



République Algérienne Démocratique et Populaire



Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

## Faculté d’Informatique

Département d'intelligence artificielle et sciences des données

Mémoire de Master

## Filière : Informatique

Spécialité : Systèmes Informatiques Intelligents

## Thème

---

Développement d'un système de détection de discours de haine pour la langue  
Arabe : Une approche basée sur l'apprentissage profond

---

Sujet Proposé par :

- Dr. HADJ AMEUR Mohamed
- AOUICHAZ Asma

Présenté par :

- KHELKHAL Kenza
- SMAALI Rim

Soutenu le 30/06/2024

Devant le jury composé de :

- Mme. TAMEN Zahia  
Mme. BERKANI Lamia

Présidente  
Membre

# **Remerciement**

Nous rendons d'abord grâce à Dieu le Tout-Puissant et Miséricordieux, pour nous avoir accordé la force et la patience nécessaires à la réalisation de ce modeste travail.

Nous tenons à exprimer notre profonde gratitude à Dr. Hadj Ameur Mohamed, Aouichat Asma, Dr. Berkani Lamia et Prof. Gessoum Ahmed pour leur précieuse guidance, leurs conseils avisés et leur soutien constant tout au long de ce projet. Leurs encouragements et leur rigueur scientifique ont été essentiels à la progression de ce travail.

Nous remercions également l'ensemble des enseignants et des membres du corps académique de l'USTHB pour leur enseignement et leur disponibilité. Leur expertise et leurs encouragements ont largement contribué à enrichir nos connaissances et à développer nos compétences.

Nos sincères remerciements vont également à tous les membres du jury pour le temps et l'attention qu'ils ont consacrés à la lecture de notre mémoire et pour leur engagement à évaluer notre travail.

Enfin, merci à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce modeste travail.

À tous, merci du fond du cœur.

# Dédicace

*Avant tout, je tiens à exprimer ma gratitude envers Allah le Tout-Puissant pour m'avoir accordé la santé, la volonté et le courage nécessaires à la réalisation de ce travail.*

*Je dédie ce modeste travail : À mes chers parents, pour leurs conseils, leurs encouragements et leur soutien indéfectible. À mes sœurs et à mon frère, pour leur présence et leur soutien constant.*

*Un merci particulier à tous mes proches pour leur appui inestimable.  
À tous ceux qui ont contribué à ce travail, je témoigne de ma reconnaissance, de mon respect et de ma gratitude.*

***Kenza***

# Dédicace

*À la mémoire de mon père,*

*À ma mère, pour son amour inconditionnel, son soutien indéfectible tout au long  
de ce parcours.*

*À ma sœur Nouha, ma première supportrice, sans qui je ne serais pas arrivée là où  
je suis aujourd’hui.*

*À ma sœur Lilia et mon frère Moanis.*

*À tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.*

***Rim***

## Résumé

L'expansion rapide des médias sociaux a facilité la diffusion massive des discours haineux, amplifiant ainsi leur portée et leur impact. Cette prolifération souligne l'urgence de systèmes efficaces de détection et de modération, en particulier dans les régions arabophones. La diversité linguistique de l'arabe, qui englobe de nombreux dialectes, et la rareté des ensembles de données annotées compliquent encore davantage ces efforts. Pour répondre à cette problématique, nous avons appliqué plusieurs approches basées sur les modèles de machine learning (ML), deep learning (DL) et d'approches hybrides pour détecter les discours haineux en arabe, en se concentrant spécifiquement sur le dialecte algérien. Nous avons aussi expérimenté avec divers modèles préentraînés pour améliorer les performances de détection. Des expérimentations et des tests comparatifs approfondis ont été menés pour évaluer l'efficacité de ces méthodes, fournissant des informations sur les meilleures pratiques pour la détection des discours haineux.

**Mots clés :** Détection des discours de haine, TALN, Apprentissage Automatique, Apprentissage Profond, Dialecte algérien, Arabe

## **Abstract**

The rapid expansion of social media has facilitated the widespread of hate speech, amplifying its reach and impact. This proliferation underscores the urgent need for effective detection and moderation mechanisms, particularly in Arabic-speaking regions. The linguistic diversity of Arabic, encompassing numerous dialects, and the scarcity of labeled datasets further complicate these efforts. This study addresses these critical issues by applying a comprehensive range of machine learning (ML) baselines, deep learning (DL) models, and hybrid approaches to detect hate speech in Arabic, with a specific focus on the Algerian dialect. We also experimented with various pretrained models to enhance detection performance. Experiments and comparative tests were conducted to evaluate the effectiveness of these methods, providing insights into the best practices for hate speech detection.

**Keywords :** Hate Speech Detection, NLP, Machine Learning, Deep Learning, Algerian Dialect, Arabic

# Table des matières

<b>Liste des tableaux</b>	v
<b>Table des figures</b>	vi
<b>Abréviations et acronymes</b>	viii
<b>Introduction Générale</b>	1
<b>1 TALN et Apprentissage Automatique</b>	3
1.1 Introduction . . . . .	3
1.2 Le traitement du langage naturel (TALN) . . . . .	3
1.2.1 Définition . . . . .	3
1.2.2 Applications de traitement du langage naturel . . . . .	4
1.2.3 Traitement de la langue naturelle Arabe . . . . .	4
1.3 Approches d'apprentissage automatique . . . . .	6
1.3.1 Définition . . . . .	6
1.3.2 Apprentissage supervisé . . . . .	6
1.3.3 Apprentissage non supervisé . . . . .	8
1.3.4 Apprentissage par renforcement . . . . .	8
1.4 Approches d'apprentissage profond . . . . .	8
1.4.1 Définition . . . . .	8
1.4.2 Apprentissage profond vs Réseaux de neurones . . . . .	8
1.4.3 Les réseaux de neurones convolutifs (CNN) . . . . .	9
1.4.4 Les réseaux de neurones récurrents (RNN) . . . . .	10
1.4.5 Les réseaux de neurones à mémoire à court terme (LSTM) . . . . .	10
1.4.6 Les unités récurrentes à portes (GRU) . . . . .	11
1.5 Transformers . . . . .	11
1.5.1 Mécanisme d'attention . . . . .	12
1.5.2 Encodeur . . . . .	12
1.5.3 Décodeur . . . . .	13
1.6 Large Language Models (LLMs) . . . . .	13
1.6.1 Définition . . . . .	13
1.6.2 Architecture . . . . .	13
1.6.3 Entraînement des LLMs . . . . .	15
1.6.4 Le fine-tuning des LLMs . . . . .	16
1.7 Conclusion . . . . .	17

<b>2 Travaux liés</b>	<b>18</b>
2.1 Introduction . . . . .	18
2.2 Introduction au discours de haine . . . . .	18
2.3 Importance de la détection du discours haineux en arabe . . . . .	19
2.4 Le dialecte Algérien . . . . .	20
2.5 Recherches et approches existantes . . . . .	20
2.5.1 Approches précédentes de la détection du discours haineux en arabe . . . . .	20
2.5.2 Approches précédentes de la détection du discours haineux en dialecte algérien . . . . .	21
2.5.3 Ensembles de données pour le discours haineux en arabe . . . . .	22
2.6 Conclusion . . . . .	24
<b>3 Conception</b>	<b>25</b>
3.1 Introduction . . . . .	25
3.2 Architecture générale . . . . .	25
3.3 Prétraitement Commun . . . . .	26
3.3.1 Nettoyage des données . . . . .	27
3.3.2 Détection des langues . . . . .	28
3.3.3 Traitement des abréviations . . . . .	28
3.3.4 Encodage phonétique . . . . .	29
3.3.5 Correction des mots . . . . .	30
3.3.6 Traduction . . . . .	31
3.3.7 Translittération . . . . .	31
3.4 Approches proposées . . . . .	31
3.4.1 Machine Learning baselines . . . . .	31
3.4.2 Approches basées sur l'apprentissage profond . . . . .	34
3.4.3 Approches basées sur les modèles pré-entraînés (Transformers) . . . . .	36
3.4.4 Approches hybrides . . . . .	39
3.4.5 Conclusion . . . . .	40
<b>4 Réalisation et résultats</b>	<b>41</b>
4.1 Introduction . . . . .	41
4.2 Présentation de l'environnement . . . . .	41
4.2.1 Google Colaboratory . . . . .	41
4.2.2 Langage (python) . . . . .	42
4.2.3 Bibliothèques . . . . .	42
4.3 Expérimentations . . . . .	44
4.3.1 Description des données expérimentales . . . . .	44
4.3.2 Mesures de performance pour l'évaluation . . . . .	48
4.3.3 Evaluation . . . . .	48
4.4 Analyse et discussion de résultats . . . . .	53
4.4.1 Performance . . . . .	53
4.4.2 Entraînement et Temps d'exécution . . . . .	55
4.4.3 Analyse des erreurs . . . . .	57
4.4.4 Comparaison de résultats . . . . .	60
4.5 Présentation de l'application . . . . .	65
4.6 Conclusion . . . . .	67



# Liste des tableaux

1.1	Comparaison des différents LLMs . . . . .	15
2.1	Les corpus de discours haineux en arabe . . . . .	24
3.1	Extrait de tableau des abréviations . . . . .	28
4.1	Répartition d'ensemble des données . . . . .	44
4.2	Répartition des classes dans les ensembles d'entraînement et de test dans Algerian dialect toxicity speech dataset . . . . .	45
4.3	Répartition des ensembles d'entraînement et de test dans OffensEval2020 dataset . . . . .	46
4.4	Exemple de commentaire généré par le modèle AraGPT-2 . . . . .	46
4.5	Performances des modèles ML sur les ensembles de données . . . . .	49
4.6	Performances des modèles DL sur les ensembles de données . . . . .	50
4.7	Comparaison des différents LLMs utilisés . . . . .	51
4.8	Métriques de performance pour différents datasets et LLMs . . . . .	52
4.9	Métriques de performance pour différents datasets et modèles . . . . .	53
4.10	Configuration d'entraînement pour différents modèles sur les deux ensembles de données Algerian Dialect Toxicity et OffensEval2020 . . . . .	55
4.11	Temps d'entraînement pour différents modèles sur les deux ensembles de données . . . . .	57
4.12	Prédictions d'AraBERTv02 et classes réelles des faux positifs provenant de l'ensemble de test Algerian Dialect Toxicity Speech . . . . .	58
4.13	Prédictions d'AraBERTv02 et classes réelles des faux positifs provenant de l'ensemble de test OffensEval 2020 . . . . .	59
4.14	Prédictions d'AraBERTv02 et classes réelles des faux négatifs provenant de l'ensemble de test OffensEval2020 . . . . .	60
4.15	Résultats des classificateurs ML . . . . .	60
4.16	Performance of DL Classifiers using FastText Word Embeddings . . . . .	61
4.17	Les résultats obtenus par Alami et al. sur le dataset OffensEval 2020 Arabe . . . . .	61
4.18	Performance moyenne de chaque modèle avec la validation croisée. . . . .	62
4.19	Performance de la combinaison de systèmes pour l'ensemble de test officiel en arabe . . . . .	62
4.20	Résultats de F1 score pour la langue arabe Sub-Task A . . . . .	63
4.21	Performance de classification avec différentes caractéristiques et modèles . . . . .	63
4.22	Performance des différents classificateurs avec Mazajak embeddings . . . . .	64
4.23	Résultats de F1-score des trois meilleures équipes du concours et Flan-T5-large . . . . .	64
4.24	Tableau comparatif des résultats des meilleurs modèles . . . . .	64

# Table des figures

1.1	Relation entre le IA, ML, DL, NLP et LLM . . . . .	6
1.2	Un exemple de modèle SVM pour des vecteurs de caractéristiques bidimensionnels . . . . .	7
1.3	Feed forward neural network . . . . .	9
1.4	Les couches des réseaux de neurones convolutifs [15] . . . . .	9
1.5	Réseaux de neurones récurrents [17] . . . . .	10
1.6	Cellule de réseaux de neurones à mémoire à court terme[19] . . . . .	10
1.7	Cellule de réseaux de neurones à portes[21] . . . . .	11
1.8	La structure encodeur-décodeur de l'architecture Transformer [22] . . . . .	12
1.9	L'architecture de BERT pour le pre-training et le Fine-Tuning [24] . . . . .	14
1.10	Processus de fine-tuning . . . . .	16
3.1	Architecture générale . . . . .	26
3.2	Prétraitement pipeline . . . . .	27
3.3	L'algorithme de distance de Levenshtein . . . . .	30
3.4	Architecture CNN . . . . .	35
3.5	Architecture modèle Bi-GRU . . . . .	36
3.6	Architecture modèle Bi-LSTM . . . . .	36
3.7	Architecture du modèle hybride CNN+Bi-GRU . . . . .	39
3.8	Architecture du modèle hybride AraBERT+CNN . . . . .	40
3.9	Architecture du modèle hybride AraBERT+SVM . . . . .	40
4.1	Logos des différentes bibliothèques Python utilisées . . . . .	42
4.2	Répartition des classes discours haineux et non discours haineux dans le dataset Algerian dialect toxicity speech. . . . .	45
4.3	La distribution des classes dans l'ensemble d'entraînement OffessEval2020 avant l'augmentation des données . . . . .	47
4.4	La distribution des classes dans l'ensemble d'entraînement OffessEval2020 après l'augmentation des données . . . . .	47
4.5	Courbes de l'exactitude et de perte en fonction des époques pour le modèle CNN+GRU utilisant le dataset Algerian Toxicity speech pour 2 époques. . . . .	56
4.6	Courbes de l'exactitude et de perte en fonction des époques pour le modèle CNN+GRU utilisant le dataset Algerian Toxicity speech pour 10 époches . . . . .	56
4.7	Matrice de confusion de AraBERTv02 avec Algerian Dialect Toxicity Speech . . . . .	58
4.8	Matrice de confusion de AraBERTv02 avec OffensEval2020 . . . . .	59
4.9	Interface de détection des discours haineux . . . . .	65
4.10	Illustration de l'analyse d'un commentaire en arabe contenant un discours haineux . . . . .	65

4.11 Illustration de l'analyse d'un commentaire en dialecte algérien qui ne contient pas de discours haineux. . . . .	66
4.12 Illustration de l'analyse d'un commentaire écrit avec Arabizi en dialecte algérien qui ne contient pas de discours haineux. . . . .	66

# Abréviations et acronymes

<b>IA</b>	Intelligence Artificielle
<b>TALN</b>	Traitement Automatique du Langage Naturel
<b>MSA</b>	Modern Standard Arabic
<b>ML</b>	Machine Learning
<b>SVM</b>	Support vector machine
<b>RF</b>	Random Forest
<b>SGD</b>	Stochastic Gradient Descent
<b>TF-IDF</b>	Term Frequency-Inverse Document Frequency
<b>CBow</b>	Continuous Bag Of Words
<b>DL</b>	Deep Learning
<b>CNN</b>	Convolutional Neural Network
<b>RNN</b>	Recurrent Neural Network
<b>LSTM</b>	Long Short Term Memory
<b>Bi-LSTM</b>	Bidirectional Long Short-term Memory
<b>GRU</b>	Gated Recurrent Unit
<b>Bi-GRU</b>	Bidirectional Gated Recurrent Unit
<b>LLM</b>	Large Language Model
<b>PEFT</b>	Parameter-Efficient Fine-Tuning
<b>Acc</b>	Accuracy
<b>API</b>	Application Programming Interface
<b>GPU</b>	Graphics Processing Unit
<b>TPU</b>	Tensor Processing Unit
<b>RAM</b>	Random Access Memory

# Introduction Générale

Dans le monde numérique d'aujourd'hui, le discours de haine est devenu une menace croissante pour la cohésion sociale et la sécurité. Avec l'augmentation exponentielle de l'utilisation des réseaux sociaux et des plateformes en ligne, le contenu haineux peut se propager rapidement, affectant de nombreuses personnes et communautés, il peut provoquer des dommages sociaux significatifs, notamment en attisant les tensions et les conflits, en perpétuant les préjugés et en causant des torts psychologiques aux victimes. Ce phénomène pose des défis importants en matière de régulation et de modération des contenus, particulièrement dans les langues à ressources limitées telles que l'arabe et ses dialectes. La capacité de détecter et de gérer efficacement ces discours de haine est essentielle pour maintenir un environnement en ligne sain et sécurisé.

Le traitement de la langue arabe présente des défis uniques en raison de sa complexité morphologique, et le manque relatif de ressources annotées disponibles pour la recherche et le développement d'outils de traitement automatique du langage naturel (TALN). La langue arabe est caractérisée par une riche morphologie, comprenant une grande variété de formes verbales et nominales, des systèmes de racines et de patrons, et des procédés de dérivation complexes. Ces caractéristiques rendent le traitement automatique de la langue particulièrement difficile.

En outre, la diversité dialectale de l'arabe ajoute une couche supplémentaire de complexité. Chaque région arabe utilise un dialecte spécifique qui peut différer considérablement de l'arabe standard moderne (MSA). Ces dialectes ne sont pas seulement des variations mineures de l'MSA, mais peuvent avoir des vocabulaires, des grammaires et des phonologies distinctes. Parmi ces dialectes, le dialecte algérien qui pose un défi particulier en raison de son mélange unique d'influences linguistiques berbères, arabes, françaises et autres. Cela complique la tâche de création d'outils TALN qui doivent être adaptables à cette diversité linguistique.

Cependant, il existe certaines limitations dans la recherche actuelle. Par exemple, certaines études se contentent de tester des algorithmes traditionnels d'apprentissage automatique (ML) et d'apprentissage profond (DL) sans exploiter les modèles pré-entraînés plus avancés. De plus, certaines recherches sur le discours de haine en Algérie n'utilisent que l'arabe standard sans prendre en compte l'usage de l'arabizi, une forme hybride de l'arabe écrit en caractères latins, fréquemment utilisée dans les communications en ligne. Cette omission peut réduire l'efficacité des modèles de détection dans des contextes réels où l'arabizi est courant.

L'objectif de ce projet est de développer des méthodes de TALN et d'apprentissage automatique pour améliorer la détection du discours de haine en arabe, avec un focus particulier sur le dialecte algérien. Nous proposons d'exploiter des modèles pré-entraînés, et de les affiner (fine-tuning) pour répondre aux spécificités de la langue arabe. Cette approche permet de surmonter les limitations

dues au faible volume de données annotées en utilisant des modèles puissants et déjà bien entraînés sur de grandes quantités de données.

Notre mémoire sera organisé comme suit :

- **Chapitre 01 : TALN et Apprentissage Automatique**

Ce premier chapitre explore le Traitement Automatique du Langage Naturel (TALN) et l'Apprentissage Automatique, en introduisant leurs concepts fondamentaux, applications, et défis spécifiques au traitement de la langue arabe.

- **Chapitre 02 : État de l'art**

Ce chapitre présente une revue de la littérature existante et des travaux précédents liés au sujet du projet.

- **Chapitre 03 : Conception**

Ce chapitre détaille la conception du système ou de la solution proposée, y compris l'architecture, les algorithmes et les modèles utilisés.

- **Chapitre 04 : Réalisation et résultats**

Ce chapitre décrit la mise en œuvre concrète du projet, les expérimentations réalisées, ainsi que les résultats obtenus et leur analyse.

# Chapitre 1

## TALN et Apprentissage Automatique

### 1.1 Introduction

Ce premier chapitre se consacre à deux domaines interconnectés et en pleine expansion : le Traitement Automatique du Langage Naturel (TALN) et l'apprentissage automatique. Ces technologies sont au cœur de nombreuses innovations récentes en intelligence artificielle et jouent un rôle crucial dans la manière dont les machines interagissent avec les données textuelles et linguistiques. La première section du chapitre introduit les concepts fondamentaux du TALN, en commençant par une définition claire et en explorant les diverses applications de cette technologie. Une attention particulière est portée au traitement de la langue arabe, illustrant les défis spécifiques à cette langue. Ensuite, les méthodes d'apprentissage supervisé et non supervisé sont abordées. Ces approches constituent les bases de l'apprentissage automatique et sont essentielles pour comprendre les techniques les plus avancées abordées ultérieurement. Les techniques modernes d'apprentissage profond sont ensuite approfondies, notamment les réseaux de neurones convolutifs (CNN), les réseaux de neurones récurrents (RNN), les réseaux de neurones à mémoire à court terme (LSTM), les unités récurrentes à portes (GRU) et les transformers. Chacune de ces architectures est détaillée pour montrer comment elles contribuent à améliorer les capacités des systèmes de TALN. Enfin, le chapitre met en lumière les modèles de langage de grande taille (LLMs), leur architecture, leur processus d'entraînement, et les techniques de fine-tuning. Les LLMs représentent une avancée majeure dans le domaine de l'IA, offrant des capacités de compréhension et de génération de texte sans précédent.

### 1.2 Le traitement du langage naturel (TALN)

#### 1.2.1 Définition

Le Traitement Automatique du Langage Naturel (TALN) constitue une subdivision de l'informatique, et plus précisément du domaine de l'Intelligence Artificielle (IA). Cette discipline se concentre sur l'objectif de doter les ordinateurs de la capacité de comprendre les textes et les discours de la même manière que les humains. Actif depuis plus de 50 ans, le TALN combine les principes de linguistique, de statistiques, ainsi que d'apprentissage automatique et profond afin d'automatiser le traitement et la compréhension du langage humain sous sa forme textuelle. [1]

## 1.2.2 Applications de traitement du langage naturel

- **Détection des spams** : un filtre anti-spam est un programme utilisé pour détecter les courriels non sollicités, non désirés et infectés par des virus et empêcher ces messages d'atteindre les boîtes de réception. Les meilleures technologies de détection du spam utilisent les capacités de classification de texte du TALN pour analyser les courriels à la recherche d'un langage qui indique souvent la présence de spam ou de phishing.
- **Analyse des sentiments** : elle consiste à identifier les informations subjectives dans un texte afin d'extraire l'opinion et les sentiments de l'auteur.
- **Traduction automatique** : processus par lequel un ordinateur traduit un texte d'une langue à une autre, par exemple de l'anglais à l'arabe, sans intervention humaine.
- **Résumés de textes** : il s'agit d'une tâche de traitement du langage naturel qui consiste à condenser un long document textuel en résumés, en une version plus compacte tout en conservant les informations et le sens les plus importants, et à le présenter dans un format lisible par l'homme.
- **Les agents virtuels et les chatbots** : comme Siri d'Apple et Alexa d'Amazon, utilisent la reconnaissance vocale pour identifier des modèles dans les commandes vocales. Ils utilisent la génération de langage naturel pour proposer des actions appropriées ou des réponses utiles. Les méthodes TALN sont au cœur du fonctionnement des chatbots d'aujourd'hui, qui peuvent facilement gérer des tâches standard telles qu'informer les clients sur les produits ou les services, répondre à leurs questions, etc. Ils sont utilisés sur de multiples canaux, notamment l'internet, les applications et les plateformes de messagerie.

## 1.2.3 Traitement de la langue naturelle Arabe

L'arabe est la langue officielle de 25 pays, plus de 400 millions de personnes dans le monde parlent cette langue. L'arabe est également reconnu comme la quatrième langue la plus utilisée sur internet[2]. La variété et la complexité de la langue arabe posent certaines difficultés aux chercheurs en TALN arabe pour détecter les discours haineux sur les réseaux sociaux.

### 1. Caractéristiques de la langue arabe

- **L'orthographe arabe** : l'orthographe étudie le système d'écriture d'une langue, comprenant des symboles et des règles d'utilisation. En arabe, l'orthographe présente une certaine complexité du fait de ses 28 lettres, un nombre comparable à celui de l'anglais. Contrairement aux langues latines, l'arabe s'écrit de droite à gauche et utilise des diacritiques pour représenter les voyelles, facilitant la prononciation et résolvant les ambiguïtés lexicales et sémantiques permettant ainsi une compréhension plus précise du texte. Les marques diacritiques fréquemment rencontrées incluent فتحة (Fathah), ضمة (Dammah) et كسرة (Kasrah).
- **La morphologie arabe** : la morphologie étudie la structure interne des mots et comment les morphèmes se combinent pour former de nouveaux mots. En arabe, la morphologie comprend l'infexion et la dérivation. L'infexion crée de nouveaux mots en ajoutant des morphèmes qui expriment des propriétés grammaticales. L'arabe est une langue fortement inflexionnelle, avec deux genres, trois nombres et trois temps verbaux. La morphologie nominale arabe est plus complexe que la verbale, incluant le genre, le nombre, l'état et le cas. La dérivation, quant à elle, consiste à créer de nouveaux mots à partir de mots existants en ajoutant des préfixes,

des suffixes ou des infixes. Ces nouveaux mots appartiennent généralement à une catégorie grammaticale différente de celle du mot de base.

- **La syntaxe arabe :** la syntaxe étudie les mécanismes responsables de l'organisation des mots au sein d'une langue pour créer des phrases significatives. En arabe, la syntaxe est fortement liée à la morphologie, plusieurs aspects syntaxiques étant exprimés à la fois par l'ordre des mots et par la morphologie. L'arabe comprend deux types de phrases : verbales et nominales, avec des structures de phrases différentes de l'anglais. Les phrases nominales en arabe commencent par un nom et peuvent avoir un sujet et un prédicat, tandis que les phrases verbales commencent par un verbe et suivent généralement un ordre Verbe-Sujet-Objet.

## 2. Défis dans le traitement du texte arabe

Les complexités et les caractéristiques de la langue arabe ont rendu le traitement du texte arabe un défi majeur, où les chercheurs doivent faire face à plusieurs difficultés, telles que :

- **Variation dialectale :** une question majeure dans l'identification du discours haineux en arabe est la grande variation dialectale. Par exemple, contrairement à l'arabe standard moderne (MSA), qui est écrit et parlé avec une orthographe standardisée, les dialectes sont souvent parlés et rarement écrits. Cela conduit à de vastes variations dans l'orthographe et la grammaire, lorsque ces dialectes sont transcrits dans la communication numérique. Ces types de variations présentent de réels défis pour les systèmes automatiques de détection du discours haineux, car elles sont subtiles et sensibles, et donc échappent aux modèles linguistiques habituels.
- **Code-switching :** il a été prouvé que la langue arabe est une langue où le code-switching est largement utilisé, alternant entre l'arabe standard moderne (MSA) et les dialectes locaux, ou même mélangeant l'arabe avec d'autres langues comme l'anglais et le français. Cela pourrait également rendre les problèmes de code-switching parmi les arabophones potentiellement bien plus graves par rapport à d'autres langues en ce qui concerne les systèmes de traitement automatique du langage, qui doivent identifier et interpréter avec précision plusieurs systèmes linguistiques en une seule conversation ou texte.  
Exemple : "Trop tard Je ne vous supporte pas" روحی کرھنا منک انتی ماکیش انسان کامل
- **Facteurs sociolinguistiques :** la limite de ce qui constitue un discours haineux peut varier considérablement selon les contextes sociolinguistiques. En effet, certains propos peuvent être considérés comme offensants dans une communauté ou un dialecte particulier, mais pas dans un autre.
- **Ressources limitées :** par rapport à l'anglais, l'arabe dispose de moins de ressources pour le TAL, y compris les corpus annotés et les modèles de langage avancés. Cette pénurie de ressources entrave le développement d'outils efficaces pour détecter le discours haineux dans les dialectes arabes.
- **Ambiguïté et sarcasme :** l'utilisation du sarcasme et du langage ambigu peut obscurcir l'intention derrière une déclaration. Détecter de telles nuances dans le texte, en particulier à travers divers dialectes, ajoute une autre couche de complexité à la reconnaissance du discours

haineux.

Aborder les difficultés dans les outils de TALN tels que la tokenisation, la lemmatisation et l'analyse morphologique est encore un autre défi auquel sont confrontés les chercheurs et les développeurs. Ces problèmes ont été largement étudiés et ont fait l'objet de recherches [3][4][5][6].

## 1.3 Approches d'apprentissage automatique

### 1.3.1 Définition

L'apprentissage automatique est une branche de l'intelligence artificielle (IA) qui se concentre sur la construction de systèmes informatiques qui apprennent à partir de données en imitant la façon dont les humains apprennent. Le large éventail de techniques d'apprentissage automatique permet aux applications et logicielles d'améliorer leurs performances au fil du temps.

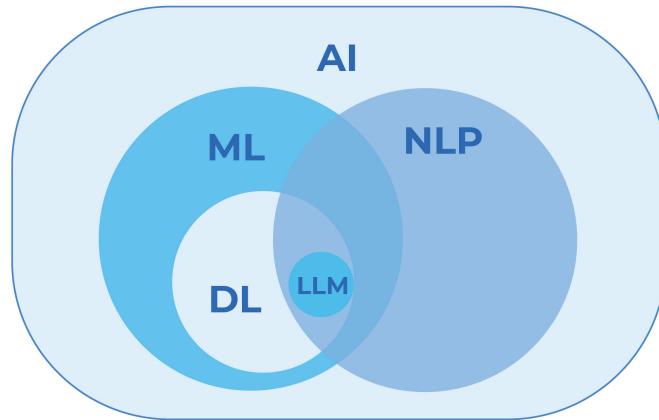


FIGURE 1.1 – Relation entre le IA, ML, DL, NLP et LLM

### 1.3.2 Apprentissage supervisé

L'apprentissage supervisé est une branche de l'apprentissage automatique qui permet aux algorithmes d'apprendre à partir d'exemples. Ces exemples, appelés données d'apprentissage, sont constitués de paires d'entrées et de sorties. L'algorithme analyse ces paires pour identifier les relations entre les entrées et les sorties, et il apprend à prédire les sorties pour de nouvelles entrées qu'il n'a jamais vues auparavant. Voici quelques exemples de modèles supervisés :

- **Naïve Bayes** : les classificateurs de bayes naïfs sont un ensemble d'algorithmes d'apprentissage supervisé basés sur l'application du théorème de Bayes avec l'hypothèse "naïf" de l'indépendance conditionnelle entre chaque paire de caractéristiques compte tenu de la valeur de la variable de classe. Les Naïves bayes sont utilisés pour des tâches de classification, telles que la classification de textes et le filtrage de spam.[7]
- **Régression linéaire** : la régression linéaire consiste à estimer des valeurs réelles (telles que le coût des maisons, le nombre d'appels, etc.) en fonction d'une variable continue ou de plusieurs variables

continues. Pour ce faire, il est nécessaire d'établir la relation entre les variables indépendantes et dépendantes en ajustant la meilleure ligne, également appelée ligne de régression, qui est représentée par une équation linéaire.[8]

- **Support vector machines (SVM)** : le SVM est un classificateur discriminant qui utilise un hyperplan comme ligne de séparation entre les données. Il cherche à trouver un ou plusieurs hyperplans optimaux qui divisent les ensembles de données en fonction de leur classe. Dans un espace à deux dimensions, l'hyperplan sépare le plan en deux sections avec chaque classe de chaque côté. La marge, qui est la distance entre les surfaces de décision, sépare généralement les points de données des différentes classes. La surface de décision est une ligne reliant les points de données les plus proches de la limite de classe, appelés vecteurs de support, comme le montre la figure 1.2.[9]

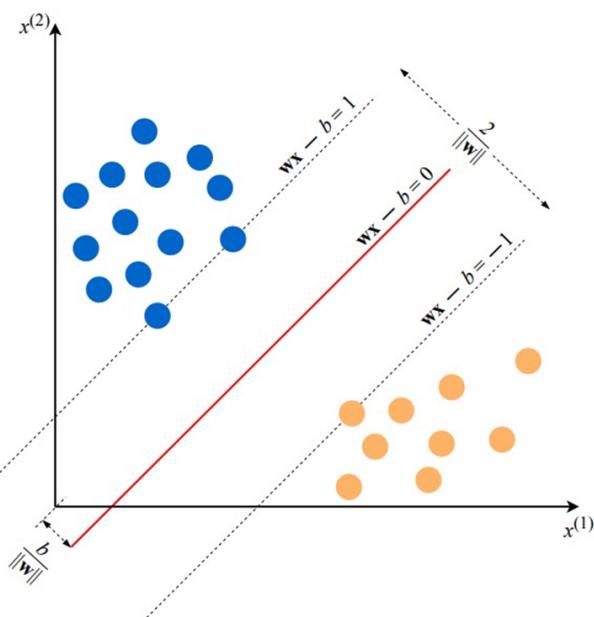


FIGURE 1.2 – Un exemple de modèle SVM pour des vecteurs de caractéristiques bidimensionnels

- **Les forêts aléatoires (Random Forest)** : une forêt aléatoire est un méta-estimateur qui construit plusieurs classificateurs d'arbres de décision, sur différents sous-échantillons du jeu de données, et utilise la moyenne, pour améliorer la précision prédictive et réduire le surapprentissage. Chaque arbre dans la forêt est construit en utilisant la meilleure stratégie de division, garantissant que les décisions les plus optimales sont prises à chaque noeud. En combinant les prédictions de plusieurs arbres, une forêt aléatoire améliore la robustesse et la fiabilité du modèle par rapport à un seul arbre de décision.[10]
- **La descente de gradient stochastique (SGD)** : La SGD est une méthode simple mais très efficace pour ajuster des classificateurs linéaires et des régressions sous des fonctions de perte convexes telles que les machines à vecteurs de support (linéaires) et la régression logistique. Bien que la SGD soit utilisée dans la communauté de l'apprentissage automatique depuis longtemps, elle a récemment suscité un intérêt considérable dans le contexte de l'apprentissage à grande échelle. La SGD a été appliquée avec succès à des problèmes d'apprentissage automatique à grande échelle et à des données clairsemées, souvent rencontrés dans la classification de texte et le traitement du langage naturel.[11]

### **1.3.3 Apprentissage non supervisé**

L'apprentissage non supervisé en intelligence artificielle est un type d'apprentissage automatique qui apprend à partir de données sans supervision humaine. Contrairement à l'apprentissage supervisé, l'apprentissage non supervisé utilise des algorithmes pour analyser et regrouper des ensembles de données non étiquetées. Ces algorithmes découvrent des modèles cachés ou des regroupements de données sans intervention humaine. L'apprentissage non supervisé est principalement utilisé pour la modélisation de sujets, les représentations de mots, le regroupement, la détection d'anomalies, la génération de texte, la réduction de dimensionnalité, la modélisation de langage, etc...[12]

### **1.3.4 Apprentissage par renforcement**

L'apprentissage par renforcement est une branche de l'intelligence artificielle où un agent apprend à prendre des décisions en interagissant avec un environnement. À travers un processus d'essai-erreur, l'agent reçoit des récompenses ou des punitions en fonction de ses actions, ce qui l'aide à améliorer ses stratégies pour atteindre des objectifs spécifiques. Ce type d'apprentissage est particulièrement utile dans des situations complexes où il est difficile de programmer explicitement toutes les possibilités.

## **1.4 Approches d'apprentissage profond**

### **1.4.1 Définition**

L'apprentissage profond est un sous-domaine de l'apprentissage automatique qui s'intéresse aux algorithmes inspirés de la structure et de la fonction du cerveau, appelés réseaux neuronaux artificiels. Ces réseaux neuronaux tentent de simuler le comportement de l'être humain, ce qui leur permet d'apprendre à partir de grandes quantités de données.

### **1.4.2 Apprentissage profond vs Réseaux de neurones**

Les termes d'apprentissage profond et de réseaux de neurones sont souvent utilisés de manière interchangeable car tous les systèmes d'apprentissage profond sont constitués de réseaux neuronaux. Cependant, les détails techniques varient. Les réseaux de neurones feedforward, ou réseaux de neurones simples, traitent les données dans une seule direction, de l'entrée à la sortie, comme le montre la figure 1.3. En revanche, les systèmes d'apprentissage profond ont plusieurs couches cachées, ce qui les rend profonds. Les CNN ont des couches convolutionnelles, des couches de regroupement et des couches entièrement connectées, tandis que les RNN sont composés d'unités récurrentes liées en série. Les neurones simples ont moins de paramètres et sont moins complexes que les systèmes d'apprentissage profond, mais ces derniers sont plus adaptés pour les tâches complexes grâce à leur capacité à capturer des dépendances temporelles et à utiliser des autoencodeurs. Enfin, les réseaux de neurones simples sont plus rapides à entraîner mais moins performants pour des tâches complexes, tandis que les systèmes d'apprentissage profond peuvent traiter de vastes volumes de données et sont plus performants pour des tâches telles que le traitement du langage naturel et la reconnaissance vocale.[13]

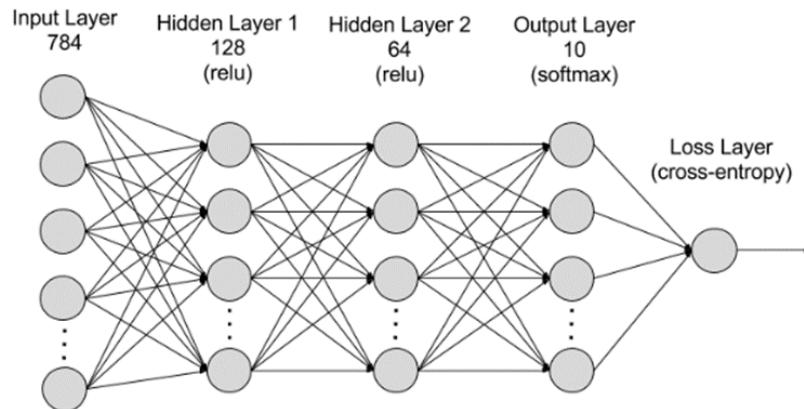


FIGURE 1.3 – Feed forward neural network

#### 1.4.3 Les réseaux de neurones convolutifs (CNN)

Les réseaux neuronaux convolutifs (CNN) sont un type de modèle d'apprentissage profond particulièrement adapté aux tâches impliquant des données structurées, telles que les images, l'audio ou le texte dans le traitement automatique du langage naturel (TALN). Ils sont appelés "convolutifs" en raison de l'utilisation d'une opération mathématique appelée convolution pour traiter les données d'entrée. Dans les tâches de TALN, les données d'entrée sont généralement des phrases ou des documents représentés sous forme de matrice, chaque ligne de la matrice correspondant à un token (généralement un mot, mais aussi un caractère). En plus de la convolution, les CNN utilisent une opération appelée pooling, qui réduit la dimensionnalité des données et extrait les caractéristiques les plus importantes, rendant le modèle plus efficace et moins sensible au bruit (voir figure 1.4). Les CNN sont particulièrement bien adaptés aux tâches de classification telles que l'analyse des sentiments, la détection de spams ou la catégorisation de documents.[14]

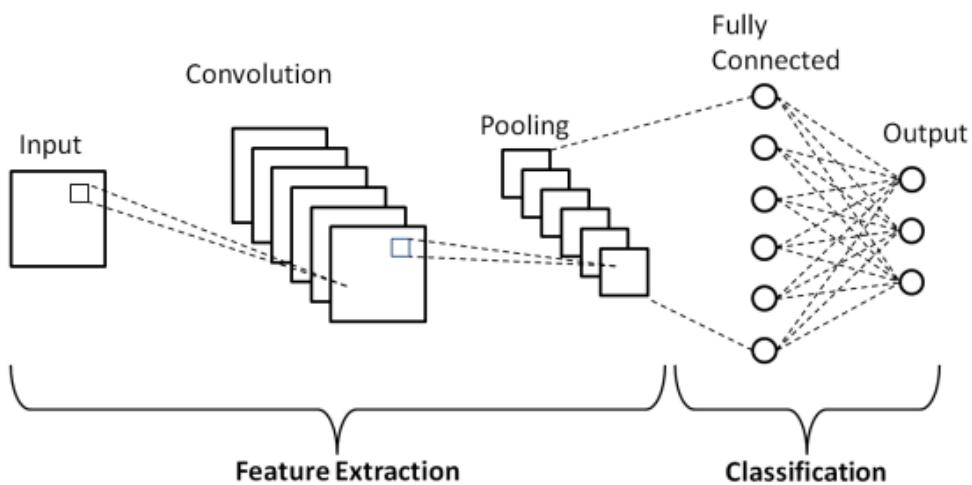


FIGURE 1.4 – Les couches des réseaux de neurones convolutifs [15]

#### 1.4.4 Les réseaux de neurones récurrents (RNN)

Un réseau neuronal récurrent (RNN) est un modèle d'apprentissage profond conçu pour traiter et convertir des données séquentielles en sorties spécifiques. Ces algorithmes sont couramment utilisés pour résoudre des problèmes ordinaux ou temporels, comme la traduction linguistique, le traitement du langage naturel ou la reconnaissance vocale. Les RNN se distinguent par leur capacité à "mémoriser", utilisant des informations provenant des entrées précédentes pour influencer les entrées et sorties actuelles, comme illustré dans la figure 1.5.[16]

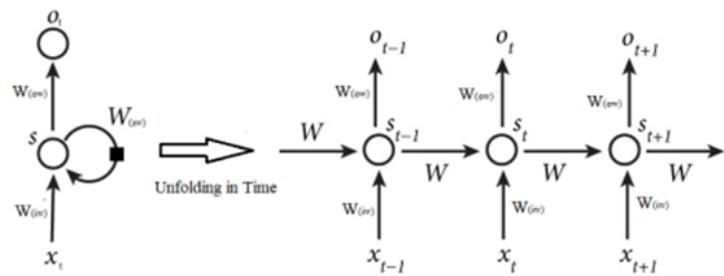


FIGURE 1.5 – Réseaux de neurones récurrents [17]

#### 1.4.5 Les réseaux de neurones à mémoire à court terme (LSTM)

Les réseaux de neurones à mémoire à court terme sont un type de réseau neuronal récurrent capable d'apprendre la dépendance d'ordre dans les problèmes de prédiction de séquence, ce qui le rend adapté aux données textuelles. Chaque unité prend comme entrée le vecteur d'intégration du mot actuel et la sortie de l'unité précédente, mettant à jour sa cellule de mémoire interne, comme le montre la figure 1.6. Ce processus récursif permet d'accumuler des informations sur tous les autres mots du texte. Contrairement aux réseaux neuronaux traditionnels, les LSTM intègrent des connexions de rétroaction, ce qui leur permet de traiter des séquences entières de données, et pas seulement des points de données individuels. Cela les rend très efficace pour comprendre et prédire des modèles dans des données séquentielles telles que des séries temporelles, des textes et des discours.[18]

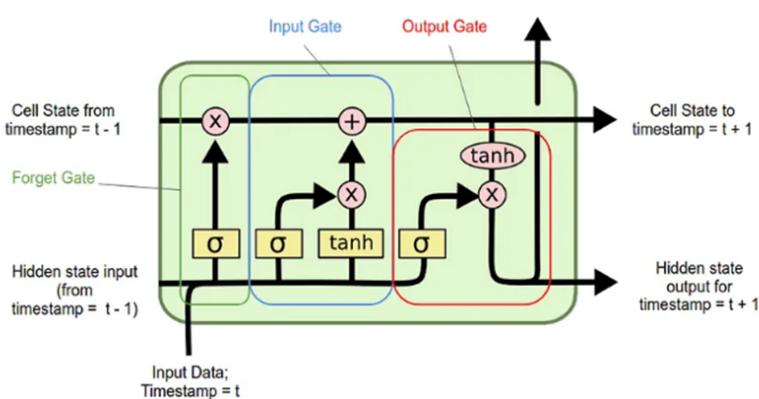


FIGURE 1.6 – Cellule de réseaux de neurones à mémoire à court terme[19]

#### 1.4.6 Les unités récurrentes à portes (GRU)

Le GRU (Gated Recurrent Unit) représente une simplification de l'architecture LSTM, où les portes d'entrée et d'oubli sont combinées en une unique porte de mise à jour, et les connexions pephole ainsi que les fonctions d'activation en sortie sont éliminées (figure 1.7). La fonction de la porte de sortie, également nommée porte de réinitialisation, est ajustée pour ne gérer que la transmission des connexions récurrentes à l'entrée du bloc.[20]

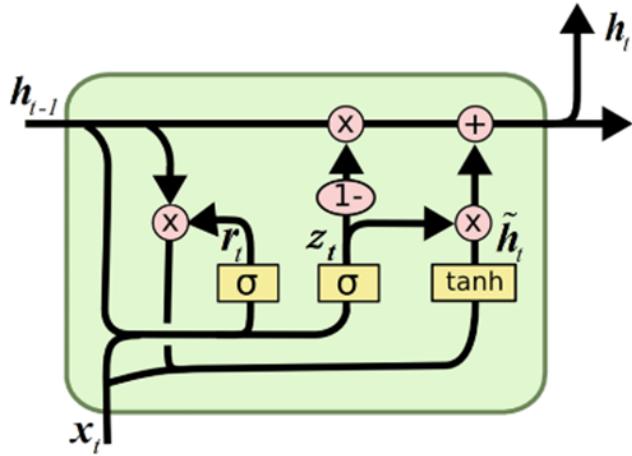


FIGURE 1.7 – Cellule de réseaux de neurones à portes[21]

où :

- $h_t$  : État actuel à l'instant  $t$
- $h_{t-1}$  : État précédent à l'instant  $t - 1$
- $z_t$  : Porte de mise à jour
- $r_t$  : Porte de réinitialisation
- $x_t$  : Entrée

### 1.5 Transformers

Les transformers sont un type d'architecture de réseau neuronal qui transforme ou modifie une séquence d'entrée en une séquence de sortie. Pour ce faire, ils apprennent le contexte et suivent les relations entre les composants de la séquence. Contrairement aux modèles traditionnels basés sur des réseaux neuronaux récurrents, les Transformers ne s'appuient pas sur des connexions séquentielles et sont capables de capturer des relations à long terme dans un texte. L'architecture Transformer a été présentée en juin 2017 dans l'article "Attention Is All You Need" [22], elle suit une structure codeur-décodeur, comme illustré dans la figure 1.8. Le fonctionnement d'un Transformer repose sur deux éléments clés : l'attention et les blocs transformer. L'attention permet au modèle d'attribuer des poids différents aux mots d'une séquence et de concentrer son attention sur les parties les plus pertinentes du texte. Les blocs de transformation sont des couches qui appliquent des transformations non linéaires aux représentations d'entrée et aident le modèle à apprendre des modèles et des structures linguistiques.

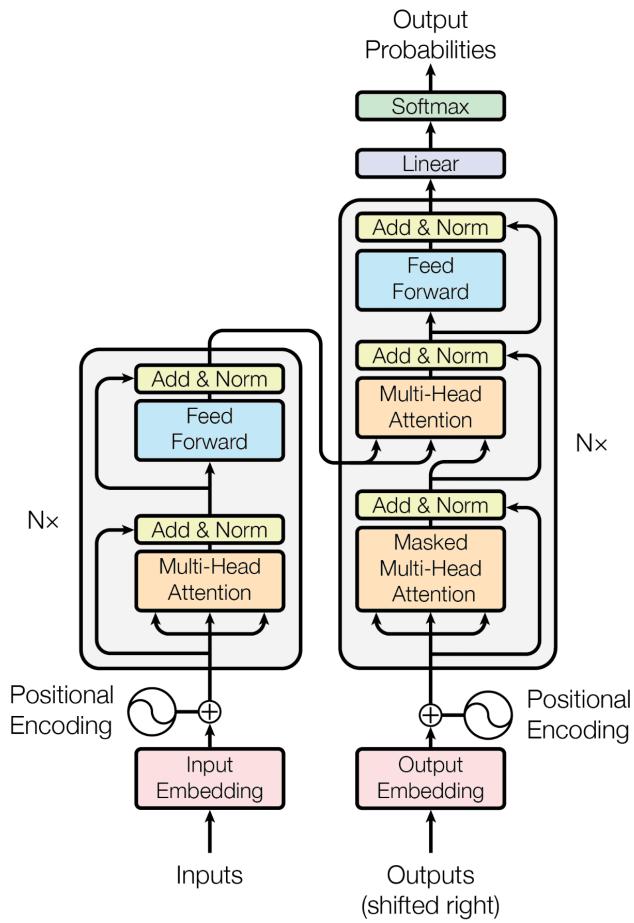


FIGURE 1.8 – La structure encodeur-décodeur de l’architecture Transformer [22]

### 1.5.1 Mécanisme d’attention

Un mécanisme d’attention est une méthode informatique inspirée de la cognition humaine qui aide les modèles d’intelligence artificielle à se concentrer sur des parties spécifiques des données d’entrée lors de leur traitement. L’attention permet à un modèle de se focaliser sur des éléments importants d’une phrase, facilitant ainsi la compréhension du contexte des données textuelles ou de toute autre donnée séquentielle.

Le modèle Transformer a révolutionné la mise en œuvre de l’attention en supprimant la récurrence et les convolutions, en se basant uniquement sur un mécanisme d’auto-attention. Ce mécanisme consiste à examiner différents mots dans une même phrase et à déterminer l’importance de chacun pour en comprendre le sens. Cela aide le modèle à prendre en compte les relations entre les mots dans le texte. [23]

### 1.5.2 Encodeur

L’encodeur consiste en une pile de  $N$  couches identiques, chacune contenant deux sous - couches. La première sous-couche incorpore un mécanisme d’auto-attention à têtes multiples, tandis que la seconde implique un réseau direct entièrement connecté par position. Des connexions résiduelles

entourent chacune de ces deux sous-couches, suivies d'une normalisation des couches. En d'autres termes, la sortie de chaque sous-couche est soumise à  $\text{LayerNorm}(x + \text{Sublayer}(x))$ , où  $\text{Sublayer}(x)$  représente la fonction mise en œuvre par la sous-couche elle-même. Pour faciliter ces connexions résiduelles, toutes les sous-couches du modèle, ainsi que les couches d'intégration, génèrent des sorties de dimension  $d_{\text{model}}$ . [22]

### 1.5.3 Décodeur

De même, le décodeur est composé d'une pile de  $N$  couches identiques. Outre les deux sous-couches de chaque couche du codeur, le décodeur introduit une troisième sous-couche responsable de l'attention multi-têtes sur la sortie de la pile du codeur. Suivant l'approche de l'encodeur, des connexions résiduelles entourent chacune des sous-couches du décodeur, suivies d'une normalisation des couches. En outre, la sous-couche d'auto-attention de la pile du décodeur est modifiée pour empêcher les positions de s'occuper des positions suivantes. Ce masquage, combiné au fait que les encastrements de sortie sont décalés d'une position, garantit que les prédictions pour la position  $i$  reposent uniquement sur les sorties connues aux positions inférieures à  $i$ .[22]

## 1.6 Large Language Models (LLMs)

### 1.6.1 Définition

Un LLM est un modèle d'intelligence artificielle entraîné par des algorithmes d'apprentissage profond pour comprendre, générer, traduire et résumer des volumes importants de langage humain écrit et de données textuelles. Ces modèles font preuve d'une flexibilité remarquable, puisqu'un seul LLM peut exécuter un large éventail de tâches. En outre, les LLM font preuve d'une capacité impressionnante à faire des prédictions sur la base d'invites ou d'entrées minimales, soulignant ainsi leur adaptabilité et leur efficacité.

### 1.6.2 Architecture

Les architectures de Modèles de Langage avec Transformers introduits précédemment, ont redéfini le paysage du Traitement du Langage Naturel grâce à leur conception révolutionnaire. Parmi ces modèles, l'un des plus utilisés en traitement automatique du langage est :

BERT (Bidirectional Encoder Representations from Transformers) : un modèle de transformer développé par une équipe de Google en 2018 [24]. Il est conçu pour pré-entraîner des représentations bidirectionnelles profondes à partir d'un texte non étiqueté en conditionnant conjointement les contextes gauche et droit afin d'avoir une compréhension plus profonde du contexte linguistique. BERT est un transformer bidirectionnel pré-entraîné à l'aide d'une combinaison d'objectifs de modélisation du langage masqué et de prédiction de la phrase suivante sur un large corpus comprenant le Toronto Book Corpus et Wikipedia. BERT, se décline en deux versions : Base et Large. La distinction réside dans le nombre de couches, la taille cachée et le nombre de têtes d'auto-attention. Le modèle peut traiter à la fois une phrase unique et une paire de phrases sous forme de séquence de tokens, ce qui le rend adapté aux tâches de traitement automatique du langage naturel (TALN) sur des données conversationnelles[25].

Contrairement aux modèles Transformer complets, BERT n'utilise que l'encodeur. Ce choix s'explique par la nature des objectifs de pré-apprentissage employés, tels que le modèle de langage masqué (MLM) et la prédiction de la phrase suivante (NSP), qui nécessitent une compréhension contextuelle bidirectionnelle. Le modèle de langage masqué (MLM) consiste à masquer aléatoirement certains tokens dans la phrase d'entrée et à entraîner le modèle à prédire ces tokens masqués, ce qui permet à BERT de comprendre le contexte à la fois à gauche et à droite des mots masqués. La prédiction de la phrase suivante (NSP), quant à elle, implique de donner au modèle deux phrases et de l'entraîner à prédire si la seconde phrase suit logiquement la première. Cette tâche aide BERT à capturer les relations entre les phrases, ce qui est crucial pour des applications telles que la compréhension de texte et les tâches de question-réponse. Ainsi, le décodeur, habituellement employé pour produire des séquences de sortie dans des tâches telles que la traduction automatique, n'est pas intégré dans BERT.

Le processus de fine-tuning de BERT est également crucial pour son efficacité. Après le pré-entraînement sur de vastes corpus de texte, BERT peut être adapté à des tâches spécifiques en ajoutant une ou plusieurs couches de sortie appropriées et en continuant l'entraînement sur un jeu de données plus petit et spécifique à la tâche. Cela permet de tirer parti de la riche représentation du langage apprise durant le pré-entraînement tout en s'ajustant aux particularités de la nouvelle tâche.

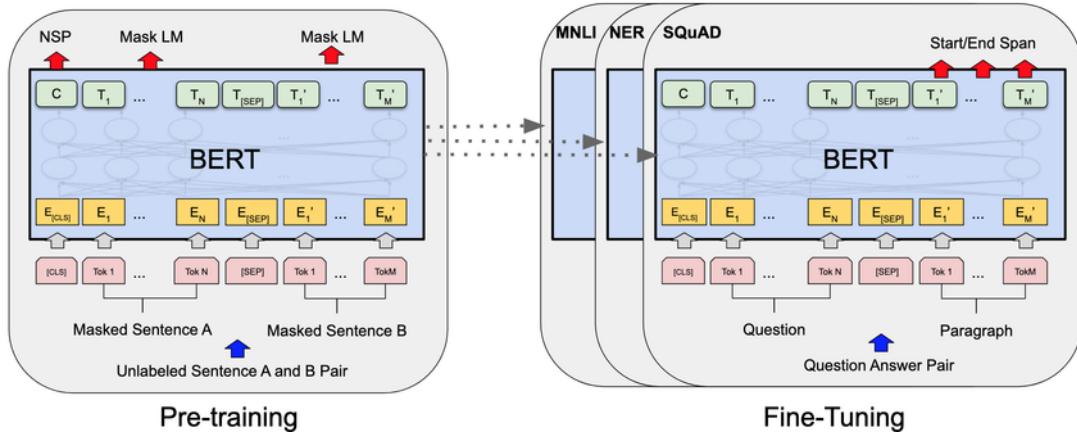


FIGURE 1.9 – L'architecture de BERT pour le pre-training et le Fine-Tuning [24]

Ci-dessous se trouve un tableau comparatif (table 1.1) illustrant plusieurs modèles de langage existants (LLM) :

Model	GPT-3	BERT	T5	GPT-4	LLAMA 3
Year Released	2020	2018	2019	2023	2024
Architecture	Transformer (Dec)	Transformer (Enc)	Transformer (Enc-Dec)	Transformer (Dec)	Transformer (Enc)
Parameters	175 billion	110 million - 340 million	220 million - 11 billion	1 trillion	8 Billion - 70 billion
Training Data	Common Crawl, Books, Wikipedia	Wikipedia, BooksCorpus	C4 (Colossal Clean Crawled Corpus)	Diverse internet text	15T tokens of public text data
Key Features	Generative text, Few-shot learning	Contextual embeddings, NLP tasks	Text-to-text format, Multi-task learning	Generative text, Advanced reasoning	Robust performance
Performance	High on generative tasks	High on understanding tasks	Versatile across multiple tasks	State-of-the-art on generative tasks	High on NLP benchmarks
Use Cases	Content creation, Chatbots	Text classification, Entity recognition	Translation, Summarization	Advanced AI applications, Content creation	AI assistants, Content creation
Open Source	No	Yes	Yes	No	Yes

TABLE 1.1: Comparaison des différents LLMs

### 1.6.3 Entraînement des LLMs

Les réseaux neuronaux basés sur les transformateurs sont très volumineux. Ces réseaux contiennent plusieurs nœuds et couches. Chaque nœud dans une couche est connecté à tous les nœuds de la couche suivante, chacun ayant un poids et un biais. Les poids et les biais, ainsi que les embeddings, sont appelés paramètres du modèle. Les grands réseaux neuronaux basés sur les transformateurs peuvent avoir des milliards de paramètres. La taille du modèle est généralement déterminée par une relation empirique entre le nombre de paramètres et la taille des données d'entraînement. L'apprentissage est réalisé à l'aide d'un corpus volumineux de données, au cours de ce processus, le modèle ajuste de manière itérative les valeurs des paramètres jusqu'à ce que le modèle prédise correctement le token suivant à partir de la séquence précédente de tokens d'entrée. Il le fait à travers des techniques d'auto-apprentissage qui apprennent au modèle à ajuster les paramètres pour maximiser la probabilité des tokens suivants dans les exemples d'entraînement. Une fois entraînés, les LLMs peuvent être facilement adaptés pour effectuer plusieurs tâches en utilisant des ensembles de données supervisées relativement petits, un processus appelé fine-tuning. Trois modèles d'apprentissage courants existent :

- **L'apprentissage zéro-shot** : les LLM de base peuvent répondre à une large gamme de requêtes sans entraînement explicite, souvent par le biais de stimuli, bien que la précision des réponses puisse varier.

- **L'apprentissage few-shot** : en fournissant quelques exemples d'entraînement pertinents, les performances du modèle de base s'améliorent significativement dans ce domaine spécifique.
- **Le fine-tuning** : il s'agit d'une extension de l'apprentissage few-shot, dans laquelle un modèle de base est entraîné pour ajuster ses paramètres avec des données supplémentaires pertinentes pour une application spécifique[26].

#### 1.6.4 Le fine-tuning des LLMs

Le fine-tuning consiste à prendre des modèles pré-entraînés et à les entraîner davantage sur des ensembles de données plus petits et spécifiques afin de peaufiner leurs capacités et d'améliorer leurs performances dans une tâche ou un domaine particulier. Le fine-tuning vise à transformer des modèles polyvalents en modèles spécialisés. Il comble le fossé entre les modèles pré-entraînés génériques et les exigences uniques des applications spécifiques, garantissant que le modèle de langage s'aligne étroitement sur les attentes humaines. Le fine-tuning des LLM implique un processus d'apprentissage supervisé où les poids des LLM sont ajustés en utilisant un ensemble de données d'exemples étiquetés, ce qui permet d'améliorer la capacité du modèle pour des tâches spécifiques. Voici quelques approches de fine-tuning à considérer :

- **Le fine-tuning par instruction** : il consiste à modifier des modèles pré-entraînés en fournissant des instructions spécifiques ou des contraintes pendant le processus de fine-tuning.[27]
- **Le fine-tuning complet** : fait référence au processus de fine-tuning d'un modèle pré-entraîné entier sur de nouvelles données ou tâches sans aucune contrainte ou instruction spécialisée. [28]
- **Le fine-tuning à efficacité paramétrique (PEFT)** : se concentre sur l'optimisation du processus de fine-tuning pour minimiser le nombre de paramètres ajustés dans le modèle pré-entraîné tout en obtenant néanmoins des performances satisfaisantes sur la tâche cible. Des techniques telles que gradual unfreezing ou selective layer freezing sont couramment utilisées pour obtenir une efficacité paramétrique et éviter l'overfitting. [29]

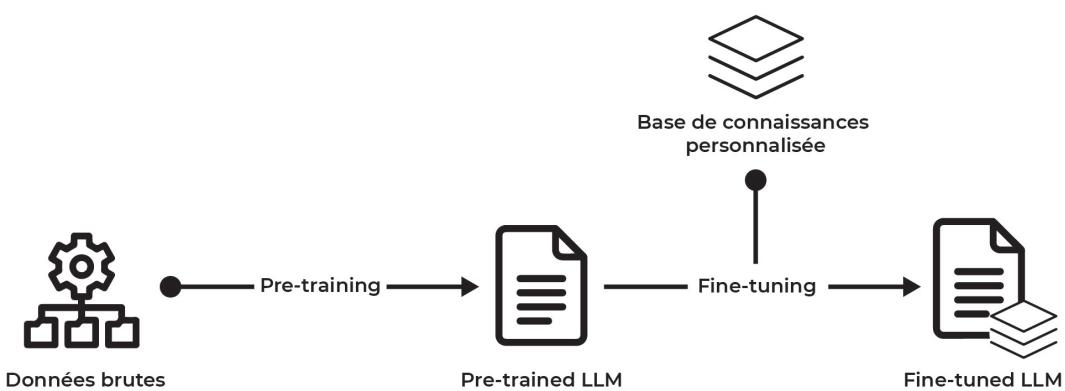


FIGURE 1.10 – Processus de fine-tuning

## 1.7 Conclusion

Ce chapitre a fourni une vue d'ensemble approfondie des principaux concepts et techniques dans le domaine du Traitement Automatique du Langage Naturel et de l'Apprentissage Automatique. En commençant par les bases du TALN et en progressant vers des méthodes d'apprentissage automatique plus avancées, ce chapitre a établi un cadre solide pour comprendre comment ces technologies peuvent être appliquées et optimisées. En résumé, ce chapitre a présenté les concepts clés et les techniques actuelles, et il a également préparé le terrain pour des discussions plus détaillées et des applications pratiques dans les chapitres suivants.

# Chapitre 2

## Travaux liés

### 2.1 Introduction

De nos jours, avec la prolifération des médias sociaux et des plateformes en ligne, la lutte contre le discours haineux est devenue un enjeu majeur. Cependant, cette tâche se heurte à de nombreux défis linguistiques et techniques. La richesse de la langue arabe, ses multiples variantes dialectales, ainsi que le manque de ressources annotées constituent des obstacles majeurs. Ce chapitre dresse un état des lieux des recherches existantes sur la détection du discours haineux en arabe, en mettant l'accent sur le dialecte algérien. Nous aborderons les approches proposées jusqu'à présent ainsi que les ensembles de données disponibles.

### 2.2 Introduction au discours de haine

La définition du discours haineux varie considérablement selon les organisations, les cultures et les systèmes juridiques, reflétant ainsi la diversité des valeurs sociétales et des cadres réglementaires en vigueur. Cette notion complexe ne fait pas l'objet d'un consensus universel, comme en témoignent ces quelques exemples de définitions :

- **Les Nations Unies** : "Dans le langage courant, le « discours de haine » désigne un discours injurieux visant un groupe ou un individu sur la base de caractéristiques intrinsèques (telles que la race, la religion ou le genre) et pouvant menacer la paix sociale"[30].
- **Le Comité des Ministres du Conseil de l'Europe** : "discours public qui exprime la haine ou encourage la violence à l'égard d'une personne ou d'un groupe sur la base d'éléments tels que la race, la religion ou le sexe"[31].
- **Meta** : "Une attaque directe contre des personnes, plutôt que contre des concepts ou des institutions, fondée sur ce que nous appelons des caractéristiques protégées : l'origine ethnique, l'origine nationale, le handicap, la religion, la caste, le sexe, l'identité de genre, et les maladies graves"[32]
- **Youtube** : "13 « attributs » où il est spécifiquement interdit de promouvoir la violence ou la haine contre des individus ou des groupes : « âge, caste, handicap, ethnie, identité et expression de genre, nationalité, race, statut d'immigrant, religion, sexe/genre, orientation sexuelle, victimes d'un événement violent majeur et leurs proches, et statut de vétéran. "[33]
- **X** : "des propos qui ciblent, intimident ou utilisent un langage menaçant ou déshumanisant en raison de caractéristiques protégées." [34]

Bhikhu Parekh[35], a répertorié des exemples de ce que divers pays ont pénalisé ou tenté de pénaliser comme discours de haine. Ces exemples illustrent les approches diverses de la régulation du discours de haine à travers le monde, reflétant les différents cadres juridiques et sensibilités culturelles :

- Crier « [N-words] rentrez chez vous », imiter des bruits de singe, et scander des slogans racistes lors de matchs de football.
- « L'islam hors de Grande-Bretagne. Protégez le peuple britannique. »
- Une affiche représentant une femme en burka avec un texte qui dit : « Qui sait ce qu'elles ont sous leurs vêtements sinistres et laids : des objets volés, des armes, voire des bombes ? »
- « Les Noirs sont naturellement inférieurs, lubriques, prédisposés aux activités criminelles et ne devraient pas être autorisés à s'installer dans des zones respectables. »

Le discours haineux se manifeste souvent sous forme de remarques déplaisantes, d'insultes, de menaces, de stéréotypes dommageables, de micro-agressions, et de la diffusion d'images ou de symboles offensants. L'impact négatif de ce phénomène est amplifié par la facilité d'accès, l'agressivité sous couvert d'anonymat, et la capacité à inciter à des actions concrètes telles que la discrimination et la violence.

## 2.3 Importance de la détection du discours haineux en arabe

Le discours haineux, par sa nature divisive et potentiellement nocive, représente un défi majeur dans les sociétés contemporaines. Sa propagation peut non seulement fragmenter les communautés et aggraver les tensions sociales. La détection et la modération efficaces de ce type de discours revêtent donc une importance cruciale. Voici quelques points clés soulignant l'importance de cette détection :

- **Préservation de la Cohésion Sociale** : le discours haineux peut fragmenter les communautés et exacerber les tensions sociales. Sa détection et sa modération contribuent à maintenir la cohésion et le respect mutuel au sein des sociétés diversifiées.
- **Protection des Groupes Vulnérables** : les minorités ethniques, religieuses et autres groupes vulnérables sont souvent les cibles du discours haineux. Une détection efficace aide à les protéger contre les discours qui pourraient encourager la discrimination ou la violence à leur égard.
- **Inégalités linguistiques** : les langues minoritaires, comme l'arabe, peuvent être sous-représentées dans les systèmes de détection automatique, ce qui peut entraîner une détection inadéquate ou même pas de détection du discours haineux.

Assurer la surveillance et la modération du discours haineux, en particulier en arabe, est devenu indispensable pour endiguer sa diffusion et encourager la création d'un espace virtuel empreint de bienveillance et de respect mutuel.

## 2.4 Le dialecte Algérien

L'analyse du dialecte algérien dans le cadre de la détection des discours haineux présente des défis uniques mais cruciaux pour l'avancement de la recherche dans ce domaine. Le dialecte algérien, avec sa richesse linguistique issue d'un mélange de l'arabe, du tamazight et du français, reflète la complexité culturelle et historique de l'Algérie. Il est également essentiel de reconnaître que le dialecte algérien varie considérablement d'une région à l'autre. Les variations dialectales entre l'est, l'ouest, le nord et le sud de l'Algérie ajoutent une couche supplémentaire de complexité à la détection précise du discours haineux.

Les différences dans le vocabulaire se manifestent avec des expressions comme "Sh7al yaswa hada" pour demander le prix de quelque chose, contre "bgedah hada" dans une autre région.

Un autre exemple est la manière dont on nomme les œufs, avec "El bidh" , "Wled el djedj", et "Le3dham".

En outre, les locuteurs algériens ont tendance à conjuguer des mots français de manière unique, comme dans la phrase "ta9der tconseillili restaurant familial ?" qui veut dire "est-ce que tu peux me conseiller un restaurant familial?", conjuguant "conseiller" en "tconseillili". Ils modifient également des mots français, comme dans "nro7 lekhedma fe tomobil" qui veut dire "je pars au travail avec l'automobile", transformant "automobile" en "tomobil".

Cette diversité linguistique pose des obstacles significatifs pour les algorithmes de détection de discours haineux, souvent conçus autour de langues largement documentées et uniformes.

## 2.5 Recherches et approches existantes

La reconnaissance automatique du discours haineux en langue arabe, qu'il s'agisse de l'arabe standard moderne (MSA) ou des différents dialectes arabes, a suscité un intérêt croissant dans la recherche ces dernières années. Cette attention est motivée par la propagation inquiétante des discours haineux sur les plateformes de médias sociaux. De nombreuses études ont été consacrées à l'élaboration de modèles et d'approches permettant de relever ce défi de la détection automatique des discours haineux en arabe.

### 2.5.1 Approches précédentes de la détection du discours haineux en arabe

**Areej Al Hassan et Hmood Al Dossari (2022)** [36] pour cette étude un ensemble de données de 11 000 tweets a été collecté et étiqueté, un modèle SVM a été utilisé comme référence pour être comparé à quatre modèles de deep learning : LSTM, CNN+LSTM, GRU et CNN+GRU. Les résultats ont montré que les quatre modèles de deep learning surpassent le modèle SVM dans la détection. Bien que le SVM atteigne un rappel global de 74%, les modèles de deep learning ont une moyenne de 75% de rappel. L'ajout d'une couche de CNN au LSTM améliore la performance globale avec une précision de 72%, un rappel de 75% et un score F1 de 73%.

**Khezzar et al. (2023)** [37] ont proposé un cadre 'arHateDetector' qui détecte le discours haineux dans les tweets arabes, il utilise différents modèles d'apprentissage automatique et d'apprentissage

profond. Ils ont utilisé différents jeux de données pour créer leur propre jeu de données appelé arHateDataset, composé de 34 000 tweets, dont 32% sont des tweets haineux et les 68% restants sont des tweets normaux. Les expérimentations ont montré qu'AraBERT a surpassé les autres modèles, atteignant une précision de 93% sur le jeu de données compilé.

**Alrashidi et al. (2023)** [38] se sont concentrés sur le développement d'un framework pour détecter le contenu abusif en arabe, en utilisant un ensemble de données qui fait partie d'un jeu de données multilingue construit par Ousidhoum et al[39]. contenant 3353 tweets annotés sous plusieurs aspects et une combinaison de modèles d'apprentissage automatique, d'apprentissage profond et de modèles de langage préentraînés.

Ils introduisent un modèle d'apprentissage multitâche (MTL), tirant parti des modèles de langage préentraînés en arabe dialectal, et démontrent sa performance supérieure dans l'identification du contenu abusif arabe par rapport aux modèles d'apprentissage profond existants. Le cadre comprend la collecte et l'annotation des données, la sélection des caractéristiques, la formation du modèle en utilisant les classificateurs SVM et NB (approche ML), les modèles LSTM et CNN (approche DL), et le fine-tuning des modèles BERT empilés (modèles préentraînés). Ils valident également le modèle en utilisant des techniques d'augmentation des données comme NLPaug pour faire face aux contraintes de ressources. Leur architecture MTL, incorporant le modèle de langue MARBERT avec des couches supplémentaires comme LSTM et CNN. Le résultat obtenu a surpassé quatre des cinq attributs en atteignant des scores macro-F1 de 71% pour l'attribut cible, 91% pour l'attribut groupe, 34% pour l'attribut hostilité et 23% pour l'attribut annotateur.

**Ahmad et al. (2023)** [40] ont présenté une étude sur la détection du discours haineux dans les tweets en dialecte jordanien arabe, ont examiné un ensemble de données ‘Jordanian Hate Speech Corpus (JHSC)’ de 403 688 tweets en utilisant des modèles de représentation de texte comme TF-IDF, Word2Vec et AraBERT.

Leur recherche, utilisant des classificateurs d'apprentissage automatique tels que SVM, la régression logistique, CatBoost et AdaBoost. Les meilleurs résultats ont été obtenus en utilisant le modèle CatBoost qui a rapporté un rappel de 51%, une précision de 51% et un score F1 de 50%. Le modèle AraBERT fine-tuned a atteint un rappel de 60% , une précision de 60% et un score F1 de 60% .

**Magnossão et al. (2023)** [41] ont évalué la performance de six modèles transformers : AraBERT, AraELECTRA, Albert-Arabic, AraGPT2, mBERT et XLMRoBERTa et leurs combinaisons d'ensemble. Ils ont utilisé le jeu de données Aracovid [42], qui a été collecté sur Twitter et se compose de 10 828 tweets. Les meilleurs résultats sur le jeu d'entraînement, ont été obtenus en utilisant l'approche d'ensemble basée sur le vote majoritaire , atteignant un score F1 de 60% et une précision de 86%.

## 2.5.2 Approches précédentes de la détection du discours haineux en dialecte algérien

**Guellil et al. (2021)** [43] ont présenté une approche pour détecter le discours haineux contre les femmes dans la communauté arabe sur des plateformes de réseaux sociaux. Ils proposent un nouveau corpus de discours haineux nommé Arabic\_fr\_en, annoté par trois annotateurs différents. Le corpus est évalué en utilisant divers algorithmes, y compris le réseau de neurones convolutifs (CNN), le réseau LSTM et le réseau Bi-LSTM. Les résultats des simulations indiquent que le modèle CNN surpassé les LSTM et Bi-LSTM, atteignant un score F1 allant jusqu'à 86% .

**Lanasri et al. (2023)** [2] ont utilisé un ensemble de données de 13,5K documents, provenant de trois réseaux sociaux distincts (YouTube, Facebook et X). Les auteurs évaluent neuf modèles différents, incluant des architectures d'apprentissage automatique traditionnelles et d'apprentissage profond avancées. Leur approche globale englobe la collecte de données, l'annotation utilisant à la fois des méthodes automatiques et manuelles, et des techniques d'extraction de caractéristiques. Le papier rapporte le modèle Dzarashield comme le plus efficace, atteignant une précision de 87%, un rappel de 87% et un score F1 de 87%.

**Boucherit et Abainia (2022)** [44] ont élaboré un nouveau corpus de plus de 8,7k textes annotés manuellement comme normaux, abusifs et offensants. L'étude évalue divers classificateurs, y compris BiLSTM, CNN, FastText, SVM et NB. Notamment, les résultats expérimentaux ont révélé que les classificateurs SVM et Multinomial NB ont démontré une performance supérieure par rapport à tous les autres classificateurs avec une précision de 74% et 75% respectivement, soulignant leur efficacité dans la catégorisation précise de texte dans ce contexte linguistique.

**Mazari et Kheddar (2023)** [45] ont présenté un nouveau jeu de données annoté à étiquettes multiples, comprenant 14 150 commentaires issus de Facebook, YouTube et Twitter pour détecter les textes toxiques en dialecte algérien. Les étiquettes comprennent le discours de haine, le langage offensant et la cyberintimidation. Plusieurs tests ont été réalisés avec des modèles d'apprentissage automatique et d'apprentissage profond. Le modèle Bi-GRU a obtenu les meilleures performances, avec une précision de 73,6% et un score F1 de 75,8%.

### 2.5.3 Ensembles de données pour le discours haineux en arabe

Dans notre exploration des approches et des ressources pour la détection du discours de haine en arabe, nous reconnaissons le rôle pivot que jouent les ensembles de données complets et bien annotés dans l'avancement du domaine. La complexité du discours de haine, combinée aux nuances linguistiques et culturelles de l'arabe, nécessite des ensembles de données qui sont riches en diversité. Pour illustrer les ressources existantes dans ce domaine, nous nous référons à un tableau d'ensembles de données élaboré par Ahmad et al.[40] (table 2.1). Les ensembles de données répertoriés ont été compilés pour couvrir un large éventail de dialectes et de contextes, reflétant la variabilité complexe de l'utilisation de la langue arabe à travers différentes régions et communautés.

References	Dialect	Source	Dataset size	Labeling process	Classes	Best classifier	Results
[46]Omar et al. (2020)	Mixed	Facebook, Twitter, Instagram, YouTube	20,000	Manual	Hate, Not hate	RNN	Acc : 98.7%, F1-score : 98.7%, Recall : 98.7%, Precision : 98.7%
[47]Alshalan and Al-Khalifa (2020)	Saudi	Twitter	9,316	Manual	hateful, abusive, or normal	CNN	Acc : 83%, F1-score : 79%, Recall : 78%, Precision : 81%

References	Dialect	Source	Dataset size	Labeling process	Classes	Best classifier	Results
[42]Ameur and Aliane (2021) (AraCOVID19-MFH)	Mixed	Twitter	10,828	Manual	Yes, No, Cannot decide	Arabert Cov19	F1-score : 98.58%
[48]Duwairi et al. (2021) (ArHS)	Levantine	Twitter	9,833	Manual	Hate or normal; hate, abusive or normal	Binary class : CNN, Ternary class : BiLSTM-CNN and BERT	F1-score : 81%, F1-score : 74%, F1-score : 56%
[49]Faris et al. (2020)	Mixed	Twitter	3,696	NaN	Hate, neutral, or normal	LSTM+ CNN, with word embedding Aravec (N-grams...)	F1-score : 71.68%
[50]Anezi (2022) (arHateDataset)	Mixed	Variaty	4,203	NaN	Racism, Against religion, gender inequality, violence, offense, bullying, normal positive, and normal negative	RNN architectures : DRNN-1 : binary classification, DRNN-2 : multi-labeled classification	Validation accuracy : 83.22%, 90.30%
[37]Khezzar et al. (2023)	Mixed	Twitter (public datasets)	34,107	Unifying annotation in datasets	Hate, no hate	AraBERT	Accuracy : 93%
[51]Alsaafari and Sadaoui (2021)	Standard and Gulf	Twitter	Training : 9,345, Unlabeled : 5M, Testing : 4,002	Semi-supervised learning	Clean, or offensive	CNN + Skip gram	F1-score : 88.59%, Recall : 89.60%, Precision : 87.69%
[40]Aref et al. (2020)	Mixed	Twitter	3,232	Manual	Hate or Not hate	CNN-FastText	Acc : 71%, F1-score : 52%, Recall : 69%, Precision : 42%
[52]Alshaalan and Al-Khalifa (2020)	Saudi	Twitter	9,316	NaN	Hate or not hate	CNN	Acc : 83%, F1-score : 79%, Recall : 78%, Precision : 81%
[53]Salomon et al. (2022)	Tunisian	Twitter	10,000	Manual	Hateful or Normal	AraBERT	F1-Score : 99%

References	Dialect	Source	Dataset size	Labeling process	Classes	Best classifier	Results
[54] Alsafari et al. (2020b)	Gulf	Twitter	5,361	Manual	2-classes : Clean or Offensive/Hate, 3-classes : Clean, Offensive or Hate, 6-classes : Clean, Offensive, Religious Hate, Gender Hate, Nationality Hate or Ethnicity Hate	CNN + mBERT	2-classes : 87.03%, 3-classes : 78.99%, 6-classes : 75.51%
[55] Mursi et al. (2022)	Mixed	Twitter	3,000	Manual	Extremist or non-extremist	SVM	Acc : 92%, F1-score : 92%, Recall : 95%, Precision : 89%

TABLE 2.1: Les corpus de discours haineux en arabe

Les différentes études sur le discours de haine en arabe présentent plusieurs limitations notables. La taille des corpus varie considérablement, allant de quelques milliers à plus de trente mille échantillons, influençant ainsi la fiabilité et la généralisabilité des résultats. La majorité des études utilisent un étiquetage manuel, introduisant des biais subjectifs et des incohérences. De plus, la focalisation sur des dialectes spécifiques, tels que le saoudien ou le tunisien, limite la généralisation des modèles à d'autres dialectes. Les classifications varient également, allant de simples binaires à des classifications multi-étiquettes complexes, compliquant la comparaison directe des résultats. Les modèles de classification utilisés diffèrent, incluant des RNN, CNN et BERT, avec des performances dépendant fortement de l'architecture du modèle et des techniques de prétraitement. Enfin, une limitation importante est l'absence de considération de l'Arabizi.

## 2.6 Conclusion

Dans ce chapitre, nous avons étudié la détection du discours haineux en arabe. Après avoir défini ce type de discours et souligné son importance, nous avons examiné les recherches existantes, en distinguant les approches appliquées à l'arabe standard et à l'arabe dialectal algérien. En outre, nous avons présenté un tableau récapitulatif des ensembles de données disponibles.

# Chapitre 3

## Conception

### 3.1 Introduction

Ce chapitre expose les diverses approches que nous avons élaborées pour détecter les discours haineux dans les données textuelles en arabe, incluant ses différents dialectes, notamment le dialecte algérien.

Nous commencerons par décrire l'architecture générale de nos systèmes de détection des discours haineux, qui intègrent divers composants et techniques. Cette architecture repose sur une série de techniques de prétraitement commun, essentielles pour préparer les données textuelles avant de les soumettre aux modèles d'apprentissage. Parmi ces techniques, nous inclurons le nettoyage des données, le traitement des abréviations, l'encodage phonétique, la correction des mots, la détection des langues, la traduction et la translittération. Enfin, nous allons détailler les approches spécifiques que nous avons mises en place pour la détection. Ces approches comprennent l'utilisation d'algorithmes d'apprentissage automatique et profond, des approches hybrides, ainsi que l'exploitation de modèles pré-entraînés.

### 3.2 Architecture générale

La figure ci-dessous illustre l'architecture globale proposée dans notre projet. Elle repose sur un pipeline allant du prétraitement des données à l'évaluation des différents modèles envisagés.

Le processus débute par l'utilisation d'un corpus de données étiquetées, c'est-à-dire un ensemble de textes annotés comme discours haineux ou non discours haineux. Une étape cruciale de prétraitement est ensuite appliquée pour nettoyer et normaliser ces données textuelles brutes.

S'ensuit une phase de représentation du texte, différentes représentations vectorielles sont exploitées dans notre projet, telles que TF-IDF, word2vec, AraVec, FastText et les embeddings contextuels.

Le cœur de l'architecture réside dans les différents modèles d'apprentissage automatique envisagés, allant des approches classiques de machine learning aux modèles neuronaux profonds et hybrides, en passant par les puissants modèles de langues pré-entraînés.

Enfin, une évaluation des performances de ces modèles, sur la base de métriques standard (la précision, le rappel, le score F1 et l'exactitude), permet d'identifier les approches les plus prometteuses pour la tâche de détection du discours haineux.

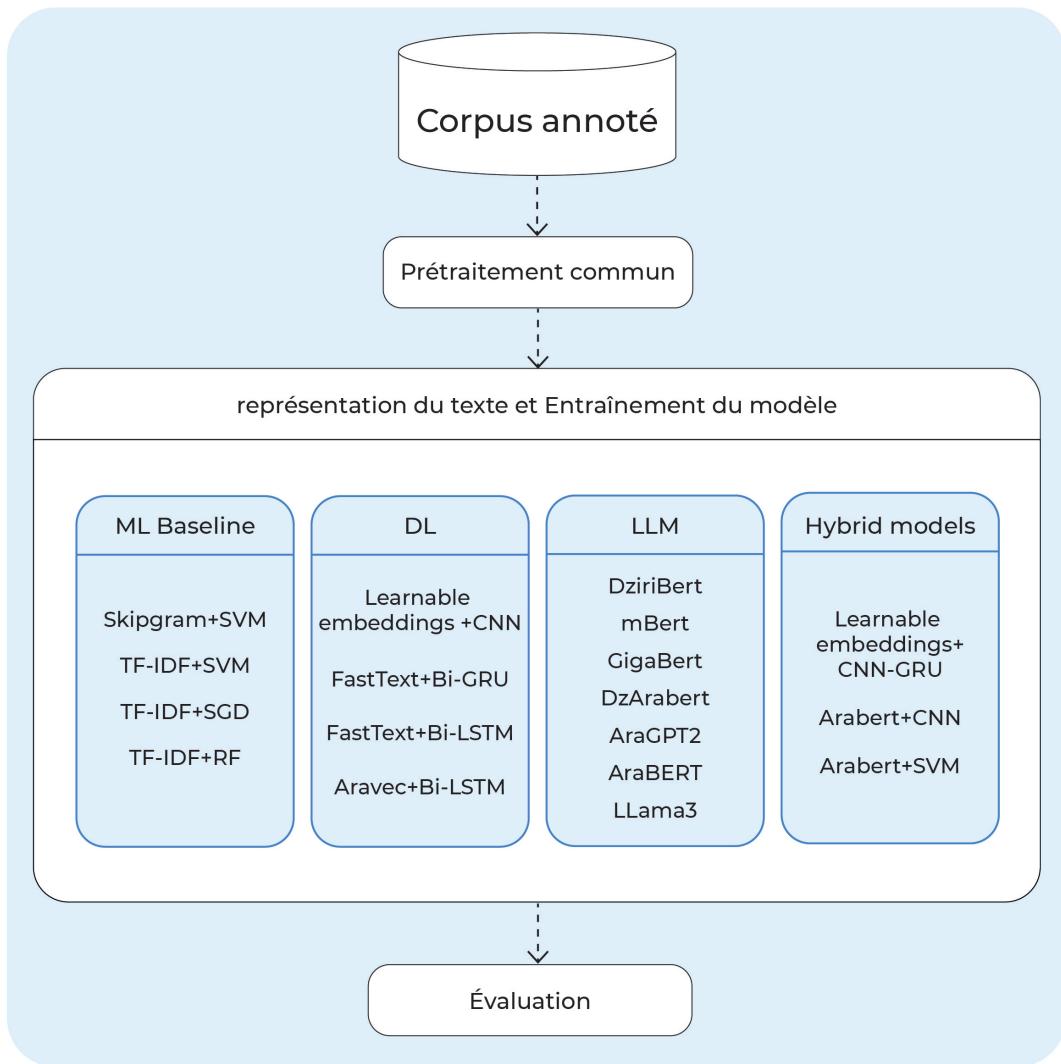


FIGURE 3.1 – Architecture générale

### 3.3 Prétraitement Commun

La langue arabe sur les réseaux sociaux se caractérise par une complexité linguistique significative, nécessitant des techniques de prétraitement robustes pour détecter les discours haineux. Le nettoyage et le prétraitement des données, sont essentiels en TALN pour gérer les irrégularités comme l’orthographe non standard, les abréviations, l’argot et les emojis.

Cette section décrit les différentes techniques de prétraitement commun employées pour tous les modèles proposés dans ce chapitre. Ces techniques améliorent la qualité des données, ce qui est essentiel car cela permet aux modèles d’apprendre plus efficacement et de mieux généraliser à partir des données traitées.

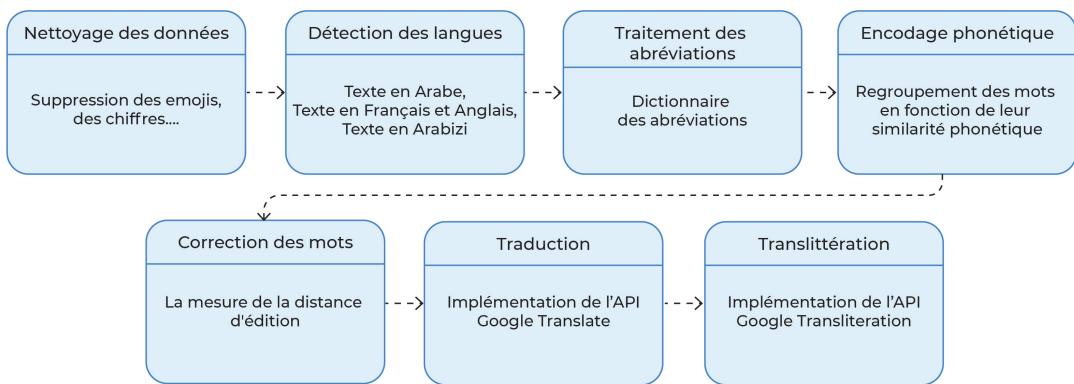


FIGURE 3.2 – Prétraitement pipeline

### 3.3.1 Nettoyage des données

Le nettoyage des données a été une étape essentielle dans la préparation du texte. Voici les étapes que nous avons appliquée pour le nettoyage dans notre cas :

- **Suppression des emojis** : les emojis peuvent transmettre des émotions et des sentiments qui ne sont pas pertinents pour l'identification des discours haineux.
- **Conversion en minuscules** : convertir tout le texte en minuscules pour éviter que des mots comme "Chat" et "chat" soient traités différemment.
- **Suppression des chiffres** : les chiffres représentent souvent des informations non pertinentes comme des âges, des dates ou des statistiques, ce qui pourrait distraire de la détection des discours haineux.
- **Suppression de la ponctuation** : les signes de ponctuation peut être supprimée, car leur utilisation incohérente dans les dialectes arabes peut créer des complications dans l'analyse.
- **Normalisation des caractères** : dans l'écriture arabe, il existe différentes représentations pour certains caractères, comme les différentes formes de l'alif (أ ء ئ ؔ).

La normalisation permet d'unifier ces variantes en une forme standard, facilitant ainsi l'analyse et évitant les redondances.

- **Suppression des diacritiques** : les diacritiques (harakat), peuvent être supprimés sans affecter la détection des discours haineux, car ils ne changent pas le sens des mots dans ce contexte.
- **Suppression des mots vides** : les mots vides (articles, prépositions, etc.) n'apportent généralement pas d'information pertinente pour la détection des discours haineux. Leur suppression peut aider à se concentrer sur les mots clés importants.
- **Suppression des caractères répétés** : dans les textes en dialectes arabes, il est courant de trouver des répétitions de caractères pour exprimer une emphase ou une insistance. Ces répétitions peuvent être supprimées pour normaliser le texte.

- **Suppression des sauts de ligne et des espaces inutiles** : le nettoyage des sauts de ligne et des espaces superflus facilite le traitement du texte et évite les erreurs d'analyse.
- **Suppression des indicateurs de retweet et des noms d'utilisateur** : la suppression des indicateurs de retweet (RT @USER :) et des noms d'utilisateur permet de se concentrer uniquement sur le contenu du message.

### 3.3.2 Détection des langues

Dans le cadre de notre étude, la détection de la langue des commentaires était une étape essentielle de notre processus de prétraitement. Pour répondre aux défis spécifiques posés par le dialecte algérien et son intégration avec d'autres langues, nous avons utilisé des expressions régulières et la bibliothèque "Langid" de Python. Ces techniques nous ont permis de classifier les commentaires selon leur langue de manière précise, facilitant ainsi une analyse adaptée aux particularités de chaque groupe linguistique. Plus précisément, notre approche nous a permis d'identifier et de catégoriser les commentaires comme suit :

- **Textes en Arabe** : nous avons isolé les commentaires contenant uniquement des caractères arabes.
- **Textes en Français et en Anglais** : ce groupe comprend des commentaires principalement en français et en anglais.
- **Textes en Arabizi** : étant donné l'utilisation répandue de l'Arabizi au sein des espaces de discussion en ligne algériens, nous avons séparé les commentaires qui incorporent à la fois des scripts arabes et latins.

### 3.3.3 Traitement des abréviations

Le traitement des abréviations est essentiel compte tenu de leur prévalence dans la communication quotidienne sur les réseaux sociaux et autres plateformes en ligne. Les abréviations, qu'elles soient des troncations de mots, des acronymes ou des formes argotiques propres au dialecte algérien, posent un défi pour les systèmes de traitement automatique des langues. Pour relever ce défi, nous avons développé et intégré un dictionnaire d'abréviations spécifiquement conçu pour le dialecte algérien. Ce dictionnaire comprend une collection des formes abrégées les plus fréquemment utilisées.

L'intégration de ce dictionnaire d'abréviations dans notre pipeline de traitement permet d'étendre automatiquement les formes abrégées en leurs formes complètes correspondantes.

Abréviation	Mot original
slm	salam
mrc	merci
dz	algérie
b1	bien
pk	pourquoi

TABLE 3.1 – Extrait de tableau des abréviations

### 3.3.4 Encodage phonétique

Le dialecte algérien, variant régionalement et sans forme écrite standard, présente un autre défi pour le TALN. Dans un corpus typique, on peut trouver plusieurs orthographies pour le même mot, ce qui complique considérablement la tâche des modèles. Pour relever ce défi, nous avons adopté une approche basée sur le hachage phonétique.

Le hachage phonétique consiste à représenter les mots en se basant sur leur prononciation plutôt que sur leur orthographe exacte. Cela implique d'utiliser des algorithmes phonétiques qui regroupent les mots ayant des sonorités similaires et leur attribuent le même code de hachage. En d'autres termes, même si les mots sont orthographiés différemment, s'ils ont une prononciation similaire, ils recevront le même code de hachage.

Par exemple, le mot « Bienvenue » peut être transcrit de diverses manières telles que « mar7ba », « marhba », « mer7aba », « merhaba ».

Cette approche offre plusieurs avantages dans le cadre du TALN. Premièrement, elle permet de contourner les variations orthographiques en se concentrant sur la prononciation commune des mots. Deuxièmement, elle réduit la dimensionnalité de l'espace des mots en regroupant les variantes orthographiques.

Pour mettre en œuvre le hachage phonétique, nous avons utilisé l'algorithme « Soundex », spécialement conçu pour traiter les spécificités phonétiques. L'algorithme Soundex est un algorithme phonétique utilisé pour indexer les noms par leur son. Il est principalement utilisé pour associer des noms qui sonnent de manière similaire malgré de petites différences dans l'orthographe. Voici comment il fonctionne [56] :

1. La première lettre du nom est conservée telle quelle
2. Toutes les autres lettres du mot sont converties en nombres selon leur groupe de son :
  - B, F, P, V → 1
  - C, G, J, K, Q, S, X, Z → 2
  - D, T → 3
  - L → 4
  - M, N → 5
  - R → 6
  - H, W, Y → non codé
3. Supprimez toutes les voyelles.
4. Continuez jusqu'à ce que le code comporte une lettre et trois chiffres. Si la longueur du code est supérieure à quatre, tronquez-le à quatre lettres. Si le code est plus court que quatre lettres, complétez-le par des zéros.

Pour regrouper les mots en fonction de leur similarité phonétique dans un ensemble de données, nous avons créé un dictionnaire où chaque clé correspond à un code Soundex et chaque valeur est une liste de mots partageant ce code. En traitant chaque mot dans l'ensemble de données, nous avons calculé son code Soundex puis ajouté le mot à la liste associée à son code Soundex respectif dans le dictionnaire.

Exemples :

M610 : ['mar7ba', 'marhba', 'mer7aba', 'merhaba']

### 3.3.5 Correction des mots

Bien que le codage phonétique permette de traiter les variations basées sur la prononciation, il peut ne pas capturer adéquatement des différences orthographiques, telles que les substitutions ou les transpositions de caractères. Pour compléter l'approche de codage phonétique, nous avons incorporé l'utilisation de métriques de distance d'édition.

La distance d'édition mesure la similarité entre deux chaînes de caractères en calculant le nombre minimal d'opérations (insertions, suppressions ou substitutions) nécessaires pour transformer une chaîne en l'autre.

---

```

Function LEVDIST (a, b) is
    n  $\leftarrow$  a.size();
    m  $\leftarrow$  b.size();
    previous[j]  $\leftarrow$  j  $\forall$  j  $\in$  0..m;
    for i  $\leftarrow$  0 to n - 1 do
        current[0]  $\leftarrow$  i + 1 ;
        for j  $\leftarrow$  0 to m - 1 do
            indicator  $\leftarrow$  a[i]  $\neq$  b[j] ? 1 : 0;
            current[j + 1]  $\leftarrow$  Minimum(
                previous[j + 1], current[j + 1], previous[j] + indicator);
            previous[j + 1]  $\leftarrow$  current[j + 1];
            current[j]  $\leftarrow$  previous[j];
            previous[j]  $\leftarrow$  current[j];
        end
        previous  $\leftarrow$  current; /* copy current row to previous
        row for next iteration */
    end
    return previous[m]; /* result is in the previous row
    after the last swap */
end

```

---

FIGURE 3.3 – L'algorithme de distance de Levenshtein

Nous avons utilisé l'algorithme de distance de Levenshtein [57] pour évaluer la distance d'édition entre les mots au sein de chaque groupe phonétique, après avoir testé plusieurs valeurs de seuil, nous avons déterminé que lorsque cette distance d'édition entre deux mots est inférieure ou égale à quatre, ces mots sont alors considérés comme des correspondances étroites.

En associant les codes phonétiques à la distance d'édition, nous avons identifié et regroupé les mots qui étaient à la fois phonétiquement similaires et orthographiquement proches.

Exemple :

W200 : ['wach', 'wachia', 'wsh', 'wejeh', 'wajhahe', 'wjouh']

Groupe 1 : ['wach', 'wachia', 'wsh']

Groupe 2 : ['wejeh', 'wajhahe', 'wjouh']

Pour chaque groupe, nous avons sélectionné le mot le plus fréquent comme représentant pour ce groupe. En cas d'égalité, le mot qui vient en premier alphabétiquement est choisi comme représentant. Tous les mots du groupe sont ensuite associés à ce représentant.

### 3.3.6 Traduction

La diversité linguistique de notre données exige une traduction précise afin de faciliter une analyse complète du contenu.

Nous avons utilisé la bibliothèque "googletrans" de Python, qui implémente l'API Google Translate, pour automatiser la traduction des commentaires en français, anglais et arabizi. Cet outil a offert une solution rapide et évolutive pour gérer les ensembles de données impliqués.

Google Translate est alimenté par des modèles d'apprentissage automatique avancés, notamment la neural machine translation (NMT). Cette technologie de pointe aborde la traduction de manière holistique plutôt que par parties isolées, ce qui se traduit par des traductions plus fluides et contextuellement adaptées.[58]

Exemple :

Avant : "rabi y3awnak wrabi yachfik allah yba3d 3lik l3ayn ou yahfadlk al3a2ila"

Après : "ربی یعاونک و ربی یشفیک الله یبعد علیک لعین او یحفظلك العائلة"

### 3.3.7 Translittération

Lorsque l'API Google Translate ne parvient pas à traduire certains mots en arabizi, nous utilisons la translittération. Pour cela, nous avons recours à la bibliothèque "google-transliteration" de Python qui utilise l'API Google Transliterate. Cette approche permet de transformer les lettres latines en caractères arabes tout en préservant la prononciation approximative. Cela est particulièrement utile pour les termes spécifiques, les noms propres, ou les expressions techniques qui ne sont pas bien gérées par les outils de traduction automatique.

Exemple :

Texte : "Rabi me3ak inchallah mais se peuple aimes les voleurs"

Après traduction : "Rabi Me3ak Inchallah

Après translittération : "ربی معاك إنشالله لكن الناس يحبون اللصوص"

## 3.4 Approches proposées

Dans cette section, nous décrivons les différentes approches adoptées dans notre étude pour la détection des discours haineux, couvrant un éventail de techniques allant des modèles traditionnels d'apprentissage automatique aux approches modernes basées sur l'apprentissage profond et les modèles pré-entraînés. Chaque approche est minutieusement décrite, en débutant par les phases de prétraitement des données et les techniques de représentation textuelle, avant de passer à la présentation de l'architecture de chaque modèle.

### 3.4.1 Machine Learning baselines

Cette partie se consacre à l'exploration des approches de référence couramment utilisées en machine learning pour évaluer la performance des modèles. Les baselines servent de points de comparaison pour déterminer si les modèles avancés apportent une amélioration significative. Nous aborderons les étapes essentielles, y compris le prétraitement des données, la représentation du texte et la présentation de cette approche.

### 3.4.1.1 Prétraitement

La phase de prétraitement dans ce processus comprend prétraitement commun mentionné précédemment, ainsi que trois autres étapes spécifiques à cette approche, réalisées à l'aide de la bibliothèque NLTK :

- **La tokenisation** : cette étape divise le texte en mots individuels ou en unités significatives appelées tokens, cela permet de séparer les mots distincts et de les traiter comme des unités indépendantes.

Ci-dessous, un exemple d'un commentaire avant et après la tokenisation :

Avant tokenisation :

الصح الحدود ماشي داخل بلاد ميكي

Après tokenisation :

[الصح ، الحدود ، ماشي ، داخل ، بلاد ، ميكي]

- **Le stemming** : le stemming vise à réduire les mots à leur forme de base en supprimant les suffixes redondants. Cette étape est pour simplifier les mots et les regrouper selon leur racine commune.
- **La lemmatisation** : la lemmatisation va plus loin que le stemming pour le but de convertir les mots en leur forme de dictionnaire, appelée lemme. Cela permet de s'assurer que tous les mots sont réduits à leur forme la plus fondamentale, en tenant compte des règles grammaticales et de la morphologie de la langue.

En combinant ces trois étapes, le prétraitement du texte avec NLTK permet de réduire les variations orthographiques et grammaticales, de se concentrer sur le sens principal des mots et d'améliorer la cohérence globale des données textuelles. Cela facilite une analyse ultérieure plus efficace et précise.

### 3.4.1.2 Représentation de texte

La représentation textuelle pour les modèles d'apprentissage automatique implique deux méthodes principales : le TF-IDF et le word2vec skip-gram, que nous allons détailler dans ce qui suit.

- **TF-IDF** : la méthode TF-IDF permet de capturer l'importance relative des mots dans un document donné par rapport à un ensemble de documents. Elle attribue à chaque mot un score qui reflète à la fois sa fréquence d'apparition dans le document et sa rareté dans l'ensemble des documents. Ce score permet de transformer le texte brut en un vecteur numérique représentant la distribution des mots dans le document.[59]

Le TF-IDF est le produit de deux facteurs comme le montre la formule suivante :

$$\text{TFIDF}(m, d) = \text{TF}(m, d) \times \text{IDF}(m) \quad (3.1)$$

- $\text{TF}(m, d)$  : est la fréquence du mot  $m$  dans un document  $d$ .
- $\text{IDF}(m)$  : représente la fréquence de documents inverse du mot  $m$ , plus les documents contiennent le mot  $m$ , plus IDF diminue.

L>IDF se calcule avec la formule suivante :

$$IDF(m) = \log \left( \frac{N}{df(m)} \right) \quad (3.2)$$

- N : est le nombre total de documents.
- df( $m$ ) : est le nombre de documents contenant  $m$

- **Word2Vec (Skip-gram) :** le modèle Skip-gram est une méthode de réseau neuronal pour l'apprentissage des représentations des mots et fait partie de l'architecture Word2Vec pour le calcul des embeddings de mots. Contrairement à CBow Word2Vec, qui utilise les mots environnants pour prédire le mot central, Skip-gram utilise le mot central pour prédire les mots environnants. Le sous-échantillonnage des mots fréquents permet d'atténuer le problème posé par le fait que ces mots fournissent moins d'informations que les mots rares. Le modèle Skip-gram capture les relations sémantiques et syntaxiques entre les mots et peut représenter des phrases. Sa fonction objective somme les probabilités logarithmiques des mots environnants à gauche et à droite du mot cible  $w(t)$  pour produire l'objectif suivant [60] :

$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log P(w_{t+j} | w_t) \quad (3.3)$$

où :

- $T$  est le nombre total de mots dans le corpus.
- $w_t$  est le mot cible à la position  $t$ .
- $w_{t+j}$  sont les mots du contexte dans une fenêtre de taille  $c$  autour du mot cible  $w_t$ .
- $P(w_{t+j} | w_t)$  est la probabilité du mot du contexte  $w_{t+j}$  étant donné le mot cible  $w_t$ .

#### 3.4.1.3 Présentation de l'approche

L'approche basée sur l'apprentissage automatique pour la détection des discours de haine repose sur l'utilisation de différentes combinaisons de techniques de vectorisation de texte et de modèles de classification.

Une première combinaison utilise un SVM (Support Vector Machine) avec la vectorisation tf-idf (Term Frequency-Inverse Document Frequency), une méthode efficace pour transformer le texte en vecteurs pondérés en fonction de la fréquence des termes.

Une seconde méthode utilise un SVM avec des vecteurs de mots générés par le modèle Word2Vec en mode skip-gram. Ce dernier est entraîné sur les commentaires tokenisés, où chaque mot est représenté par des vecteurs denses de taille fixe. Ensuite, la représentation vectorielle de chaque commentaire est obtenue en faisant la moyenne des vecteurs de tous les mots présents dans le commentaire, ce qui permet de capturer les relations sémantiques entre les mots.

Une troisième combinaison utilise un classificateur de descente de gradient stochastique (SGD) avec tf-idf. La SGD est une méthode rapide et efficace pour les grandes quantités de données.

Enfin, la quatrième méthode associe une forêt d'arbres décisionnels (Random Forest) avec tf-idf, combinant la robustesse d'un ensemble d'arbres décisionnels avec la vectorisation basée sur la fréquence des termes.

Ces différentes combinaisons permettent de comparer l'efficacité des approches classiques et avancées pour la tâche critique de détection des discours de haine.

### 3.4.2 Approches basées sur l'apprentissage profond

L'approche basée sur l'apprentissage profond consiste tout d'abord à aborder le prétraitement des données spécifiquement adapté à cette méthode. Ensuite, nous passerons en revue différentes techniques de représentation du texte utilisées, incluant FastText, AraVec, ainsi que les embeddings apprenables. Enfin, les architectures utilisées dans cette approche seront introduites, notamment les modèles BI-GRU, BI-LSTM et CNN.

#### 3.4.2.1 Prétraitement

Une fois les étapes de prétraitement commun déjà mentionnées sont réalisées, nous procérons à la tokenization en utilisant le tokenizer de la bibliothèque Keras. Avant de commencer à mettre en œuvre les architectures qui seront présentées par la suite.

Le tokenzier Keras est un outil utilisé pour convertir des données textuelles en un format adapté aux modèles d'apprentissage automatique, en particulier aux modèles d'apprentissage profond construits avec Keras.

Le tokenzier Keras fonctionne en deux étapes principales. Tout d'abord, il divise les phrases textuelles en mots individuels ou tokens (comme la séparation d'une phrase en mots). Ensuite, il attribue un entier unique à chaque mot en fonction de sa fréquence d'apparition. Cela crée un dictionnaire qui fait correspondre chaque mot à un nombre entier unique.

#### 3.4.2.2 Représentation de texte

La représentation du texte pour les modèles de deep learning comporte trois méthodes principales : FastText, AraVec et les embeddings apprenables qui seront présentées ci-dessous :

- **FastText** : Le modèle d'embeddings de mots FastText, est une technique populaire développée par le laboratoire de recherche en intelligence artificielle de Facebook, capture les relations sémantiques entre les mots en tenant compte de l'information de sous-mots, particulièrement bénéfique pour les langues avec une morphologie riche comme la langue arabe. Il représente les mots comme des vecteurs dans un espace à haute dimension, où les mots similaires ont tendance à avoir des vecteurs plus proches, capturant ainsi les relations sémantiques. Contrairement aux embeddings de mots traditionnels, FastText intègre les n-grammes de caractères lors de la génération des vecteurs. En permettant l'entraînement sur des données multilingues, il gère efficacement les mots de différents langages. [61]
- **AraVec** : AraVec est un projet open source de représentation distribuée des mots (word embedding) pré-entraîné, destiné à fournir à la communauté de recherche en NLP arabe des modèles de word embedding puissants et gratuits. La troisième version d'AraVec propose 16 modèles de word embedding différents, comprenant des modèles de unigrammes et de n-grammes, construits à partir de deux domaines de contenu arabe : les Tweets et les articles en arabe de Wikipedia. Pour notre étude, nous avons utilisé la version entraînée sur les Tweets.[62]
- **Learnable embeddings** : Les embeddings apprenables, ou couche d'embedding, constituent un élément clé des modèles de traitement du langage naturel profonds. Leur fonction principale est de convertir chaque mot d'une séquence textuelle en une représentation vectorielle dense,

capturant ainsi son sens et ses relations avec les autres mots du contexte.

Cette conversion s'effectue en deux étapes :

- **Initialisation du dictionnaire d'embeddings** : La couche d'embedding commence par créer un dictionnaire d'embeddings, qui attribue à chaque mot unique du vocabulaire un vecteur numérique aléatoire. La dimension de ces vecteurs, appelée "dimension d'embedding", détermine la complexité et la granularité de la représentation des mots.
- **Apprentissage des embeddings** : Ces embeddings sont "apprenables" car ils sont ajustés automatiquement pendant l'apprentissage du modèle, permettant ainsi au réseau neuronal de découvrir des représentations vectorielles qui capturent les relations sémantiques et contextuelles entre les mots dans les données d'entraînement. En d'autres termes, les embeddings apprenables sont capables de capturer des informations significatives sur la signification et le contexte des mots.[63]

### 3.4.2.3 Présentation de l'approche

Dans le cadre des approches d'apprentissage profond, nous allons élaborer trois architectures distinctes : deux d'entre elles sont basées sur les réseaux de neurones récurrents (RNN), à savoir les modèles Bi-GRU et Bi-LSTM, tandis que la troisième utilise les réseaux de neurones convolutifs (CNN). Chaque architecture présente des caractéristiques spécifiques qui la rendent adaptées à la tâche de classification des discours haineux.

- **CNN** : Ce modèle commence par une couche d'embedding qui convertit chaque mot en sa représentation vectorielle dense, ces embeddings étant ajustés automatiquement pendant l'apprentissage du modèle pour capturer les relations sémantiques entre les mots, suivie d'une couche dropout avec un taux de 0.5 pour régulariser la sortie. Ensuite, une couche de convolution unidimensionnelle, une taille de noyau de 3, une fonction d'activation ReLU et un padding qui conserve les dimensions d'entrée sont ajoutés. Cette couche est suivie d'une couche de GlobalAveragePooling1D pour réduire la dimensionnalité spatiale. Une autre couche dropout avec un taux de 0.5 est ensuite appliquée pour la régularisation. Une couche dense est ajoutée avec des régularisations L1, L2 et d'activité pour éviter l'overfitting. Une normalisation en batch est appliquée suivie d'une autre couche dropout pour régulariser davantage le modèle. Enfin, une couche dense avec une activation sigmoïde est ajoutée pour produire une distribution de probabilité sur les classes de classification.



FIGURE 3.4 – Architecture CNN

- **Bi-GRU** : Le modèle Bi-GRU utilise l'embedding fastText pour capturer les relations sémantiques entre les mots, en les représentant sous forme de vecteurs denses. L'architecture du modèle comprend deux couches Bi-GRU, capables de traiter des séquences dans les deux sens. Cela permet au modèle de capturer efficacement les informations contextuelles cruciales pour l'identification des discours haineux. Pour éviter l'overfitting et améliorer la stabilité du modèle, des couches de Batch normalization sont introduites après chaque couche Bi-GRU, suivies de couches dropout avec un taux de 0.5. La couche finale est une couche dense avec une seule unité et une fonction d'activation sigmoïde, permettant une classification binaire.

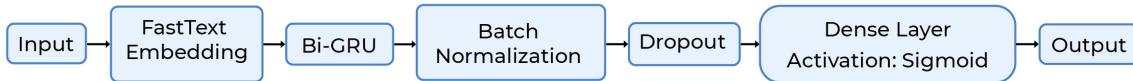


FIGURE 3.5 – Architecture modèle Bi-GRU

- **Bi-LSTM :** Le modèle Bi-LSTM utilise également l'embedding fastText, similaire au modèle Bi-GRU précédent. L'architecture du modèle suit la même structure avec deux couches Bi-LSTM, capables de traiter les séquences dans les deux directions. Afin de contrer l'overfitting et de garantir la stabilité du modèle, des couches de Batch normalization sont insérées après chaque couche Bi-LSTM, suivies de couches dropout avec un taux de 0.25. Le dropout joue un rôle crucial en réduisant l'influence des poids individuels sur la sortie du modèle, ce qui contribue à prévenir l'overfitting et à améliorer ses performances. Enfin, une couche dense avec une seule unité et une fonction d'activation sigmoïde est utilisée pour la classification binaire.

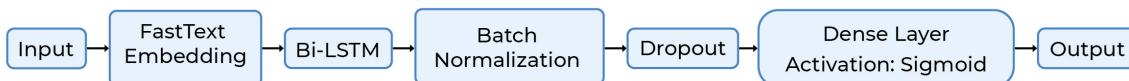


FIGURE 3.6 – Architecture modèle Bi-LSTM

### 3.4.3 Approches basées sur les modèles pré-entraînés (Transformers)

Cette section explore l'approche basée sur les modèles pré-entraînés, notamment les Transformers. Nous commencerons par une présentation générale des caractéristiques des modèles de langage pré-entraînés (LLMs) utilisés dans notre étude, qui incluent ceux spécifiques au dialecte algérien, multilingues et autres. Ensuite, les techniques de prétraitement spécifiques nécessaires pour l'utilisation de ces modèles seront abordées. Enfin, on passe à l'explication du processus de fine-tuning, montrant comment ces modèles sont ajustés pour la tâche de détection des discours haineux.

#### 3.4.3.1 Prétraitement spécifique aux modèles pré-entraînés

Pour cette approche basée sur les modèles pré-entraînés, après avoir effectué le prétraitement précédemment mentionné, la tokenization est effectuée en utilisant le tokenizer intégré au LLM lui-même. Cela signifie que le processus de tokenization est réalisé en utilisant les mécanismes internes du modèle linguistique, ce qui garantit une cohérence et une compatibilité optimales avec le modèle.

Nous illustrons ici des exemples de commentaires écrits en dialecte algérien avant et après la tokenization avec le tokeniser de DziriBERT.

Avant tokenization	Après tokenizaion
قاطع لبحر متعرفش تعم اقعد بلادك خير	[اقعد, '#م', 'تعو', 'متعرفش', 'لبحر', 'قاطع', 'خير', 'بلادك']

Prenons aussi l'exemple suivant d'un commentaire tokenisé avec AraBERT

Avant tokenization	Après tokenizaion
بتغيطكم ارهابين بتوغ المناخ التشاءعي	[‘مٰى’ ‘بٰت’ ‘اَرْهَابِيٌّ’ ‘الْمَنَاخُ’ ‘التَّشَاءُعُ’ ‘كٰمٰطٰ’ ‘وٰعٰ’ ‘يٰ’ ‘نٰنٰ’ ‘كٰمٰطٰ’ ‘وٰعٰ’ ‘بٰتْغٰيْتُكُمْ’]

### 3.4.3.2 Présentation de l'approche

L'approche proposée repose sur le fine-tuning de divers Modèles de Langage pré-entraînés (LLMs), incluant à la fois des modèles spécifiques au dialecte algérien et d'autres modèles couvrant la langue arabe dans son ensemble, y compris ses différents dialectes. En utilisant des données de discours de haine pour le fine-tuning, cette méthode vise à créer des modèles robustes de classification. Cette approche capitalise sur la capacité des LLMs à saisir les nuances linguistiques, tout en adaptant leur compréhension aux spécificités régionales et dialectales, offrant ainsi une solution prometteuse pour la détection précise du discours de haine. Les modèles utilisés dans cette étude comprennent :

- **DziriBert** : DziriBERT est le premier modèle de langue algérien pré-entraîné sur plus d'un million de tweets algériens. DziriBERT surpassé les modèles existants, en particulier avec l'alphabet latin, malgré son entraînement sur un ensemble de données relativement petit (150 Mo). Le dialecte algérien, influencé par plusieurs langues, présente des défis pour les modèles existants en raison de ses caractéristiques uniques. Le jeu de données collecté, obtenu à partir de l'API Twitter, contient 1,1 million de tweets après prétraitement. DziriBERT, basé sur l'architecture BERT, utilise un Tokenizer WordPiece et est entraîné à l'aide de la tâche de modélisation de langue masquée (MLM). Notre objectif principal était de déterminer dans quelle mesure DziriBERT gère efficacement le dialecte algérien dans la classification et la détection du discours de haine.[64]
- **Multilingual Bert (mBert)** : mBERT est un modèle pré-entraîné basé sur l'architecture Transformer, largement utilisé pour les tâches de Traitement Automatique du Langage Naturel (TALN). Ce modèle est entraîné sur un vaste ensemble de données textuelles et de code, lui permettant de traiter des textes en plusieurs langues sans nécessiter de pré-entraînement spécifique à une langue particulière. mBERT adopte une approche d'apprentissage auto-supervisé, ce qui signifie qu'il n'a pas besoin de données annotées manuellement pour le pré-entraînement. Cela lui permet de profiter de grandes quantités de données textuelles disponibles publiquement dans de nombreuses langues. Notamment, il est pré-entraîné en utilisant des corpus monolingues de Wikipédia couvrant 104 langues, y compris l'arabe.[51]
- **GigaBert** : Selon Lan et al. (2020) [65], GigaBERT-v4 possède la même configuration que BERT base, avec 12 couches d'attention, 12 opérations d'auto-attention, des dimensions cachées de 768 par couche et 110 millions de paramètres. En outre, GigaBERT-v4 utilise une taille de vocabulaire arabe de 26 000 pour 4,3 milliards de données d'entraînement en arabe provenant de diverses sources (Arabic Gigaword Parker et al., 2009, Wikipedia, Oscar Javier et al., 2019 avec

commutation de code).[66]

- **DzArabert** : DzaraBert est une version élaguée du modèle de langage basé sur les Transformers, spécialement pré-entraînée pour le dialecte algérien (DziriBERT). Cette version est conçue pour traiter les textes algériens écrits exclusivement en caractères arabes. [67]
- **AraGPT2-base** : AraGPT2 est le premier modèle avancé de génération de langage conçu spécifiquement conçu pour l'arabe, a été entraîné à partir de zéro sur un grand corpus arabe de textes internet et d'articles. Il s'agit d'un modèle empilé de transformateurs-décodeurs, similaire à l'architecture de GPT-2, il est entraîné en utilisant l'objectif de modélisation de langage causale, visant à prédire le mot suivant dans une séquence en fonction du contexte précédent. AraGPT2 est entraîné sur un ensemble de données texte arabe massif de 77 Go. Les auteurs soulignent le succès d'AraGPT2 en réponse à des questions sans exemple, démontrant ainsi sa capacité à s'adapter à de nouvelles tâches au-delà de son objectif d'entraînement initial (la génération de langage).[68]
- **Bert-base-arabertv02-twitter** : AraBERT est un modèle de représentation textuelle contextualisée spécialement conçu pour la langue arabe. Construit sur l'architecture BERT (Bidirectional Encoder Representations from Transformers), ce modèle excelle dans diverses tâches de compréhension du langage naturel (NLU). Il atteint des résultats de pointe sur plusieurs tâches NLU en arabe, notamment l'analyse de sentiment, la reconnaissance d'entités nommées et la réponse aux questions. AraBERT est disponible publiquement pour une utilisation dans les bibliothèques populaires de traitement du langage naturel. Le modèle est pré-entraîné sur un large corpus arabe comprenant des extraits de Wikipedia en arabe, des articles de sites web d'actualités et des collections de textes arabes disponibles publiquement.  
AraBERTv02-Twitter-base est une version récente du modèle, adaptée aux dialectes arabes et aux tweets. Elle a été entraînée en poursuivant le pré-entraînement à l'aide de la tâche MLM (Masked Language Model) sur environ 60 millions de tweets en arabe, filtrés à partir d'une collection de 100 millions. Ce nouveau modèle a intégré des emojis dans le vocabulaire, en plus des mots courants qui n'étaient pas présents au départ. [69]
- **LLama 3** : Llama3 est une famille de modèles développée par Meta AI, offrant des capacités de pointe. Ces modèles existent en deux tailles : 8 milliards et 70 milliards de paramètres. Une innovation technique clé est l'utilisation d'un vocabulaire de 128 000 token avec un tokeniseur, qui encode efficacement le langage pour de meilleures performances. De plus, l'« attention groupée par requête » (GQA) améliore l'efficacité du modèle pendant l'inférence, le rendant plus rapide à utiliser. Cette combinaison de fonctionnalités fait de Llama3 un outil puissant pour diverses tâches de traitement du langage naturel.[70]

#### 3.4.3.3 Fine-tuning des modèles pré-entraînés

Dans le processus de fine-tuning des LLMs, la première étape consiste à sélectionner méticuleusement les hyperparamètres, tels que le taux d'apprentissage, la taille des batches et le nombre d'époques d'entraînement, en expérimentant avec diverses valeurs pour déterminer celles qui optimisent les performances pour l'ensemble de données et le modèle en question. Ensuite, en adoptant

une méthodologie de fine-tuning spécifique avec "Trainer API" de HuggingFace, nous exploitons les connaissances générales du pré-entraînement tout en les adaptant à la tâche particulière de détection des discours de haine. Tout au long du processus, une évaluation continue est cruciale, impliquant une surveillance régulière des performances du modèle à l'aide de métriques appropriées telles que la précision, le rappel et le F1-score, afin de détecter les problèmes potentiels comme l'overfitting ou l'underfitting, et d'ajuster la stratégie en conséquence.

Enfin, une optimisation itérative est réalisée en analysant en profondeur les résultats obtenus après chaque itération de fine-tuning, identifiant ainsi les faiblesses spécifiques du modèle, telles que les types de discours de haine qu'il a du mal à détecter, pour raffiner la stratégie de fine-tuning et améliorer continuellement les performances globales du modèle.

### 3.4.4 Approches hybrides

Les approches hybrides abordées incluent l'utilisation de CNN avec BI-GRU, les embeddings AraBERT associés à un classifieur SVM, ainsi que l'approche combinant les embeddings AraBERT avec CNN. Les architectures des modèles hybrides sont détaillées ci-dessous :

- **CNN + Bi-GRU** : ce modèle hybride CNN-GRU utilise une approche combinée pour la classification. Il commence par une couche d'embedding pour convertir les données d'entrée en vecteurs de dimension dense, ajustés pendant l'apprentissage pour capturer les relations sémantiques. Ensuite, deux couches de convolution unidimensionnelle sont appliquées, suivies de couches de max pooling, afin d'extraire les caractéristiques locales importantes des données textuelles. Pour régulariser le modèle et éviter l'overfitting, des couches de dropout sont insérées après chaque couche de pooling. De plus, une couche de dropout spatiale est introduite pour encore plus de régularisation. Ensuite, une couche bidirectionnelle GRU est ajoutée pour capturer les dépendances séquentielles dans les deux directions. Après cela, les caractéristiques extraites sont aplatis et passées à travers une couche dense avec une activation ReLU et une régularisation L2 pour réduire l'overfitting. Enfin, une couche dense de sortie avec une activation sigmoïde est utilisée pour la classification en deux classes, à savoir le discours haineux et le discours non haineux.

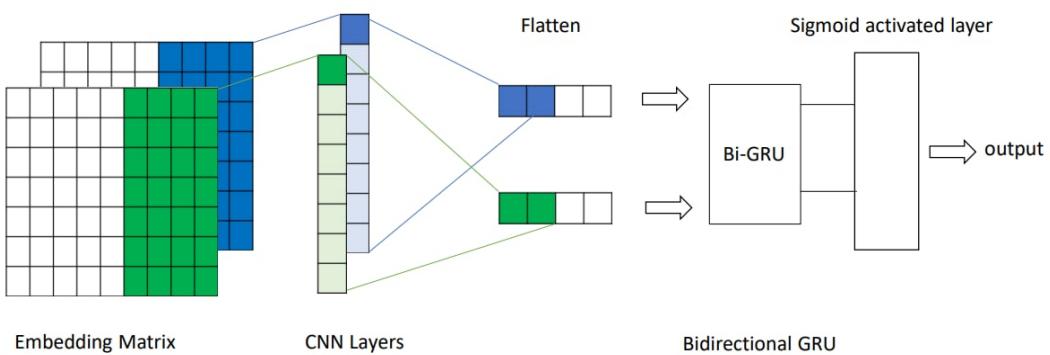


FIGURE 3.7 – Architecture du modèle hybride CNN+Bi-GRU

- **AraBERT + CNN** : Ce modèle utilise un modèle AraBERT pour générer des embeddings contextuels, qui sont ensuite traités par plusieurs couches convolutionnelles avec différentes tailles de filtres pour capturer divers motifs. Le pooling maximal et le dropout sont appliqués pour la

sélection des caractéristiques et la régularisation. Enfin, une couche entièrement connectée avec une activation sigmoïde produit la sortie de classification.



FIGURE 3.8 – Architecture du modèle hybride AraBERT+CNN

- **AraBERT + SVM :** L'architecture du modèle AraBERT + SVM pour la détection des discours de haine repose sur l'utilisation du modèle AraBERT pour extraire des embeddings contextuels, et d'un classificateur SVM. Les mots du texte prétraité sont convertis en embeddings à l'aide du modèle de langage AraBERT, puis envoyés au SVM, qui les classe en discours de haine ou non. Les embeddings d'AraBERT sont contextuels et plus robustes que les embeddings non contextuels, ce qui peut améliorer les performances du classificateur SVM.



FIGURE 3.9 – Architecture du modèle hybride AraBERT+SVM

### 3.4.5 Conclusion

En conclusion, ce chapitre a présenté un éventail d'approches pour détecter les discours haineux dans les textes en arabe, y compris différents dialectes tels que l'algérien. Après la phase de prétraitement, différentes techniques de représentation et de traitement du texte ont été explorées. Le choix de l'approche optimale dépendra des caractéristiques des données, des objectifs visés et des ressources disponibles. Les détails des expérimentations menées ainsi que les résultats obtenus pour chacune des approches proposées seront présentés et analysés dans le prochain chapitre.

# Chapitre 4

## Réalisation et résultats

### 4.1 Introduction

Ce chapitre offre une vue détaillée de l'environnement de travail, des expérimentations menées, ainsi que de l'analyse et de la discussion des résultats obtenus dans l'application de méthodes d'apprentissage automatique et profond à des ensembles de données spécifiques.

D'abord, les outils et technologies utilisés sont introduits, tels que Google Colaboratory, le langage de programmation Python, et diverses bibliothèques employées pour les expérimentations. Après, la description des datasets utilisés est abordée, notamment l'Algerian Dialect Toxicity Speech Dataset et l'OffensEval2020 Dataset. Les mesures de performance telles que l'exactitude, la précision, le rappel et le F1-Score sont également présentées pour évaluer les modèles. Cette partie se termine par une évaluation des différents modèles testés, incluant les baselines de machine learning, les modèles d'apprentissage profond, les modèles hybrides et les Large Language Models (LLMs).

Ensuite, nous procéderons à une analyse détaillée des performances des modèles, des durées d'entraînement et d'exécution, ainsi qu'une évaluation des erreurs. De plus, une comparaison des résultats obtenus sur les deux ensembles de données étudiés est offerte afin de fournir une vision claire des forces et des faiblesses de chaque approche.

Enfin, l'application pratique développée à partir des modèles et techniques étudiés est mise en lumière, illustrant leur utilité et leur efficacité dans la détection des discours de haine.

### 4.2 Présentation de l'environnement

Dans cette section, nous allons explorer les outils et les technologies qui constituent l'environnement de travail de notre étude. L'objectif est de fournir une vue d'ensemble des composants clés et de leur rôle dans le développement et l'exécution des modèles de détection des discours de haine.

#### 4.2.1 Google Colaboratory

Google Colaboratory, aussi connu sous le nom de Google Colab, est un service gratuit en ligne qui offre un environnement de notebook Jupyter basé sur le cloud. Il permet aux utilisateurs d'écrire et d'exécuter du code Python, d'entraîner des modèles d'apprentissage automatique, et d'analyser

des données, le tout sans avoir à installer de logiciel ou à configurer du matériel.

Nos modèles d'apprentissage profond ont été entraînés sur la plateforme Google Colab en utilisant un GPU Tesla T4, 12.7 Go de RAM système et 15 Go de RAM GPU.

#### 4.2.2 Langage (python)

Python est un langage de programmation de haut niveau, interprété et multiparadigme, largement utilisé dans divers domaines tels que le développement web, l'analyse de données, l'apprentissage automatique et la recherche scientifique. Il est connu pour sa lisibilité, sa polyvalence et sa communauté active. Sa large gamme de bibliothèques en fait un choix populaire pour une grande variété de tâches de TALN.

#### 4.2.3 Bibliothèques



FIGURE 4.1 – Logos des différentes bibliothèques Python utilisées

- **NumPy** : Une bibliothèque Python pour le calcul numérique. Elle fournit des objets multi-dimensionnels puissants et efficaces pour travailler avec des données numériques. NumPy est la base de nombreux autres packages scientifiques Python, tels que Pandas et Matplotlib.
- **Pandas** : Une bibliothèque Python pour la manipulation et l'analyse de données, elle offre des structures de données performantes telles que DataFrame et Series pour gérer des données tabulaires. Elle facilite l'importation, le nettoyage, la transformation et l'analyse de données, en permettant également la création de visualisations de données informatives.
- **Matplotlib** : Une bibliothèque Python de visualisation de données 2D très populaire. Elle permet de créer une grande variété de graphiques, y compris des lignes, des barres, des histogrammes, des scatter plots, des heatmaps, et bien plus encore. Matplotlib est particulièrement appréciée pour sa flexibilité et sa capacité à produire des visualisations de haute qualité qui peuvent être personnalisées de manière approfondie.

- **Scikit-learn** : Scikit-learn (sklearn) est une bibliothèque Python pour l'apprentissage automatique et l'analyse de données, fournissant un large éventail d'algorithmes d'apprentissage automatique supervisé et non supervisé. Elle simplifie la construction, l'entraînement et l'évaluation de modèles de machine learning, permettant d'effectuer des tâches telles que la classification, la régression, le clustering et la réduction de dimensionnalité.
- **Tensorflow** : TensorFlow est une bibliothèque open-source pour le calcul numérique à grande échelle, principalement utilisée pour le deep learning. Elle fournit des outils puissants pour la construction, l'entraînement et le déploiement de modèles de deep learning, permettant de tirer parti des GPU et des TPU pour accélérer les calculs, tout en offrant une flexibilité pour construire des modèles complexes.
- **Keras** : Keras est une bibliothèque Python pour le deep learning, construite sur TensorFlow, offrant une interface de haut niveau pour la construction de réseaux neuronaux profonds. Elle simplifie la définition de l'architecture du réseau, l'entraînement et l'évaluation, permettant ainsi de créer des modèles de deep learning pour des tâches telles que la classification d'images, le traitement du langage naturel et la génération de texte.
- **PyTorch** : Une bibliothèque open-source pour le deep learning, similaire à TensorFlow, elle propose un paradigme de programmation dynamique et basé sur les graphes pour la construction de modèles de deep learning. Elle offre une grande flexibilité et un contrôle précis sur le calcul des gradients, étant largement utilisée dans la recherche et le développement en deep learning.
- **Hugging Face** : La plateforme Hugging Face est un écosystème complet pour le développement, le partage et le déploiement de modèles de traitement du langage naturel et d'autres types de modèles d'apprentissage automatique. Elle comprend notamment la bibliothèque transformers.
- **Transformers** : Transformers est une bibliothèque Python pour le traitement du langage naturel, basée sur l'architecture de réseau neuronal Transformer. Elle fournit des modèles pré-entraînés et des outils pour des tâches de TALN telles que la classification de texte, la traduction automatique et la génération de texte, permettant d'obtenir des performances remarquables dans divers domaines du traitement automatique du langage naturel.
- **NLTK** : NLTK (Natural Language Toolkit) est une bibliothèque Python complète pour le traitement du langage naturel, offrant un large éventail d'outils pour des tâches du TALN telles que la tokenisation, la lemmatisation et l'analyse syntaxique. Elle permet de traiter et d'analyser de grandes quantités de texte de manière efficace.
- **Fuzzywuzzy** : Une bibliothèque Python pour le calcul de la similarité entre des chaînes de caractères. Elle fournit des algorithmes pour mesurer la ressemblance phonétique et sémantique entre des mots ou des phrases, utile pour des tâches telles que la correction d'orthographe et la détection de doublons. Fuzzywuzzy offre aussi des fonctions telles que ratio(), partial\_ratio(), token\_set\_ratio() et fuzz.levenshtein\_distance() pour calculer les différentes mesures de similarité.
- **Langid** : La bibliothèque langid en Python est un outil d'identification automatique des langues d'un texte. Elle utilise des modèles d'apprentissage automatique pour classer le texte dans un ensemble prédéfini de langues en fonction de son contenu, fournissant à la fois la langue détectée

et un score de confiance.

- **Gensim :** Gensim est une bibliothèque Python pour le traitement du langage naturel qui offre un large éventail d'outils pour modéliser et analyser des textes. Elle est particulièrement connue pour ses fonctionnalités de modélisation de sujets, d'apprentissage automatique et de traitement vectoriel du langage. Gensim est utilisée dans une variété de domaines, notamment l'analyse de sentiments, la recommandation de systèmes et la classification de textes.
- **Streamlit :** Un cadre Python pour la création d'applications Web interactives. Il permet de créer des interfaces utilisateur riches et réactives avec peu de code. Streamlit est idéal pour le prototypage rapide d'idées d'apprentissage automatique et de science des données.

## 4.3 Expérimentations

Cette section détaille les diverses expérimentations menées pour développer et évaluer des modèles de détection des discours de haine. Deux ensembles de données spécifiques ont été décrits : Algerian Dialect Toxicity Speech Dataset et OffensEval2020 Dataset. Ensuite, nous aborderons les mesures de performance utilisées pour évaluer les modèles. Enfin, les résultats des différentes approches évaluées seront présentés, incluant les baselines de machine learning, les modèles basés sur l'apprentissage profond, les modèles hybrides, et les Les modèles de langage pré-entraînés (LLMs).

### 4.3.1 Description des données expérimentales

Dans cette étude, deux ensembles de données ont été utilisés, l'un pour le dialecte algérien pur [45] et l'autre qui n'est pas biaisé envers un dialecte arabe spécifique [71]. Ci-dessous se trouve une description détaillée de leurs caractéristiques.

#### 4.3.1.1 Algerian Dialect Toxicity Speech Dataset

- **Description des données :** Le dataset de Mazari et Kheddar (2023)[45] a été créé en collectant des données textuelles en dialecte algérien à partir de Facebook, YouTube et Twitter. Il couvre 11 sujets variés, tels que la politique et les problèmes sociaux. Ce corpus se compose de 14 150 commentaires rédigés en lettres arabes, en arabe transcrit en alphabet latin (Arabizi), parfois en français ou en anglais, ou encore d'une combinaison de ces langues. Trois annotateurs, maîtrisant le dialecte algérien, ont manuellement étiqueté les données pour déterminer si chaque commentaire contenait un discours de haine, de la cyberintimidation ou un langage offensant (un commentaire pouvant contenir les trois simultanément). Cependant, notre étude se concentre exclusivement sur la détection du discours haineux. Ils ont identifié 7573 instances de discours haineux et 6577 instances de discours non-haineux. Les statistiques détaillées sont présentées dans le tableau ci-dessous :

Hate Speech	Offensive Language	Cyberbullying			
Yes	No	Yes	No	Yes	No
7573	6577	5766	8384	4524	9626

TABLE 4.1 – Répartition d'ensemble des données

La répartition des classes "discours haineux" et "non discours haineux" dans le dataset est illustrée dans l'histogramme ci-après.

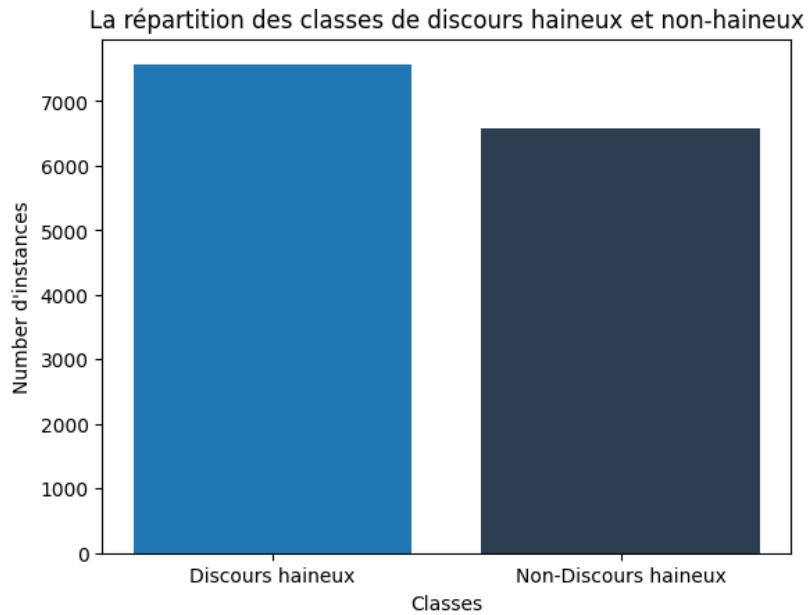


FIGURE 4.2 – Répartition des classes discours haineux et non discours haineux dans le dataset Algerian dialect toxicity speech.

- **Préparation des données :** Le dataset est réparti en 80% pour l'entraînement, 20% pour les tests. Le nombre d'instances de chaque classe pour les ensembles d'entraînement et de test est présenté dans le tableau suivant :

Ensemble d'entraînement		Ensemble de test	
Yes	No	Yes	No
6013	5307	1561	1270

TABLE 4.2 – Répartition des classes dans les ensembles d'entraînement et de test dans Algerian dialect toxicity speech dataset

#### 4.3.1.2 OffensEval2020 Dataset

- **Description des données :** L'ensemble de données arabe OffensEval2020 ciblait spécifiquement le contenu potentiellement offensant collecté sur Twitter par Zampieri et al. (2020)[71], inclut plusieurs langues, dont l'arabe qui nous intéresse dans notre étude. Les auteurs mentionnent que l'ensemble de données ne privilie pas de dialectes, de sujets ou de genres spécifiques, afin d'enrichir le corpus avec des tweets susceptibles d'être offensants. La collecte des données a été réalisée en utilisant l'API Twitter.

Le schéma d'annotation propose une modélisation hiérarchique du langage offensant. Il classe chaque tweet en utilisant la hiérarchie à trois niveaux pour l'anglais, Cependant, d'après les auteurs la seule annotation effectuée pour la langue arabe concerne le niveau A, à savoir la détection du langage offensant qui sera utilisé dans notre étude.

**Niveau A (subtask\_A) :** Détection du langage offensant

Le texte est-il offensant (OFF) ou non offensant (NOT) ? NOT : texte qui n'est ni offensant, ni grossier ; OFF : texte contenant du langage inapproprié, des insultes ou des menaces.

**Niveau B (subtask\_B)** : Catégorisation du langage offensant

Le texte offensant est-il ciblé ou non ciblé ?

**Niveau C (subtask\_C)** : Identification de la cible du langage offensant

Qui ou quoi est la cible du contenu offensant ?

L'ensemble de données est constitué de 10 000 instances, divisé par les auteurs en un ensemble d'entraînement et un ensemble de test contenant respectivement 8000 tweets et 2000 tweets. Le nombre de chaque classe dans chaque ensemble est montré par le tableau suivant :

Ensemble d'entraînement		Ensemble de test	
Yes	No	Yes	No
1 589	6 411	402	1 598

TABLE 4.3 – Répartition des ensembles d'entraînement et de test dans OffensEval2020 dataset

— **Préparation des données** : L'ensemble de données est considérablement déséquilibré, la classe offensive ayant significativement moins d'instances comparée à la classe non-offensive. Pour remédier à ce déséquilibre dans les données d'entraînement, nous avons appliqué une approche d'augmentation de données. Cela a impliqué la génération de nouvelles instances pour la classe offensive en utilisant le modèle AraGPT-2[68] qui est un Transformer pré-entraîné pour la génération de langage en arabe, à partir des instances originales. Les auteurs de AraGPT-2 ont mentionné que le modèle a été évalué en utilisant la mesure de perplexité, qui évalue la capacité d'un modèle probabiliste à prédire un échantillon. Les résultats montrent que AraGPT-2 est capable de produire un texte arabe de haute qualité, cohérent, grammaticalement correct et syntaxiquement solide. Par ailleurs, l'ensemble officiel de test est resté inchangé.

Nous avons commencé par concaténer la classe et le texte pour créer les instances d'entrée, puis nous les avons transmis au modèle afin d'apprendre les relations de dépendance entre les mots et entre la classe et le texte. Ce qui permet de guider le modèle pour contrôler la génération de texte et garantir qu'il reste spécifique à la classe offensive.

Cette approche s'inspire de l'étude de Habbat et al. (2023) intitulée "Using AraGPT and Ensemble Deep Learning Model for Sentiment Analysis on Arabic Imbalanced Dataset" [72].

Voici un exemple de commentaire généré par le modèle AraGPT-2 à partir d'un commentaire de classe offensive provenant de l'ensemble d'entraînement original :

Commentaire original	Commentaire généré avec AraGPT-2
gross الفاشل بىشى في مليون داهيه	طويت رجلك جross الفاشل بىشى في مليون
ذهب بلا عوده الله لا يرجعك أبدا لزمالك تانى	داهيه لا يا ريت الراجل ده لا يرجع
يا فاشل يا حمار يا حمار يا حمار	أبدا لزمالك تانى يا عاجز مافيش فايدة منك أبدأ

TABLE 4.4 – Exemple de commentaire généré par le modèle AraGPT-2

Les figures ci-dessous montrent la répartition des classes offensive et non offensive sur l'ensemble d'entraînement d'OffensEval2020 avant et après l'augmentation avec AraGPT-2.

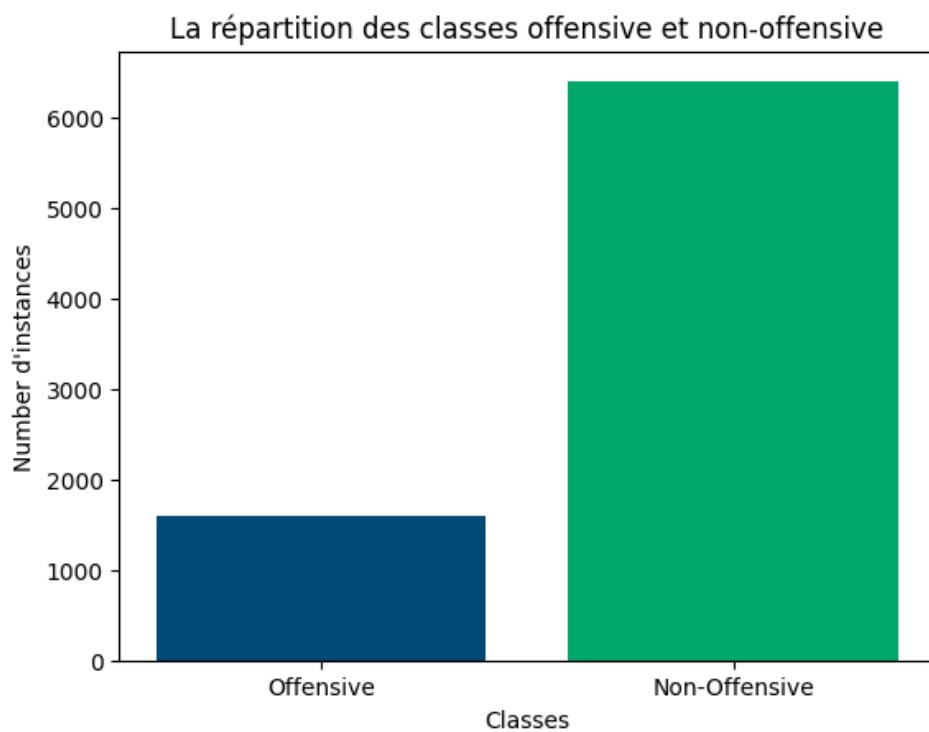


FIGURE 4.3 – La distribution des classes dans l’ensemble d’entraînement OffessEval2020 avant l’augmentation des données

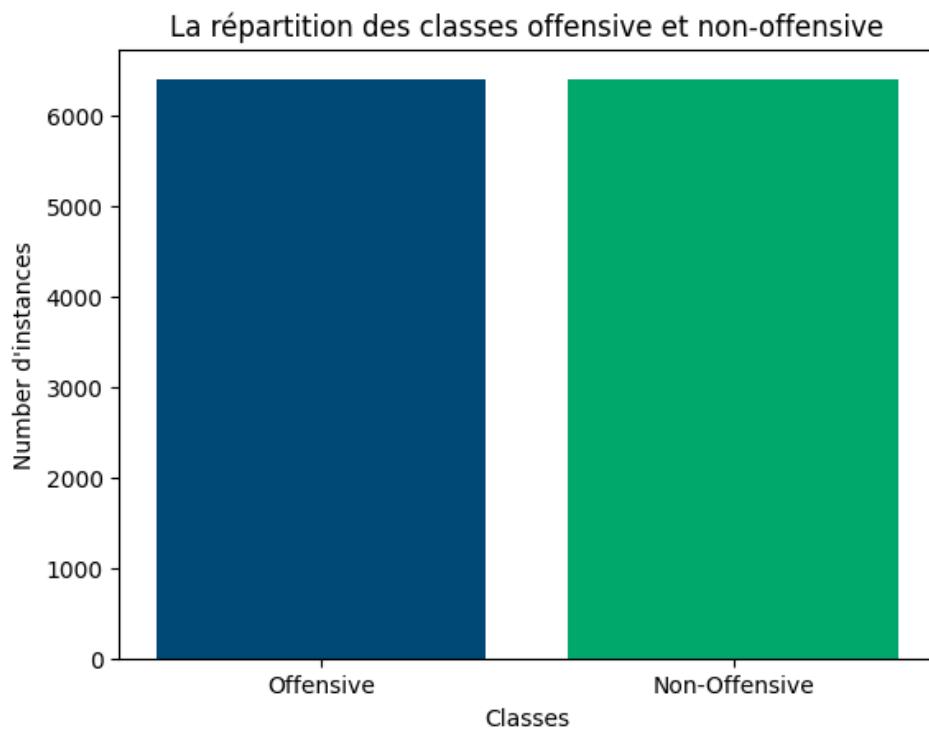


FIGURE 4.4 – La distribution des classes dans l’ensemble d’entraînement OffessEval2020 après l’augmentation des données

### 4.3.2 Mesures de performance pour l'évaluation

Les métriques utilisées pour évaluer cette étude comprennent l'exactitude, la précision, le rappel et le score F1. Leurs définitions et leurs formules sont présentées ci-après.

- **Exactitude (Accuracy)** : L'exactitude est la proportion de prédictions correctes parmi l'ensemble des prédictions effectuées. C'est la métrique d'évaluation la plus simple et intuitive, souvent utilisée pour mesurer la performance globale d'un modèle de classification. Cependant, elle peut être trompeuse dans les situations où il existe un déséquilibre significatif entre les classes dans le jeu de données. Dans de tels cas, un modèle pourrait obtenir une haute exactitude en favorisant la prédiction de la classe majoritaire, sans pour autant fournir une bonne performance sur la classe minoritaire. Sa formule se présente comme suit :

$$\text{Exactitude} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (4.1)$$

- **Précision (Precision)** : La précision mesure la proportion de prédictions positives qui sont effectivement correctes. En d'autres termes, elle indique la fiabilité des résultats positifs en montrant combien de prédictions positives sont réellement des cas positifs. Elle se calcule comme le rapport entre le nombre de vrais positifs et le nombre total de prédictions positives, sous la formule suivante :

$$\text{Précision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (4.2)$$

- **Rappel (Recall)** : Le rappel mesure la proportion des véritables éléments positifs qui ont été correctement identifiés par le modèle. En d'autres termes, il indique l'exhaustivité des résultats positifs en montrant combien de cas positifs réels ont été correctement prédits comme tels. Il se calcule comme le rapport entre le nombre de vrais positifs et le nombre total d'éléments positifs réels à l'aide de la formule suivante :

$$\text{Rappel} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (4.3)$$

- **F1-Score** : Le F1-Score est la moyenne harmonique de la précision et du rappel. Il s'agit d'une mesure équilibrée qui combine à la fois la fiabilité (précision) et l'exhaustivité (rappel) des résultats. Le F1-Score est particulièrement utile lorsque les classes sont déséquilibrées, car il fournit une évaluation unique qui prend en compte les deux aspects de la performance du modèle, sous la formule suivante :

$$\text{F1 Score} = \frac{2 \cdot \text{Précision} \cdot \text{Rappel}}{\text{Précision} + \text{Rappel}} \quad (4.4)$$

où :

- TP : True Positives
- FP : False Positives
- TN : True Negatives
- FN : Flase Negatives

### 4.3.3 Evaluation

L'évaluation des modèles de machine learning, deep learning, LLMs et des modèles hybrides est effectuée sur les ensembles de données Algerian Toxicity Speech et OffensEval2020. Ce dernier est

évalué avant et après l'augmentation de données avec AraGPT-2, ce qui permet de mesurer l'impact de l'augmentation des données sur la capacité des modèles à détecter et à classer correctement les discours offensifs.

#### 4.3.3.1 Machine learning baselines

Ce tableau (Table 4.5) présente les performances de différents classificateurs de machine learning traditionnels sur les deux ensembles de données utilisés dans cette étude. Les classificateurs évalués comprennent les machines à vecteurs de support (SVM), les forêts aléatoires (Random Forest ou RF) et la descente de gradient stochastique (Stochastic Gradient Descent ou SGD). Pour le classificateur SVM, nous avons expérimenté avec différentes techniques de représentation des textes, notamment le TF-IDF (Term Frequency-Inverse Document Frequency) et les embeddings Skip-Gram issus de l'approche Word2Vec. Les mesures de performance présentées sont l'exactitude (Accuracy), la précision, le rappel et le score F1. Ces résultats de référence serviront de base de comparaison avec les modèles d'apprentissage profond et les modèles de langages pré-entraînés (LLM) explorés ultérieurement. L'objectif est d'évaluer les forces et les faiblesses de ces approches classiques pour la tâche de détection de discours haineux en langue arabe, y compris pour le dialecte algérien spécifique présent dans le premier ensemble de données.

	<b>Acc</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>
<b>Algerian Dialect Toxicity Speech Dataset</b>				
SVM + TF-IDF	0.69	0.69	0.69	0.69
SVM + Skip Gram	0.61	0.62	0.61	0.61
RF	0.66	0.66	0.66	0.66
SGD	0.69	0.69	0.69	0.69
<b>OffensEval2020 Dataset Before Augmentation</b>				
SVM + TF-IDF	0.89	0.88	0.89	0.89
SVM + Skip Gram	0.80	0.63	0.80	0.70
RF	0.88	0.88	0.88	0.86
SGD	0.89	0.89	0.88	0.89
<b>OffensEval2020 Dataset After Augmentation</b>				
SVM + TF-IDF	0.89	0.90	0.90	0.89
SVM + Skip Gram	0.80	0.64	0.80	0.71
RF	0.88	0.88	0.88	0.87
SGD	0.89	0.90	0.89	0.89

TABLE 4.5 – Performances des modèles ML sur les ensembles de données

#### 4.3.3.2 Modèles basés sur l'apprentissage profond

Le tableau qui suit (Table 4.6) illustre les résultats obtenus par différents modèles de deep learning sur les deux ensembles de données utilisés. Les modèles évalués comprennent deux architectures dérivées des réseaux neuronaux récurrents (RNN) : Bidirectional Long Short-Term Memory (Bi-LSTM), où nous avons expérimenté avec deux embeddings, FastText et AraVec, pour déterminer la meilleure représentation textuelle, et Bidirectional Gated Recurrent Unit (Bi-GRU) avec FastText, ainsi que les réseaux de neurones convolutifs (CNN) où des embeddings apprenables ont été utilisés. Les mesures de performance présentées incluent l'exactitude (Accuracy), la précision, le rappel et le

score F1. L'objectif est d'évaluer les performances des architectures RNN reconnues pour leur robustesse dans le traitement des données séquentielles en particulier les données textuelles, par rapport aux CNN pour la tâche de classification des discours haineux en langue arabe avec ses différents dialectes.

	<b>Acc</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>
	<b>Algerian Dialect</b>	<b>Toxicity Speech Dataset</b>		
CNN	0.68	0.68	0.68	0.68
Bi-LSTM + FastText	0.70	0.70	0.70	0.70
Bi-LSTM + AraVec	0.67	0.68	0.67	0.67
Bi-GRU	0.71	0.70	0.70	0.70
<b>OffensEval2020 Dataset Before Augmentation</b>				
CNN	0.89	0.88	0.87	0.89
Bi-LSTM + FastText	0.90	0.90	0.89	0.90
Bi-LSTM + AraVec	0.90	0.90	0.90	0.89
Bi-GRU	0.91	0.91	0.91	0.91
<b>OffensEval2020 Dataset After Augmentation</b>				
CNN	0.91	0.90	0.90	0.91
Bi-LSTM + FastText	0.92	0.92	0.92	0.92
Bi-LSTM + AraVec	0.92	0.92	0.92	0.92
Bi-GRU	0.92	0.92	0.92	0.92

TABLE 4.6 – Performances des modèles DL sur les ensembles de données

#### 4.3.3.3 LLMs

Le tableau suivant (Table 4.7) présente une comparaison des LLMs utilisés dans cette étude, notamment ceux décrits dans la section conceptuelle du chapitre 3, en termes de l'année de sortie, de l'architecture, du nombre de paramètres, des données d'entraînement et des caractéristiques.

Model	Year Released	Architecture	Parameters	Training Data	Features
<b>AraBERT-V2</b>	2020	Transformer (Enc) Same as BERTBase architecture	136M	Trained on Arabic tweets Sentences : 200M size : 77GB nWords : 8.6B	for Arabic dialects and tweets
<b>DziriBERT</b>	2021	Transformer (Enc), Same as BERTBase architecture	124M	1.1M tweets (20M tokens)	Handles Algerian text contents written using both Arabic and Latin characters

<b>DzAraBERT</b>	2023	Transformer (Enc) Based on DziriBERT architecture	113M	1.1M tweets (20M tokens)	A pruned model of DziriBERT, It handles Al- gerian text contents written using Arabic letters
<b>GigaBERT- V4</b>	2020	Transformer (Enc) Same as BERTBase architecture	110 M	It was pre- trained in a large-scale cor- pus (Gigaword+ Oscar+ WikiPe- dia) with 10B tokens	A customized bilingual BERT for English and Arabic, State-of-the-art zero-shot trans- fer performance from English to Arabic on information extraction (IE) tasks.
<b>Multilingual BERT-base</b>	2019	Transformer (Enc)	179M	BooksCorpus : 800M words	Pretrained mo- del on the top 104 languages
<b>AraGPT-2- base</b>	2021	Transformer (Dec)	135M	The data- set consists of 77GB / 200,095,961 lines / 8,655,948,860 words / 82,232,988,358 chars	Pre-Trained Transformer for Arabic Language Gene- ration
<b>Llama3</b>	2024	Transformer (Enc)	8 Billion - 70 bil- lion	15T tokens of public text data	State-of-the-art performance, able to handle longer context

TABLE 4.7: Comparaison des différents LLMs utilisés

Le tableau ci-après (Table 4.8) expose les performances des modèles de langage pré-entraînés déjà évoqués dans le tableau précédent (Table 4.7), suite au finetuning effectué sur ces modèles avec les deux ensembles de données utilisés dans notre étude. Leurs performances sont évaluées en termes d'exactitude (Accuracy), de précision, de rappel et de score F1. L'objectif est d'exploiter la robustesse des modèles pré-entraînés qui reposent sur les Transformers pour leur fonctionnement interne qui constitue une avancée majeure en traitement du langage naturel, ce qui en fait une approche performante pour la détection des discours haineux dans plusieurs dialectes arabes.

	<b>Acc</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>
<b>Algerian Dialect Toxicity Speech Dataset</b>				
AraBERTv02-base	0.79	0.79	0.79	0.79
DziriBERT	0.77	0.76	0.77	0.77
DzaraBERT	0.76	0.76	0.76	0.76
GigaBERT-V4	0.75	0.75	0.75	0.75
MultilingualBERT-base	0.74	0.74	0.74	0.74
AraGPT-2-base	0.65	0.64	0.64	0.64
Llama3	0.76	0.76	0.76	0.76
<b>OffensEval2020 Dataset Before Augmentation</b>				
AraBERTv02-base	0.94	0.94	0.94	0.94
DziriBERT	0.92	0.92	0.92	0.92
DzaraBERT	0.92	0.92	0.92	0.92
GigaBERT-V4	0.90	0.90	0.90	0.90
MultilingualBERT-base	0.89	0.89	0.89	0.89
AraGPT-2-base	0.88	0.82	0.83	0.83
Llama3	0.93	0.93	0.93	0.93
<b>OffensEval2020 Dataset After Augmentation</b>				
AraBERTv02-base	0.95	0.95	0.95	0.95
DziriBERT	0.94	0.94	0.94	0.94
DzaraBERT	0.94	0.94	0.95	0.94
GigaBERT-V4	0.91	0.91	0.91	0.91
MultilingualBERT-base	0.90	0.90	0.90	0.90
AraGPT-2-base	0.89	0.82	0.85	0.83
Llama3	0.94	0.94	0.94	0.94

TABLE 4.8: Métriques de performance pour différents datasets et LLMs

#### 4.3.3.4 Modèles hybrides

Dans le tableau ci-dessous, les performances des modèles hybrides sont exposées, ces modèles visent à enrichir le classificateur SVM et le modèle CNN en tirant parti des embeddings contextuels d’AraBERT. Ces embeddings offrent davantage de robustesse par rapport aux embeddings non contextuels tels que Word2Vec, utilisés précédemment. Nous avons également évalué le modèle combiné CNN + Bi-GRU, qui associe la capacité du CNN à saisir les hiérarchies spatiales dans les données grâce aux couches de convolutions avec la robustesse du Bi-GRU dans la gestion des dépendances séquentielles. Les performances sont évaluées à l'aide de quatre métriques : l'exactitude (Accuracy), la précision, le rappel et le score F1.

	<b>Acc</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>
<b>Algerian Dialect Toxicity Speech Dataset</b>				
AraBERTv02 + CNN	0.77	0.77	0.77	0.77
AraBERTv02 + SVM	0.73	0.74	0.73	0.73
CNN + Bi-GRU	0.70	0.70	0.70	0.70
<b>OffensEval2020 Dataset Before Augmentation</b>				
AraBERTv02 + CNN	0.93	0.92	0.93	0.93
AraBERTv02 + SVM	0.89	0.90	0.89	0.89
CNN + Bi-GRU	0.89	0.90	0.89	0.89
<b>OffensEval2020 Dataset After Augmentation</b>				
AraBERTv02 + CNN	0.94	0.94	0.94	0.94
AraBERTv02 + SVM	0.91	0.90	0.89	0.89
CNN + Bi-GRU	0.91	0.91	0.91	0.91

TABLE 4.9: Métriques de performance pour différents datasets et modèles

## 4.4 Analyse et discussion de résultats

Cette section se consacre à l’analyse et à la discussion détaillée des résultats des expérimentations. Elle débute par une analyse de la performance des modèles évalués, mettant en lumière les performances des différents modèles proposés pour la détection des discours de haine. Ensuite, les aspects relatifs à l’entraînement et au temps d’exécution des modèles sont abordés, puis une analyse des erreurs est menée afin d’identifier où le modèles les plus performants ont des difficultés à classifier correctement les tweets comme discours de haine ou non. Enfin, une comparaison de nos résultats avec ceux des études antérieures portant sur les mêmes ensembles de données.

### 4.4.1 Performance

Les modèles pré-entraînés (LLMs) surpassent globalement les modèles de deep learning et de machine learning en raison de leur capacité à saisir les nuances des dialectes arabes, notamment le dialecte algérien, qui présente un vocabulaire étendu reflétant les particularités du dialecte algérien, ainsi qu’une diversité de sujets comprenant 11 catégories distinctes sont inclus dans le l’ensemble de données Algerian Dialect Toxicity Speech et de nombreuses subtilités, rendant ainsi leur classification difficile.

Les résultats issus de divers modèles expérimentaux mettent en évidence les subtilités impliquées dans la détection des discours de haine dans le dialecte algérien. Alors que les approches traditionnelles de machine learning et de deep learning ont fourni quelques insights précieux, elles ont échoué à saisir les caractéristiques nuancées du dialecte algérien. Contrairement au dataset OffensEval2020, où les modèles de deep learning et de machine learning ont obtenu des résultats satisfaisants. En revanche, le modèle AraBERTv02 entraîné sur des tweets en arabe s’est imposé comme l’approche la plus réussie, soulignant le rôle crucial des modèles basés uniquement sur des encodeurs dans le

domaine des études de cette nature.

DziriBERT et DzaraBERT, une version modifiée de DziriBERT, ont démontré des performances quasiment équivalentes sur les deux ensembles de données, se plaçant ainsi comme les deuxièmes meilleurs modèles après AraBERT, ainsi que Llama3 pour le cas de l'ensemble de données OffensEval2020.

MultilingualBERT et GigaBERT ont donné des résultats comparables, bien que GigaBERT présente une légère supériorité. Cette différence pourrait s'expliquer par le fait que GigaBERT se spécialise exclusivement dans l'arabe et l'anglais, tandis que MultilingualBERT est conçu pour traiter 104 langues différentes.

Les tests effectués sur deux ensembles de données ont révélé que les modèles de langage pré-entraînés basés sur l'architecture BERT surpassent le modèle AraGPT-2-base en termes de performance. Par ailleurs, Llama3 a été parmi les modèles les plus performants pour les deux ensembles de données. En raison des limitations de la version gratuite de Google Colab, nous avons utilisé les versions « base » des modèles AraGPT2 et AraBERT. Des performances accrues pourraient être obtenues avec un entraînement plus long et l'utilisation de modèles plus volumineux « large ».

Les architectures issues des réseaux récurrents (RNN), telles que Bi-LSTM et Bi-GRU, surpassent les performances de CNN. Cette supériorité démontre la pertinence des RNN pour l'analyse de texte. En effet, leur capacité à traiter les séquences d'entrée dans les deux sens leur confère une plus grande robustesse dans la capture du sens du texte.

TF-IDF s'est avérée être l'approche la plus performante par rapport à Word2Vec skip-gram pour les classificateurs traditionnels d'apprentissage automatique. Sur l'ensemble de données OffensEval2020, les embeddings fastText et AraVec ont obtenu des résultats identiques avec le modèle Bi-LSTM. Tandis que fastText a surpassé AraVec sur le Algerian Dialect Toxicity Speech Dataset ce qui démontre que FastText est mieux adapté au dialecte algérien, car ce modèle de word embedding est entraîné sur des données textuelles collectées sur la plateforme Facebook, incluant le dialecte algérien. Pour les modèles SVM et CNN, les embeddings contextuels d'AraBERTv02 ont obtenu les meilleurs résultats, surpassant les autres options.

L'augmentation de données appliquée à l'ensemble de données OffensEval2020 a permis d'obtenir des améliorations des performances pour la quasi-totalité des modèles évalués, contribuant à enrichir l'ensemble d'entraînement avec de nouveaux exemples synthétiques, ce qui a permis aux modèles d'apprendre des patterns plus subtils et d'améliorer leur capacité à généraliser à de nouvelles données.

Les modèles examinés dans notre étude pour la détection des discours de haine ont montré de meilleures performances avec l'ensemble de données OffensEval2020 qu'avec le dataset Algerian Toxicity speech. Cela indique la complexité du dialecte algérien, qui présente des défis uniques pour le traitement automatique du langage. Ces résultats soulignent la nécessité de développer des ressources linguistiques plus robustes et des techniques adaptées pour traiter les particularités des dialectes régionaux.

## 4.4.2 Entraînement et Temps d'exécution

Cette section se concentre sur les aspects liés à l'entraînement des modèles et le temps d'exécution associé. Tout d'abord, nous explorerons le processus d'entraînement des modèles, en incluant les configurations spécifiques des modèles d'apprentissage profond et des modèles de langage pré-entraînés (LLMs) utilisés, ainsi que les courbes d'entraînement illustrant la perte et l'exactitude. Ensuite, le temps nécessaire pour entraîner chaque modèle sera fourni.

### 4.4.2.1 Entraînement

En raison du temps d'exécution GPU court et de la RAM GPU limitée sur Colab, nous n'avons pas pu utiliser de techniques de recherche d'hyperparamètres. À la place, nous avons varié manuellement les hyperparamètres, en basant les intervalles sur des études précédentes menées pour certains modèles de langage (LLM). Les hyperparamètres incluant le nombre d'époques, le batch size et taux d'apprentissage (Learning rate) utilisés pour obtenir les résultats précédemment présentés sont répertoriés dans ce tableau :

Model	Algerian Dialect Toxicity Dataset			OffensEval2020 Dataset		
	Epochs	Batch-size	Learning-rate	Epochs	Batch-size	Learning-rate
CNN	10	32	1e-3	10	32	1e-3
Bi-LSTM + Fast-Text	10	16	1e-3	8	16	1e-3
Bi-LSTM + Ara-Vec	5	16	1e-3	5	16	1e-3
Bi-GRU	10	16	1e-3	3	32	1e-3
AraBERTv02-base	2	32	1e-4	6	16	2e-5
DziriBERT	5	16	2e-5	5	16	2e-5
DzaraBERT	5	16	2e-5	5	16	2e-5
GigaBERT-V4	5	32	1e-5	5	32	1e-5
MultilingualBERT-base	4	16	2e-5	4	16	2e-5
AraGPT-2-base	5	32	1e-5	5	16	1e-5
CNN + Bi-GRU	2	32	1e-3	3	32	1e-3
AraBERT + CNN	2	32	2e-5	2	16	5e-5
LLama3	1	1	2e-4	1	1	2e-4

TABLE 4.10 – Configuration d'entraînement pour différents modèles sur les deux ensembles de données Algerian Dialect Toxicity et OffensEval2020

Pour le modèle CNN+GRU, 2 époques ont été suffisantes pour obtenir de meilleurs résultats, car lors de l'entraînement du modèle pendant plus longtemps, il surapprend rapidement et produit de faibles performances.

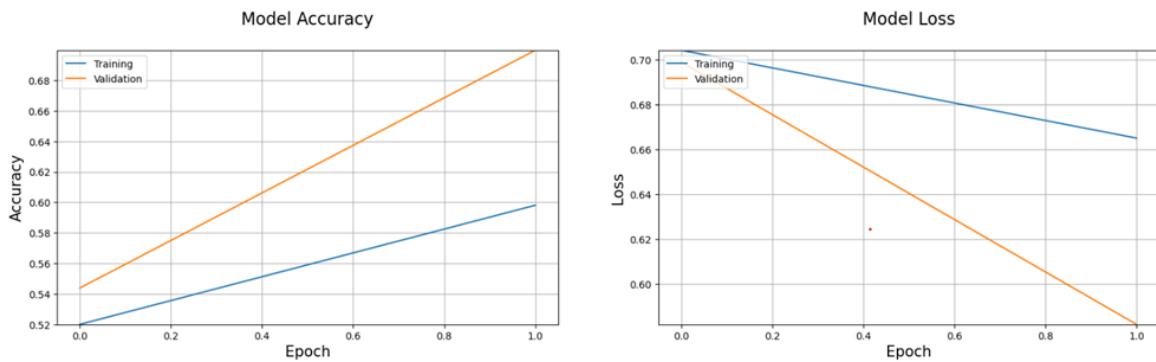


FIGURE 4.5 – Courbes de l'exactitude et de perte en fonction des époques pour le modèle CNN+GRU utilisant le dataset Algerian Toxicity speech pour 2 époques.

Lorsque le modèle est entraîné avec 10 époques, il présente un surajustement et devient moins performant comme le montre la courbe ci-dessous :

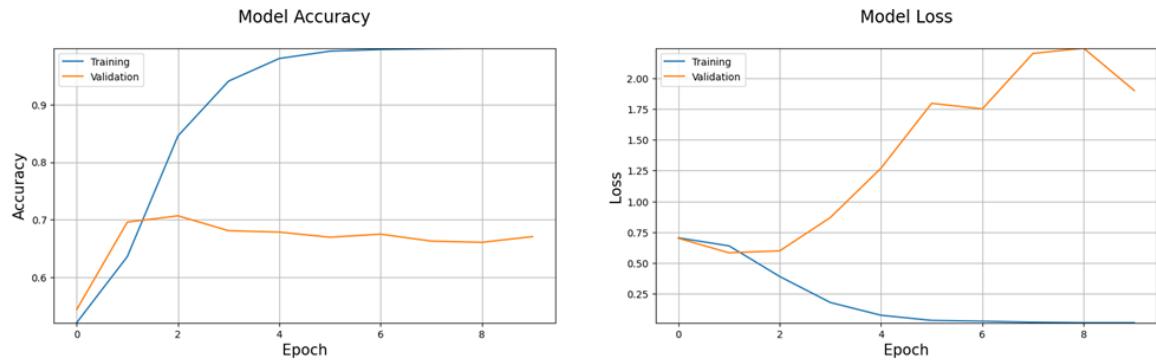


FIGURE 4.6 – Courbes de l'exactitude et de perte en fonction des époques pour le modèle CNN+GRU utilisant le dataset Algerian Toxicity speech pour 10 époques

#### 4.4.2.2 Temps d'exécution

Les tests ont été effectués sur Google Colab en utilisant un GPU Tesla T4. Le tableau ci-dessous présente le temps d'exécution en minutes de chaque modèle sur chaque ensemble de données :

Model	Algerian Toxicity Dataset (min)	Dialect Speech Dataset (min)	OffensEval2020 Dataset (min)
CNN	3	3	3
Bi-LSTM	11	9	
Bi-LSTM + AraVec	5	4	
Bi-GRU	8	3	
AraBERTv02-base	6	30	
DziriBERT	25	23	
DzaraBERT	26	23	
GigaBERT-V4	22	25	
MultilingualBERT-base	19	16	
AraGPT-2-base	21	19	
CNN + Bi-GRU	2	3	
AraBERT + CNN	50	45	
AraBERT + SVM	55	40	
LLama3	149	200	

TABLE 4.11: Temps d’entraînement pour différents modèles sur les deux ensembles de données

#### 4.4.3 Analyse des erreurs

L’objectif de cette analyse est d’examiner les erreurs de classification rencontrées par le modèle le plus performant AraBERTv02 sur les deux ensembles de données utilisés. Nous allons analyser les matrices de confusion pour identifier les tendances et les causes possibles des faux positifs et des faux négatifs. Ces analyses fournissent des insights précieux sur les limitations du modèle et les défis posés par les erreurs d’étiquetage ainsi que les nuances linguistiques des tweets analysés.

En commençant par la matrice de confusion obtenue par AraBERTv02 sur l’ensemble de données Algerian Toxicity Speech montrée dans la figure suivante :

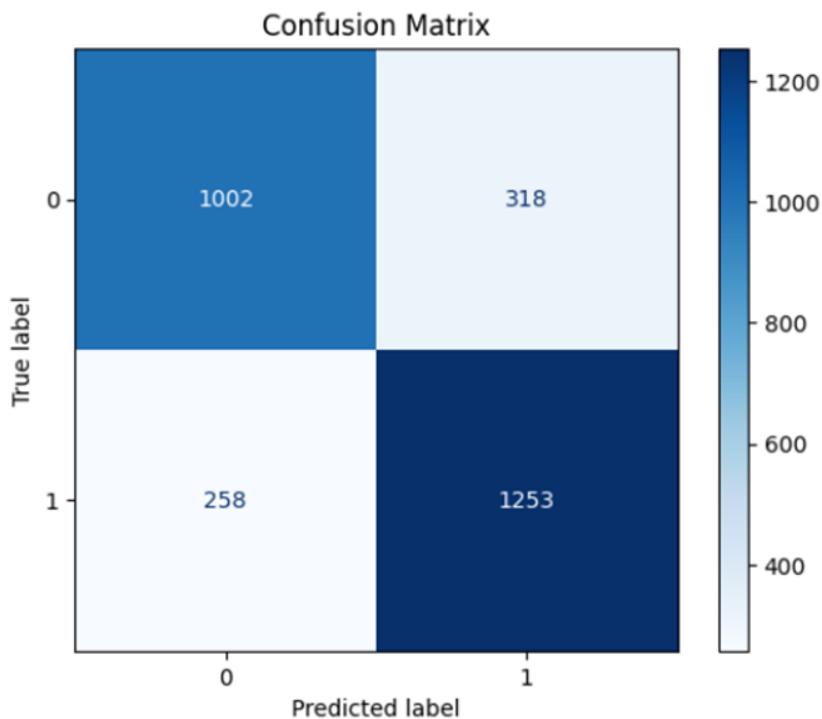


FIGURE 4.7 – Matrice de confusion de AraBERTv02 avec Algerian Dialect Toxicity Speech

Après l'analyse des faux positifs observés sur l'ensemble de données Algerian Toxicity Speech avec AraBERTv02, il est constaté que les instances classées comme discours haineux alors que le label réel est non haineux sont dans la plupart des cas mal étiquetées. Cela signifie qu'elles abordent souvent des discours haineux ou parfois contiennent des mots de discours haineux, mais l'intention n'est pas de propager de la haine.

Ci-dessous se trouvent quelques instances de faux positifs :

Tweet	Classe réelle	Classe prédictée
فليم هندي كان خاص غي لعن والرقص الله ابهدلكم يا كذاين	NOT	OFF
الخوارج والإخوان أينما حلوا حل الخراب والشر	NOT	OFF

TABLE 4.12 – Prédictions d'AraBERTv02 et classes réelles des faux positifs provenant de l'ensemble de test Algerian Dialect Toxicity Speech

Pour les faux négatifs, il n'y a pas de particularité notable pour les instances que le modèle ne parvient pas à détecter comme discours haineux.

Examinons à présent la matrice de confusion d'AraBERTv02 obtenue sur le dataset OffensEval2020, illustrée ci-dessous.

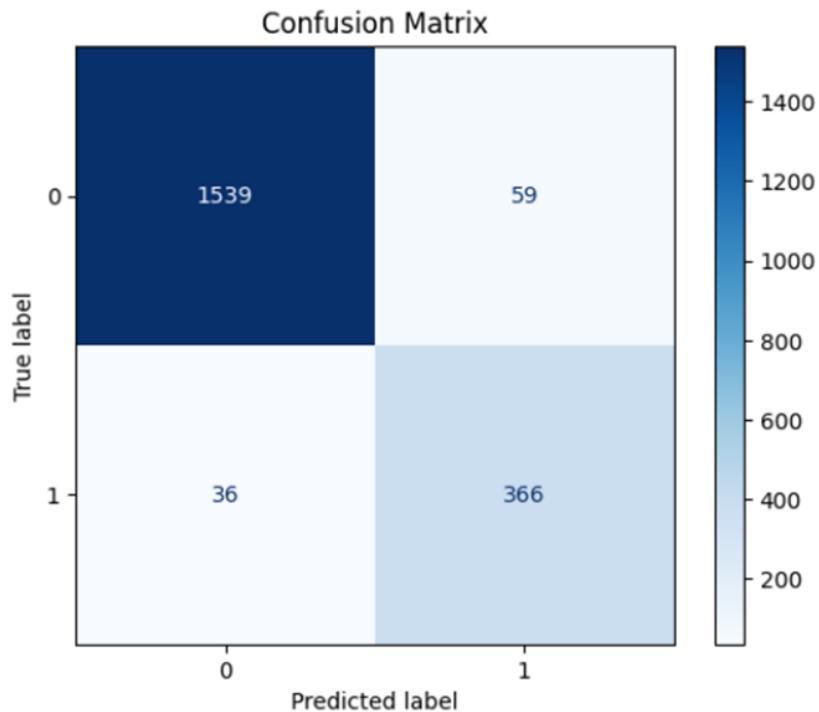


FIGURE 4.8 – Matrice de confusion de AraBERTv02 avec OffensEval2020

Les instances non offensantes mal classées par AraBERTv02 sur le dataset OffensEval2020 sont parfois dues à des erreurs d'étiquetage. Dans d'autres cas, ces instances semblent offensantes car elles contiennent des mots à connotation offensive, bien qu'elles soient annotées comme non offensantes.

Des exemples de faux positifs provenant de l'ensemble de test officiel sont présentés dans ce tableau, avec leur classe réelle et la classe prédictive avec AraBERTv02 correspondante.

Tweet	Classe réelle	Classe prédictive
لا متكمش أقف عندك يا خروف يا إخوانجي يا منساق يا مغيب يا بتاع تركيا وقطر يعني مثلًا ف السياسه تيجي تقول الإخوان كانو كويسين في كذ	NOT	OFF
دا منظر بروفايل واحد هيقدا دكتور؟ ! يا فضيحتك يا وزارة الصحة يا فضيحتك يا مصر	NOT	OFF
يا فحامتك يا ثعبان	NOT	OFF

TABLE 4.13 – Prédiction d'AraBERTv02 et classes réelles des faux positifs provenant de l'ensemble de test OffensEval 2020

Les instances classées comme non offensantes alors qu'elles sont en réalité de classe offensive

sont dues au fait que certains tweets ne contiennent pas de termes offensants, mais sont classés comme tels, bien qu'ils puissent avoir une autre signification ou intention.

Le tableau ci-dessous illustre quelques exemples de faux négatifs :

Tweet	Classe réelle	Classe prédictive
يلا يا انسه يلا يا انسه على مشاريعك	OFF	NOT
يا شريفه يا شريفه يا شريفه	OFF	NOT

TABLE 4.14 – Prédictions d’AraBERTv02 et classes réelles des faux négatifs provenant de l’ensemble de test OffensEval2020

#### 4.4.4 Comparaison de résultats

Dans cette comparaison, nous allons présenter les résultats de différentes études menées sur chacun des deux ensembles de données. L’objectif est de mettre en lumière les performances et les approches variées employées pour chaque ensemble de données en identifiant les plus robustes pour la détection des discours de haine en langue arabe et ses différents dialectes, et de positionner nos propres résultats par rapport à ceux obtenus dans la littérature existante.

##### 4.4.4.1 Algerian Dialect Toxicity Speech Dataset

La seule étude que nous avons trouvée sur le jeu de données Algerian Dialect Toxicity Speech est celle réalisée par les auteurs du jeu de données lui-même, Mazari et Kheddar (2023), intitulée "Deep Learning-based Analysis of Algerian Dialect Dataset Targeted Hate Speech, Offensive Language and Cyberbullying" [45]. Cette étude vise à détecter le discours haineux, le langage offensant et le cyberharcèlement. Cependant, notre étude se concentre exclusivement sur la détection des discours haineux.

Les résultats obtenus par les auteurs sont présentés à l'aide de l'accuracy (exactitude) et du score F1 pour les classificateurs de machine learning et les modèles de deep learning, comme indiqué dans les tableaux ci-dessous :

ML classifiers	TF-IDF		Word embedding (Word2Vec)			
			Skip-Gram		CBOW	
	Acc.	F1-Score	Acc.	F1-Score	Acc.	F1-Score
Random forest	64.7	66.2	55.5	59.5	56.1	60.3
Multinomial NB	66.3	73.6	53.4	68.2	55.1	62.7
SVC	66.6	69.9	55.7	63.4	56.3	63.1
SGD	71.6	72.3	54.3	51.0	56.9	72.4
LR	66.5	71.7	55.6	64.6	55.8	62.9

TABLE 4.15 – Résultats des classificateurs ML

DL classifiers	Word embeddings (FastText)	
	Accuracy	F1-score
LSTM	64.3	68.7
GRU	65.8	65.9
Bi-LSTM	68.3	69.8
Bi-GRU	73.6	75.8
CNN	62.7	64.2

TABLE 4.16 – Performance of DL Classifiers using FastText Word Embeddings

L'étude de Mazari et Kheddar(2023)[45] se concentre exclusivement sur l'utilisation de modèles d'apprentissage profond, sans exploiter la robustesse des modèles préentraînés (LLMs), qui pourraient améliorer les résultats pour les tâches de détection de la toxicité dans le dialecte algérien.

#### 4.4.4.2 OffensEval2020 Dataset

Plusieurs études ont été menées sur l'ensemble de données OffensEval2020, en utilisant diverses approches. Chaque étude est accompagnée d'une description de l'approche utilisée par ses auteurs, ainsi que des résultats obtenus ci-après.

Alami et al. (2020) [73] ont exploré trois approches de modélisation. Le premier modèle, appelé Vanilla, ne procède à aucun prétraitement des emojis. Dans cette méthode, la phrase est directement segmentée à l'aide du segmenteur Farasa, puis tokenisée à l'aide du tokeniseur AraBERT. Le deuxième modèle est AraBERTEmojisIn, substitue les emojis par leurs équivalents en arabe. Quant au dernier modèle, AraBERTEmojisOut, il remplace les emojis par le token spécial [MASK] tout en incluant une nouvelle entrée contenant le sens arabe de chaque emoji dans le tweet, lesquels sont séparés par le token spécial [SEP].

Les résultats obtenus par les auteurs avec les modèles Vanilla, AraBERTEmojisIN et AraBERTEmojisOUT en ce qui concerne l'exactitude (Accuracy), la précision, le rappel et le score F1 sont répertoriés dans le tableau suivant.

Model	Accuracy	Precision	Recall	F1-Score
Weighted average				
Vanilla	93.3	93.19	93.3	93.23
AraBERTEmojisIN	93.3	93.41	93.3	93.35
AraBERTEmojisOUT	93.9	94.38	93.9	94.06
Macro average				
Vanilla	93.3	89.58	88.10	88.82
AraBERTEmojisIN	93.3	87.83	89.04	88.42
AraBERTEmojisOUT	93.9	87.11	91.42	89.06

TABLE 4.17 – Les résultats obtenus par Alami et al. sur le dataset OffensEval 2020 Arabe

L'étude menée par Hassan et al. (2020) [74] a mis en œuvre une variété de modèles, dont le classificateur SVM, le modèle Convolutional-Long Short Term Memory (C-LSTM) et BERT multilingue. Chacun de ces modèles exploitait des caractéristiques distinctes, pour SVM ils ont expérimenté avec différents intervalles de n-grammes de caractères et de mots. Le modèle Convolutional-Long Short Term Memory (C-LSTM) utilise les embeddings de mots pré-entraînés Mazajak. Ces embeddings sont combinés avec la représentation des mots au niveau des caractères avant d'être transmis à une couche LSTM bidirectionnelle.

Les résultats de ces différentes approches sont présentés dans les tableaux suivants (Table 4.18) et (Table 4.19), où les performances sont évaluées en termes d'exactitude, de précision, de rappel et de score F1. Le premier tableau résume les performances moyennes obtenues lors de la validation croisée, tandis que le deuxième tableau expose les performances des systèmes combinés sur l'ensemble de test officiel.

No.	Model	Features	Accuracy	Precision	Recall	F1-Score
1	fastText		88.4	84.0	77.0	79.8
2	Multi-BERT		89.9	82.6	83.1	82.9
3	C-LSTM		92.4	86.8	87.3	87.1
4	SVM	C[2-7]	91.4	87.5	84.6	85.9
5	SVM	W[1-3]	90.3	87.53	80.3	83.2
6	SVM	C [2-7] + W[1-3]	91.6	89.5	82.8	85.5
7	SVM	C [2-7] + W[1-3] + Mazajak	92.9	89.3	87.9	88.6
8	7 & ValenceList		93.0	88.9	89.0	88.9

TABLE 4.18 – Performance moyenne de chaque modèle avec la validation croisée.

Model	Accuracy	Precision	Recall	F1-Score
SVM & ValenceList + C-LSTM + Mult-BERT	93.85	91.36	89.08	90.16

TABLE 4.19 – Performance de la combinaison de systèmes pour l'ensemble de test officiel en arabe

Wang et al. (2020)[75] ont mis en œuvre un ensemble de modèles XLM-RoBERTa-base et XLM-RoBERTa-large (Conneau et al., 2019) [76] dans le but de développer une méthode unifiée pour détecter le langage offensant dans toutes les langues de OffensEval2020, et non pas uniquement pour l'arabe. Cette approche se divise en deux étapes distinctes. Dans un premier temps, le modèle est formé sur un vaste corpus de textes non supervisés multilingues, afin d'obtenir un modèle unifié capable d'embrasser toutes les nuances linguistiques. Par la suite, dans la deuxième étape, ce modèle pré-entraîné est ajusté sur des données annotées spécifiques à chaque langue. Les auteurs ont ensuite comparé les performances des méthodes entraînées sur des données multilingues à celles entraînées sur des données monolingues.

Les performances en termes de score F1 pour XLM-RoBERTa-base et XLM-RoBERTa-large, après finetuning avec des données multilingues et monolingues, sont présentées dans le tableau suivant :

<b>XLM-R BASE</b>		<b>XLM-R LARGE</b>		<b>Submitted Result (Ensemble)</b>	
Single	Multi	Single	Multi	Multi	Rank in all teams
0.8649	0.8730	0.8969	0.9015	0.8989	3

TABLE 4.20 – Résultats de F1 score pour la langue arabe Sub-Task A

Mubarak et al. (2021) [77] ont effectué une analyse approfondie visant à décrire les caractéristiques du langage offensant arabe, tout en explorant différentes techniques de représentation textuelle. Les embeddings statiques utilisés par les auteurs comprennent ceux pré-entraînées tels que FastText, AraVec, Mazajak et des embeddings FastText personnalisés entraînés sur l'ensemble de données OffensEval2020, ainsi que les caractéristiques lexicales qui impliquent le comptage des termes positifs et négatifs à partir d'un lexique de polarité.

Les auteurs ont étudié une gamme de modèles comprenant l'Arbre de Décision (Decision Tree), la Forêt Aléatoire (Random Forest), le Naïve Bayes Gaussien (Gaussian NB), le Perceptron, AdaBoost, le Gradient Boosting, la Régression Logistique (Logistic Regression) ainsi que les Machines à Vecteurs de Support (SVM) avec différentes représentations de texte. Comme illustré dans le tableau 4.21, en plus des modèles pré-entraînés BERT Base multilingue et AraBERT.

Les tableaux suivants présentent les résultats en termes de précision, rappel et score F1 obtenus par les auteurs avec les différents modèles qu'ils ont proposés.

<b>Model / Classifier</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>
<b>Lexical Features</b>			
SVM	68.5	35.3	46.6
<b>Pre-trained Static Embeddings</b>			
fastText/SVM	76.7	43.5	55.5
AraVec/SVM	85.5	69.2	76.4
Mazajak/SVM	88.6	72.4	79.7
<b>Embedding Trained on Their Data</b>			
fastText/fastText	82.1	68.1	74.4
<b>Contextualized Embeddings</b>			
BERT Base multilingual	78.3	74.0	76.0
AraBERT	84.6	82.4	83.2

TABLE 4.21 – Performance de classification avec différentes caractéristiques et modèles

<b>Model</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>
Decision Tree	51.2	53.8	52.4
Random Forest	82.4	42.4	56.0
Gaussian NB	44.9	86.0	59.0
Perceptron	75.6	67.7	66.8
AdaBoost	74.3	67.0	70.4
Gradient Boosting	84.2	63.0	72.1
Logistic Regression	84.7	69.5	76.3
SVM	88.6	72.4	79.7

TABLE 4.22 – Performance des différents classificateurs avec Mazajak embeddings

Zampieri et al. (2023) [71] ont mené une étude approfondie des compétitions OffensEval, visant à évaluer l’efficacité des méthodes d’identification des langages offensants, ainsi qu’une analyse des performances de divers modèles pré-entraînés (LLM) sur les ensembles de données OffensEval2019 et OffensEval2020, comparant leurs résultats à ceux des systèmes les plus performants issus des compétitions. Cependant, pour l’ensemble de données OffensEval2020 en langue arabe, ils n’ont inclus que les résultats des trois meilleures équipes du concours, ainsi que le score obtenus par le modèle pré-entraîné Flan-T5-large.

Le tableau suivant présente le score F1 dans un ordre décroissant des trois meilleures équipes du concours ainsi que du modèle Flan-T5-large proposé par les auteurs.

<b>Model</b>	<b>F1-score</b>
OffensEval Rank 1	0.902
OffensEval Rank 2	0.901
OffensEval Rank 3	0.899
Flan-T5-large	0.530

TABLE 4.23 – Résultats de F1-score des trois meilleures équipes du concours et Flan-T5-large

Notre modèle surpassé les meilleurs modèles des études mentionnées précédemment. En effet, notre meilleur modèle AraBERTv02 atteint un score F1 de 95,3 %, une exactitude de 95,3 %, une précision de 95,4 % et un rappel de 95,2 %. comme le met en évidence ce tableau comparatif :

<b>Model</b>	<b>Accuracy</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>
Notre meilleur model (AraBERTv02)	0.953	0.954	0.952	0.953
Alami et al. (2020)	93.9	94.38	93.9	94.06
Hassan et al. (2020)	93.85	91.36	89.08	90.16
Wang et al. (2020)	-	-	-	0.8989
Mubarak et al. (2021)	-	88.6	86.0	79.7
Zampieri et al. (2023)	-	-	-	0.530

TABLE 4.24 – Tableau comparatif des résultats des meilleurs modèles

## 4.5 Présentation de l'application

L'interface est conçue avec la bibliothèque Streamlit, offre la possibilité d'ajouter un commentaire, ceci permet de le soumettre à une analyse après avoir effectué le prétraitement décrit dans le chapitre 3. Ensuite, elle détermine le pourcentage de discours haineux et non haineux en utilisant notre modèle le plus performant AraBERTv02, comme présenté dans la figure ci-dessous.

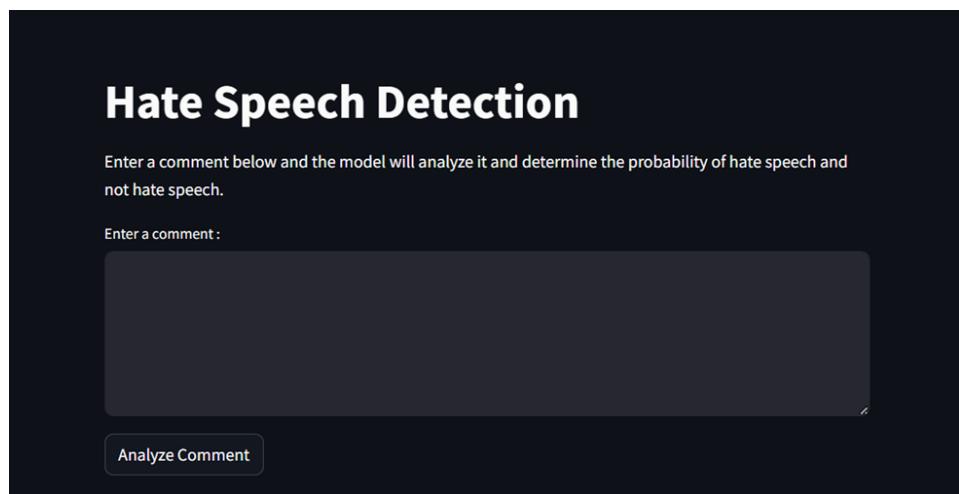


FIGURE 4.9 – Interface de détection des discours haineux

Une illustration est présentée dans la figure suivante, où le commentaire est classé comme discours haineux avec un pourcentage de 99,17 %.

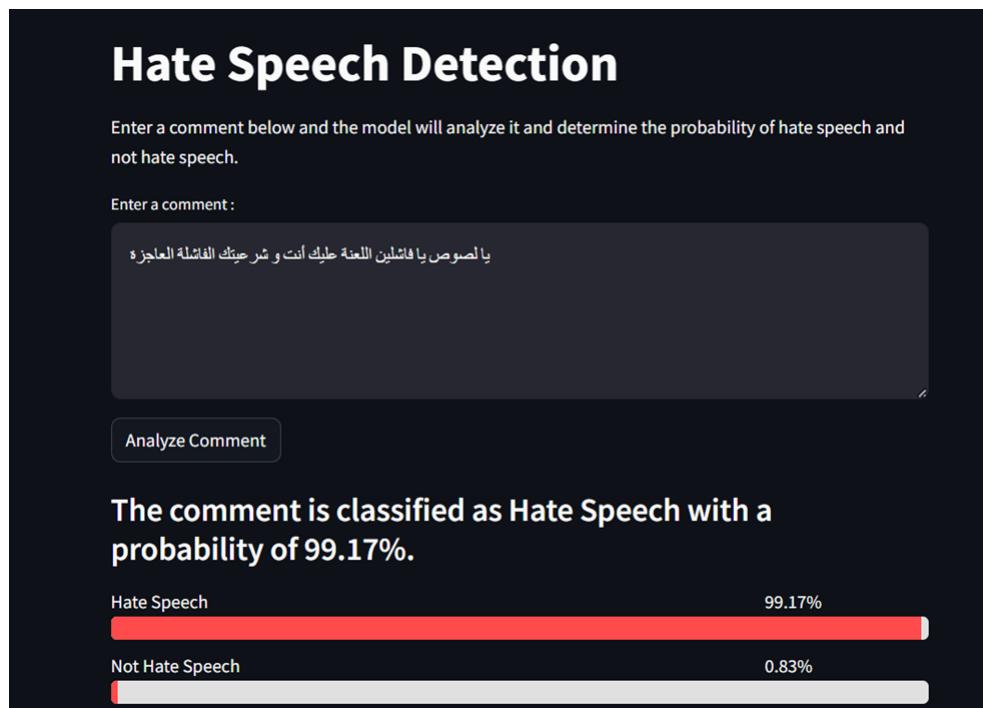


FIGURE 4.10 – Illustration de l'analyse d'un commentaire en arabe contenant un discours haineux

Voici un exemple de classification d'un texte écrit en dialecte algérien en arabe : Ci-dessous un

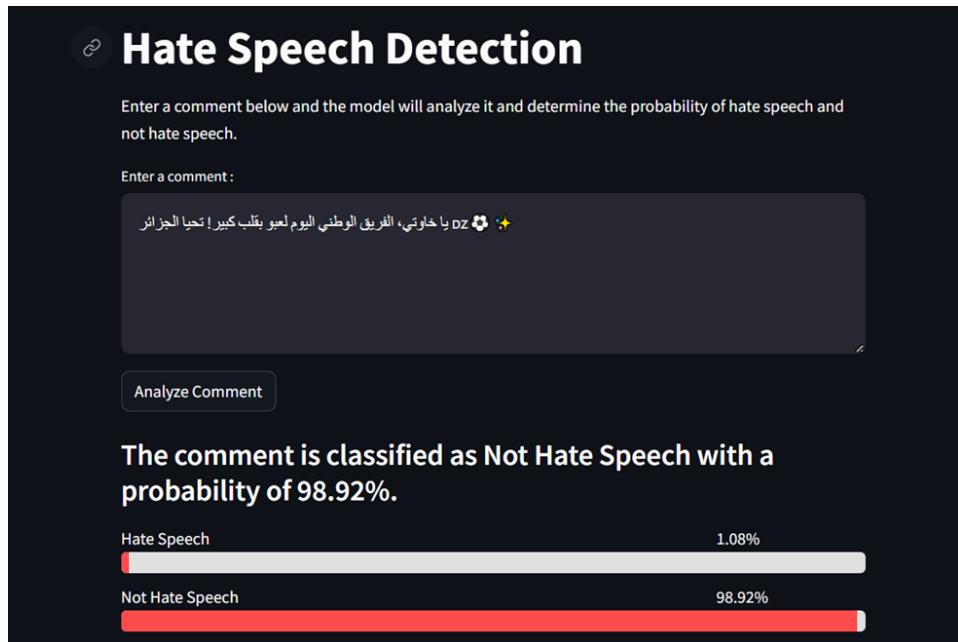


FIGURE 4.11 – Illustration de l'analyse d'un commentaire en dialecte algérien qui ne contient pas de discours haineux.

autre exemple de classification d'un texte écrit en dialecte algérien en arabizi :

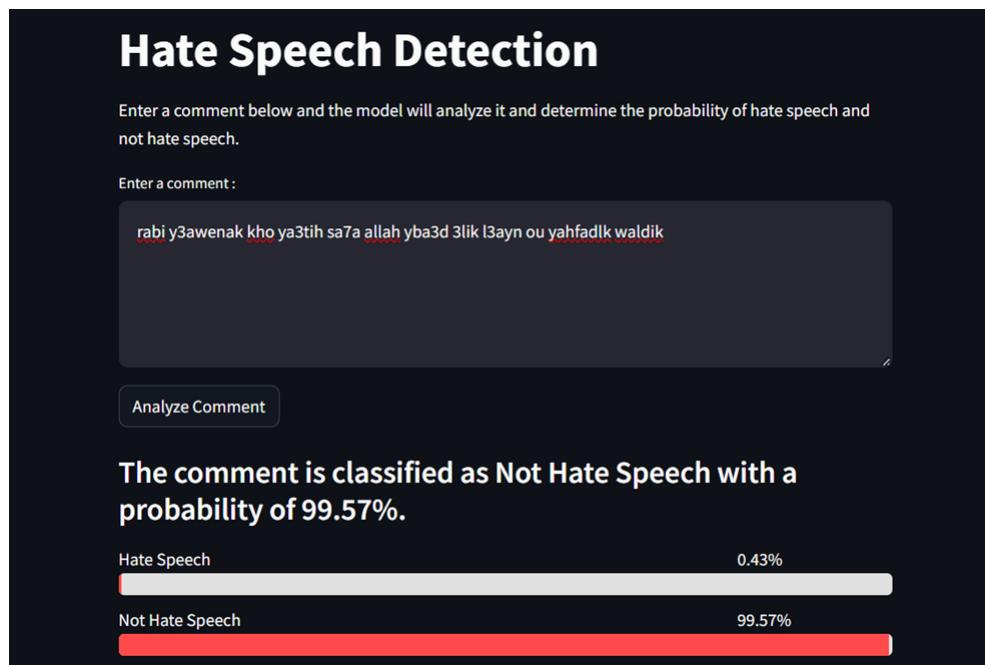


FIGURE 4.12 – Illustration de l'analyse d'un commentaire écrit avec Arabizi en dialecte algérien qui ne contient pas de discours haineux.

## 4.6 Conclusion

Ce chapitre a fourni une vue d'ensemble complète des différentes étapes nécessaires pour mener des expérimentations approfondies dans le domaine du traitement automatique du langage naturel et de l'apprentissage automatique. En somme, ce chapitre a non seulement illustré le processus complet de mise en place et d'évaluation de modèles de traitement du langage naturel, mais a également préparé le terrain pour une exploration plus approfondie des applications potentielles et des améliorations futures dans ce domaine dynamique.

# Conclusion Générale

À travers cette étude, nous avons exploré le traitement automatique du langage naturel ainsi que les approches d'apprentissage automatique et d'apprentissage profond appliquées à la détection du discours haineux en langue arabe, avec un focus particulier sur le dialecte algérien.

Dans le premier chapitre, nous avons souligné l'importance de cette tâche ainsi que les défis spécifiques liés à la langue arabe et aux dialectes. Nous avons présenté les concepts fondamentaux du TALN et de l'apprentissage automatique, en détaillant les différentes approches telles que l'apprentissage supervisé, non supervisé, et profond, ainsi que les modèles de réseaux de neurones avancés, notamment les CNN, RNN, LSTM et les Transformers, qui sous-tendent les modèles de langage pré-entraînés (LLMs). Le deuxième chapitre a été dédié à l'état de l'art sur la détection du discours haineux en arabe, ainsi qu'aux différentes approches et ensembles de données existants.

Le troisième chapitre a détaillé la conception de notre approche, incluant l'architecture générale, les étapes de prétraitement des données et les différentes méthodes proposées, allant des baselines de machine learning aux approches avancées basées sur l'apprentissage profond et les modèles basés sur les Transformers.

Enfin, le quatrième chapitre a présenté les expérimentations et les résultats obtenus. Nous avons décrit notre environnement de travail et les outils utilisés, évalué nos modèles à l'aide de mesures de performance standard, et discuté l'analyse des résultats, incluant la performance, le temps d'entraînement et la comparaison des résultats, ainsi que l'illustration de l'application pratique de notre approche dans une application dédiée.

En conclusion, cette étude montre clairement que les approches modernes de TALN et d'apprentissage profond, notamment les modèles pré-entraînés, offrent des outils puissants pour la détection du discours haineux en arabe, même dans des dialectes spécifiques comme l'algérien. Les résultats obtenus sont prometteurs et ouvrent la voie à des applications concrètes pouvant contribuer à la modération des contenus en ligne et à la lutte contre le discours haineux. Cependant, des défis subsistent en termes de complexité linguistique, nécessitant des efforts continus de recherche et d'innovation.

Certaines limitations sont à noter, telles que la fiabilité des annotations, qui peut introduire du bruit dans les données d'entraînement et impacter les performances des modèles, ainsi que la taille relativement restreinte des corpus. Par ailleurs, l'utilisation de plateformes comme Google Colaboratory limite les expérimentations en fonction de la disponibilité et de la capacité des ressources fournies.

Pour l'avenir, il serait bénéfique d'aborder une classification multi-classe des discours de haine au lieu de se limiter à une classification binaire. L'intégration de sources de données supplémentaires, telles que les métadonnées contextuelles, pourrait également améliorer la précision et la pertinence

des détections. En outre, augmenter la taille du corpus et maintenir des annotations fiables, tout en explorant des architectures alternatives de deep learning, sont des pistes prometteuses pour surmonter les défis actuels.

# Références

- [1] *What is natural language processing (NLP) ?* en. 2024. URL : <https://www.ibm.com/natural-language-processing> (visité le 19/04/2024).
- [2] Dihia LANASRI et al. *Hate speech detection in algerian dialect using deep learning*. 2023. arXiv : 2309.11611 [cs.CL].
- [3] Ali FARGHALY et al. “Arabic Natural Language Processing : Challenges and Solutions”. In : *ACM Transactions on Asian Language Information Processing (TALIP)* 8 (jan. 2009), p. 1-.
- [4] Imane GUELLIL et al. “Arabic Natural Language Processing : an overview”. In : *Journal of King Saud University - Computer and Information Sciences* 33 (fév. 2019). DOI : 10.1016/j.jksuci.2019.02.006.
- [5] Khaled SHAALAN et al. “Challenges in Arabic Natural Language Processing”. In : nov. 2018, p. 59-83. ISBN : 978-981-322-938-9. DOI : 10.1142/9789813229396\_0003.
- [6] Oumayma OUESLATI et al. “A review of sentiment analysis research in Arabic language”. In : *Future Generation Computer Systems* 112 (mai 2020). DOI : 10.1016/j.future.2020.05.034.
- [7] F. PEDREGOSA et al. *Scikit-learn : Machine Learning in Python*. 2011.
- [8] AKASH. *What is Supervised Learning and its different types ?* 2022. URL : <https://www.edureka.co/blog/supervised-learning/> (visité le 19/04/2024).
- [9] A. BURKOV. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019. ISBN : 9781999579517. URL : <https://books.google.dz/books?id=0jbxwQEACAAJ>.
- [10] *Random Forest Classifier*. URL : <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html> (visité le 05/04/2024).
- [11] *Stochastic Gradient Descent*. URL : <https://scikit-learn.org/stable/modules/sgd.html> (visité le 08/03/2024).
- [12] *Support Vector Machines*. URL : <https://cloud.google.com/discover/what-is-unsupervised-learning> (visité le 20/04/2024).
- [13] *What's the Difference Between Deep Learning and Neural Networks ?* URL : <https://aws.amazon.com/compare/the-difference-between-deep-learning-and-neural-networks/> (visité le 20/04/2024).
- [14] Samaya MADHAVAN. *Introduction to Convolutional Neural Networks*. 2021. URL : <https://developer.ibm.com/articles/introduction-to-convolutional-neural-networks/> (visité le 23/04/2024).
- [15] PHUNG et RHEE. “A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets”. In : *Applied Sciences* 9 (oct. 2019), p. 4500. DOI : 10.3390/app9214500.

- [16] IBM. *What are recurrent neural networks ?* URL : <https://www.ibm.com/topics/recurrent-neural-networks> (visité le 23/04/2024).
- [17] Pushparaja MURUGAN. *Learning The Sequential Temporal Information with Recurrent Neural Networks.* Juill. 2018.
- [18] DATASCIENTEST. *Long Short Term Memory (LSTM) : de quoi s'agit-il ?* 2023. URL : <https://datascientest.com/long-short-term-memory-tout-savoir> (visité le 23/04/2024).
- [19] Ryan T. J. J. *LSTMs Explained : A Complete, Technically Accurate, Conceptual Guide with Keras.* 2020. URL : <https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2> (visité le 24/04/2024).
- [20] MEDDAH Yasmine KLOUL NAWEL. *Classification de polarité d'opinions à base d'aspects à l'aide de l'apprentissage profond.* 2020.
- [21] Guoqiang JIN et al. “An Adaptive Anti-Noise Neural Network for Bearing Fault Diagnosis Under Noise and Varying Load Conditions”. In : *IEEE Access* 8 (avr. 2020), p. 74793-74807. DOI : [10.1109/ACCESS.2020.2989371](https://doi.org/10.1109/ACCESS.2020.2989371).
- [22] Ashish VASWANI et al. *Attention Is All You Need.* 2023. arXiv : 1706.03762 [cs.CL].
- [23] Sagar PATIL. *Attention Mechanism in the Transformers.* 2023. URL : <https://medium.com/@sagarpatiler/attention-mechanism-in-the-transformers-fd067df25ea> (visité le 26/04/2024).
- [24] Jacob DEVLIN et al. *BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2019. arXiv : 1810.04805 [cs.CL].
- [25] *BERT.* URL : [https://huggingface.co/docs/transformers/model\\_doc/bert](https://huggingface.co/docs/transformers/model_doc/bert) (visité le 26/04/2024).
- [26] *What are Large Language Models (LLM) ?* URL : <https://aws.amazon.com/what-is/large-language-model/> (visité le 26/04/2024).
- [27] Jérémie ROBERT. *Instruction Tuning : En quoi consiste cette technique de fine-tuning ?* URL : <https://datascientest.com/instruction-tuning-tout-savoir> (visité le 27/04/2024).
- [28] Kai Lv et al. *Full Parameter Fine-tuning for Large Language Models with Limited Resources.* 2023. arXiv : 2306.09782 [cs.CL].
- [29] Sayak Paul SOURAB MANGRULKAR. *PEFT : Parameter-Efficient Fine-Tuning of Billion-Scale Models on Low-Resource Hardware.* 2023. URL : <https://huggingface.co/blog/peft> (visité le 27/04/2024).
- [30] UNITED NATIONS. *What is Hate Speech ?* URL : <https://www.un.org/fr/hate-speech/understanding-hate-speech/what-is-hate-speech> (visité le 30/03/2024).
- [31] CONSEIL DE L’EUROPE. *Le discours de haine.* URL : <https://www.coe.int/fr/web/freedom-expression/hate-speech> (visité le 28/04/2024).
- [32] META. *Règles communautaires : Discours de haine.* URL : <https://transparency.meta.com/fr-fr/policies/community-standards/hate-speech/> (visité le 28/04/2024).
- [33] YOUTUBE. *Hate Speech Policy.* URL : <https://support.google.com/youtube/answer/2801939?hl=en> (visité le 30/04/2024).

- [34] X. X's policy on hateful conduct. URL : <https://help.x.com/en/rules-and-policies/hateful-conduct-policy> (visité le 30/04/2024).
- [35] Luvell ANDERSON et Michael BARNES. "Hate Speech". In : *The Stanford Encyclopedia of Philosophy*. Sous la dir. d'Edward N. ZALTA et Uri NODELMAN. Fall 2023. Metaphysics Research Lab, Stanford University, 2023.
- [36] A. AL-HASSAN et H. AL-DOSSARI. "Detection of hate speech in Arabic tweets using deep learning". In : *Multimedia Systems* 28 (2022), p. 1963-1974. DOI : 10.1007/s00530-020-00742-w.
- [37] Ramzi KHEZZAR, Abdelrahman MOURSI et Zaher AL AGHBARI. "arHateDetector : detection of hate speech from standard and dialectal Arabic Tweets". In : *Discover Internet of Things* 3 (mars 2023). DOI : 10.1007/s43926-023-00030-9.
- [38] Bedour ALRASHIDI, Amani JAMAL et Ali ALKHATHLAN. "Abusive Content Detection in Arabic Tweets Using Multi-Task Learning and Transformer-Based Models". In : *Applied Sciences* 13 (mai 2023), p. 5825. DOI : 10.3390/app13105825.
- [39] Nedjma OUSIDHOUM et al. "Multilingual and Multi-Aspect Hate Speech Analysis". In : *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Sous la dir. de Kentaro INUI et al. Hong Kong, China : Association for Computational Linguistics, nov. 2019, p. 4675-4684. DOI : 10.18653/v1/D19-1474. URL : <https://aclanthology.org/D19-1474>.
- [40] Ashraf AHMAD et al. "Hate Speech Detection in the Arabic Language : Corpus Design, Construction and Evaluation". In : (sept. 2023). DOI : 10.20944/preprints202309.0497.v1.
- [41] Angel Felipe Magnossão de PAULA et al. *Transformers and Ensemble methods : A solution for Hate Speech Detection in Arabic languages*. 2023. arXiv : 2303.09823 [cs.CL].
- [42] Mohamed Seghir Hadj AMEUR et Hassina ALIANE. *AraCOVID19-MFH : Arabic COVID-19 Multi-label Fake News and Hate Speech Detection Dataset*. 2021. arXiv : 2105.03143 [cs.CL].
- [43] Imane GUELLIL et al. *Sexism detection : The first corpus in Algerian dialect with a code-switching in Arabic/ French and English*. 2021. arXiv : 2104.01443 [cs.CL].
- [44] Oussama BOUCHERIT et Kheireddine ABAINIA. *Offensive Language Detection in Under-resourced Algerian Dialectal Arabic Language*. 2022. arXiv : 2203.10024 [cs.CL].
- [45] Ahmed Cherif MAZARI et Hamza KHEDDAR. "Deep Learning-based Analysis of Algerian Dialect Dataset Targeted Hate Speech, Offensive Language and Cyberbullying". In : 13 (avr. 2023), p. 965-972. DOI : 10.12785/ijcds/130177.
- [46] Ahmed OMAR, Tarek M. MAHMOUD et Tarek ABD-EL-HAFEEZ. "Comparative Performance of Machine Learning and Deep Learning Algorithms for Arabic Hate Speech Detection in OSNs". In : *Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020)*. Sous la dir. d'Aboul-Ella HASSANIEN et al. Cham : Springer International Publishing, 2020, p. 247-257. ISBN : 978-3-030-44289-7.
- [47] Raghad ALSHALAN et Hend AL-KHALIFA. "A Deep Learning Approach for Automatic Hate Speech Detection in the Saudi Twittersphere". In : *Applied Sciences* 10.23 (2020). ISSN : 2076-3417. DOI : 10.3390/app10238614. URL : <https://www.mdpi.com/2076-3417/10/23/8614>.

- [48] Rehab DUWAIRI, Amena HAYAJNEH et Muhannad QUWAIDER. “A Deep Learning Framework for Automatic Detection of Hate Speech Embedded in Arabic Tweets”. In : *Arabian Journal for Science and Engineering* 46.4 (2021), p. 4001-4014. DOI : 10.1007/s13369-021-05383-3.
- [49] Hossam FARIS et al. “Hate Speech Detection using Word Embedding and Deep Learning in the Arabic Language Context.” In : *ICPRAM*. 2020, p. 453-460.
- [50] Faisal Yousif Al ANEZI. “Arabic hate speech detection using deep recurrent neural networks”. In : *Applied Sciences* 12.12 (2022), p. 6010.
- [51] Safa ALSAFARI, Samira SADAQUI et Malek MOUHOUB. *Deep Learning Ensembles for Hate Speech Detection*. Nov. 2020. DOI : 10.1109/ICTAI50040.2020.00087.
- [52] Raghad ALSHAALAN et Hend AL-KHALIFA. “Hate speech detection in saudi twittersphere : A deep learning approach”. In : *Proceedings of the fifth Arabic natural language processing workshop*. 2020, p. 12-23.
- [53] Palé Ollo SALOMON, Zied KECHAOU et Ali WALI. “Arabic hate speech detection system based on AraBERT”. In : *2022 IEEE 21st International Conference on Cognitive Informatics Cognitive Computing (ICCI\*CC)*. 2022, p. 208-213. DOI : 10.1109/ICCI57084.2022.10101577.
- [54] Safa ALSAFARI, Samira SADAQUI et Malek MOUHOUB. “Hate and offensive speech detection on Arabic social media”. In : *Online Social Networks and Media* 19 (2020), p. 100096.
- [55] Khalid T MURSI et al. “Detecting islamic radicalism arabic tweets using natural language processing”. In : *IEEE Access* 10 (2022), p. 72526-72534.
- [56] Luke OTWELL. *The Soundex Algorithm*. 2021. URL : <https://medium.com/@lukehenryotwell/the-soundex-algorithm-d39c2f8d8756> (visité le 12/05/2024).
- [57] Sergey GRASHCHENKO. *Levenshtein Distance Computation*. 2023. URL : <https://www.baeldung.com/cs/levenshtein-distance-computation> (visité le 12/05/2024).
- [58] Yonghui WU et al. *Google’s Neural Machine Translation System : Bridging the Gap between Human and Machine Translation*. 2016. arXiv : 1609.08144 [cs.CL].
- [59] Anirudha SIMHA. *Understanding TF-IDF for Machine Learning*. 2021. URL : <https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/> (visité le 01/05/2024).
- [60] Tomas MIKOLOV et al. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. arXiv : 1310.4546 [cs.CL].
- [61] Piotr BOJANOWSKI et al. *Enriching Word Vectors with Subword Information*. 2017. arXiv : 1607.04606 [cs.CL].
- [62] Abu Bakr MOHAMMAD, Kareem EISSL et Samhaa EL-BELTAGY. “AraVec : A set of Arabic Word Embedding Models for use in Arabic NLP”. In : *Procedia Computer Science* 117 (nov. 2017), p. 256-265. DOI : 10.1016/j.procs.2017.10.117.
- [63] TENSORFLOW. *Word embeddings*. 2023. URL : [https://www.tensorflow.org/text/guide/word\\_embeddings](https://www.tensorflow.org/text/guide/word_embeddings) (visité le 03/05/2024).
- [64] Amine ABDAOUI et al. *DziriBERT : a Pre-trained Language Model for the Algerian Dialect*. 2021. arXiv : 2109.12346. URL : <https://arxiv.org/abs/2109.12346>.
- [65] Wuwei LAN et al. *An Empirical Study of Pre-trained Transformers for Arabic Information Extraction*. 2020. arXiv : 2004.14519 [cs.CL].

- [66] Wuwei LAN et al. “GigaBERT : Zero-shot Transfer Learning from English to Arabic”. In : *Proceedings of The 2020 Conference on Empirical Methods on Natural Language Processing (EMNLP)*. 2020.
- [67] Dzarabert. URL : <https://huggingface.co/Sifal/dzarabert> (visité le 29/04/2024).
- [68] Wissam ANTOUN, Fady BALY et Hazem HAJJ. *AraGPT2 : Pre-Trained Transformer for Arabic Language Generation*. 2021. arXiv : 2012.15520 [cs.CL].
- [69] Wissam ANTOUN, Fady BALY et Hazem HAJJ. “AraBERT : Transformer-based Model for Arabic Language Understanding”. In : *LREC 2020 Workshop Language Resources and Evaluation Conference 11–16 May 2020*, p. 9.
- [70] AI@META. “Llama 3 Model Card”. In : (2024). URL : [https://github.com/meta-llama/llama3/blob/main/MODEL\\_CARD.md](https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md).
- [71] Marcos ZAMPIERI et al. “OffensEval 2023 : Offensive language identification in the age of Large Language Models”. In : *Natural Language Engineering* 29.6 (2023), p. 1416-1435. DOI : 10.1017/S1351324923000517.
- [72] Nassera HABBAT et al. “Using AraGPT and ensemble deep learning model for sentiment analysis on Arabic imbalanced dataset”. In : *ITM Web of Conferences* 52 (mai 2023), p. 02008. DOI : 10.1051/itmconf/20235202008.
- [73] Hamza ALAMI et al. “LISAC FSDM-USMBA Team at SemEval-2020 Task 12 : Overcoming AraBERT’s pretrain-finetune discrepancy for Arabic offensive language identification”. In : *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. Sous la dir. d’Aurelie HERBELOT et al. Barcelona (online) : International Committee for Computational Linguistics, déc. 2020, p. 2080-2085. DOI : 10.18653/v1/2020.semeval-1.275. URL : <https://aclanthology.org/2020.semeval-1.275>.
- [74] Sabit HASSAN et al. “ALT at SemEval-2020 Task 12 : Arabic and English Offensive Language Identification in Social Media”. In : *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. Sous la dir. d’Aurelie HERBELOT et al. Barcelona (online) : International Committee for Computational Linguistics, déc. 2020, p. 1891-1897. DOI : 10.18653/v1/2020.semeval-1.249. URL : <https://aclanthology.org/2020.semeval-1.249>.
- [75] Shuohuan WANG et al. “Galileo at SemEval-2020 Task 12 : Multi-lingual Learning for Offensive Language Identification Using Pre-trained Language Models”. In : *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. Sous la dir. d’Aurelie HERBELOT et al. Barcelona (online) : International Committee for Computational Linguistics, déc. 2020, p. 1448-1455. DOI : 10.18653/v1/2020.semeval-1.189. URL : <https://aclanthology.org/2020.semeval-1.189>.
- [76] Alexis CONNEAU et al. *Unsupervised Cross-lingual Representation Learning at Scale*. 2020. arXiv : 1911.02116 [cs.CL].
- [77] Hamdy MUBARAK et al. “Arabic Offensive Language on Twitter : Analysis and Experiments”. In : *Proceedings of the Sixth Arabic Natural Language Processing Workshop*. Sous la dir. de Nizar HABASH et al. Kyiv, Ukraine (Virtual) : Association for Computational Linguistics, avr. 2021, p. 126-135. URL : <https://aclanthology.org/2021.wanlp-1.13>.