dans l'apprentissage supervisé, on peut également considérer deux catégories différentes :

— La classification, dont le but est d'identifier la catégorie à laquelle appartient l'instance.

— La régression, où l'on cherche à prédire une quantité continue.

Une approche courante pour la classification du discours de haine consiste à différencier les instances de texte haineux de celles qui ne le sont pas. Cependant, le nombre de classes considérées peut varier selon les différentes approches, comme cela est illustré dans la section des travaux connexes.

1.5 La classification des textes

La classification de texte est une tâche en Traitement Automatique du Langage Naturel (TALN), également connu sous le nom de Natural Language Processing en anglais (NLP). Qui consiste à attribuer automatiquement des classe (aussi appelée étiquettes ou labels) prédéfinies à des documents texte en fonction de leur contenu. La classification de texte utilise des algorithmes d'apprentissage automatique et des techniques de traitement du langage naturel pour analyser les données textuelles et extraire des caractéristiques pertinentes qui peuvent être utilisées pour classer le texte [14].

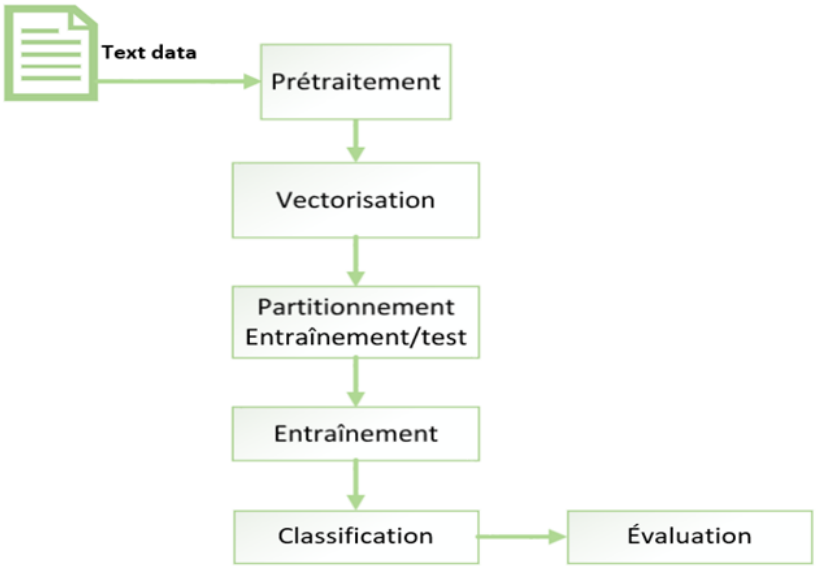

FIGURE 1.1 – Processus de classification de texte

La classification automatique de texte passe par un certain nombre d'étapes . On peut résumer ce processus en trois grandes étapes :

1. **Préparation des données** : cette étape implique la collecte et la préparation de données. Les données doivent être collectées à partir de sources diverses et représentatives, suivies d'un prétraitement pour nettoyer le texte, le normaliser et supprimer les caractères inutiles.
2. **Entraînement du modèle** : cette étape implique l'entraînement d'un modèle de classification de texte à partir des données préparées. Les techniques NLP sont utilisées pour extraire des caractéristiques du texte, puis former des modèles de classification à l'aide d'algorithmes d'apprentissage automatique.

3. **Évaluation du modèle** : cette étape consiste à évaluer l'efficacité du modèle de classification de texte à l'aide de mesures telles que la précision, le rappel, le score F1. Cette étape identifie les améliorations nécessaires pour améliorer la qualité du modèle.

Chacune de ces étapes est essentielle pour garantir la qualité et l'efficacité des modèles de classification de discours de haine. La figure suivante illustre le processus de classification de texte.

1.5.1 Le prétraitement des données

Est essentiel pour le traitement du langage naturel, car il permet d'identifier et de générer des données propres et pures pour l'évaluation lors de l'analyse. Ce processus comprend un ensemble de techniques qui permettent aux techniques de traitement du langage naturel de traiter et d'analyser les données pour les convertir en code compréhensible par la machine. Parmis Les techniques de prétraitement des données :

1. **Tokenization** : c'est le premier processus dans lequel la phrase est divisée en unités plus petites (appelées jetons).
2. **Nettoyage** : supprimer les caractères et les mots inutiles tels que les symboles, la ponctuation, les hashtags et les URL de texte si nécessaire.
3. **Mots vides** : sont supprimés car ils ne portent pas d'informations pertinentes dans la plupart des contextes.
4. **Minuscules** : ils sont supprimés car les techniques de NLP ne font pas la distinction entre les lettres minuscules et majuscules.
5. **Lemmatisation** : retourner les verbes à leurs racines.

Les traitements spécifiques propre à notre cas seront détaillés dans la partie suivante.

1.5.2 Vectorisation

Le fonctionnement des ordinateurs diffère de celui du cerveau humain, tout comme les algorithmes d'apprentissage automatique. Pour un algorithme, un texte n'est qu'une série de caractères binaires dénuée de signification. Afin de donner du sens à un texte du point de vue de l'algorithme d'apprentissage, il est nécessaire de le transformer en vecteur [19]. Cette étape est cruciale dans les algorithmes de classification de texte et dans les projets de traitement du langage naturel. Plusieurs méthodes existent pour effectuer cette transformation.

1.5.2.1 Terme Fréquence-Inverse Document Fréquence (TF-IDF)

Pour Term Frequency-Inverse Document Frequency (TF-IDF) [20]. Ce calcul évalue l'importance d'un terme contenu dans un document, relativement à une collection. Si un mot rare est

particulièrement présent dans un document, son TF-IDF sera très élevé. La pondération TFIDF est calculée avec la formule suivante :

$$\text{TF}(t,d) = \frac{\text{Nombre d'apparitions du terme } t \text{ dans le document } d}{\text{Nombre total de termes dans le document}}$$

$$\text{IDF}(t) = \log\left(\frac{\text{Nombre de documents dans le corpus}}{\text{Nombre de documents où } t \text{ apparaît}}\right)$$

D'où

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \times \text{IDF}(t)$$

donne la pertinence de chaque document vis-à-vis d'un mot.

1.5.2.2 Word2vec

Word2vec [21] est un réseau de neurones à deux couches formées de manière supervisée pour générer des représentations distributionnelles de mots. Il prend en entrée un grand nombre de mots et génère un espace vectoriel de faible dimension (entre 50 et 300). Les vecteurs de mots sont positionnés dans l'espace vectoriel de sorte que les mots qui partagent un contexte commun dans le corpus soient côte à côte dans l'espace vectoriel. Word2vec utilise deux types d'architectures : Les modèles de sacs de mots continus ou Continuous Bag Of Words (CBOW) sont conçus pour apprendre les incorporations en formant un réseau qui prédit les mots à partir du contexte. Le contexte peut être formé, par exemple, par cinq mots à droite et cinq mots à gauche du mot à prédire. Le modèle Skip-Gram (SG) a une architecture symétrique et est conçu pour apprendre les imbrications en formant un réseau qui prédit les mots de contexte en fonction d'un mot d'entrée.

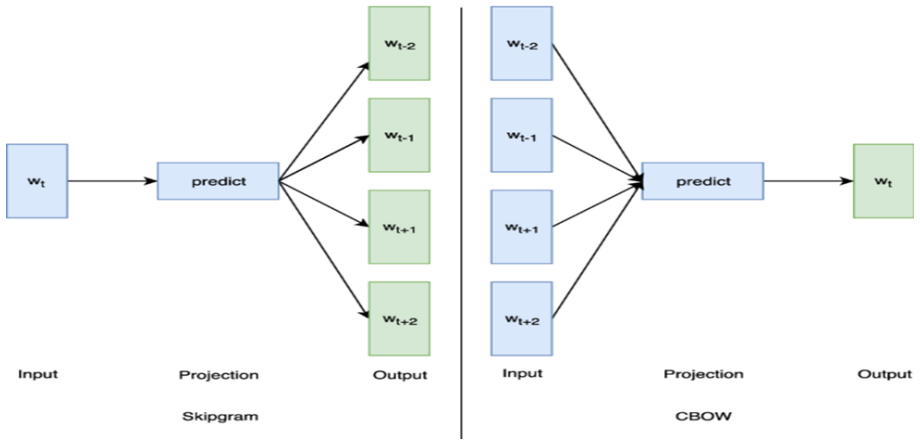

FIGURE 1.2 – L'architecture de CBOW et SG [1]

1.5.2.3 Bag of words

Un sac de mots ou bag of words [22] est une représentation d'un texte qui décrit les occurrences des mots dans un document. Le bag-of-words est tout simplement un tableau. Le

nombre de colonnes correspond au nombre de mots présents dans l'ensemble des textes du corpus, chaque ligne correspond à un document. A l'intersection d'une colonne et d'une ligne, nous savons si un mot est présent ou non dans un document. Ce paramètre peut être le nombre d'occurrences du mot dans le document, ou bien une pondération (**??**). Cette méthode est appelée bag-of-words car les mots sont conservés sans aucune structure, sans aucune dépendance les uns avec les autres. Le modèle permet juste de savoir si un mot est présent ou non et à quelle fréquence. Cette méthode est utilisée pour l'extraction de sujets, l'analyse de sentiments, mais est inefficace pour la traduction ou le résumé de textes où la place des mots dans la phrase a son importance.

	about	bird	heard	is	the	word	you
About the bird, the bird, bird bird bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0

FIGURE 1.3 – Exemple Sac de mots

1.5.2.4 Les embeddings contextuels

Les embeddings contextuels ont été développés pour surmonter les limitations des embeddings traditionnels, mentionnées précédemment.
Contrairement aux représentations vectorielles fixes des mots, les embeddings contextuels sont calculés de manière dynamique en fonction des mots qui les entourent, prenant en compte la structure des phrases et le fait que le sens d'un mot est fortement influencé par son contexte. Par exemple, dans les phrases "la souris grise" et "la souris d'ordinateur", le mot "souris" a des significations différentes en raison des mots qui l'accompagnent.
Ces Embeddings sont générés par des architectures complexes comme les Transformers. Ces modèles sont entraînés sur de vastes corpus de texte afin de capturer les relations entre les mots et d'obtenir des représentations plus riches et contextuellement adaptées. Dans cette partie, nous avons donné un aperçu général de divers algorithmes d'embedding de mots. Résumons-les ci-dessous :

En conclusion, cette partie a mis en évidence les étapes et les méthodes clés pour la préparation et la vectorisation des données textuelles. La prochaine partie se concentrera sur les différents algorithmes d'apprentissage utilisés pour la classification des textes, en explorant leur fonctionnement et leurs applications dans ce domaine.

Technique	Caractéristiques principales	Cas d'utilisation
TF-IDF	Méthode statistique pour capturer la pertinence des mots par rapport au corpus de texte. Il ne capture pas les associations de mots sémantiques.	Meilleur pour la recherche d'informations et l'extraction de mots-clés dans les documents.
Word2Vec	Architectures CBOW et Skip-gram basées sur des réseaux neuronaux, plus efficaces pour capturer des informations sémantiques.	Utile dans la tâche d'analyse sémantique.
BERT	Mécanisme d'attention basé sur les transformateurs pour capturer des informations contextuelles de haute qualité.	Traduction de langue, système de questions-réponses. Déployé dans le moteur de recherche Google pour comprendre les requêtes de recherche.

TABLE 1.1 – Techniques de traitement du langage naturel

1.6 Les approches utilisées pour la classification des textes

La classification des textes fait appel à différentes approches pour analyser et catégoriser automatiquement les données textuelles. Dans cette section, nous allons explorer les principales techniques utilisées, en mettant l'accent sur les algorithmes d'apprentissage automatique, les algorithmes d'apprentissage en profondeur et les transformers.

1.6.1 Les algorithmes de l'apprentissage automatique

L'apprentissage automatique (ML) offre diverses techniques pour la classification des textes. Ces méthodes consistent à entraîner un modèle sur un ensemble de données textuelles annotées, afin de lui permettre de généraliser et de classer de nouveaux textes. Parmi les algorithmes couramment utilisés, on retrouve :

1.6.1.1 Machine à vecteurs de support

Support Vector Machine (SVM) est l'un des algorithmes d'apprentissage supervisé les plus populaires, qui est utilisé pour les problèmes de classification et de régression. Cependant, il est principalement utilisé pour les problèmes de classification dans l'apprentissage automatique [23]. L'objectif de l'algorithme SVM est de trouver la meilleure limite de décision ou l'hyperplan qui peut séparer les données d'entraînement en différentes classes afin de pouvoir classer les

nouvelles données correctement dans le futur. Les points extrêmes qui sont utilisés pour construire cet hyperplan sont appelés vecteurs de support. Il existe deux types de SVM :

SVM linéaire : Lorsque les données d'entraînement peuvent être séparées linéairement, c'est-à-dire qu'elles peuvent être divisées en deux classes en utilisant une seule ligne droite, le SVM linéaire est utilisé pour construire la limite de décision. **SVM non linéaire** : Lorsque les données ne peuvent pas être séparées linéairement, un SVM non linéaire est utilisé pour construire la limite de décision. Il utilise des fonctions noyau pour transformer les données dans un espace de dimension supérieure où elles peuvent être séparées linéairement.

Dans le schéma qui suit, on détermine un hyperplan qui sépare les deux ensembles de points. Les points les plus proches, qui seuls sont utilisés pour la détermination d'hyperplan, sont appelés vecteurs de support (Figure1.4).

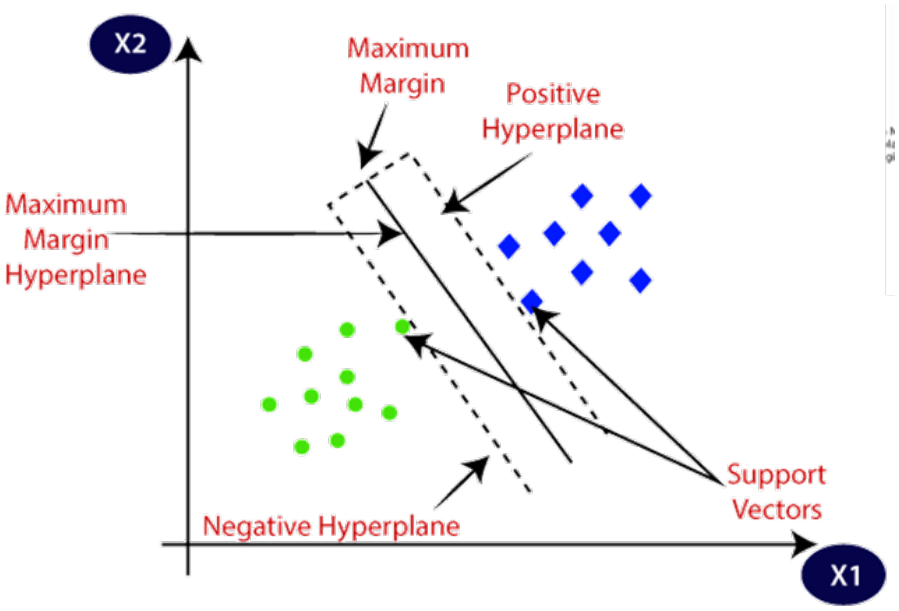

FIGURE 1.4 – SVM[2]

Kernel Trick est largement utilisé dans le modèle de machines à vecteurs de support (SVM) pour relier la linéarité et la non-linéarité [23]. Il convertit l'espace de dimension inférieure non linéaire en un espace de dimension supérieure, ce qui nous permet d'obtenir une classification linéaire. Nous projetons donc les données avec des fonctionnalités supplémentaires afin qu'elles puissent être converties dans un espace de dimension supérieure (figure1.5).

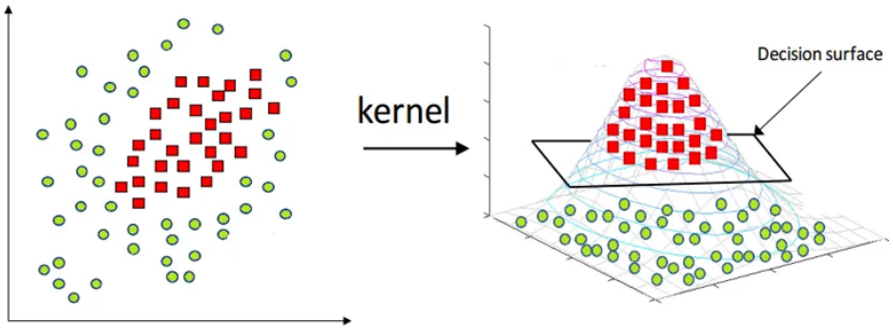

FIGURE 1.5 – Kernel trick[3]

1.6.1.2 Algorithme de classification d'arbre de décision

L'arbre de décision ou Decision Tree (DT) est une méthode d'apprentissage supervisé utilisée principalement pour les problèmes de classification. Il représente graphiquement une série de décisions basées sur les caractéristiques des données, afin de prédire la classe ou la valeur cible d'un nouvel exemple. L'arbre de décision se compose de nœuds, tels que le nœud racine, les nœuds de décision et les nœuds feuilles. Le nœud racine est le point de départ de l'arbre, les nœuds de décision sont utilisés pour prendre des décisions basées sur les caractéristiques, et les nœuds feuilles représentent les résultats ou les prédictions.
Les décisions sont prises en fonction des caractéristiques des données. À chaque nœud de décision, une caractéristique est évaluée et une branche est suivie en fonction de la valeur de cette caractéristique. Ce processus est répété jusqu'à atteindre un nœud feuille, où la prédiction finale est faite.
La construction de l'arbre de décision se fait à l'aide d'algorithmes tels que CART (Classification and Regression Tree) [24]. Ces algorithmes évaluent les caractéristiques des données et sélectionnent la meilleure caractéristique pour diviser l'arbre de manière à maximiser la pureté des sous-ensembles résultants.

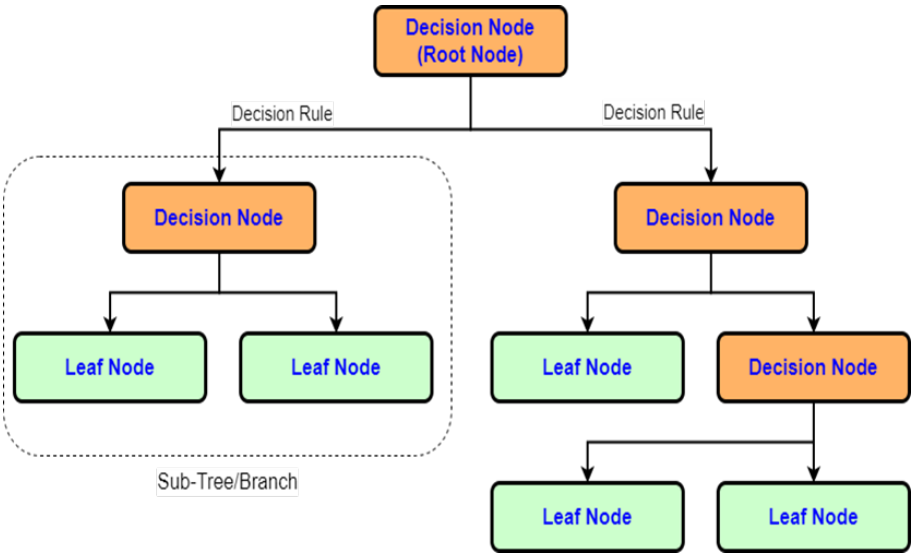

FIGURE 1.6 – Arbre de décision [4]

1.6.1.3 La régression logistique

La régression logistique (LR) Est un algorithme populaire de l'apprentissage supervisé utilisé pour prédire une variable dépendante catégorielle en utilisant un ensemble de variables indépendantes [5]. Contrairement à la régression linéaire, qui est utilisée pour résoudre des problèmes de régression, la régression logistique est utilisée pour résoudre des problèmes de classification. Elle ajuste une fonction logistique en forme de "S" pour prédire des valeurs maximales de 0 ou 1. La régression logistique peut fournir des probabilités et classer de nouvelles données à l'aide d'ensembles de données continus et discrets, et peut également déterminer les variables les plus efficaces pour la classification.
La fonction "S" fait référence à la fonction sigmoïde, qui est utilisée dans la régression logistique. Est définie comme suit :

$$S(x) = \frac{1}{1 + e^{-x}}$$

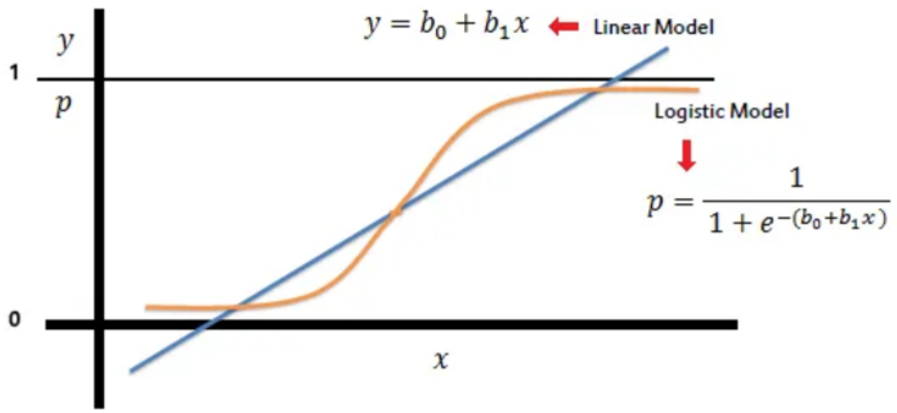

FIGURE 1.7 – Logistic regression [5]

1.6.2 L'Apprentissage profond - Deep learning

Le Deep Learning (DL) est une branche de l'apprentissage automatique qui utilise des réseaux neuronaux artificiels, inspirés de la structure et de la fonction du cerveau humain, pour résoudre des problèmes complexes. Cette méthode a suscité un grand intérêt dans le domaine de l'intelligence artificielle. Contrairement aux réseaux neuronaux traditionnels, le Deep Learning utilise plusieurs couches cachées pour apprendre des expériences ou des données enregistrées, comme le ferait un être humain. Les réseaux neuronaux qui ne contiennent pas de couches cachées ne sont pas considérés comme faisant partie du domaine du Deep Learning [25].

1.6.2.1 Le neurone artificiel

Le modèle mathématique du neurone artificiel est inspiré par le fonctionnement des neurones biologiques. C'est un opérateur qui reçoit des entrées de l'environnement externe ou d'autres neurones, chacune de ces entrées étant pondérée par un poids synaptique. Le neurone artificiel fournit une sortie uniquement si la somme pondérée des entrées dépasse un certain seuil interne. Lorsqu'on interconnecte ces neurones, on obtient un réseau de neurones artificiels [26].

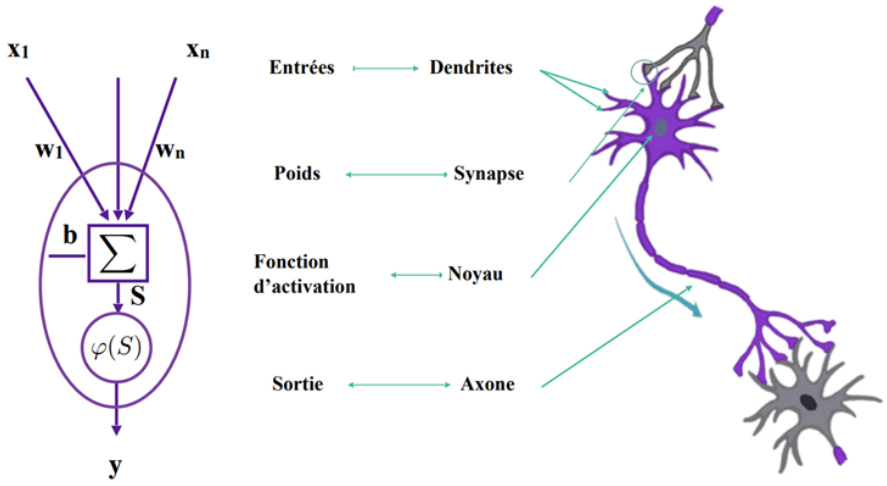

FIGURE 1.8 – Comparaison entre un neurone biologique et un neurone formel

Le principe général du neurone artificiel est de calculer une somme pondérée des entrées, qui peuvent provenir des sorties d'autres neurones dans un réseau. Chaque entrée est notée $x_1, \ldots, x_n$ et est associée à un poids $w_i \in \mathbb{R}$, en plus de la valeur de biais notée x_0 associée au poids w_0.

1.6.2.2 Les réseaux neuronaux artificiels

Les réseaux neuronaux artificiels sont des modèles de calcul qui consistent en plusieurs couches de neurones interconnectées. Ces réseaux sont utilisés pour la classification et la prédiction des données d'entrée. Ils sont composés d'une couche d'entrée, une ou plusieurs

couches cachées, et une couche de sortie. Chaque couche contient des nœuds, et chaque nœud est associé à un poids qui est utilisé lors du traitement de l'information d'une couche à l'autre. Chaque nœud est une unité qui est généralement exprimée par une fonction d'activation, telle que la fonction sigmoïde [27].

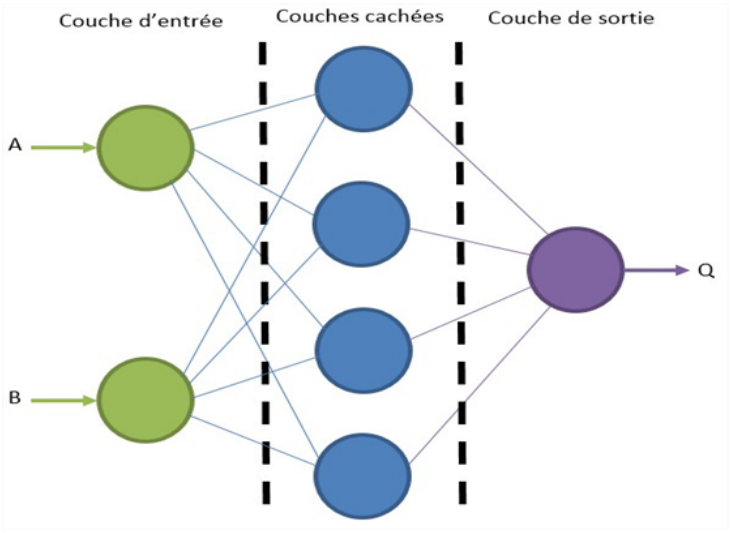

FIGURE 1.9 – Réseaux de neurone [[6]]

Un réseau de neurones f est une succession de transformations mathématiques permettant de passer de l'espace des features a l'espace des prédictions :

$$\begin{array}{rccc} f: & \mathbb{R}^{n_x} & \to & \mathbb{R}^{n_y} \\ & x & \mapsto & \hat{y} \end{array}$$

Le calcul de la sortie (y) à partir de l'entrée x s'appelle la phase forward du réseau. Un réseau de neurones très simple peut par exemple consister en une simple transformation linéaire, on aura alors $(y) = f(x) = Wx$ avec W une matrice de taille $n_y \times n_x$. Cette transformation est la couche de base des réseaux de neurones classiques. On utilise d'ailleurs souvent une transformation affine $f(x) = Wx + b$.

1.6.2.3 Les fonctions d'activation

Généralement, on fait suivre chaque transformation linéaire d'une fonction d'activation qui est une fonction mathématique non-linéaire appliquées aux sorties des neurones dans les réseaux de neurones artificiels. Elles sont utilisées pour introduire une non-linéarité dans les sorties des neurones, ce qui permet au réseau de traiter des données non linéaires ou complexes. Les fonctions d'activation peuvent également aider à régulariser le réseau et à éviter la surcharge en limitant l'amplitude des valeurs de sortie des neurones [28].
Il existe plusieurs types de fonctions d'activation couramment utilisées dans les réseaux de neurones, notamment :

1. **La fonction sigmoïde** : elle transforme les valeurs d'entrée en une sortie comprise entre 0 et 1. Elle était largement utilisée dans les premiers réseaux de neurones, mais elle est maintenant moins courante car elle peut présenter des problèmes de convergence.
2. **La fonction ReLU (Rectified Linear Unit)** : elle renvoie la valeur d'entrée si elle est positive, et zéro sinon. Elle est très populaire car elle est simple à calculer et donne de bons résultats en pratique.
3. **La fonction Tanh tangente hyperbolique** : elle est similaire à la fonction sigmoïde, mais elle renvoie des valeurs comprises entre -1 et 1. Elle est moins utilisée que la fonction ReLU car elle peut présenter des problèmes de convergence.
4. **La fonction softmax** : elle est couramment utilisée dans les réseaux de neurones pour la classification. Elle renvoie une distribution de probabilité sur les différentes classes, ce qui permet de déterminer la classe la plus probable pour une entrée donnée.

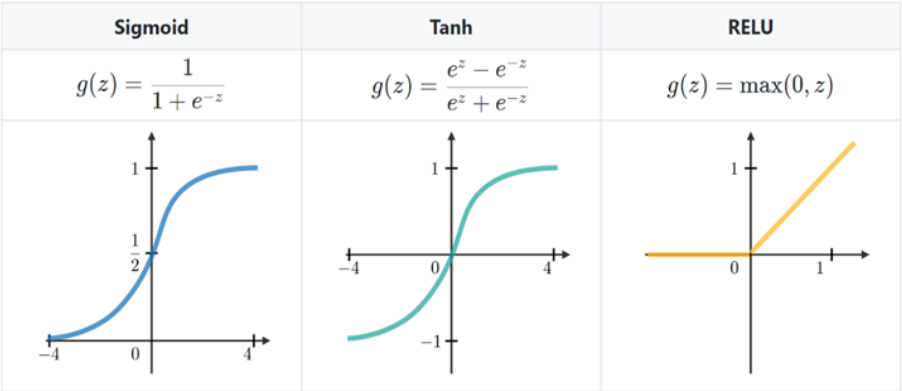

FIGURE 1.10 – Fonctions d'activation

Le choix de la fonction d'activation dépend du type de réseau de neurones et du problème que l'on cherche à résoudre. Le rôle des fonctions d'activation est crucial car elles permettent aux réseaux de neurones d'apprendre à partir des données et de produire des sorties précises et cohérentes[28].

1.6.2.4 Fonction de coût (loss)

Grâce à la phase forward, un réseau de neurones produit pour l'entrée $x^{(i)}$ une sortie $\hat{y}^{(i)}$. Pour mesurer à quel point cette sortie diffère de la cible $y^{(i)}$, on utilise une fonction de coût (loss). La plupart du temps, la fonction de coût globale $\mathcal{L}(X, Y)$ est la moyenne d'une fonction de coût unitaire $l(y^{(i)}, \hat{y}^{(i)})$ entre les prédictions et les cibles. [28].

$$\mathcal{L}(X, Y) = \frac{1}{N} \sum_{i=1}^{N} l(y^{(i)}, f(x^{(i)}))$$

1.6.2.5 L'algorithme de descente de gradient

La fonction de coût permet de mesurer l'erreur de réseau de neurones, et pour minimiser cette erreur sur l'ensemble des exemples d'apprentissage, nous devons apprendre ses paramètres en utilisant l'algorithme de descente de gradient. Cet algorithme consiste à calculer la dérivée de la fonction de coût par rapport à un paramètre w, et à faire un pas dans l'opposé de la direction du gradient [28].

$$W = W - \alpha \frac{\partial \mathcal{L}(X,Y)}{\partial W}$$

Le taux d'apprentissage (learning rate) est utilisé dans l'algorithme de descente de gradient pour contrôler la vitesse de l'optimisation.

1.6.3 Principales architectures de réseaux de neurones utilisées en apprentissage profond

Dans le cadre de notre étude, nous allons explorer différentes architectures de réseaux de neurones qui sont largement utilisées en apprentissage profond. Ces architectures comprennent les réseaux de neurones entièrement connectés, les réseaux de neurones convolutifs (CNN) et les réseaux de neurones récurrents (RNN).

1.6.3.1 Fully Connected Neural Network

Également appelé réseau de neurones entièrement connecté (FCNN) est un type de réseau de neurones artificiels dans lequel chaque neurone d'une couche est connecté à tous les neurones de la couche suivante. Cela signifie que toutes les sorties d'une couche sont des entrées pour toutes les unités de la couche suivante. Dans un réseau de neurones entièrement connecté, les neurones sont organisés en couches, avec une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie (Figure 1.11). Chaque neurone de la couche cachée reçoit des signaux d'entrée de tous les neurones de la couche précédente et calcule une sortie en fonction de sa fonction d'activation et des poids de connexion correspondants [29].

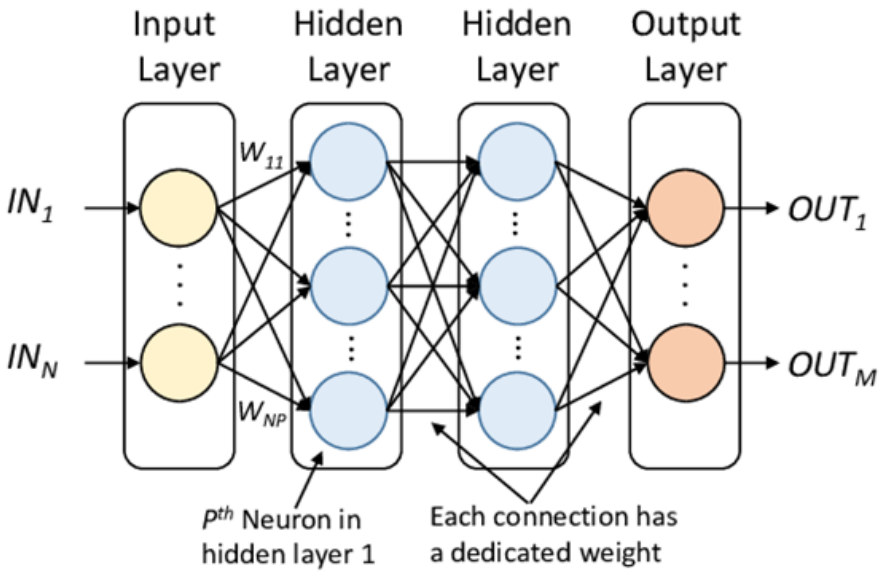

FIGURE 1.11 – Exemple de FCNN [7]

1.6.3.2 Réseau de neurones convolutifs (CNN)

Un réseau de neurones convolutifs (CNN) est un type de réseau de neurones artificiels spécialement conçu pour le traitement d'images. Il est inspiré par le fonctionnement du cortex visuel dans le cerveau humain et utilise des filtres de convolution pour extraire des caractéristiques importantes de l'image. Dans un CNN, les couches de convolution sont utilisées pour extraire des caractéristiques de l'image d'entrée, telles que des bords, des coins, des textures et des formes. Ces caractéristiques sont ensuite traitées par des couches de pooling pour réduire la dimension de l'image et éliminer le bruit. Enfin, les caractéristiques restantes sont acheminées vers une ou plusieurs couches de neurones entièrement connectées pour la classification ou la régression [30].

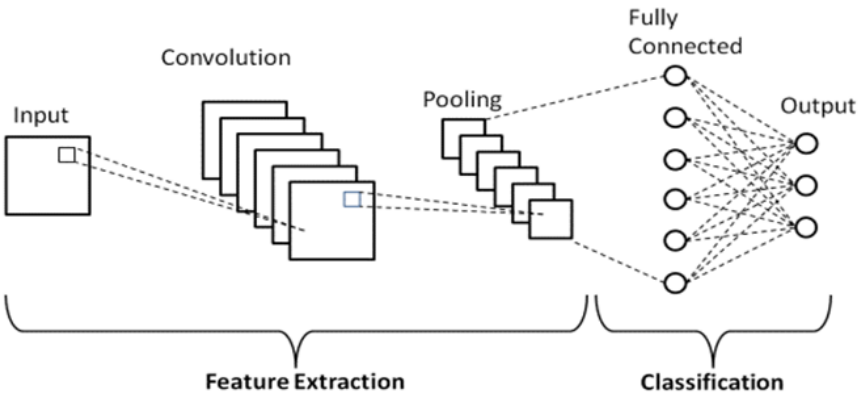

FIGURE 1.12 – CNN architecture

Chacune de ces couches joue un rôle clé dans la représentation et l'analyse des données. Examinons maintenant ces couches en détail.

a) La couche convolutive C'est la couche la plus importante convolution. Une couche de convolution est caractérisée par :

— Les dimensions des noyaux de convolution, généralement une convolution à une dimension égale à 2 avec des noyaux carrés.

— Le nombre des filtres de convolution C, c'est le nombre de cartes d'activations, ou cartes de caractéristiques, en sortie de la couche. Ces cartes sont représentées sous la forme de tenseurs de dimension 3 (H, W, C) avec H la hauteur des cartes, W la largeur et C le nombre de canaux.

— Le pas de convolution ou stride(s). C'est le pas de décalage du noyau de convolution à chaque calcul.

— Le padding p. C'est le paramètre permettant de dépasser la taille de l'image pour appliquer la convolution en ajoutant des pixels autour de l'image.

b) Couche d'échantillonnage (Pooling) Semblable à la couche de convolution, la couche d'échantillonnage est chargée de réduire la taille spatiale des cartes de caractéristiques, mais elle conserve les informations les plus importantes. Il existe différents types d'échantillonnage dont l'échantillonnage maximum-ou Max Pooling-, l'échantillonnage moyen ou Average Pooling.
c) Couche complètement connectée (fully connected) : La couche complètement connectée est un Perceptron multicouches traditionnel qui utilise une fonction d'activation (par exemple softmax) sur le vecteur de sortie afin d'ajouter la non-linéarité.
Il est également courant d'utiliser des couches de normalisation, des couches de régularisation et des couches de dropout pour améliorer les performances et éviter le surapprentissage.

1.6.3.3 Les réseaux de neurones récurrents RNN

Les réseaux de neurones récurrents (RNN), sont un type de réseau de neurones largement utilisé dans le domaine de l'apprentissage en profondeur [31]. Les RNN utilisent les sorties précédentes comme entrées supplémentaires et sont parfaitement adaptés au traitement de données séquentielles. Généralement, elles se présentent sous la forme suivante 1.13

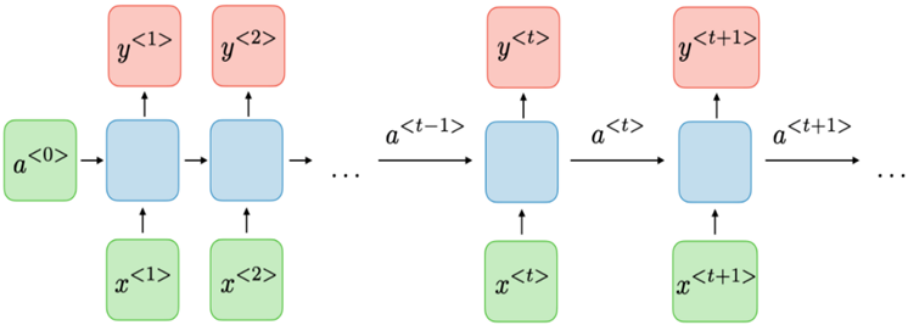

FIGURE 1.13 – Principe des RNN [38]

À chaque instant t, le passage vers l'avant (forward pass) est modélisé par les équations suivantes

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{1}$$

Et

$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y) \tag{2}$$

Où $x^{<t>}$ et $x^{<t>}$ sont respectivement le vecteur d'entrée du réseau et le vecteur d'activation à l'instant t, g_1, g_2 sont des fonctions d'activation, les W et b sont respectivement les poids et les biais à apprendre durant l'entrainement du réseau. La valeur de sortie à l'instant t $y^{<t>}$ est calculée par l'équation (2) en fonction de la valeur d'activation $a^{<t>}$ calculée par l'équation (1). Nous constatons clairement l'aspect récurrent dans ces calculs (le calcul à l'instant t est à base de l'information apportée de l'instant t-1, elle-même calculée à partir de l'information apportée de t-2 etc.), contrairement à un réseau de neurones classique ANN (Artificial Neural Network) où la sortie dépend uniquement des valeurs d'entrées.

1.6.4 Les Transformers

1.6.4.1 Définition de transformer

Le transformer Est un modèle de Deep Learning (donc un réseau de neurones) de type Sequence-to-sequence (Seq2Seq) qui a la particularité de n'utiliser que Les transformers sont une classe de modèles de réseaux de neurones artificiels utilisés pour le traitement du langage naturel (NLP) et d'autres tâches liées au traitement de séquences. Les transformers ont été introduits en 2017 par Vaswani et al. Dans leur article "Attention Is All You Need" [8]. Contrairement aux réseaux de neurones récurrents (RNN), les transformers n'utilisent pas de boucles récurrentes pour traiter les séquences, mais se basent sur un mécanisme d'attention pour accorder une pondération différente à chaque élément de la séquence en fonction de son importance pour la tâche en cours. Cette architecture permet aux transformers de traiter efficacement des séquences de longueurs variables et d'atteindre des performances de pointe dans de nombreuses tâches de

NLP, telles que la traduction automatique, la génération de texte, la compréhension de la langue naturelle, etc.

1.6.4.2 L'architecture du Transformer

L'architecture du Transformer est composée de deux parties principales 1.14 :

— **L'encodeur** est la partie du Transformer qui encode la séquence d'entrée en une représentation de haute qualité qui peut être utilisée pour résoudre différentes tâches de NLP. L'encodeur est constitué de plusieurs couches identiques. Chaque couche contient deux modules principaux : la couche de transformation de l'attention et la couche feed-forward. Dans la couche de transformation de l'attention, l'encodeur utilise un mécanisme d'attention multi-têtes pour accorder différentes pondérations à chaque token de la séquence d'entrée en fonction de sa relation avec les autres tokens. Ensuite, la couche feed-forward transforme la représentation de chaque token en une représentation de haute qualité [8].

— - **Le décodeur** est la partie du Transformer qui génère une séquence de sortie à partir de la représentation de l'entrée encodée par l'encodeur. Le décodeur est également constitué de plusieurs couches identiques, chacune contenant trois modules principaux : la couche de transformation de l'attention, la couche feed-forward et la couche d'attention croisée. La couche de transformation de l'attention dans le décodeur utilise une attention masquée pour éviter que les tokens de la sortie ne "voient" les tokens futurs de la séquence de sortie lors de la génération de chaque token. La couche d'attention croisée permet au décodeur de prendre en compte la représentation encodée de l'entrée pour générer une sortie cohérente [8].

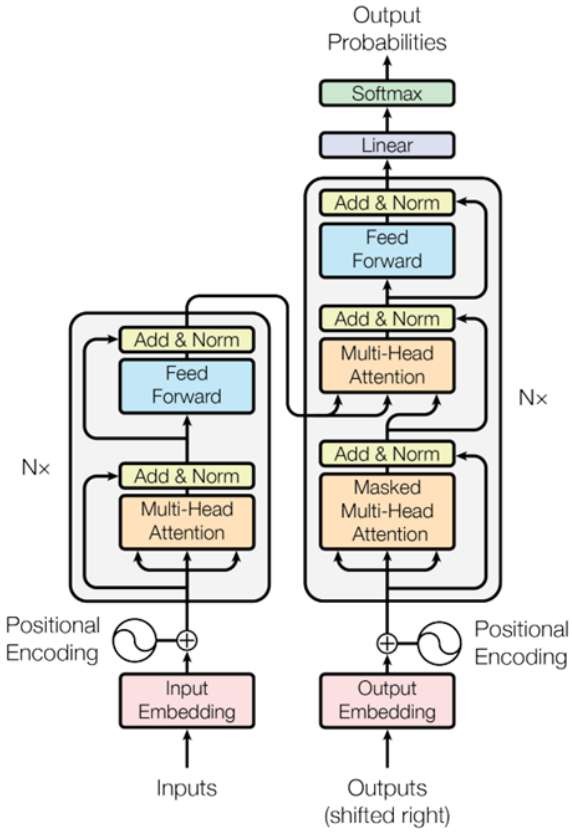

FIGURE 1.14 – L'architecture des transformers [8]

1.6.4.3 Mécanisme d'attention

L'architecture Transformer a révolutionné l'utilisation de l'attention en se passant des récurrences et des convolutions sur lesquelles les modèles précédents s'appuyaient largement. Une fonction d'attention est un mécanisme qui associe une requête à un ensemble de paires clé-valeur pour produire une sortie. La requête, les clés, les valeurs et la sortie sont tous des vecteurs. La sortie est obtenue en calculant une somme pondérée des valeurs, où les poids sont déterminés par une fonction de compatibilité entre la requête et chaque clé correspondante. En d'autres termes, la fonction d'attention mesure l'importance de chaque valeur en fonction de sa relation avec la requête [8].

1.6.4.4 BERT

BERT (Bidirectional Encoder Representations from Transformers) est un modèle de traitement de langage naturel pré-entraîné basé sur l'architecture Transformer. Il a été développé par Google en 2018 et a rapidement établi de nouveaux records de performance dans de nombreuses tâches de traitement de langage naturel, telles que la classification de textes, la réponse à des questions, et la compréhension de textes [32].

L'architecture de BERT utilise des blocs d'encodeurs Transformer empilés les uns sur les autres, chacun composé d'un module d'attention multi-tête suivi d'une couche feed-forward. Le

module d'attention multi-tête permet de prendre en compte le contexte de chaque token dans la séquence en regardant tous les autres tokens de la séquence, à la fois avant et après lui. Cela permet au modèle de capturer les relations sémantiques complexes entre les mots, ce qui est essentiel pour la plupart des tâches de traitement du langage naturel.

Le modèle BERT est également pré-entraîné sur de grandes quantités de données de texte brut, en utilisant des tâches d'apprentissage non supervisées telles que la prédiction de mots masqués et la prédiction de la phrase suivante, afin de capturer la structure du langage naturel. Ensuite, le modèle est finement ajusté (fine-tuned) sur des tâches spécifiques en ajoutant une couche de classification supplémentaire et en entraînant le modèle sur des données étiquetées pour cette tâche 1.15.

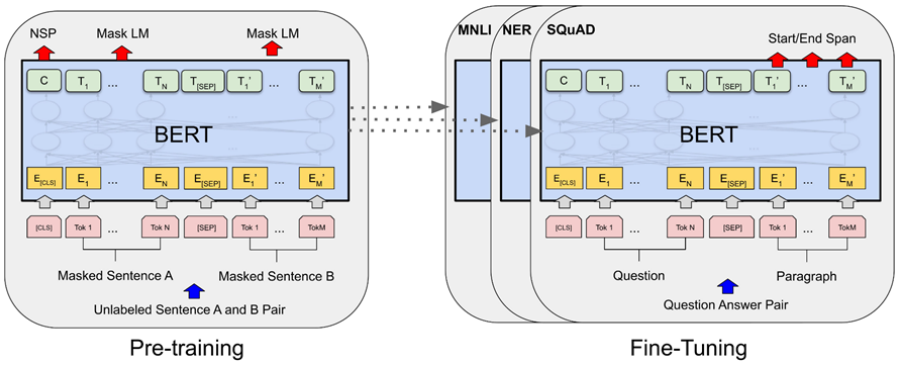

FIGURE 1.15 – Procédures globales de pré-entraînement et d'ajustement fin pour BERT

1.6.4.5 Utilisation de BERT (Fine-Tuning)

Le fine-tuning consiste à utiliser une version pré-entraînée de BERT dans l'architecture d'un modèle pour une tâche de NLP spécifique. Pour une tâche de classification de texte par exemple, plus précisément pour l'analyse de sentiments des avis de cinéphiles, l'architecture du modèle « fine-tune » peut ressembler à ça :

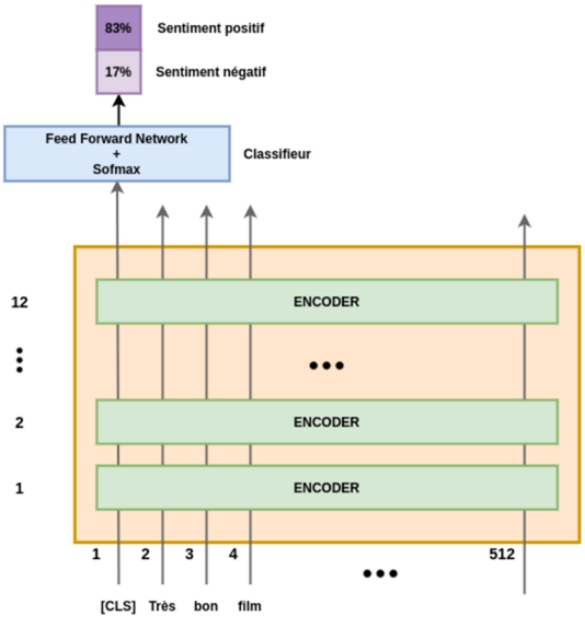

FIGURE 1.16 – BERT fine tuning [41]

1.6.4.6 AraBERT

AraBERT est un modèle de représentation de la langue arabe conçu pour améliorer les performances dans diverses tâches de traitement automatique du langage naturel (NLP) en arabe. Il est développé en se basant sur le modèle BERT [33].

FIGURE 1.17 – AraBERT logo

La configuration d'AraBERT, appelée BERTbase, comprend 12 blocs d'encodeurs, des dimensions cachées de 768, 12 têtes d'attention, une dimension maximale de séquence de 512, et un total de 110 millions de paramètres [33]. Le tableau 1.2 présente les deux dernières versions d'AraBERT avec leurs caractéristiques :

Modèle	**Taille** (Mo/Paramètres)	**Données** (Phrases/Taille/Nombres de mots)
AraBERTv0.2-base	543 Mo / 136 M	200 M / 77 Go / 8,6 Md
AraBERTv0.2-Twitter-base	543 Mo / 136 M	Identique à v0.2 + 60 M Tweets Multi-Dialectes

TABLE 1.2 – Tableau des caractéristiques des deux dernières versions d'AraBERT .

1.7 Le discour de haine et la langue arabe

Ces dernières années, le nombre d'utilisateurs de médias sociaux a augmenté de manière significative, en particulier dans la région arabe où les utilisateurs peuvent librement exprimer leur opinion et leur pensée, ce qui a entraîné une augmentation élevée du discours de haine et des crimes de haine. Cela pose un défi pour détecter et éliminer ce langage offensant des contenus arabes. La langue arabe est parlée par des millions de personnes dans le monde. On estime qu'il y a entre 280 et 422 millions de locuteurs natifs et non natifs. L'arabe utilise un script lu et écrit de droite à gauche, avec 28 lettres pouvant varier en fonction de leur position dans le mot. Les voyelles sont indiquées par des diacritiques. Il existe deux formes principales d'arabe : l'arabe classique et l'arabe standard moderne [34].

L'arabe classique, également connu sous le nom d'arabe littéraire ou coranique, est la langue écrite du Coran. Bien qu'il ne soit plus couramment parlé, il est encore utilisé à des fins religieuses en raison de son ancienneté. En revanche, l'arabe standard moderne est la langue officielle du monde arabe, largement utilisée dans les médias et la culture. Il est basé sur l'arabe classique en termes de syntaxe, de morphologie et de phonologie, mais présente des aspects lexicaux plus modernes.

Les arabophones utilisent une variété de dialectes, souvent estimés à au moins 30. Chaque dialecte est suffisamment distinct pour que les locuteurs aient souvent recours à l'arabe formel pour se comprendre mutuellement. Les dialectes arabes peuvent différer géographiquement, socialement, politiquement ou culturellement.

Ces dialectes arabes sont fréquemment utilisés dans les contenus générés par les utilisateurs sur les réseaux sociaux tels que Twitter, Facebook, Instagram, etc. Les dialectes les plus couramment utilisés comprennent l'arabe égyptien, l'arabe levantin (Liban, Syrie, Jordanie, Palestine), l'arabe dialectal du Golfe (Koweït, Émirats arabes unis, Bahreïn, Qatar, Arabie saoudite et Oman) et l'arabe nord-africain (Maroc, Algérie, Tunisie, Mauritanie et Libye).

Cependant, les dialectes arabes posent des défis tels que des erreurs d'orthographe, l'utilisation d'un langage informel sur les médias sociaux, l'absence de majuscules pour distinguer les caractéristiques et l'absence de grammaire standardisée dans les dialectes locaux [35]. Ces difficultés rendent encore plus complexe la détection des discours de haine en arabe.

1.8 Travaux connexes

Dans cette section, nous examinons les études les plus récentes sur la détection de discours de haine en ligne, principalement dans le contexte des langues anglaise et arabe.

La recherche de [36] a proposé un ensemble de données collecté à partir de Twitter compose de 14 100 tweets. Chaque tweet est associé à trois étiquettes indiquant s'il est offensant ou non, ciblé ou non, et si la cible est un individu ou un groupe. Les auteurs ont comparé le modèle hybride Unigrams+SVM aux réseaux de neurones CNN et BiLSTM. Les résultats obtenus par les deux modèles de réseau de neurones sont supérieurs à ceux obtenus par SVM pour toutes les tâches. Cependant, CNN a surpassé BiLSTM avec un score F1 de 80%, 69% et 47% pour les trois tâches respectivement.

Dans [37] les auteurs ont appliqué LR, NB, les arbres de décision, les forêts aléatoires et SVM en tant que modèles de classification pour l'identification de discours de haine sur un ensemble de données comprend 24 802 tweets étiquetés comme étant des discours de haine, du langage offensant ou aucun des deux. De plus, les auteurs ont utilisé une combinaison de Unigrams, Bigrams, Trigrams, le meilleur F1-macro de 90% a été obtenu en utilisant LR.

Les auteurs de [38] Ils ont utilisé deux ensembles de données disponibles publiquement pour étudier l'impact de l'utilisation d'un modèle basé sur BERT pré-entraîné avec différentes stratégies de fine-tuning sur la tâche de détection de discours de haine. La valeur F1-mesure la plus élevée est obtenue par BERTbase + CNN, qui est de 88% et 92% respectivement sur les ensembles de données Waseem-dataset et Davidson-dataset.

Dans l'étude de [39], un ensemble de données de discours de haine spécifique au dialecte levantin a été collecté, comprenant plus de 5800 tweets étiquetés. Les chercheurs se sont penchés sur l'utilisation des caractéristiques N-grammes pour détecter les tweets haineux. Les résultats de leurs expériences de classification binaire et multi-classes ont montré que le modèle Naïve Bayes (NB) surpassait le modèle SVM. En effet, le modèle Naïve Bayes a obtenu le meilleur score F1 avec une performance remarquable de 89,6%.

Dans une autre étude [40], plusieurs modèles ont été évalués pour classer les discours de haine en arabe. Les chercheurs ont utilisé un ensemble de données composé de 10 000 tweets étiquetés comme étant offensifs ou non. Ils ont constaté que le meilleur score F1 atteignait 83,2% en utilisant le modèle AraBERT.

Dans un autre contexte [41] , une étude s'est concentrée sur la détection des discours de haine et offensants en dialecte tunisien à partir d'un ensemble de données de 6 000 tweets. Pour

extraire les caractéristiques des tweets, les chercheurs ont appliqué une méthode basée sur les N-grammes avec une pondération de la fréquence des termes (TF). Les modèles SVM et Naive Bayes (NB) ont ensuite été utilisés pour la classification. Les résultats ont été remarquables, avec un score F1 de 83,6% pour la classification binaire et de 92,3% pour la classification ternaire.

Dans une étude récente [42] , les auteurs se sont penchés sur l'utilisation des modèles basés sur BERT pour détecter les discours de haine en arabe dans les commentaires des médias sociaux. Ils ont entraîné et évalué plusieurs modèles basés sur BERT pour cette tâche. Parmi ces modèles, BERT-EN (BERT English) s'est démarqué en obtenant les meilleurs résultats avec un score F1 de 98%. AraBERT a également réalisé une performance remarquable avec un score F1 de 95%, le plaçant juste derrière BERT-EN.

Un ensemble de données de discours de haine religieux en arabe a été collecté par [43] , comprenant environ 6600 tweets étiquetés comme étant haineux ou non haineux. Les chercheurs ont utilisé cet ensemble de données pour évaluer leur modèle de détection. Selon leurs résultats, le modèle GRU (Gated Recurrent Unit) a affiché les performances les plus élevées, atteignant un score F1 de 0,77. Le modèle SVM (Support Vector Machine) a également obtenu une performance solide avec un score F1 de 0,72, le plaçant juste derrière le modèle GRU.

Dans une étude comparative [44] visant à détecter les phrases de haine en arabe, cinq modèles d'embedding de mots ont été évalués, notamment CNN, GRU, BILSTM, ainsi qu'un modèle hybride CNN+BILSTM. Les chercheurs ont observé que les modèles basés sur le skip-gram produisaient des représentations plus efficaces que les autres embeddings de mots. En ce qui concerne les architectures de réseaux de neurones, les résultats ont montré que le modèle CNN surpassait les autres modèles pour les trois tâches de classification, à savoir la classification en 2 classes, 3 classes et 6 classes. Cette constatation met en évidence la performance supérieure du modèle CNN dans la détection des phrases de haine en arabe.

Dans leur étude [45] , les chercheurs ont utilisé un modèle d'apprentissage profond pré-entraîné appelé MarBERT pour classifier les discours de haine. Les tests ont été réalisés sur l'ensemble de données osact4, et les résultats ont révélé un score F1 de 92,3% pour la tâche HS (Hate Speech) et de 88,73% pour la tâche OFF (Offensive Speech). Les modèles multi-tâches, qui traitent simultanément plusieurs tâches liées aux discours de haine, ont montré de meilleures performances que les modèles mono-tâche.

Dans la tâche partagée OSACT 4 [46] , qui vise à détecter le langage offensant et les discours de haine sur la sphère Twitter arabe, le système gagnant pour la sous-tâche A a combiné plusieurs approches. Il a utilisé un ensemble de SVM avec des n-grammes et des embeddings pré-entraînés (Mazajak), ainsi que des modèles DNN tels que FastText, CNN+RNN et des

embeddings contextuels (multilingue BERT). Pour prendre une décision sur l'étiquette de chaque entrée, un vote majoritaire a été utilisé. Pour la sous-tâche B, des modèles linéaires SVM ont été utilisés avec un count vectorizer basé sur des caractères de 2 à 5. Les scores F1 les plus élevés obtenus pour les sous-tâches A et B étaient respectivement de 90,5% par [47] et 95,2 par [48].

Dans la tâche partagée OSACT5, décrite dans [49], un ensemble de données comprenant 12 698 tweets a été utilisé, avec trois sous-tâches distinctes : la détection de l'offensivité (Subtask A), la détection des discours de haine (Subtask B) et la détection du type de discours de haine (Subtask C). Pour la sous-tâche A, le score F1 le plus élevé atteint était de 0,852, réalisé par [50] en utilisant un ensemble de modèles (MARBERTV2+MARBERT+QARiB). Pour les sous-tâches B et C, les scores F1 les plus élevés étaient respectivement de 0,831 et 0,528, obtenus en utilisant MARBERT avec des réseaux de neurones quasi-récursifs, comme rapporté par [51].

L'auteur de l'article [52] a présenté le modèle Arabic BERT-Mini (ABMM) pour la détection des discours de haine sur les réseaux sociaux. Dans cette étude, le modèle BERT a été utilisé pour analyser les données provenant de Twitter et classifier les résultats en trois catégories : normal, abusif et haineux. Les performances du modèle ABMM se sont révélées très prometteuses, avec un score F1 de 0,986, surpassant ainsi les autres modèles évalués.

Dans l'article [53], les auteurs ont proposé une approche pour la détection des discours de haine dirigés contre les femmes dans la communauté arabe sur les réseaux sociaux. Ils ont développé un nouveau corpus de discours de haine, et utilisé trois algorithmes d'apprentissage automatique différents pour valider ce corpus. Les algorithmes utilisés étaient un réseau de neurones convolutif profond (CNN), un réseau de mémoire à court terme (LSTM) et un réseau LSTM bidirectionnel (Bi-LSTM). Les résultats ont révélé que le modèle CNN obtenait la meilleure performance, avec un score F1 de 86%.

Dans l'article [54], les chercheurs ont travaillé sur un nouveau corpus appelé DziriOFN, spécifiquement conçu pour la détection de langage offensant dans le dialecte algérien. Ce corpus contient 8 700 textes annotés manuellement comme étant normaux, offensants ou abusifs. Les résultats des modèles ML montrent que les SVM et les classificateurs multinomiaux Naive Bayes atteignent respectivement une accuracy de 74,4 % et 75,2 % pour la classification avec deux étiquettes, et 66,9 % et 66,2 % pour la classification avec trois étiquettes. En ce qui concerne les modèles DL, les résultats montrent que le CNN, le BiLSTM et FastText atteignent respectivement une accuracy de 52,3 %, 52,0 % et 71,6 % pour la classification avec deux étiquettes, et 34,7 %, 40,0 % et 64,8 % pour la classification avec trois étiquettes.

Dans l'article [55], les auteurs se sont intéressés à la détection de texte toxique en dialecte algérien. Ils ont créé un ensemble de données composé de 14 150 commentaires extraits de

différentes plateformes sociales telles que Facebook, YouTube et Twitter. Ces commentaires ont été étiquetés selon trois catégories : discours de haine, langage offensant et cyberintimidation. Plusieurs tests ont été réalisés en utilisant à la fois des modèles de classification d'apprentissage automatique traditionnels (ML) et des modèles d'apprentissage profond (DL). Parmi ces modèles, le modèle Bi-GRU (Bidirectional Gated Recurrent Unit) a obtenu les meilleurs résultats pour la classification avec trois étiquettes., avec un score F1 de 75,8%.
Le tableau suivant 1.3 résume les différentes études mentionnées et leurs résultats.

Article	DataSet	Size	Classes	Modèle et méthode de vectorisation	F1score (%)
[36]	OLID EN	14100	OFF, NOT-OFF	CNN ,Fast Text	80
			TIN, UNT		69
			GRP, IND, OTH		47
[37]	Davidson EN	24802	HS, OFF, NOT-OFF	LR N-Gram	90
[38]	Waseem EN	19697	Racism, Sexism, Neither	BERT + CNN	88
	Davidson EN	24783	Hate, Offensive, Neither		92
[39]	L-HSAB	5846	Abusive, Normal	Naive Bayes,N-Gram	89.6
			Abusive,Hate,Normal		74.4
[40]	MSA, DA	10000	OFF, NOT-OFF	ARABERT	83.2
				SVM ,Skip-Gram	79.7
[41]	T-HSAB	6039	Abusive, Normal	Naive Bayes, UNI+BI	83.6
			Abusive, Normal Hate		92.3
[42]	MSA, DA	11268	Hate, Non-Hate	BERTen	98
				ARABERT	95
[43]	MSA	6600	Hate, Non Hate	GRU Based RNN, Aravec	77
[44]	MSA, DA	5361	Hate, Clean	CNN, W2V Skip-Gram	87.22
			OFF, Hate, Clean	CNN ,FT Skip-Gram	76.09
			Clean,OFF,Rel,Gen,Nat,Eth	CNN ,FT Skip-Gram	71.06
[45]	MSA, DA	10000	OFF, NON-OFF	MTL-MARBERT	92.3
			HS, NON-HS	MTL-MARBERT	88.73
[46]	MSA	10,000	OFF NOT-OFF	SVM+CNN+RNN+MBERT	90.5
			HS NOT-HS	SVM,count-vectorizer	95.2
[49]	MSA	12698	OFF, NOT-OFF	MARBERTV2+MARBERT+QARIB	85.2
			HS, NOT-HS	MARBERT QRNN	83.1
			NON,HS1,HS2,....,HS6	MARBERT QRNN	52.8
[52]	MSA	9352	Normal, Abusive, Hate	ARABIC BERT-MINI	98.6
[53]	Algerian	3798	HS, NON-HS	CNN, SG	86
[55]	Algerian	14150	HS, OFF, CB	BI-GRU, Fast Text	75.8

TABLE 1.3 – Travaux Connexe

1.8.1 Discussion générale

Les travaux précédents dans le domaine de la détection de discours de haine en arabe ont exploré différentes approches basées sur la machine learning, en mettant l'accent sur l'utilisation de réseaux de neurones convolutifs (CNN), de modèles de transfert d'apprentissage (transfer learning) et d'apprentissage multi-tâches (multitask learning).

Les réseaux de neurones convolutifs (CNN) ont été largement utilisés en raison de leur capacité à extraire des caractéristiques pertinentes à partir de données textuelles. Ils sont capables de détecter des motifs locaux et globaux dans les séquences de mots, ce qui les rend adaptés à la détection de discours de haine Les résultats des études ont montré que les modèles CNN ont souvent obtenu de bonnes performances dans la classification des textes en arabe, avec des scores F1 élevés.

Le transfert d'apprentissage a également été exploré avec succès dans la détection de discours de haine en arabe. Les chercheurs ont utilisé des modèles pré-entraînés tels que BERT et ses variantes(comme AraBERT) pour bénéficier des connaissances apprises sur des tâches similaires dans d'autres langues. Cela a permis d'améliorer les performances des modèles en leur fournissant une compréhension plus profonde des relations entre les mots et les contextes.

En ce qui concerne l'apprentissage multi-tâches, certains travaux ont montré que l'entraînement simultané de modèles pour différentes tâches liées à la détection de discours de haine (par exemple, détection d'offensivité, classification de types de discours de haine) peut améliorer les performances globales du système. Cela est dû au partage des connaissances et des représentations apprises entre les tâches, ce qui permet d'obtenir une meilleure généralisation et une meilleure capacité à traiter des tâches complexes.

En conclusion, les travaux antérieurs ont démontré l'efficacité des modèles basés sur les machines Learning, tels que SVM et CNN, les modèles de transfert d'apprentissage et l'apprentissage multi-tâches, pour la détection de discours de haine et de toxicité en arabe. Ces approches ont permis d'obtenir des performances prometteuses, ouvrant la voie à des systèmes de détection plus robustes et adaptés aux spécificités de la langue arabe. Cependant, il reste encore des défis à relever, notamment en termes de collecte de données de haute qualité et de prise en compte des particularités culturelles et linguistiques propres à la région arabe d'où notre intérêt à comparer différentes démarches pour trouver la plus optimale.

1.9 Conclusion

Dans ce chapitre, une revue de la littérature sur les travaux précédents sur la détection automatique de discours de haine a été présentée, mettant en évidence certains travaux menés sur les langues anglaise et arabe. En outre, des travaux portant sur les langues dialectales, en particulier l'arabe dialectal, ont été examinés. En outre, le discours de haine sur les réseaux sociaux, ses catégories et ses impacts sociaux sur les individus ont été discutés et définis

Chapitre 2

Conception d'une méthode de détection du Hate Speech

2.1 Introduction

Dans ce chapitre, nous nous intéressons à la conception des modèles de classification de texte pour notre projet. Notre objectif est d'étudier différentes approches d'extraction de caractéristiques et d'identifier le modèle le plus performant pour notre tâche spécifique. Pour atteindre cet objectif, nous présentons tout d'abord l'architecture globale de notre système, ainsi que ses différentes composantes. Nous détaillons les différentes étapes du processus de classification de texte, en mettant l'accent sur la sélection des caractéristiques pertinentes et la construction des modèles appropriés.

2.2 Architecture de système

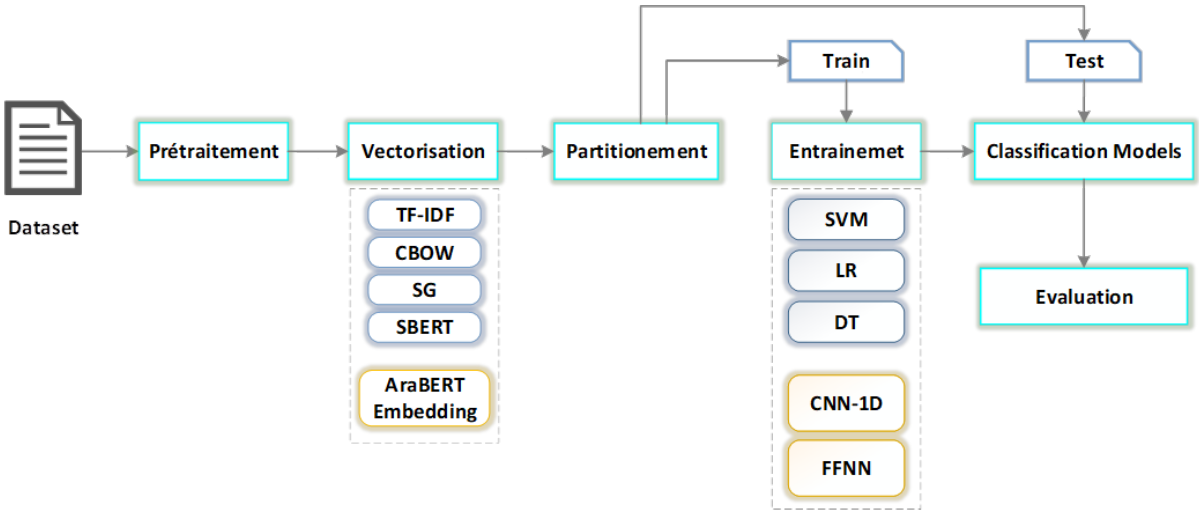

FIGURE 2.1 – Architecture du système

2.3 Description de l'architecture du système

L'architecture de notre système comprend deux tâches principales : le traitement du dataset et la construction du modèle de classification. Dans cette section, nous détaillons chaque tâche et expliquons comment elles s'articulent pour atteindre notre objectif global.

Traitement du dataset : Dans cette première étape, nous nous concentrons sur la préparation et le traitement du dataset. Cela comprend des étapes telles que le nettoyage du texte, la normalisation, la tokenisation et la création des ensembles d'entraînement, et de test.

Construction du modèle de classification : Une fois que le dataset est préparé, nous nous tournons vers la construction du modèle de classification. Cette étape implique la sélection d'une approche appropriée pour représenter les caractéristiques des textes et la construction du modèle lui-même. Nous explorons différentes méthodes d'extraction de caractéristiques, telles que les modèles de langue pré-entraînés, les méthodes basées sur les statistiques linguistiques. Nous détaillons également les différentes architectures de modèles que nous avons utilisées.

Dans les sections suivantes, nous approfondirons chaque tâche et expliquerons en détail les techniques, les méthodes et les choix spécifiques que nous avons faits pour construire notre système de classification de texte.

2.3.1 Traitement du dataset

Le dataset sur lequel nous travaillons a été collecté à partir de Twitter. Twitte en arabe présente des particularités linguistiques qui nécessitent une attention particulière lors du traitement des données. La langue arabe peut contenir des caractères spécifiques, des formes verbales diverses et une syntaxe complexe. Par conséquent, il est important de prendre en compte ces particularités lors du nettoyage et de la normalisation du texte afin d'obtenir une représentation précise des tweets.
Un autre aspect intéressant de Twitter est l'utilisation répandue des emojis dans les tweets en arabe. Les emojis sont des symboles graphiques utilisés pour exprimer des émotions, des idées ou des concepts. Leur présence dans les tweets peut fournir des informations supplémentaires pour la classification de texte.
De nombreuses techniques de prétraitement ont été utilisées dans ces étapes, les points suivants décrivent ces techniques :

2.3.1.1 Les étapes de prétraitement

L'ensemble de données sur lequel nous allons travailler passera par plusieurs étapes de prétraitement.

— **La suppression des diacritiques :** Fait référence à la procédure de suppression des signes diacritiques, également appelés "harakat" en arabe, des mots arabes. Les diacritiques sont des marques qui sont utilisées dans l'écriture de la langue arabe pour indiquer les voyelles courtes et d'autres modifications phonétiques.
Un exemple pour illustrer la suppression des diacritiques arabes :
Mot arabe avec diacritiques : مَرْحَبًا , Mot arabe sans diacritiques : مرحبا

— **La normalisation :** Dans le cadre de la normalisation du texte arabe, nous pouvons effectuer des transformations telles que la substitution de certaines lettres spécifiques par des lettres plus courantes. Par exemple, les lettres "إ" et "آ" peuvent être remplacées par la lettre" ا".
Dans le processus de suppression des lettres répétées, nous éliminons les répétitions excessives et conservons une seule occurrence de la lettre concernée. Par exemple, dans les revues en ligne, on peut trouver des mots comme "رااائع" où la lettre "ا" est répétée plusieurs fois.

— **La suppression des signes de ponctuation :** Cette opération consiste à éliminer tous les symboles de ponctuation présents dans les phrases arabes. Exemple : Phrase d'origine : "

؟مرحباً، كيف حالك" , Phrase sans ponctuation : "مرحبا كيف حالك".

— **Le filtrage de texte :** Fait référence à l'opération de suppression de certains éléments pécifiques dans notre dataset, tels que les mots non arabes, les nombres, les URLs, @USER.

— **La description des emojis :** Nous avons effectué la conversion des emojis en leur description en arabe. Cette opération consiste à traduire les emojis en arabe pour fournir une interprétation textuelle de leur signification.

— **Tokenisation :** Maintenant que nous avons des données nettoyées ne contenant que des mots significatifs, nous pouvons passer à la phase de tokenisation pour diviser le texte en mots séparés appelés tokens. La méthode la plus simple pour effectuer la tokenisation consiste à diviser le texte en fonction des espaces blancs.

— **Pading :** pour les modèles de Deep Learning la méthode "pad_sequences" est utilisée pour s'assurer que toutes les séquences d'une liste ont la même longueur.

Pour les modèles de transfert d'apprentissage :

— **Tokenisation de Transformers :** Ajouter [CLS] au début et [SEP] à la fin de la phrase. Le rôle de ces tokens est de définir les limites de la phrase. La tokenisation des phrases se fait en utilisant un tokenizer de transformer qui suit la tokenisation WordPiece.

— **Encodage des tokens :** Cette étape consiste à encoder (embedding) les tokens en utilisant la taille du vocabulaire des tokens. Cela permet d'obtenir "Inputs IDs".

— **Création de masques d'attention :** Ce masque d'attention permet de différencier les tokens de remplissage des tokens réels dans le calcul des vecteurs d'attention. Les tokens de remplissage ne seront pas pris en compte lors du calcul des vecteurs d'attention.

2.3.2 Extraction des caractéristiques

Dans cette partie, nous abordons les méthodes de représentation et de pondération de documents textuels, qui consistent à obtenir des vecteurs compréhensibles par la machine pour permettre l'apprentissage de nos modèles. Nous examinerons notamment les principales méthodes de représentation proposées pour atteindre cet objectif.

2.3.2.1 TF-IDF

Dans le cadre de notre approche d'extraction de caractéristiques, nous allons appliquer La méthode TF-IDF avec les algorithmes d'apprentissage automatique classiques afin de les comparer à l'utilisation des embeddings de mots en tant que caractéristiques. Cette méthode permet d'évaluer l'importance relative d'un terme dans un document par rapport à l'ensemble de la collection de documents. Elle prend en compte à la fois la fréquence du terme dans le

document (TF - Term Frequency) et sa fréquence inverse dans l'ensemble de la collection (IDF - Inverse Document Frequency).

2.3.2.2 Word2vec (Aravec)

Dans notre deuxième méthode de vectorisation, nous avons utilisé un modèle d'embedding de mots pré-entraîné en arabe appelé AraVec2.0 [9]. Ce modèle offre plusieurs architectures pour les embeddings de mots arabes, chacune étant entraînée sur l'un des trois ensembles de données distincts : les tweets, les pages Web et les articles arabes de Wikipédia. Pour chaque ensemble de données, deux modèles ont été construits en utilisant les approches Skip Gram et Continuous Bag of Words (CBOW). Plus précisément, pour notre étude axée sur les tweets, nous avons utilisé les modèles pré-entraînés avec les caractéristiques indiquées dans le tableau suivant 2.1.

Modèle	Documents	Vocabulaires	Dimension
Twitter-CBOW	66,900,000	331,679	300
Twitter-Skipgram	66,900,000	331,679	300

TABLE 2.1 – Les caractéristiques de CBOW et SG [9]

2.3.2.3 Contextuelle embedding

Pour la troisième méthode de vectorisation, nous avons utilisé AraBERT (Arabic Bidirectional Encoder Representations from Transformers)[33] pour encoder chaque phrase en arabe dans notre ensemble de données en utilisant son architecture de transformer. Nous avons ensuite extrait la sortie correspondante de AraBERT pour chaque phrase encodée, qui représente une représentation vectorielle de la phrase un tenseur de longueur fixe (généralement 768 pour les modèles AraBERT Base et 1024 pour les modèles AraBERT Large). Ces représentations vectorielles sont ensuite utilisées comme vecteur d'entrée pour nos différents modèles de classification de discours haineux.

2.3.3 Apprentissage des modèles

Après la préparation du jeu de données, Nous avons utilisé deux approches différentes pour l'entraînement des modèles : l'apprentissage automatique traditionnel (SVM,LR,DT) et l'apprentissage profond avec les modèles AraBERT fine-tuned et AraBERT avec CNN 1D.

2.3.3.1 Models de machine learning

Nous avons choisi les algorithmes de classification SVM, LR et Decision Tree pour évaluer la classification des discours haineux à partir des embeddings TF-IDF, CBOW Skip-gram et AraBERT embedding pour les raisons suivantes :

SVM : Les machines à vecteurs de support sont connues pour leur capacité à gérer des espaces vectoriels de grande dimension. Elles peuvent être efficaces dans la classification de textes, notamment pour la détection de discours haineux. Les SVM sont également capables de gérer des données non linéaires grâce à des noyaux appropriés. Dans notre cas, on a opté pour les SVM à kernel RBF(radial basis function). Pour le choix des paramètres C et gamma , nous utilisons l'algorithme Grid Search. Nous avons trouvé que la valeur C la plus adéquate est 10, et Gamma est 0.1.

LR (Régression Logistique) : La régression logistique est un algorithme couramment utilisé pour la classification binaire. Elle est bien adaptée à la classification de textes, et peut être efficace pour détecter les discours haineux. De plus, elle fournit des probabilités de prédiction, ce qui peut être utile pour interpréter les résultats. De la même manière que pour les SVM, nous avons opté pour l'algorithme Grid serach [56] pour el choix des paramètres optimaux en terme de Régulation (C) et de pénalité (penality) . Leurs valeurs optimales sont respectivement 100 et 12.

Decision Tree (Arbre de décision) : Les arbres de décision sont des modèles de classification populaires et faciles à interpréter. Ils peuvent capturer des relations non linéaires entre les caractéristiques et les classes cibles, ce qui peut être utile pour détecter les discours haineux. De plus, les arbres de décision sont moins sensibles aux valeurs aberrantes et aux données manquantes. Les arbres de décisions dépendent aussi d'un certain nombre de paramètres, nous avons opté pour les critères suivants :

— Fonction de mesure de l'impureté (criterion) : Pour une classification binaire, nous avons utilisé l'indice de Gini. Il mesure l'impureté des classes dans un nœud.

— Profondeur maximale de l'arbre (max_depth) : Ce paramètre contrôle la profondeur maximale de l'arbre. Une profondeur plus élevée peut permettre à l'arbre de capturer des relations plus complexes, mais cela peut également augmenter le risque de surajustement.

— Nombre minimum d'échantillons pour scinder un nœud (min_samples_split) : Ce paramètre définit le nombre minimum d'échantillons requis dans un nœud pour qu'il puisse être divisé davantage. Une valeur plus élevée peut éviter les divisions insignifiantes et aider à régulariser l'arbre. Cependant, une valeur trop élevée peut entraîner une perte d'informations. Nous ajusterons sa valeur via grid search.

En utilisant ces algorithmes sur les embeddings extraits des méthodes TF-IDF, CBOW Skip-gram et AraBERT Embedding, nous espérons obtenir une compréhension approfondie des performances de chaque méthode d'extraction de caractéristiques dans la classification des discours haineux.

2.3.3.2 Les Modèles de deep learning

Dans cette partie nous avons utilisé AraBERT la version arabe du modèle BERT de Google, développée pour prendre en charge les particularités de la langue arabe. AraBERT est conçu avec 110 millions de paramètres qui peuvent apprendre à partir de l'ensemble de données d'entrée. Il est également équipé de 12 couches d'auto-attention [33]. Nous avons utilisé la version du modèle "AraBERTv0.2-Twitter-base", qui a été pré-entraîné sur un énorme ensemble de données d'environ 77 gigaoctets de texte arabe. Cet ensemble de données se compose d'environ 200 000 000 de phrases qui génèrent 8,6 milliards de tokens. De plus, nous avons choisi ce modèle car il a été spécialement ajusté en utilisant des données multi-dialectales. Il a été entraîné sur 60 000 000 de tweets en arabe multi-dialectal, collectés et extraits de plateformes de médias sociaux de toutes les régions du monde arabe.

2.3.3.3 AraBERT avec FFNN

L'architecture de ce modèle est une combinaison du modèle AraBERT et d'un classificateur linéaire. Dans cette combinaison, le classificateur linéaire constitué de couches de réseaux de neurones entièrement connectés (FFNN). (Figure2.2).

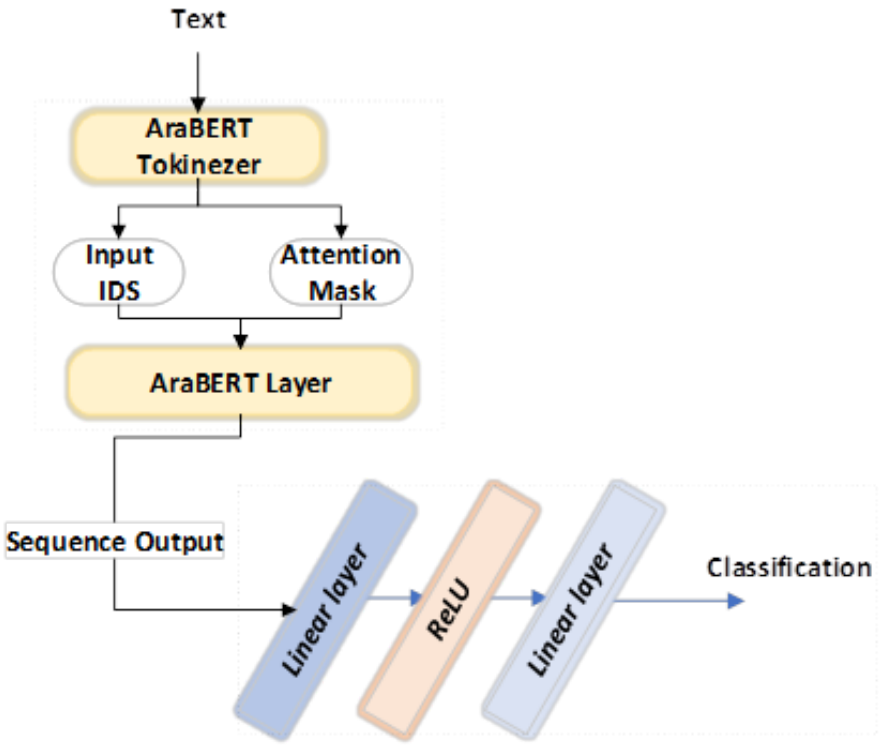

FIGURE 2.2 – L'architecture du modèle AraBERT avec FFNN

Description de l'architecture du Arabert avec FFNN

La tokenisation des textes est réalisée en utilisant AraBERT tokenizer, qui encode chaque phrase et génère les input_ids et les masques d'attention correspondants. Ces informations encodées sont ensuite renvoyées en sortie pour être utilisées en tant qu'entrées dans le modèle.
Nous avons choisi une longueur maximale (max_length) de 128. La sélection d'une longueur maximale de 128 a été motivée par le souci d'accélérer le processus d'entraînement sans sacrifier significativement l'information contenue dans les phrases. Cette valeur limite permet de traiter rapidement des séquences plus courtes et de réduire la complexité du modèle.

Le modèle BERT :

La propagation avant dans le modèle se déroule comme suit :
Le modèle prend en entrée les input_ids (identifiants d'entrée) et les attention_mask (masques d'attention) générés par le tokenizer AraBERT. Ces données d'entrée sont fournies au modèle BERT, qui traite les séquences et produit des sorties.
Les représentations cachées les plus récentes correspondant au jeton [CLS] sont extraites des sorties de BERT. Sont ensuite utilisées comme entrée pour le classificateur afin de réaliser la classification. La dimension de la sortie du modèle est de 768. Cela signifie que les représentations cachées extraites pour le jeton [CLS] ont une dimension de 768.

Le classificateur :

La dimension de la sortie du modèle est de 2. Cela signifie que le modèle génère des probabilités pour deux classes différentes dans le cadre de la tâche de classification. Le classificateur est ajouté au-dessus du modèle pour effectuer cette tâche spécifique. Il est constitué de deux couches linéaires avec une fonction d'activation ReLU appliquée entre elles. Ces couches permettent de transformer les représentations cachées en scores pour chaque classe, qui sont ensuite utilisés pour calculer les probabilités et effectuer la classification finale.

2.3.3.4 Le modèle AraBERT avec CNN-1D

Pour la deuxième approche nous avons utilisé One-Dimensional Convolutional Neural Network (CNN-1D) avec l'Embedding AraBERT comme entrée est conçu pour la classification des discours de haine. Nous avons utilisé la même tokenisation avec le tokenizer AraBERT et la même longueur maximale de séquence pour l'architecture CNN-1D que nous avons appliquée au modèle AraBERT+FFNN. La figure suivante (Figure 2.3) montre l'architecture de ce modèle.

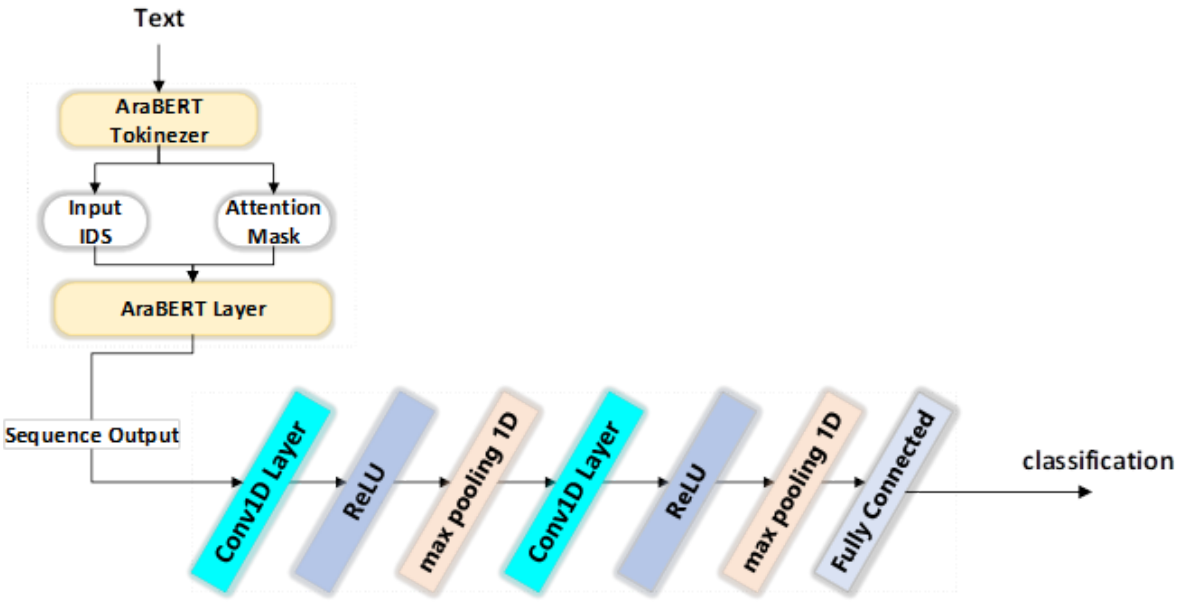

FIGURE 2.3 – L'architecture d' AraBERT avec CNN-1D

Les principales couches de ce modèle sont les suivantes :

— Arabert Tokenizer : Le modèle commence par utiliser le tokenizer Arabert pour convertir le texte en entrée en une représentation numérique adaptée à BERT. Le tokenizer utilise une longueur maximale de 128 tokens et renvoie les identifiants d'entrée (input_ids) et le masque d'attention (attention_mask).

— Modèle AraBERT :

Les input_ids et l'attention_mask générés par le tokenizer Arabert sont ensuite transmis au modèle Arabert. Ce modèle est responsable de la génération d'une représentation vectorielle pour chaque token en utilisant son dernier état caché (last hidden state). Le dernier état caché est une matrice de dimensions [batch_size(32), seq_length(128), hidden_size(768)], où batch_size représente la taille du lot d'entrée, seq_length est la longueur de la séquence d'entrée et hidden_size est la dimension de l'espace vectoriel caché de AraBERT.

— CNN-1D :

Le modèle CNN-1D est défini comme une séquence de couches de convolution 1D, de fonctions d'activation ReLU et de couches de pooling MaxPool1D. Les détails des couches sont les suivants :

1. Une couche de convolution 1D avec 768 canaux d'entrée (correspondant à la dimension cachée du modèle AraBERT), 128 canaux de sortie et une taille de noyau de 3.
2. Fonction d'activation ReLU pour introduire de la non-linéarité après la première couche de convolution.

3. Une couche de pooling MaxPool1D avec une taille de fenêtre de 2 pour réduire la taille des représentations.
4. Une autre couche de convolution 1D avec 128 canaux d'entrée (correspondant aux canaux de sortie de la première couche de convolution), 64 canaux de sortie et une taille de noyau de 3.
5. fonction d'activation ReLU pour introduire de la non-linéarité après la deuxième couche de convolution.
6. Une autre couche de pooling MaxPool1D pour réduire encore la taille des représentations.

— Couche linéaire :

Après les couches de convolution, une couche linéaire est utilisée pour effectuer la classification finale. Elle transforme les représentations vectorielles de dimension 64 (nombre de canaux de sortie de la dernière couche CNN-1D) en une sortie de dimension correspondant au nombre de classes à prédire (2 classes).

En combinant l'embedding AraBERT avec les couches de convolution 1D, ce modèle CNN 1D peut capturer des informations à différentes échelles et extraire des caractéristiques discriminantes à partir des séquences de discours de haine, permettant ainsi une classification précise des discours de haine.

2.3.4 Evaluation

Dans l'étape d l'évaluation Nous allons évaluer les performances de nos modèles en utilisant quatre mesures : la précision (Precision), le rappel (Recall), l'exactitude (Accuracy)et le score F1 (F 1 score) .

Précision (Precision) : La précision et le rappel sont deux critères de mesures statistiques évaluant les « classifieurs », aussi appelés valeur prédictive (précision) et sensibilité (rappel)[57]. Nous notons :

— **VP** : le nombre d'éléments correctement étiquetés positifs (vrai positif) .

— **FN** : le nombre de classifications incorrectes d'exemples positifs (faux négatifs).

— **FP** : le nombre d'éléments qui ont été incorrectement étiquetés positifs (faux positifs) .

— **VN** : le nombre de classifications correctes d'exemples négatifs (vrai négatif).

Dans une tâche de classification, la précision P d'une classe est le nombre de vrais positifs divisé par le nombre total d'éléments catégorisés positifs :

$$P = \frac{VP}{VP + FP}$$

Rappel (Recall) : Le Rappel R dans ce domaine est défini comme le nombre de vrais positifs divisé par le nombre total d'éléments qui appartiennent effectivement à la classe positive [57].

$$R = \frac{VP}{VP + FN}$$

Score F1 (F1 Score) : est une mesure appréciée de la performance d'un test qui combine à la fois la précision et le rappel. Elle est généralement utilisée pour comparer différents classificateurs avec une seule mesure[57].

$$\text{F1-score} = \frac{2 \cdot (\text{precision} \cdot \text{recall})}{\text{precision} + \text{recall}}$$

Exactitude (Accuracy) : L'exactitude mesure la proportion de prédictions correctes (vraies positives et vraies négatives) parmi toutes les prédictions faites par le modèle. Elle est calculée en divisant le nombre de prédictions correctes par le nombre total d'exemples [57].

$$\text{Accuracy} = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|}$$

2.4 Conclusion

Le Chapitre II a établi les fondements de notre travail de détection de contenu offensant dans le texte, qui constitue une tâche de classification. Nous avons mis en place une architecture robuste, préparé et traité le jeu de données, extrait les caractéristiques et entraîné les modèles en utilisant différentes méthodes. Ces étapes nous ont préparés pour la phase ultérieure de test et d'évaluation, qui sera abordée dans le Chapitre III.

Chapitre 3

Expérimentations et résultats

3.1 Introduction

Dans ce chapitre consacré aux tests et aux résultats, nous évaluons les performances des modèles de classification de texte pour la détection des textes offensants en arabe. Nous décrivons l'environnement de test, y compris les outils et les bibliothèques utilisés. Ensuite, nous détaillons les phases de notre expérimentation, notamment la création d'un ensemble de données homogène. Nous présentons ensuite les résultats de l'entraînement des modèles sur notre ensemble de données, en analysant les performances de chaque modèle et en discutant de leurs limites éventuelles. Cette évaluation nous permet de tirer des conclusions sur l'efficacité de notre approche pour la détection des textes offensants en arabe.

3.2 Matériel utilisé

Pour mener nos expérimentations, nous avons utilisé un ordinateur équipé des spécifications suivantes :

Processeur : Intel(R) Core (TM) i7-8665U CPU @ 1.90GHz 2.11 GHz.
Mémoire RAM installée : 16,0 Go (15,8 Go utilisable).
Type du Système : système d'exploitation 64 bits, Windows 10 Pro.

3.3 L'environnement du développement

Google Colaboratory : est un environnement interactif en ligne qui permet d'écrire et d'exécuter du code Python sans configuration requise. Colab offre un accès gratuit aux GPU et aux TPU, ce qui est particulièrement utile pour le machine learning et le deep learning. Il permet également d'installer facilement de nouvelles bibliothèques Python à l'aide de la commande "pip install". Nous avons exploité les fonctionnalités de Colab pour charger nos données, stocker nos notebooks et autres fichiers sur Google Drive, et profiter de 12 heures d'exécution continue par session[58]. Nous avons utilisé les GPU pour accélérer l'entraînement de nos modèles complexes. Colab nous a offert une plateforme pratique et puissante pour mener à bien nos expériences de test.

Python : est un langage de programmation polyvalent et largement utilisé dans le domaine de l'apprentissage automatique et du traitement du langage naturel [59]. Nous avons utilisé Python pour l'implémentation de notre modèle de détection des textes offensants en arabe.

Pandas : est une bibliothèque Python populaire pour la manipulation et l'analyse des données. Nous l'avons utilisé pour gérer et prétraiter nos données, notamment pour charger les ensembles de données, effectuer des opérations de nettoyage et de transformation des données, ainsi que pour la manipulation des structures de données tabulaires [60].

Numpy :est une bibliothèque Python essentielle pour le calcul numérique. Nous l'avons utilisé pour effectuer des opérations mathématiques sur les données, notamment pour les opérations matricielles et les calculs statistiques [61].

Sklearn : ou scikit-learn, est une bibliothèque Python populaire pour l'apprentissage automatique. Nous avons utilisé sklearn pour la mise en œuvre des algorithmes de classification et des métriques de performance, tels que la précision, le rappel, la mesure F1, etc [62].

Gensim : est une bibliothèque Python spécialisée dans le traitement du langage naturel. Nous l'avons utilisé pour l'implémentation de modèles de transformation de texte tels que le modèle Word2Vec, afin de créer des embeddings de mots [63].

Torch : est une bibliothèque open source en Python dédiée à l'apprentissage machine. Elle repose sur la bibliothèque Torch développée par Meta . PyTorch est utilisé pour effectuer des calculs tensoriels nécessaires à l'apprentissage profond, notamment les opérations sur les réseaux de neurones [64].

Transformers : est une bibliothèque Python développée par Hugging Face, qui fournit des implémentations pré-entrainées de modèles de transformation du langage tels que BERT, GPT, etc [65]. Nous avons utilisé Transformers pour intégrer le modèle pré-entrainé AraBERT dans notre architecture de modèle de détection des textes offensants.

FastAPI : est un framework web open source, basé sur Python, qui permet de développer rapidement des API REST performantes[66].

Jinja : est un moteur de templates open source pour Python. Il est utilisé pour générer du contenu dynamique dans les applications web [67].

UVicorn : est un serveur ASGI (Asynchronous Server Gateway Interface) basé sur Python, utilisé pour exécuter des applications web [68].

En utilisant cet ensemble d'outils et de bibliothèques, nous avons pu développer et évaluer notre modèles de manière efficace et précise.

3.4 Dataset

Nous avons utilisé deux ensembles de données différents pour la détection des discours offensants et haineux en arabe. Les ensembles de données utilisés sont le Corpus arabe open-source et outils de traitement de corpus (OSACT4) [69] et (OSACT5) [70].

Pour le premier ensemble de données OSACT4 , la tâche consistait à détecter les discours offensants et les discours haineux dans les médias sociaux arabes. La sous-tâche A utilisait l'ensemble de données de Twitter. OffensEval2020, qui comprenait 10 000 tweets annotés manuellement pour leur caractère offensant. La sous-tâche B se concentrait sur la détection des discours haineux dans les tweets, visant les insultes et les menaces basées sur la nationalité, l'origine ethnique, le genre, l'affiliation politique, la croyance religieuse, etc [69]. Nous avons utilisé uniquement les ensembles d'entraînement et de développement, car l'ensemble de test n'était pas étiqueté. La répartition des étiquettes dans l'ensemble de données est présentée dans le Tableau 3.1.

Subtask	Type	Total
A	NOT_OFF	6411
	OFF	1589
B	NOT_HS	7595
	HS	405

TABLE 3.1 – OSACT 4 Dataset

La deuxième source est un ensemble de données de tweets en arabe standard moderne et en dialectes arabes (OSACT5). Il s'agit du plus grand corpus annoté de tweets en arabe qui n'est pas biaisé envers des sujets spécifiques, des genres ou des dialectes. Nous avons également utilisé uniquement les ensembles d'entraînement et de développement, car l'ensemble de test n'était pas étiqueté. La répartition des étiquettes dans l'ensemble de données est présentée dans le Tableau 3.2.

Les deux ensembles de données présentaient un déséquilibre marqué dans les étiquettes, avec la sous-tâche de détection des discours haineux étant plus sévère que la sous-tâche de détection des discours offensants. Cela signifie que les exemples positifs de discours haineux étaient moins fréquents que les exemples positifs de discours offensants. Nous avons examiné la répartition des étiquettes dans les ensembles de données pour mieux comprendre les caractéristiques de chaque

Subtask	Type	Total
A	NOT_OFF	6581
	OFF	3576
B	NOT_HS	9089
	HS	1068

TABLE 3.2 – OSACT 5 Dataset

tâche.

- Exemple Sous-tâche A : الله يلعنه على هالسؤال (Que Dieu le maudisse pour cette question !)

- Exemple Sous-tâche B : أنتم شعب متخلف (Vous êtes un peuple retardé)

En utilisant ces ensembles de données, nous avons pu évaluer la performance de notre modèle de détection des discours offensants et haineux en arabe. Dans les sections suivantes, nous présentons les résultats de nos expérimentations et discutons des implications de ces résultats pour évaluer l'efficacité de notre travail.

3.4.1 Augmentation des données

Étant donné que les modèles d'apprentissage en profondeur tels que BERT sont avides de données, une grande quantité de données de chaque classe est nécessaire pour que tout apprentissage significatif se produise. Pour cette raison, nous avons appliqué une augmentation des données manuelle en fusionnant les deux ensembles de données disponibles. Tableau 3.3 présente la nouvelle répartition.

Type	Total
NOT_OFF	12992
OFF	5165
NOT_HS	16684
HS	1473

TABLE 3.3 – Dataset augmenté

Ensuite, nous équilibrons les données en supprimant aléatoirement des tweets de la classe la plus élevée. Nous obtenons de nouvelles données qui sont assez équilibrées pour la sous-tâche A, mais pas pour la sous-tâche B. Le tableau 3.4 présente les nouvelles dimensions de notre ensemble de données.

La figure suivante présente l'histogramme des sous taches "offensive" et "hate".

Type	Total
NOT_OFF	5841
OFF	5164
NOT_HS	9533
HS	1472

TABLE 3.4 – Dataset Finale

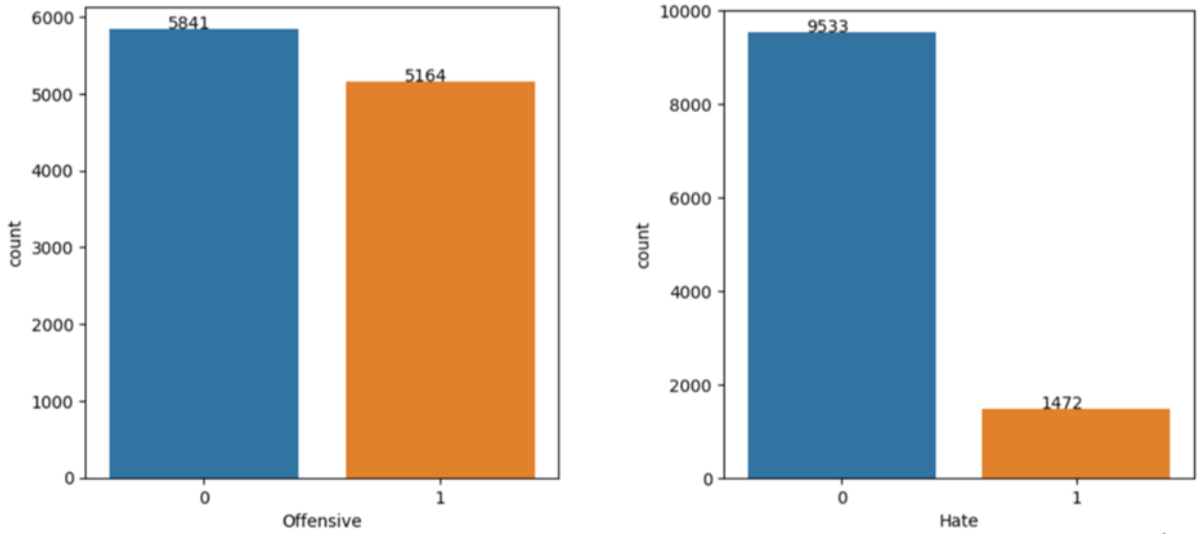

FIGURE 3.1 – L'histogramme du dataset final

3.5 La préparation des données

La préparation des données est une étape cruciale dans tous les modèles d'apprentissage. Elle implique le chargement et la préparation des données en appliquant différentes transformations. Dans notre cas, nous commençons par charger le jeu de données sous la forme d'un dataframe. Ensuite, nous utilisons une fonction de nettoyage spécifique qui prend ce dataframe de tweets et applique une série de transformations pour éliminer les signes diacritiques, normaliser les caractères, supprimer la ponctuation, réduire les caractères répétés, supprimer les mots en anglais et les chiffres, la description des emojis et nettoyer les espaces.

Nous avons choisi de comparer le rôle des emojis dans le contenu en effectuant deux comparaisons. La première comparaison a été réalisée en laissant les emojis tels quels, tandis que la seconde a été effectuée en les convertissant en leur description. Cette étape est inspirée du travail de [78]. La figure ci-dessus montre le dataset nettoyé avec et sans les descriptions des emojis.

	Tweet	Tweet_emo	OFF	HS
0	...يا رب يا واحد يا احد بحق يوم الاحد ان تهلك بني	...يا رب يا واحد يا احد بحق يوم الاحد ان تهلك بني	1	1
1	...يا لطيف يا ساتر احمدوا ربكم انها مستقعده 😐 كيف	...يا لطيف يا ساتر احمدوا ربكم انها مستقعده القمر	1	0
2	فين يا حراميه يا نصابين يا لصوص	فين يا حراميه يا نصابين يا لصوص	1	0
3	...شخصيه خياليه محونه حقوق الماحين انسان حقيقي مح	...شخصيه خياليه محونه حقوق الماحين انسان حقيقي مح	1	0
4	...يا جدعان العلق دا لسه بيقولي هذاكر الاسءله كفا	...يا جدعان العلق دا لسه بيقولي هذاكر الاسءله كفا	1	0
...	...	...	...	...
11000	...بكره حالي لما اعصب بكره حالي لما اعصب بكره حال	...بكره حالي لما اعصب بكره حالي لما اعصب بكره حال	0	0
11001	...بايع الكليجا الدين بتراجع مع انتزاع الحياء وال	...بايع الكليجا الدين بتراجع مع انتزاع الحياء وال	0	0
11002	💙 💛 🖤 يا بكايه يا بكايه	يا بكايه يا بكايه قلب ازرق قلب اصفر قلب اسود	0	0
11003	😐 👊 هذي سماجه مو مزح	هذي سماجه مو مزح القمر الجديد مع وجه لكمه	0	0
11004	...خلونا نتفق اذا شفتوا وجه الي تحبونه تاخذون تنه	...خلونا نتفق اذا شفتوا وجه الي تحبونه تاخذون تنه	0	0

11005 rows × 5 columns

FIGURE 3.2 – Dataset nettoyé

3.6 Le partitionnement de dataset

Le partitionnement des données en ensembles d'entraînement et de test est une étape importante pour s'assurer que les modèles sont évalués sur la même distribution de données, ce qui permet de faire une comparaison juste. Nous avons sauvegardé les ensembles d'entraînement et de test pour les utiliser dans tous les modèles. Cette étape a été réalisée à l'aide de la fonction split de la bibliothèque scikit-learn. Les données sont divisées en ensembles d'entraînement et de test (selon un ratio de 80 :20) avec une distribution des classes équilibre.

3.7 L'étape de vectorisation

3.7.1 Vectorisation avec TF-IDF :

Dans nos expérimentations, nous avons utilisé des algorithmes d'apprentissage machine classiques en combinaison avec la représentation TF-IDF (Term Frequency-Inverse Document Frequency). Cette représentation permet de quantifier l'importance d'un terme dans un document par rapport à l'ensemble du corpus.

Nous avons évalué les performances de chaque modèle en utilisant des mesures telles que l'accuracy, la précision, le rappel et le score F1. Nous avons employé la bibliothèque scikit-learn et les classes CountVectorizer et TfidfTransformer pour réaliser la vectorisation des textes en

utilisant des unigrammes et effectuer la transformation TF-IDF. La taille du jeu de données utilisant le TF-IDF est de (11005, 34377).

3.7.2 Vectorisation avec ARAVEC :

AraVec (cbow/skipgram) Ce sont des embeddings de mots pré-entraînés qui ont été entraînés par [9] en utilisant le framework word2vec sur un corpus de tweets. Dans notre étude, nous adoptons deux variantes d'AraVec, Continuous Bag-Of-Words (AraVec-cbow) et AraVec-skipgram. Ces modèles ont été construits en utilisant la bibliothèque Python gensim.

Dans l'objet KeyedVectors vector_size c'est la dimension des vecteurs d'embedding. Chaque terme est représenté par un vecteur de cette dimension. Il y a 331 679 clés ce sont les termes pour lesquels les embeddings ont été calculés. En d'autres termes, chaque mot du tweet est représenté par un vecteur dans un espace multidimensionnel, qui capture les caractéristiques sémantiques et syntaxiques du mot. Pour obtenir une représentation du tweet dans son ensemble, on prend la moyenne de tous ces vecteurs de mots. Cela permet de condenser l'information contenue dans le tweet en un seul vecteur, qui peut ensuite être utilisé comme entrée pour des modèles d'apprentissage machine traditionnels.

3.7.3 Vectorisation avec Sentence-BERT (SBERT)

Nous avons utilisé la bibliothèque Sentence Transformers [71] avec le modèle 'bert-base-AraBERTv02-twitter' . Ce modèle spécifique est basé sur BERT et a été pré-entraîné sur des données Twitter en langue arabe. Avec les paramètres par défaut utilisés dans ce modèle (SentenceTransformer), la sortie représente une dimension de 768 pour chaque phrase. Cela signifie que pour chaque phrase en entrée, le modèle génère une représentation vectorielle de taille 768 qui capture les informations sémantiques de la phrase. Une fois que nous avons extrait les Embeddings des phrases, nous pouvons les utiliser comme entrée pour notre modèles de classification.

3.8 Evaluation des modèles

Dans la phase d'entraînement des modèles, nous avons privilégié la tâche A en raison de l'équilibre entre les classes. Nous avons entraîné et évalué plusieurs modèles sur un ensemble de données équilibré spécifique à cette tâche. Cette approche nous a permis de sélectionner le modèle le plus performant pour la tâche A. Ensuite, nous avons évalué ce modèle sur la tâche B, caractérisée par un déséquilibre entre les classes. Cette évaluation nous a permis de comprendre comment le modèle se comporte sur des données déséquilibrées et d'évaluer son efficacité dans la détection de l'offensivité dans des situations réelles.

3.8.1 Evaluation des modèles du Machines learning

L'implémentation des modèles SVM (Support Vector Machines), LR (Logistic Regression) et DT (Decision Tree) est fait en utilisant la bibliothèque scikit-learn en Python. Nous avons utilisé la méthode GridSearchCV pour rechercher les meilleurs paramètres Cette approche nous permet de spécifier une grille de paramètres à explorer, puis de trouver les meilleures combinaisons de ces paramètres en utilisant une validation croisée.

Les hyperparamètres pour SVM sont C et gamma, pour LR ce sont C et penalty, et pour le modèle (Decision Tree) ce sont Criterion, max_depth et min_samples_split. Les meilleurs hyperparamètres obtenus après plusieurs combinaisons avec la méthode GridSearchCV et les resultat sont résumés dans les tableaux suivants.

Modèle	**Hyperparameters**	**F1-Score (%)**
SVM	C = 10, gamma = 0.1	76.06
LR	C = 100, penalty = l2	75.10
DT	Criterion : 'gini', max_depth : None, min_samples_split : 5	67.53

TABLE 3.5 – Les performances des modèles avec Tf-idf pour la tâche 'Offensive'

Les résultats montrent que le modèle SVM avec TF-IDF en utilisant les hyperparamètres C = 10 et gamma = 0.1 a obtenu le meilleur score F1 de 76.06%. Cela indique que le SVM avec ces paramètres a réussi à bien classifier les données.

Le modèle de régression logistique (LR) avec TF-IDF a également donné de bons résultats, atteignant un score F1 de 75.10% avec les hyperparamètres C = 100 et penalty = l2. Cela suggère que la régression logistique a réussi à capturer les relations linéaires entre les variables.

En revanche, l'arbre de décision avec TF-IDF a obtenu un score F1 inférieur de 67.53%, malgré l'utilisation des hyperparamètres tels que le critère 'gini', une profondeur maximale non spécifiée et un nombre minimum d'échantillons pour diviser l'arbre égal à 5. Cela peut indiquer que les décisions basées sur des seuils ne sont pas aussi efficaces que les méthodes basées sur des relations linéaires ou non linéaires pour notre problème de classification.

Les résultats montrent que pour le modèle SVM avec l'approche CBOW, les meilleurs hyperparamètres sont gamma = 0.1 et C = 10. Cela a donné une précision de 65.55%, un rappel de 65.52%, un score F1 de 65.35% et une précision globale de 65.35%. Ces résultats indiquent que le SVM avec ces paramètres a réussi à classifier les données avec une performance relativement bonne.

Pour le modèle de régression logistique (LR) avec CBOW, les meilleurs hyperparamètres sont C = 1 et penalty = 12. Cela a abouti à une précision de 64.48%, un rappel de 64.52%, un score F1 de 64.49% et une précision globale de 64.67%. Bien que légèrement inférieurs aux résultats du SVM, ces performances restent raisonnables pour la régression logistique.

En ce qui concerne l'arbre de décision avec CBOW, les hyperparamètres optimaux sont le critère 'entropy', une profondeur maximale de 5, un nombre minimum d'échantillons par feuille

Modèle	Hyperparameters	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
SVM	Gamma = 0.1, C = 10	65.55	65.52	65.35	65.35
LR	C = 1, penalty = l2	64.48	64.52	64.49	64.67
DT	Criterion : 'entropy', max_depth : 5, min_samples_leaf : 1, min_samples_split : 10	62.50	62.33	61.84	61.89

TABLE 3.6 – Les performances des modèles avec CBOW pour la tâche 'Offensive'

de 1 et un nombre minimum d'échantillons pour diviser l'arbre de 10. Cependant, les résultats obtenus sont moins satisfaisants avec une précision de 62.50%, un rappel de 62.33%, un score F1 de 61.84% et une précision globale de 61.89%. Cela suggère que l'arbre de décision n'est pas aussi performant que les autres modèles dans ce contexte particulier.

3.8.1.1 Modèles avec Skip-Gram

Modèle	Hyperparameters	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
SVM	Gamma = 0.1, C = 10	65.27	65.30	65.12	65.12
LR	C = 10, penalty = l2	64.14	64.19	64.16	64.35
DT	Criterion : 'entropy', max_depth : 5, min_samples_split : 3, min_samples_split : 10	62.66	62.90	62.09	62.16

TABLE 3.7 – Les performances des modèles avec Skip-Gram pour la tâche 'Offensive'

Pour le modèle avec l'approche skip-gram, les meilleurs hyperparamètres pour le SVM sont C = 10 et gamma = 0.1. Cela a donné une précision de 65.27%, un rappel de 65.30%, un score F1 de 65.12% et une précision globale de 65.12%. Ces résultats indiquent que le SVM avec ces paramètres a réussi à classifier les données avec une performance relativement bonne. En ce qui concerne le modèle de régression logistique (LR) avec skip-gram, les meilleurs hyperparamètres sont C = 10 et penalty = l2. Cela a abouti à une précision de 64.14%, un rappel de 64.19%, un score F1 de 64.16% et une précision globale de 64.35%. Bien que légèrement inférieurs aux résultats du SVM, ces performances restent raisonnables pour la régression logistique. En

revanche, l'arbre de décision avec skip-gram a donné des résultats moins satisfaisants. Les meilleurs hyperparamètres pour cet arbre de décision sont le critère 'entropy', une profondeur maximale de 5, un nombre minimum d'échantillons pour diviser l'arbre de 3 et un nombre minimum d'échantillons par feuille de 10. Les résultats obtenus sont une précision de 62.66%, un rappel de 62.90%, un score F1 de 62.09% et une précision globale de 62.16%. Cela suggère que l'arbre de décision n'est pas aussi performant que les autres modèles dans ce contexte spécifique, même avec l'approche skip-gram.

3.8.1.2 Modèles Avec SBERT

Modèle	Hyperparameters	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
SVM	Gamma= 0.0001,C=100	85.50	85.55	85.52	85.58
LR	C = 0.1 , penalty = l2	85.74	85.67	85.70	85.77
DT	Criterion : 'entropy', max_depth : 5, min_samples_split : 4	73.50	73.59	73.47	73.49

TABLE 3.8 – Les performances des modèles avec SBERT pour la tâche 'Offensive'

Les résultats montrent que pour le modèle SVM avec l'approche SBERT, les meilleurs hyperparamètres sont gamma = 0.0001 et C = 100. Cela a donné une précision de 85.50%, un rappel de 85.55%, un score F1 de 85.52% et une précision globale de 85.58%. Ces résultats indiquent que le SVM avec ces paramètres a obtenu une performance élevée dans la classification des données.

Pour le modèle de régression logistique (LR) avec SBERT, les meilleurs hyperparamètres sont C = 0.1 et penalty = 'l2'. Cela a abouti à une précision de 85.74%, un rappel de 85.67%, un score F1 de 85.70% et une précision globale de 85.77%. Ces performances sont similaires à celles du SVM, ce qui suggère que la régression logistique avec SBERT a également bien fonctionné dans la classification des données.

L'arbre de décision(DT) avec SBERT a obtenu des résultats inférieurs. Les meilleurs hyperparamètres pour cet arbre de décision sont le critère 'entropy', une profondeur maximale de 5 et un nombre minimum d'échantillons pour diviser l'arbre de 4. Les résultats obtenus sont une précision de 73.50%, un rappel de 73.59%, un score F1 de 73.47% et une précision globale de 73.49%. Bien que moins performant que les modèles précédents, l'arbre de décision avec SBERT a tout de même réussi à obtenir des résultats significatifs. En résumé, les modèles SVM et LR avec SBERT ont donné les meilleures performances, atteignant des scores F1 élevés et une précision globale élevée dans la classification des données. L'arbre de décision avec SBERT a

également montré des résultats décents, bien qu'il soit moins performant que les deux autres modèles.

Modèle	TF-IDF	SG	CBOW	SBERT
SVM	76.06	65.12	65.35	85.52
LR	75.10	64.16	64.49	**85.70**
DT	67.53	62.09	61.84	73.47

TABLE 3.9 – Les résultats des classificateurs en termes de F1-Score pour la tâche 'Offensive'

En général, la méthode SBERT a obtenu les meilleurs résultats, avec le SVM et la LR atteignant des scores F1 élevés. La méthode TF-IDF a également donné de bons résultats, bien que légèrement inférieurs à la méthode SBERT. Les méthodes de vectorisation Skip-gram et CBOW ont montré des performances inférieures dans la plupart des modèles, et l'arbre de décision a généralement obtenu les scores F1 les plus bas parmi les différentes méthodes de vectorisation.

3.8.2 Évaluation des modèles de réseaux de neurones

Dans cette section, nous évaluons les performances des modèles de réseau de neurones en utilisant AraBERT qui a été finement ajusté avec un FFNN (Feed-Forward Neural Network) et un CNN1D (Réseau de neurones convolutif unidimensionnel).

3.8.2.1 AraBERT avec FFNN

Le modèle AraBERT est un modèle de traitement du langage naturel pré-entraîné pour la langue arabe, qui peut être utilisé pour différentes tâches de traitement du langage. Dans ce cas, il est utilisé en conjonction avec un réseau de neurones à propagation avant (FFNN) pour effectuer une classification. L'optimiseur utilisé pour l'entraînement est AdamW, avec un taux d'apprentissage (learning rate) de 5e-5 et une décroissance (decay) de 1e-8. Avec ce modèle, nous obtenons une précision de 88,13%, un rappel de 88,25% et un score F1 de 88,16%. L'accuracy du modèle est de 88,18%. Le modèle a été entraîné avec une perte moyenne de 0,005 et a nécessité 20 époques d'entraînement (Tableau 3.10).

Model	Precision	Recall	F1 Score	Accuracy	Avg Train Loss	Epochs
AraBERT + FFNN	88.13	88.25	**88.16**	88.18	0.005	20

TABLE 3.10 – Les performances d'AraBERT +FFNN pour la tache 'Offensive'

3.8.2.2 AraBERT avec CNN1-D

Le modèle utilisant Bert-base-AraBERTv02-twitter en combinaison avec un réseau de neurones convolutifs (CNN) a obtenu des performances similaires au modèle AraBERT + FFNN

avec les mêmes hyperparamètres. Il a atteint une précision de 87.73%, un rappel de 87.85% et un score F1 de 87.75%. L'exactitude globale du modèle est de 87.77%. Le modèle a été entraîné sur une perte moyenne de 0.001 et a nécessité 20 époques pour l'entraînement (Tableau 3.11).

Model	Precision	Recall	F1 Score	Accuracy	Avg Train Loss	Epochs
AraBERT + CNN-1D	87.57	87.66	**87.60**	87.63	0.001	20

TABLE 3.11 – Les performances d' AraBERT + CNN-1D pour la tache 'Offensive'

Ces résultats indiquent que les deux modèles, malgré l'utilisation de différentes architectures (FFNN vs CNN), ont obtenu des performances comparables dans la classification. Cela suggère que la combinaison du modèle BERT spécifique à la langue arabe (Bert-base-AraBERTv02-twitter) avec des approches de réseaux de neurones différents peut aboutir à des résultats similaires en termes de précision, de rappel et de score F1.

3.8.3 Discussion sur la classification de la tâche offensive

En conclusion, les modèles SBERT+LR, AraBERT+FFNN ET AraBERT+CNN-1D ont obtenu de bonnes performances dans la tâche A (Offensive). Le modèle LR avec SBERT a atteint un F1 Score de 85.70%, tandis que le modèle AraBERT+FFNN a obtenu un F1 Score de 88.16% et le modèle AraBERT+CNN1D a obtenu un F1 Score de 87.60%. Ces résultats démontrent l'efficacité des modèles basés sur le traitement du langage naturel pour la classification de texte, en utilisant différentes méthodes de vectorisation et architectures de modèles. Le modèle AraBERT+FFNN a été légèrement supérieur en termes de F1 Score, ce qui peut être attribué à la combinaison du modèle de langue pré-entraîné AraBERT avec un réseau de neurones feedforward. Cependant, le modèle AraBERT+CNN1D a également montré de bonnes performances proches de celles du modèle AraBERT+FFNN. Les résultats obtenus suggèrent que l'utilisation de modèles pré-entraînés comme AraBERT combinés à des architectures de modèles différentes peut conduire à des performances satisfaisantes dans la classification de texte.

3.8.4 Expérimentation sur la classification des discours haineux (Hate Speech)

Pour la deuxième tâche de classification des textes contenant des discours haineux, nous avons évalué le jeu de données en utilisant les trois modèles les plus performants. Les données ont été soumises à ces modèles pour obtenir leurs prédictions et évaluer leurs performances spécifiques à cette tâche de classification.

Nous avons pris en compte la distribution déséquilibrée du jeu de données lors des tests et évaluations pour la classification des discours haineux. Le jeu de données de test comprend 1 906 échantillons de la classe "HS" (Hate Speech) et 294 échantillons de la classe "NOT-HS". Afin de mieux comprendre les résultats d'évaluation, nous avons calculé le score F1 pour chaque

classe en utilisant tous les modèles que nous avons sélectionnées. Cette approche nous permet d'analyser les performances des modèles non seulement globalement, mais également pour chaque classe individuellement.

3.8.4.1 LR avec SBERT

Class	Precision	Recall	F1-score
NOT_HS	93.68	97.17	95.39
HS	75.78	57.48	65.38

TABLE 3.12 – Les performances de SBERT + LR par classe de tâche "hate"

Modèle	Meilleurs hyperparamètres	Precision	Recall	F1-Score	Accuracy
SBERT+LR	C : 0.1, penalty : 'l2'	84.73	77.32	80.38	91.86

TABLE 3.13 – Les performances globales de model SBERT+LR pour la tache 'hate'

3.8.4.2 AraBERT avec FFNN

Class	Precision	Recall	F1-score
NOT_HS	95.45	95.65	95.55
HS	71.38	70.41	70.89

TABLE 3.14 – Les performances d AraBERT+FFNN par classe de tache 'hate'

Model	Precision	Recall	F1 score	Accuracy	Avg train loss	Epochs
AraBERT + FFNN	83.41	83.03	83.22	92.27	0.000918	40

TABLE 3.15 – Les performances globales d'AraBERT+FFNN pour la tache 'hate'

3.8.4.3 AraBERT avec CNN-1D

Class	Precision	Recall	F1-score
NOT_HS	96.15	95.59	95.87
HS	72.46	75.17	73.79

TABLE 3.16 – Les performances d'AraBERT+CNN-1D par classe de tache 'hate'

Model	Precision	Recall	F1 score	Accuracy	Avg train loss	Epochs
AraBERT + CNN-1D	84.30	85.38	**84.83**	92.86	0.000190	20

TABLE 3.17 – Les performances globales d' AraBERT+CNN-1D pour la tache 'hate'

3.8.5 Analyse des résultats

Tous les modèles obtiennent de bonnes performances dans la détection des textes non haineux, avec des scores de précision, de rappel et de F1 score élevés. Cependant, la détection des textes haineux est plus difficile et présente des performances inférieures. Pour comparer notre étude aux travaux précédents, nous devons prendre en compte le score F1, car c'est la métrique mentionnée dans la plupart des travaux cités dans le chapitre 1. Par conséquent, il est clair de choisir le modèle AraBERT+CNN-1D comme le plus performant pour cette tâche, malgré les données très déséquilibrées. Ce modèle obtient un score F1 de 95,87% pour la classe "NOT-HS" et de 73,79% pour la classe "HS", ce qui donne un score F1 global de 84,83%.

3.8.6 Choix des modèles

Les résultats d'évaluation des performances des modèles pour les tâches d'offense et de haine ont conduit à la sélection de différentes architectures. Pour la tâche'Offensive', le modèle AraBERT avec FFNN a été choisi. En revanche, pour la tâche 'Hate' AraBERT avec CNN-1D a été préféré. Cette sélection a été faite en fonction des performances observées lors des évaluations.

Pour l'ensemble de données qui contient les tweets avec la description des emojis, nous n'avons pas observé d'amélioration significative des résultats. Par conséquent, nous avons décidé de négliger cette information et de nous concentrer uniquement sur les emojis avec leur nature

3.8.7 Évaluation de la performance d'AraBERT+FFNN sur une dataset externe

Pour évaluer les capacités d'AraBERT, un modèle pré-entraîné sur la langue arabe standard et différents dialectes, nous allons le tester sur un ensemble de données externe appelé 'DZIRIOFN' [54]. Ce jeu de données est composé de deux classes : "OFF" (Offensive) et "NOT-OFF" (Not Offensive). Nous utiliserons le modèle AraBERT+FFNN que nous avons choisi pour la tâche de détection de contenu offensant. Les résultats obtenus en utilisant ce modèle sur le jeu de données 'DZIRIOFN' ont surpassé les scores obtenus dans l'étude précédente [54]. Ces résultats mettent en évidence les bonnes performances du modèle AraBERT+FFNN dans la prédiction de textes offensants (Tableau 3.8.7).

Model	Accuracy
SVM (Le meilleur modèle de [63])	0.744
AraBERT+FFNN (notre modèle)	**0.767**

TABLE 3.18 – Résultats de test sur un ensemble de données externe.

3.9 L'application

Nous avons développé une API solide et performante en utilisant FastAPI [81] comme framework principal. Pour la gestion des templates HTML, nous avons utilisé la bibliothèque Jinja2 [82], qui nous a permis de créer des templates réutilisables avec une syntaxe expressive et des fonctionnalités avancées telles que l'héritage de templates et la gestion des boucles.

Pour exécuter notre API, nous avons utilisé Uvicorn [83] comme serveur ASGI. Uvicorn est un serveur rapide et léger qui prend en charge les fonctionnalités avancées d'ASGI, nous permettant ainsi de tirer pleinement parti de l'asynchronisme et d'améliorer les performances de notre API.

Dans notre application, nous avons intégré les trois modèles les plus performants de notre étude : SBERT avec régression logistique (LR), AraBERT avec réseau neuronal entièrement connecté (FFNN) et AraBERT avec réseau neuronal convolutif à une dimension (CNN-1D).

En ce qui concerne la détection des types de discours, notre application suit une approche progressive. Nous commençons par prédire si le tweet est de type "Offensive". Si la prédiction indique que le tweet n'est pas offensant, nous concluons que le type de discours est normal et arrêtons le processus. Cependant, si la prédiction indique que le tweet est offensant, nous poursuivons en prédisant si le tweet est également de type "Hate". Cela nous permet de hiérarchiser les prédictions et de concentrer l'analyse supplémentaire sur les tweets identifiés comme offensants avant de prédire le type "Hate" le cas échéant. Cette approche progressive permet de rationaliser le processus de prédiction en évitant de prédire le type "Hate" pour les tweets qui sont déjà identifiés comme non offensants. Ainsi, nous optimisons les ressources et accélérons le traitement pour les tweets qui ne nécessitent pas d'analyse supplémentaire. Cela permet également d'obtenir des prédictions plus précises pour les tweets offensants, en se concentrant sur la détection du type "Hate" uniquement pour ceux qui ont déjà été identifiés comme offensants.

Grâce à cette approche de détection progressive, notre application peut classifier efficacement les tweets en différentes catégories (normal, offensant, haineux) en minimisant les étapes de prédiction supplémentaires lorsque cela n'est pas nécessaire.

La figure suivante illustre une prédiction d'un tweet en tant qu'offensant ou non-offensant par le modèle SBERT avec régression logistique (LR).

FIGURE 3.3 – L'interface de l'application

3.10 Conclusion

Dans ce chapitre, nous avons présenté l'environnement de travail utilisé, ainsi que les outils et les bibliothèques qui ont été employés dans notre étude. Nous avons ensuite exposé les résultats obtenus en utilisant différents modèles de classification pour les tâches de détection du discours offensant et haineux. À travers la comparaison des performances des différents classificateurs, nous avons identifié les modèles les plus performants pour chaque tâche. Le modèle AraBERT avec CNN-1D s'est révélé être le meilleur pour la tâche offensive, tandis que le modèle AraBERT avec FFNN a donné les meilleurs résultats pour la détection du discours haineux. Nous avons également présenté notre système sous la forme d'une application performante et efficace, qui repose sur ces modèles sélectionnés.

Conclusion générale et perspectives

L'objectif principal de notre travail était de détecter le discours de haine dans le contenu textuel en langue arabe, en particulier sur les réseaux sociaux, afin de fournir une solution efficace à ce problème qui affecte différentes catégories de personnes. Pour atteindre cet objectif, nous avons proposé une approche basée sur les domaines de l'Intelligence Artificielle, Spécialement de l'Apprentissage Automatique et de l'Apprentissage Profond, en appliquant différentes méthodes de classification des textes.

Nous avons préparé notre dataset en combinant deux ensembles de données collectés à partir de Twitter, avec deux tâches distinctes : l'offensant et le discours de haine. Ces ensembles de données comprenaient des tweets en arabe standard ainsi que des dialectes variés, afin d'atteindre notre objectif de couvrir un large spectre de la langue arabe.

Notre étude nous a permis d'explorer différentes techniques de Traitement Automatique du Langage et différents algorithmes tels que SVM, LR, DT, CNN-1D et FFNN, ainsi que les méthodes d'extraction de caractéristiques comme TF-IDF, Word2Vec, SBERT et AraBERT embeddings pour la classification. Nous avons constaté que l'utilisation des embeddings contextuels, notamment SBERT et AraBERT, a été la clé pour obtenir de bonnes performances avec tous les modèles, qu'ils soient basés sur l'apprentissage machine ou l'apprentissage profond. De plus, lorsque nous avons testé notre modèle sur un ensemble de données externe spécifique au dialecte algérien, nous avons obtenu des résultats très encourageants.

Nos résultats finaux ont démontré que le modèle AraBERT avec FFNN était le meilleur parmi les modèles que nous avons étudiés pour détecter les textes offensants, avec un score F1 de 88,16. De plus, le modèle AraBERT avec CNN-1D s'est révélé être le meilleur pour détecter les textes haineux, avec un score F1 de 84,83. Le prétraitement de notre dataset n'a pas été trop intensif, mais il nécessite des améliorations futures afin d'optimiser davantage les performances. Nous avons également rencontré des problèmes de performances liés à notre matériel, ce qui nous a

contraints à effectuer des entrainements sur les modèles dans des conditions de temps d'exécution et de taille de dataset limitées, en raison de la complexité des algorithmes d'apprentissage.

Malgré ces défis, nos résultats se sont avérés être très proches des meilleurs résultats obtenus dans des recherches similaires, et dans certains cas, ils ont même surpassé d'autres travaux. Cela témoigne de l'efficacité de notre approche et de la pertinence de l'utilisation des modèles de langage pré-entraînés pour la détection du discours offensant et haineux dans la langue arabe.

En tant que perspectives futures, nous envisageons de développer un script permettant d'extraire les données directement à partir de la plateforme afin de construire notre propre ensemble de données spécifiques au contenu algérien. De plus, nous prévoyons de nous pencher sur les méthodes les plus récentes dans le domaine du Traitement Automatique du Langage (TAL).

En conclusion, ce projet a été une expérience enrichissante qui nous a permis d'acquérir de nouvelles compétences et de consolider celles que nous possédions déjà. Nous avons également développé notre maîtrise du langage Python ainsi que des plateformes Google Colab et Jupyter, que nous continuerons à utiliser dans nos futurs travaux.

Bibliographie

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Proceedings of Workshop at ICLR*, vol. 2013, 01 2013.

[2] I. S. Singh, R. Bansal, A. Gupta, and A. K. Singh, "A hybrid grey wolf-whale optimization algorithm for optimizing svm in breast cancer diagnosis," *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pp. 286–290, 2020.

[3] R. Jain, "Support vector machine (svm)," *Medium*, June 2020. [Online]. Available : https://medium.com/@rishabhjain9440/support-vector-machine-svm-f23cece26419

[4] O. Sunday Samuel, "Breast cancer detection with machine learning approach," *FUDMA JOURNAL OF SCIENCES*, vol. 7, pp. 216–222, 04 2023.

[5] R. Choudhary and H. K. Gianey, "Comprehensive review on supervised machine learning algorithms," in *2017 International Conference on Machine Learning and Data Science (MLDS)*, 2017, pp. 37–43.

[6] A. Bulcke. (2020, mai) Les réseaux de neurones artificiels. [Online]. Available : https://si.blaisepascal.fr/1t-reseaux-de-neurones/

[7] M. Hanif, F. Khalid, and M. Shafique, "Cann : Curable approximations for high-performance deep neural network accelerators," 06 2019, pp. 1–6.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available : http://arxiv.org/abs/1706.03762

[9] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, "Aravec : A set of arabic word embedding models for use in arabic nlp," *Procedia Computer Science*, vol. 117, pp. 256–265, 2017, arabic Computational Linguistics. [Online]. Available : https://www.sciencedirect.com/science/article/pii/S1877050917321749

[10] K. Dinakar, R. W. Picard, and H. Lieberman, “Common sense reasoning for detection, prevention, and mitigation of cyberbullying,” in *TIIS*, 2012.

[11] R. Dredge, J. Gleeson, and X. de la Piedad Garcia, “Cyberbullying in social networking sites : An adolescent victim’s perspective,” *Computers in Human Behavior*, vol. 36, pp. 13–20, 2014. [Online]. Available : https://www.sciencedirect.com/science/article/pii/S0747563214001484

[12] T. Davidson, D. Warmsley, M. Macy, and I. Weber, “Automated hate speech detection and the problem of offensive language,” 2017.

[13] P. Fortuna and S. Nunes, “A survey on automatic detection of hate speech in text,” vol. 51, no. 4, 2018. [Online]. Available : https://doi.org/10.1145/3232676

[14] J. H. Park and P. Fung, “One-step and two-step classification for abusive language detection on Twitter,” in *Proceedings of the First Workshop on Abusive Language Online*. Vancouver, BC, Canada : Association for Computational Linguistics, Aug. 2017, pp. 41–45. [Online]. Available : https://aclanthology.org/W17-3006

[15] Y. Chen, Y. Zhou, S. Zhu, and H. Xu, “Detecting offensive language in social media to protect adolescent online safety,” in *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, 2012, pp. 71–80.

[16] K. Molek-Kozakowska, “Recenzja/review : Jim o’driscoll (2020). offensive language : Taboo, offence and social control. london : Bloomsbury academic. isbn 9781350169678,” *Res Rhetorica*, vol. 9, pp. 166–169, 12 2022.

[17]

[18] B. Mathew, A. Illendula, P. Saha, S. Sarkar, P. Goyal, and A. Mukherjee, “Hate begets hate : A temporal study of hate speech,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, pp. 1–24, 10 2020.

[19] J. MILON, “Text mining : Classification automatique de textes,” https://www.headmind.com/fr/text-mining-classification-automatique-de-textes/, HeadMind Partners Digital, 2021, publié le 14/04/2021.

[20] S.-W. Kim and J.-M. Gil, “Research paper classification systems based on tf-idf and lda schemes,” *Human-centric Computing and Information Sciences*, vol. 9, no. 1, p. 30, 2019. [Online]. Available : https://doi.org/10.1186/s13673-019-0192-7

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv :1301.3781*, 2013.

[22] B. Natanelic, “Nlp : Extraire des caractéristiques,” *Medium*, Oct 2020. [Online]. Available : https://beranger.medium.com/nlp-extraire-des-caractÃl’ristiques-pour-utiliser-des-algorithmes-ml-1c5bd9c421ba

[23] J. Yousif and M. Al-Risi, “Part of speech tagger for arabic text based support vector machines : A review,” *ICTACT Journal on Soft Computing*, vol. 09, January 2019, special Issue on Artificial Intelligence and Deep Learning. [Online]. Available : https://ssrn.com/abstract=3460178

[24] R. Genuer and J.-M. Poggi, “Arbres cart et forêts aléatoires, importance et sélection de variables,” 2017.

[25] Y. Mercadier, “Classification automatique de textes par réseaux de neurones profonds : application au domaine de la santé,” Nov. 2020. [Online]. Available : https://theses.hal.science/tel-03145856

[26] F. Rosenblatt, “The perceptron : a probabilistic model for information storage and organization in the brain.” *Psychological review*, vol. 65 6, pp. 386–408, 1958.

[27] C. Touzet, *LES RESEAUX DE NEURONES ARTIFICIELS, INTRODUCTION AU CONNEXIONNISME : COURS, EXERCICES ET TRAVAUX PRATIQUES*, ser. Collection de l’EERIE. EC2, 1992.

[28] C. Dancette, A. Saporta, and M. Cord, “Introduction aux réseaux de neurones,” Website, October 2021, retrieved from https://webia.lip6.fr/~dancette/deep-learning/.

[29] C. Hardy, “Contribution au développement de l’apprentissage profond dans les systèmes distribués,” Intelligence artificielle [cs.AI], Université de Rennes, 2019, ffNNT : 2019REN1S020ff. [Online]. Available : https://tel.archives-ouvertes.fr/tel-02284916

[30] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” *CoRR*, vol. abs/1511.08458, 2015. [Online]. Available : http://arxiv.org/abs/1511.08458

[31] M. Bouaziz, “Réseaux de neurones récurrents pour la classification de séquences dans des flux audiovisuels parallèles,” Ph.D. dissertation, 12 2017.

[32] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT : pre-training of deep bidirectional transformers for language understanding,” *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available : http://arxiv.org/abs/1810.04805

[33] W. Antoun, F. Baly, and H. Hajj, “AraBERT : Transformer-based model for Arabic language understanding,” in *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*. Marseille, France : European Language Resource Association, May 2020, pp. 9–15. [Online]. Available : https://aclanthology.org/2020.osact-1.2

[34] I. Arabic, "Statistics about the arabic language," https://industryarabic.com/arabic-facts-statistics/, March 2023, accessed on June 29, 2023.

[35] G. Alwakid, T. Osman, and T. Hughes-Roberts, "Challenges in sentiment analysis for arabic social networks," *Procedia Computer Science*, vol. 117, pp. 89–100, 2017, arabic Computational Linguistics. [Online]. Available : https://www.sciencedirect.com/science/article/pii/S1877050917321543

[36] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "Predicting the type and target of offensive posts in social media," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota : Association for Computational Linguistics, Jun. 2019, pp. 1415–1420. [Online]. Available : https://aclanthology.org/N19-1144

[37] T. Davidson, D. Warmsley, M. W. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," *CoRR*, vol. abs/1703.04009, 2017. [Online]. Available : http://arxiv.org/abs/1703.04009

[38] M. Mozafari, R. Farahbakhsh, and N. Crespi, "Hate speech detection and racial bias mitigation in social media based on BERT model," *CoRR*, vol. abs/2008.06460, 2020. [Online]. Available : https://arxiv.org/abs/2008.06460

[39] H. Mulki, H. Haddad, C. Bechikh Ali, and H. Alshabani, "L-HSAB : A Levantine Twitter dataset for hate speech and abusive language," in *Proceedings of the Third Workshop on Abusive Language Online*. Florence, Italy : Association for Computational Linguistics, Aug. 2019, pp. 111–118. [Online]. Available : https://aclanthology.org/W19-3512

[40] H. Mubarak, A. Rashed, K. Darwish, Y. Samih, and A. Abdelali, "Arabic offensive language on Twitter : Analysis and experiments," in *Proceedings of the Sixth Arabic Natural Language Processing Workshop*. Kyiv, Ukraine (Virtual) : Association for Computational Linguistics, Apr. 2021, pp. 126–135. [Online]. Available : https://aclanthology.org/2021.wanlp-1.13

[41] H. Haddad, H. Mulki, and A. Oueslati, *T-HSAB : A Tunisian Hate Speech and Abusive Dataset*, 10 2019, pp. 251–263.

[42] Z. Boulouard, M. Ouaissa, M. Ouaissa, M. Krichen, M. Almutiq, and G. Karim, "Detecting hateful and offensive speech in arabic social media using transfer learning," *Applied Sciences*, vol. 12, p. 12823, 12 2022.

[43] N. Albadi, M. Kurdi, and S. Mishra, "Are they our brothers ? analysis and detection of religious hate speech in the arabic twittersphere," in *2018 IEEE/ACM International*

Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2018, pp. 69–76.

[44] S. Alsafari, S. Sadaoui, and M. Mouhoub, "Hate and offensive speech detection on arabic social media," *Online Social Networks and Media*, vol. 19, p. 100096, 2020. [Online]. Available : https://www.sciencedirect.com/science/article/pii/S2468696420300379

[45] W. Aldjanabi, A. Dahou, M. A. A. Al-qaness, M. A. Elaziz, A. M. Helmi, and R. Damaševičius, "Arabic offensive and hate speech detection using a cross-corpora multi-task learning model," *Informatics*, vol. 8, no. 4, 2021. [Online]. Available : https://www.mdpi.com/2227-9709/8/4/69

[46] H. Mubarak, K. Darwish, W. Magdy, T. Elsayed, and H. Al-Khalifa, "Overview of OSACT4 Arabic offensive language detection shared task," in *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*. Marseille, France : European Language Resource Association, May 2020, pp. 48–52. [Online]. Available : https://aclanthology.org/2020.osact-1.7

[47] S. Hassan, Y. Samih, H. Mubarak, A. Abdelali, A. Rashed, and S. A. Chowdhury, "ALT submission for OSACT shared task on offensive language detection," in *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*. Marseille, France : European Language Resource Association, May 2020, pp. 61–65. [Online]. Available : https://aclanthology.org/2020.osact-1.9

[48] F. Husain, "OSACT4 shared task on offensive language detection : Intensive preprocessing-based approach," in *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*. Marseille, France : European Language Resource Association, May 2020, pp. 53–60. [Online]. Available : https://aclanthology.org/2020.osact-1.8

[49] H. Mubarak, H. Al-Khalifa, and A. Al-Thubaity, "Overview of OSACT5 shared task on Arabic offensive language and hate speech detection," in *Proceedinsg of the 5th Workshop on Open-Source Arabic Corpora and Processing Tools with Shared Tasks on Qur'an QA and Fine-Grained Hate Speech Detection*. Marseille, France : European Language Resources Association, Jun. 2022, pp. 162–166. [Online]. Available : https://aclanthology.org/2022.osact-1.20

[50] A. Mostafa, O. Mohamed, and A. Ashraf, "GOF at Arabic hate speech 2022 : Breaking the loss function convention for data-imbalanced Arabic offensive text detection," in *Proceedinsg of the 5th Workshop on Open-Source Arabic Corpora and Processing Tools with Shared Tasks on Qur'an QA and Fine-Grained Hate Speech Detection*. Marseille,

France : European Language Resources Association, Jun. 2022, pp. 167–175. [Online]. Available : https://aclanthology.org/2022.osact-1.21

[51] M. A. Bennessir, M. Rhouma, H. Haddad, and C. Fourati, "iCompass at Arabic hate speech 2022 : Detect hate speech using QRNN and transformers," in *Proceedinsg of the 5th Workshop on Open-Source Arabic Corpora and Processing Tools with Shared Tasks on Qur'an QA and Fine-Grained Hate Speech Detection*. Marseille, France : European Language Resources Association, Jun. 2022, pp. 176–180. [Online]. Available : https://aclanthology.org/2022.osact-1.22

[52] M. Almaliki, A. M. Almars, I. Gad, and E.-S. Atlam, "Abmm : Arabic bert-mini model for hate-speech detection on social media," *Electronics*, vol. 12, no. 4, 2023. [Online]. Available : https://www.mdpi.com/2079-9292/12/4/1048

[53] I. Guellil, A. Adeel, F. Azouaou, M. Boubred, Y. Houichi, and A. A. Moumna, "Ara-women-hate : An annotated corpus dedicated to hate speech detection against women in the Arabic community," in *Proceedings of the Workshop on Dataset Creation for Lower-Resourced Languages within the 13th Language Resources and Evaluation Conference*. Marseille, France : European Language Resources Association, Jun. 2022, pp. 68–75. [Online]. Available : https://aclanthology.org/2022.dclrl-1.9

[54] O. Boucherit and K. Abainia, "Offensive language detection in under-resourced algerian dialectal arabic language," 2022.

[55] A. C. MAZARI and H. Kheddar, "Deep learning-based analysis of algerian dialect dataset targeted hate speech, offensive language and cyberbullying," vol. 13, pp. 965–972, 04 2023.

[56]

[57] J. Brownlee, "How to calculate precision, recall, and f-measure for imbalanced classification," *Machine Learning Mastery*, January 2020. [Online]. Available : https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/

[58] "Google colab." [Online]. Available : https://colab.research.google.com/

[59] "Python." [Online]. Available : https://www.python.org/

[60] "Pandas." [Online]. Available : https://pandas.pydata.org/

[61] "Numpy." [Online]. Available : https://numpy.org/

[62] "scikit-learn." [Online]. Available : https://scikit-learn.org/stable/

[63] "Gensim." [Online]. Available : https://pypi.org/project/gensim/

[64] “Pytorch.” [Online]. Available : https://pytorch.org/

[65] “Hugging face transformers.” [Online]. Available : https://huggingface.co/docs/transformers/index

[66] “Fastapi.” [Online]. Available : https://fastapi.tiangolo.com/

[67] “Jinja.” [Online]. Available : https://jinja.palletsprojects.com/en/3.1.x/

[68] “Uvicorn.” [Online]. Available : https://www.uvicorn.org/

[69] “Osact4 : The 4th workshop on open-source arabic corpora and processing tools with shared task on offensive language detection,” Marseille, France. 12th May 2020. Co-located with LREC 2020. [Online]. Available : https://edinburghnlp.inf.ed.ac.uk/workshops/OSACT4/

[70] “Arabic hate speech 2022 shared task,” OSACT 2022 Workshop, LREC 2022, Marseille, France, 20th June 2022. [Online]. Available : https://sites.google.com/view/arabichate2022/home

[71] “sentence-transformers.” [Online]. Available : https://huggingface.co/sentence-transformers