UNIVERSITÉ DU QUÉBEC À MONTRÉAL

ANALYSE ET DÉTECTION DU CONTENU HAINEUX SUR LES RÉSEAUX SOCIAUX

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

MAÎTRISE EN INFORMATIQUE

PAR

SAFAA ENNACIRI

DÉCEMBRE 2024

REMERCIEMENTS

Je tiens d'abord à remercier mon directeur de recherche, Monsieur Hakim Lounis, qui a accepté de m'accompagner dans des circonstances particulières afin de pouvoir finaliser ce travail. Je le remercie pour sa patience, sa bienveillance, ses remarques, ses conseils et sa disponibilité. Je lui adresse toute ma gratitude.

Je remercie aussi mes chers parents qui m'ont toujours appuyé et encouragé afin d'arriver à mes fins, et j'exprime ma reconnaissance à toutes les personnes qui me sont chères et qui m'ont apporté du réconfort dans mes moments les plus difficiles.

Merci à tous.

DÉDICACE

À mes très chers parents qui m'ont toujours appuyé et encouragé

À ma chère sœur qui m'a soutenu

À mes trois enfants et aux êtres qui me sont chers

Je dédie le fruit de ce travail

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

L'utilisation des réseaux sociaux devient de plus en plus importante dans nos vies et la quantité d'information qui y circule est devenue énorme. La plupart de ces informations sont instructives et pertinentes, mais certaines d'entre elles sont nocives et incitent à la haine. La propagation sur ces réseaux, de commentaires contenant du sarcasme, de l'intimidation, de l'agressivité et du racisme, affecte négativement les utilisateurs ciblés et peut mener à de graves conséquences. C'est pourquoi il est nécessaire de trouver des solutions efficaces pour remédier à ce problème. L'identification et la détection du contenu haineux est devenue une préoccupation de plusieurs chercheurs et le sujet de diverses études. Dans ce contexte, notre projet vise à concevoir un système capable d'identifier et de détecter des commentaires haineux. Pour ce faire, nous postulons que le prétraitement des données joue un rôle déterminant dans l'efficacité d'un tel système, et nous explorons donc plusieurs combinaisons de techniques de prétraitement. Nous évaluons notre approche sur quatre ensembles de données différents, contenant des commentaires et discours haineux et nous l'appliquons à différentes approches de l'apprentissage machine, SVM, Naïve Bayes, Logistic Regression et Bidirectional Long Short-Term Memory. Nous évaluons les performances des modèles obtenus, en termes d'accuracy, de précision, de F-mesure et de rappel en comparant les différents résultats obtenus. Enfin, nous analysons et interprétons les résultats pour aboutir à une conclusion sur l'efficacité de l'approche proposée.

Mots clés : Identification de discours haineux, classification de texte, apprentissage machine, prétraitement de données, extraction de fonctionnalités.

ABSTRACT

The use of social networks is becoming more and more important in our lives and the amount of information circulating there has become enormous. Most of this information is informative and relevant, but some of it is destructive and incites violence. The spread of comments containing sarcasm, intimidation, aggression, and racism on social networks negatively influences target users and can lead to destructive consequences. This is why it is necessary to find effective solutions to remedy this problem. The identification and detection of hateful content is the major concern of several researchers and the subject of various studies. In the same context, our project aims to design a system capable of identifying and detecting hateful comments. We explore several combinations of preprocessing techniques using four different datasets containing hate comments and speech. Our method is based on the application of these techniques with machine learning algorithms, including the SVM algorithm, the Naïve Bayes algorithm, the Logistic Regression algorithm, and Bidirectional Long Short-Term Memory. We will evaluate the performance of the models in terms of accuracy, precision, F-measure and recall by comparing the different results obtained. Finally, we will analyze and interpret the results to reach a conclusion on the effectiveness of the proposed system.

Keywords: Hate speech identification, text classification, data preprocessing, feature extraction.

INTRODUCTION

L'explosion de la popularité des réseaux sociaux (Facebook, Instagram, Twitter, etc.) met en évidence un revers de la médaille. Ces plateformes peuvent véhiculer des informations importantes et utiles dans plusieurs domaines, mais elles peuvent aussi promouvoir des discours de haine, des insultes ou de l'intimidation, diffusés par certains utilisateurs malveillants envers des personnes ou des communautés. La propagation sur les réseaux sociaux de commentaires contenant du sarcasme, de l'intimidation, de l'agressivité et du racisme, a un impact négatif sur les utilisateurs ciblés et empêche d'avoir un environnement numérique sain. Pour protéger ces derniers et leur permettre d'utiliser ces plateformes de façon constructive et inclusive, il faut trouver des solutions pour identifier et bloquer à temps ce type de contenu toxique.

Les grandes quantités d'informations qui circulent sur les réseaux sociaux nécessitent d'être traitées et classées. Traiter ces informations manuellement n'est pas une tâche facile, puisque la précision des résultats de cette opération peut être influencée par plusieurs facteurs humains (fatigue, manque d'expérience, etc.). D'où l'importance d'automatiser cette procédure en utilisant les algorithmes d'apprentissage machine qui sont plus fiables en termes de précision de résultats (Domingos, 2012). Néanmoins, le processus de bâtir un modèle d'apprentissage machine est coûteux en matière d'effort et de temps. Cela est dû, entre autres, au temps nécessaire pour la création et la préparation de l'ensemble de données et pour l'entraînement et l'évaluation du modèle d'apprentissage automatique (Domingos, 2012). Un tel processus se réalise en plusieurs étapes : collecte des données, prétraitement des données, extraction des caractéristiques, entraînement puis évaluation du modèle. Les résultats obtenus sont analysés et en cas d'erreurs, des modifications sont apportées aux données ou au classificateur avant de procéder à l'entraînement et à l'évaluation une autre fois.

L'extraction des caractéristiques est une phase importante et fondamentale dans le processus de classification. Elle permet de réduire la dimensionnalité du problème, de diminuer le temps d'entraînement et d'améliorer les performances du classificateur (Liang, Sun, Yunlei, & Gao, 2017). Elle consiste à extraire les caractéristiques du texte (appelées aussi attributs) qui représentent les informations pertinentes et informatives affectant la tâche de classification. Déterminer les caractéristiques manuellement est une tâche qui est susceptible d'engendrer des erreurs, d'une part, parce que chaque caractéristique est spécifique à un domaine et d'autre part, ces caractéristiques peuvent correspondre à

plusieurs classes à la fois. Automatiser cette tâche la rend plus facile et réduit le temps et l'effort qui peuvent y être consacrés (Domingos, 2012). Le choix et le nombre des caractéristiques sont des facteurs importants qui peuvent influencer sur le succès ou l'échec d'un projet d'apprentissage machine (Domingos, 2012). Par exemple, un grand nombre de caractéristiques peut causer un sur apprentissage et le choix de caractéristiques peu pertinentes conduit à des résultats médiocres.

Il est possible de générer automatiquement les caractéristiques à partir d'un texte brut en utilisant les techniques d'extraction de caractéristiques, telles que : TF-IDF, BoW (bag of words), CountVectorizer, etc. Cependant, cela n'est pas garant de l'obtention de meilleures performances par le système. D'où l'importance de préparer et de nettoyer le texte avant de procéder à l'extraction de ces caractéristiques. L'étape de prétraitement des informations textuelles est une étape importante dans le processus de classification de textes. Il s'agit de normaliser ces données brutes en éliminant le bruit pour qu'elles soient exploitables par la suite par l'algorithme de classification (Heitz, 2006). Il faut enlever tous les termes qui ne sont pas informatifs et ne garder que ceux qui sont importants pour la classification. Cela permet d'éliminer le bruit et de diminuer la dimensionnalité du problème (Haddi, Liu, & Shi, 2013).

L'identification des commentaires haineux demeure la préoccupation majeure de plusieurs chercheurs et le sujet de diverses études. Dans ce travail, nous avons comme objectif d'étudier les meilleures techniques de prétraitement permettant d'améliorer les performances d'un système de classification de discours haineux sur les réseaux sociaux. Il s'agit donc de voir quels agencements de techniques sont les plus pertinents dans un tel contexte, en expérimentant plusieurs approches de prétraitement, combinant plusieurs techniques de prétraitement des données. Nous allons utiliser des méthodes d'apprentissage machine supervisé, tels que les algorithmes SVM, Naive Bayes, Logistic Regression et l'algorithme d'apprentissage profond Bi-LSTM (Bidirectional Long Short-Term Memory). Une comparaison sera faite entre les différents résultats obtenus pour évaluer les performances des modèles.

Le présent document est organisé comme suit :

- Dans le chapitre 1, nous présentons la problématique soulevée dans ce travail. Nous donnons un aperçu sur le discours haineux, ses définitions ainsi que ses différentes formes. Nous présentons son impact sur les utilisateurs des réseaux sociaux et l'importance de trouver des solutions pour identifier et supprimer ce type de commentaires. Enfin, nous citerons les différents objectifs de ce travail, ainsi que les démarches que nous allons suivre.

- Dans le chapitre 2, nous faisons un survol de la littérature de ce domaine. D'abord, nous définissons les différents concepts liés à la classification de texte et nous couvrons quelques travaux reliés au même sujet. Ensuite, nous présentons les algorithmes d'apprentissage machine qui sont expérimentés dans ce travail. Enfin, nous présentons quelques travaux sur la classification du contenu toxique, les grandes questions qui y sont soulevées ainsi que les solutions proposées.
- Dans le chapitre 3, nous présentons la méthodologie utilisée dans ce travail. Ainsi, nous citons les hypothèses que nous allons vérifier, puis, nous présentons les ensembles de données utilisés ainsi qu'une description des différentes étapes suivies dans le processus de classification de commentaires toxiques. Enfin, nous fournissons un aperçu des démarches adoptées dans la réalisation de ce travail.
- Dans le chapitre 4, nous donnons un aperçu sur les méthodes et outils utilisés dans la réalisation de ce travail. Nous présentons d'abord l'environnement de travail, les bibliothèques et les logiciels exploités ainsi que les différentes techniques utilisées dans les phases de prétraitement, d'entraînement et de classification. Nous utilisons trois algorithmes d'apprentissage machine supervisé, SVM, Logistic Regression, Naive Nayes et un algorithme d'apprentissage profond Bi-LSTM (Bidirectional Long Short-Term Memory). Diverses méthodes de prétraitement, telles que, la racinisation (stemming), la lemmatisation, la correction orthographique, etc. sont aussi exploitées. Ensuite, nous présentons les différentes approches expérimentées, chacune d'entre elles est constituée d'un agencement de plusieurs techniques de prétraitement.
- Dans le chapitre 5, nous dévoilons d'abord les résultats obtenus. Ensuite, nous établirons pour chaque ensemble de données, une comparaison entre les résultats obtenus par chaque algorithme de classification et avec les différentes approches expérimentées. Cette comparaison est faite en termes d'accuracy, de précision, de rappel et de F-mesure. Nous présentons aussi les meilleurs résultats obtenus dans la littérature pour chaque ensemble de données. Enfin, nous concluons sur les meilleures combinaisons qui peuvent améliorer les performances d'un système de classification de discours haineux.

CHAPITRE 1

PROBLÉMATIQUE ET OBJECTIFS

Dans ce chapitre, nous donnons un aperçu sur le discours haineux, ses définitions ainsi que ses différentes formes. Ensuite, nous présenterons l'impact de ce type de commentaires sur les utilisateurs des réseaux sociaux. Enfin, nous citerons les différents objectifs de ce travail.

1.1 Problématique

De nos jours les réseaux sociaux (Facebook, Twitter, LinkedIn, etc.) et les médias sociaux (blogues, forums de discussion, etc.) sont utilisés de plus en plus dans la vie quotidienne. Le contenu publié peut-être ordinaire, comme il peut montrer un comportement anormal de l'utilisateur (des menaces, des insultes, des harcèlements, etc.). Il s'agit des commentaires agressifs et des discours haineux qui circulent dans le Web et qui peuvent être une source de nuisance pour plusieurs utilisateurs.

Il existe diverses définitions du terme « discours de haine » qui diffèrent légèrement d'une organisation à une autre, Instagram[1] a défini le discours de haine comme étant des attaques ou des abus fondés sur la race, l'origine ethnique, l'origine nationale, le sexe, le genre, l'identité de genre, l'orientation sexuelle, la religion, la déficience physique ou la maladie[2]. Facebook[3] définit le discours de haine comme suit :

> Nous définissons les discours haineux comme 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, l'orientation sexuelle, le sexe, l'identité de genre, et les maladies graves. Nous définissons une attaque comme un discours violent ou déshumanisant, des stéréotypes offensants, une déclaration d'infériorité, une expression de mépris, de dégoût ou de renvoi, une insulte ou un appel à l'exclusion ou à la ségrégation. Nous interdisons également l'utilisation de stéréotypes préjudiciables, que nous définissons comme des comparaisons déshumanisantes historiquement utilisées pour attaquer, intimider ou exclure des groupes

1 https://www.instagram.com/

2 https://transparency.fb.com/policies/community-standards/hate-speech/?source=https%3A%2F%2Fwww.facebook.com%2Fcommunitystandards%2Fhate_speech

3 https://www.facebook.com/

> spécifiques et souvent liées à la violence hors ligne. Nous considérons l'âge comme une caractéristique protégée lorsqu'il est mentionné parallèlement à une autre caractéristique protégée[4].

Quant à Twitter[5], il le définit comme suit :

> Il est interdit d'attaquer directement d'autres personnes en vous fondant sur la race, l'origine ethnique, l'origine nationale, la caste, l'orientation sexuelle, le sexe, l'identité sexuelle, l'appartenance religieuse, l'âge, un handicap ou toute maladie grave[6].

YouTube[7] l'a défini comme suit :

> Nous supprimons tout contenu incitant à la violence ou à la haine contre des individus ou des groupes d'individus en fonction des caractéristiques suivantes : Âge, caste, handicap, origine ethnique, identité et expression de genre, nationalité, groupe ethnique, statut d'immigration, religion, sexe/genre, orientation sexuelle, statut de victime d'un événement violent majeur ou de proche d'une victime et statut d'ancien combattant[8].

Anne Weber explique dans son livre « Manal of hate speech » (Weber, 2009) qu'il n'existe pas une définition universellement acceptée du terme « discours de haine ». Elle a aussi précisé que selon le conseil des comités des ministres de l'Europe, un discours de haine est tout discours qui propage, encourage ou défend les formes de haine fondées sur le racisme, antisémitisme, xénophobie ou toutes autres formes de haine basées sur le nationalisme agressif, l'antipathie et la discrimination envers un groupe minoritaire, des immigrants et des personnes d'origine immigrants. Elle a précisé aussi que tout commentaire qui est dirigé contre une seule personne ou un groupe particulier de personnes est considéré comme discours de haine.

4 https://transparency.fb.com/policies/community-standards/hate-speech/?source=https%3A%2F%2Fwww.facebook.com%2Fcommunitystandards%2Fhate_speech

5 https://twitter.com/

6 https://help.twitter.com/fr/rules-and-policies/hateful-conduct-policy

7 https://www.youtube.com/

8 https://support.google.com/youtube/answer/2801939?hl$=$en

Un discours de haine peut se présenter sous plusieurs formes (Jahan & Oussalah, 2021) :

- Cyberintimidation : C'est un acte d'intimidation produit dans le cyberespace[9]. La loi sur l'instruction publique présente l'intimidation comme suit :

  > Tout comportement, parole, acte ou geste délibéré ou non à caractère répétitif, exprimé directement ou indirectement, y compris dans le cyberespace, dans un contexte caractérisé par l'inégalité des rapports de force entre les personnes concernées, ayant pour effet d'engendrer des sentiments de détresse et de léser, blesser, opprimer ou ostraciser[10].

- Racisme : l'encyclopédie canadienne défini le racisme comme suit :

  > Le racisme est une croyance selon laquelle les humains peuvent être divisés suivant une hiérarchie du pouvoir en fonction de leurs différences de race et d'ethnicité. Ainsi, certains groupes sont vus comme supérieurs à d'autres uniquement en raison de caractéristiques raciales ou ethniques. Le racisme est souvent exprimé sous forme de préjugés ou de discrimination[11].

- Sexisme, discrimination fondée sur le genre : selon l'office québécois de la langue française (OQLF)[12], le sexisme est défini comme suit : « Attitude ou comportement discriminatoires fondés sur le sexe ou sur des stéréotypes liés au genre. »[13]. Il peut prendre la forme d'un comportement explicitement négatif comme il peut être plus subtile (Jha & Mamidi, 2017).
- Radicalisation : sur le site du gouvernement de Canada, la radicalisation est définie comme suit :

  > Un processus extrêmement personnalisé qui amène une personne à devenir convaincue que la violence est un moyen légitime (et même une obligation personnelle) de défendre sa cause ou ses convictions idéologiques[14].

9 https://www.quebec.ca/famille-et-soutien-aux-personnes/violences/intimidation/cyberintimidation

10 https://www.legisquebec.gouv.qc.ca/fr/document/lc/I-13.3

11 https://www.thecanadianencyclopedia.ca/fr/article/racisme

12 https://www.oqlf.gouv.qc.ca/

13 https://vitrinelinguistique.oqlf.gouv.qc.ca/fiche-gdt/fiche/8975953/sexisme

14 https://www.canada.ca/fr/service-renseignement-securite/organisation/publications/recherche-sur-la-mobilisation-a-la-violence-terrorisme-principaux-resultats.html

- Langage offensant, langage abusif : c'est le langage verbal violent qui est utilisé pour rabaisser, humilier ou nuire psychologiquement ou émotionnellement à une personne[15]. Il existe différentes formes de langage abusif[16] : le rejet répétitif des sentiments d'une personne, les jugements, les reproches, les injures, la condescendance, les fausses accusations, etc.

- Discours de haine religieux : cela comprend toutes les formes de discrimination religieuse. Selon l'EEOC (U.S. Equal Employment Opportunity Commission), la discrimination religieuse est un comportement discriminatoire fondé sur les convictions religieuses[17].

Selon l'article[18] publié sur le site ici.radio-canada.ca le 08 février 2022, le taux de cyberintimidation chez les jeunes a augmenté de 37 % en 2021 par rapport à l'année 2020. La même source précise que ce sont surtout les enfants âgés entre 12 ans et 15 ans qui sont les plus humiliés en divulguant leurs données personnelles (des images et des vidéos). Selon la même source, le rapport[19] « Cyberbullying in Plain Sight» publié en 2022 par l'entreprise McAfee, montre que la cyberintimidation raciste a atteint 42% en Inde, 34% aux USA et 25% au Canada. Il précise aussi que 50% des actes de cyberintimidation ont été faits sur Facebook, 30% sur Instagram et 18% sur Twitter. Selon la même étude, le nombre de Canadiens qui ont subi une cyberintimidation sur Facebook est de 59% et ceux intimidés sur Instagram est de 57%. Les victimes sont surtout des enfants et adolescents âgés entre 10 ans et 18 ans.

Selon l'article [20] publié sur le site lapresse.ca le 30 mai 2019, l'équipe de Marie-Claude Geoffroy, chercheuse au département de psychiatrie de l'Université McGill, a montré que l'intimidation peut causer le développement de comportements suicidaires chez les enfants victimes. Elle a montré aussi que l'intimidation en ligne ou cyberintimidation peut provoquer plus de risques suicidaires chez les victimes.

15 https://psychcentral.com/health/what-is-verbal-abuse#signs

16 https://www.medicalnewstoday.com/articles/327346#types-of-v

17 https://www.eeoc.gov/overview

18 https://ici.radio-canada.ca/nouvelle/1860615/cyberintimidation-rapport-internet-sur-journee

19 https://www.mcafee.com/content/dam/consumer/en-us/docs/reports/rp-cyberbullying-in-plain-sight-2022-global.pdf

20 https://www.lapresse.ca/actualites/sciences/2019-05-30/l-intimidation-laisse-des-traces-dans-l-adn-des-jeunes-victimes

Selon un autre article[21] publié sur le site ici.radio-canada.ca le 1 mars 2021, le service de police de Montréal (SPVM) a noté une forte hausse des commentaires haineux en ligne et une augmentation des nombres de dossiers traités en relation avec ce sujet.

Pour pouvoir traiter le discours de haine à l'échelle mondiale, les nations unies[22] le définie comme suit :

> Tout type de communication, orale ou écrite, ou de comportement, constituant une atteinte ou utilisant un langage péjoratif ou discriminatoire à l'égard d'une personne ou d'un groupe en raison de leur identité, en d'autres termes, de l'appartenance religieuse, de l'origine ethnique, de la nationalité, de la race, de la couleur de peau, de l'ascendance, du genre ou d'autres facteurs constitutifs de l'identité[23].

Dans sa stratégie et plan d'action pour la lutte contre les discours de haine[24] lancés en juin 2019, l'ONU (Organisation des Nations Unies) a précisé que le discours de haine n'est pas interdit en tant que tel, mais que c'est l'incitation à la discrimination, à l'hostilité ou à la violence qui est interdite. Elle a remarqué également qu'une vague de contenu haineux et de désinformations a été déclenchée durant la pandémie de coronavirus dans les médias sociaux. L'ONU précise que les conséquences des discours haineux durant la pandémie peuvent être catastrophiques. Ils peuvent causer des divisions et des troubles sociaux puisqu'ils peuvent exposer les minorités et les étrangers à la discrimination et la violence. Le secrétaire général des Nations Unies a lancé en mai 2020, un appel à une action mondiale contre les discours de haine alimentés par cette pandémie ». En 2023, il a affirmé que le discours de haine est une sonnette d'alarme qui précède la violence et qui annonce une menace de génocide. Le Tableau 1.1 contient quelques exemples de commentaires contenant des discours haineux propagés durant la pandémie de COVID-19 contre les peuples asiatiques et les citoyens d'origine asiatique, dans différents pays, dont l'Amérique et le Canada.

21 https://ici.radio-canada.ca/nouvelle/1774030/propos-haineux-medias-sociaux-facebook-twitter-instagram

22 https://www.un.org/fr/observances/countering-hate-speech

23 https://www.un.org/fr/hate-speech/understanding-hate-speech/what-is-hate-speech

24 https://www.un.org/en/genocideprevention/documents/advising-and-mobilizing/Action_plan_on_hate_speech_FR.pdf

Tableau 1.1. Exemples de commentaires haineux.

# commentaires	Commentaires
1	@BarackObama Fuck you . It's a Chinese virus moron. It's not the result of climate it's the result of countries that eat weird shit like bats.
2	@dancezaizai @SteveTheDuckBoi @NineAttributes @RealSaavedra Keep eating bats, and rats, and dogs, and cats and tell us what a civilized country you come from. China owns this pandemic and the entire works knows it! #STFU #KungFlu #FU 🖕 https://t.co/6YEhQRSTkw
3	"After experimenting on animals, if they aren't dead, they will take the rats and bats and snakes to a local meat market, and sell them for extra money. This could be how the virus got into humans." - A doctor, on TV, just said this. What the fuck? #coronavirus #ChinaVirus
4	@tom_cruise31 @hussain_imtiyaz What u want. I am not going to feed in Chinese propaganda. They eat shit and spread virus. This is not the 1st time. Who the hell is going to pay for the loss of lives, economic damage and debt burden in so many countries. CPC is an evil regime & must rightfully be maligned.
5	@WilsonLeungWS 😴 Chinese products are well known for their poor and cheap quality but all of a suddent this chineese virus proved the world wrong. Fucking chinese 🖕 you made the world hell. Just stop eating fucking animals

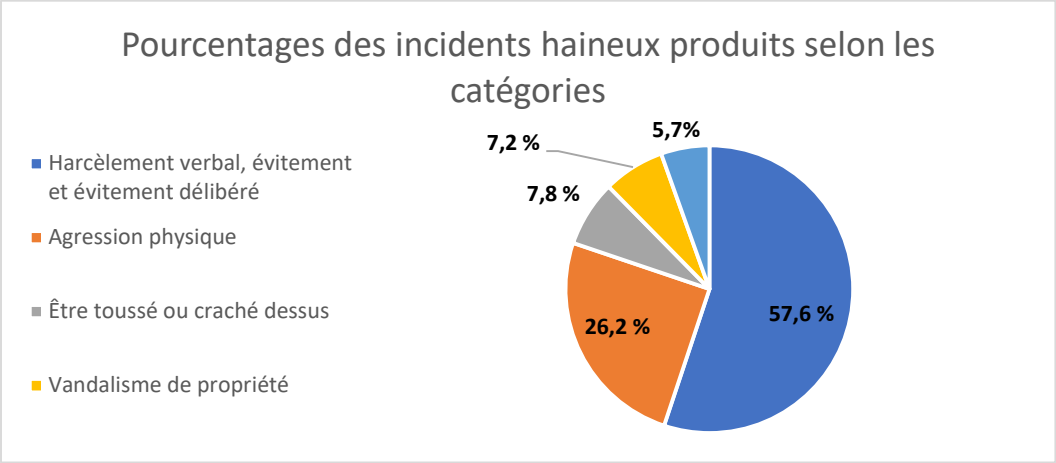
L'ONU a précisé aussi que l'étude du phénomène de la propagation des discours de haine à l'échelle mondiale s'avère difficile et ce en raison de l'absence d'une définition universelle du discours de haine, dans un contexte où les lois nationales, les communautés en ligne et les discours de haine propagés sont diversifiés. Cette difficulté est dû aussi au non-signalement des discours de haine par les victimes, aux autorités compétentes[25] .

Ce phénomène peut entraîner des conséquences fatales sur la société. Selon le rapport publié par l'ONU

25 https://www.un.org/fr/hate-speech/impact-and-prevention/challenges-of-tracking-hate

en 2020, les incidents racistes contre des Américains d'origine asiatique aux États-Unis ont augmenté depuis la pandémie de COVID19. Plus de 1 800 attaques racistes ont été signalés entre mars 2020 et mai 2020. Ces attaques comprennent de la violence physique et verbale contre des Américains d'origine chinoise, asiatique, coréenne, japonaise, vietnamienne, philippine et aussi birmane[26]. Dans son rapport national, le centre de signalement Stop AAPI Hate[27] a précisé que 10 905 incidents haineux contre des personnes asiatiques américaines et insulaires du Pacifique (AAPI) ont été signalés entre mars 2020 et décembre 2021, dont 824 se sont produits contre des ainés américains d'origine asiatique, âgés de 60 ans et plus. Selon le même rapport, 36,7% des incidents ont été signalés dans les rues publiques, tandis que 26,7% ont été produits dans les entreprises. La Figure 1.1 donne une idée sur les types d'incidents produits durant la même période avec leurs pourcentages[28].

Figure 1.1. Pourcentages des incidents haineux produits selon les catégories.



L'impact de ce type de contenu toxique peut s'avérer dangereux sur les individus et la société en général. C'est pourquoi il est nécessaire de l'identifier et le supprimer si nous voulons maintenir des conversations ouvertes et saines. Mais ce n'est pas facile lorsque le nombre d'utilisateurs est grand, d'où l'idée de trouver

26 https://spcommreports.ohchr.org/TMResultsBase/DownLoadPublicCommunicationFile?gId=25476

27 https://stopaapihate.org/our-mission/

28 https://stopaapihate.org/2022/05/24/release-elder-report-2022/

des solutions efficaces pour identifier ces commentaires toxiques et pouvoir les supprimer à temps.

1.2 Objectif

L'objectif de ce travail est de déterminer les meilleures techniques de prétraitement qui peuvent améliorer les performances d'un système de classification de texte. Pour atteindre cet objectif, nous divisons le travail en plusieurs étapes. La première étape consiste à choisir les ensembles de données que nous allons utiliser. Dans la deuxième étape, nous faisons l'échantillonnage pour les ensembles de données déséquilibrés. La troisième étape consiste à définir les deux catégories pour la classification (normal et toxique) et convertir le problème de classification en un problème binaire pour les ensembles de données multi-classes ou multi-labels. Ensuite, nous définissons les approches de prétraitement qui seront expérimentées dans ce travail. Puis, nous faisons le prétraitement des données en appliquant les différentes approches définies dans l'étape précédente. En dernier lieu, nous faisons une comparaison entre les différents résultats obtenus, à l'aide de plusieurs métriques comme la précision, l'accuracy, le rappel et la F1-mesure. La dernière étape va se focaliser sur la discussion des résultats et la comparaison de l'impact de chaque approche sur les performances du système.

1.3 Conclusion

Nous avons présenté une vue d'ensemble du discours de haine, ainsi que son impact sur les utilisateurs des réseaux sociaux. Nous avons expliqué l'importance grandissante de trouver des solutions pour identifier et supprimer ce type de commentaires. Enfin, nous avons cité les différents objectifs de ce travail, ainsi que les démarches que nous allons entreprendre.

CHAPITRE 2

REVUE DE LITTERATURE

2.1 L'apprentissage automatique

L'apprentissage automatique est une branche de l'intelligence artificielle qui a pour but de donner aux machines la capacité de l'auto-apprentissage et l'auto-évaluation afin de simuler l'apprentissage humain (Lv & Tang, 2011). La première machine capable d'apprendre l'expérience a été proposée par le mathématicien Alan Turing et elle remonte au milieu du 20ème siècle (Turing, 1995). Les chercheurs ont continué à développer ce concept durant des décennies et ils ont proposé plusieurs modèles, approches et ensembles de données. Grâce à ces efforts, l'apprentissage automatique est devenu un domaine puissant qui a connu un grand succès. Ses techniques sont utilisées de plus en plus par les chercheurs pour faire la reconnaissance, la classification ou la prédiction. Elles permettent d'automatiser la détection d'anomalies et de résoudre beaucoup de problèmes de classification dans le monde réel, comme la détection des fraudes par carte de crédit, la détection des courriels indésirables, la détection de l'hameçonnage sur Internet et la détection des intrusions (Subasi, Chapter 5 - Other classification examples, 2020). Elles ont été aussi utilisées dans d'autres domaines comme l'industrie et le domaine médicale (Syeda, et al., 2021), (Crowston & Bolici, 2019) et (Peng, Jury, Dönnes, & Ciurtin, 2021).

Il existe plusieurs étapes pour créer un modèle d'apprentissage automatique efficace. D'abord, il faut concevoir l'environnement de travail afin de remplir les objectifs. Cela étant, il faut collecter les données et les préparer pour respecter un format spécifique. Ensuite, le modèle choisi est entraîné avec les données prétraitées. L'étape finale consiste à évaluer les performances du modèle. Si ce dernier n'est pas performant, il sera entraîné et évalué à plusieurs reprises jusqu'à ce que ses performances soient optimales (Peng, Jury, Dönnes, & Ciurtin, 2021).

L'apprentissage automatique évoque plusieurs approches, non exclusives. Nous nous arrêterons sur quatre d'entre-elles : L'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage profond et l'apprentissage par renforcement.

L'apprentissage automatique supervisé consiste à utiliser un ensemble de données étiquetées pour entraîner un modèle dans le but de prédire ou de classer des résultats. Il analyse ces données pour trouver les relations entre les entrées et les sorties et applique par la suite les règles apprises pour développer un

modèle d'apprentissage automatique (Peng, Jury, Dönnes, & Ciurtin, 2021). Ce type d'apprentissage utilise deux approches : La classification et la régression. La classification est utilisée lorsque les données d'entrée sont catégorielles pour prédire la catégorie de la donnée de test (Robinson & Akins, 2021). La régression permet de déterminer la valeur de sortie en analysant l'ensemble de données d'entrée (Robinson & Akins, 2021). Ce type d'apprentissage est appliqué dans plusieurs domaines comme la prédiction du pronostic d'une maladie (Peng, Jury, Dönnes, & Ciurtin, 2021). Quant à l'apprentissage automatique non supervisé, il utilise des données d'entraînement non étiquetées, et permet d'analyser et de regrouper ces données pour découvrir des modèles cachés ou des structures intrinsèques (Peng, Jury, Dönnes, & Ciurtin, 2021). Le clustering est la technique utilisée dans ce type d'apprentissage, et consiste à analyser les données et chercher des similitudes ou des dissemblances entre eux pour les regrouper par la suite en groupes (clusters) en fonction de leurs similarités et leurs différences. Cette approche est appliquée, par exemple, dans le domaine médical (Peng, Jury, Dönnes, & Ciurtin, 2021).

L'apprentissage par renforcement n'a pas besoin de recevoir en entrée des données d'entraînement. Il collecte ses données directement de l'environnement réel dans lequel le système fonctionne. Dans ce type d'apprentissage, le système agit de manière autonome, il interagit avec l'environnement et apprend à partir d'expériences pour trouver la solution idéale. Il est capable d'explorer plusieurs solutions et d'exploiter le résultat de ces explorations pour adapter son comportement. Il apporte des récompenses locales selon le résultat obtenu (positif ou négatif) pour maximiser par la suite la solution qui a le cumul de récompenses le plus important (Robinson & Akins, 2021). L'apprentissage par renforcement est appliqué, entre autres, dans la conception des systèmes robotiques ou autonomes qui interagissent avec l'environnement (Robinson & Akins, 2021), et il est moins répandu dans le domaine médical (Peng, Jury, Dönnes, & Ciurtin, 2021).

L'apprentissage profond simule dans son comportement le comportement du cerveau humain. Il permet d'analyser les données en utilisant un réseau neuronal multicouches afin de trouver par la suite des modèles qui seront utilisés dans la prise de décision. Parmi les algorithmes d'apprentissage profond les plus populaires, il y a les réseaux de neurones convolutifs (CNN) et les réseaux de neurones récurrents (RNN) (Peng, Jury, Dönnes, & Ciurtin, 2021). L'apprentissage profond est utilisé dans le pronostic (Klang, et al., 2020), la reconnaissance en l'imagerie médicale (Le, Liu, & Huang, 2009), la détection d'objets visuels et la reconnaissance vocale et textuelle (Robinson & Akins, 2021).

2.2 La classification de texte

La classification de texte est une tâche fondamentale du traitement du langage naturel (NLP) (Li, et al., 2022). Elle consiste à étiqueter un texte non classifié et lui attribuer une catégorie en respectant certains critères et règles à l'aide des algorithmes d'apprentissage automatique (Ma, Li, Wu, & Zhang, 2018).

Par exemple, si nous avons un ensemble d'entraînement contenant des livres $L = (l_1, l_2 \dots l_n)$ déjà étiquetés avec $Classe_1$ et $Classe_2$ qui sont « adultes » et « enfants », la tâche de classification consiste à attribuer automatiquement à chaque nouveau livre une des deux catégories « adultes » ou « enfants », selon son contenu (Korde, 2012).

La notion d'Intelligence Artificielle (IA) a été introduite par Alan Turing dès 1950. En 1959, Arthur Samuel a introduit le terme d'apprentissage automatique (machine learning) pour son programme capable d'apprendre à jouer aux dames. Les années 1990 ont connu l'apparition des modèles tels que Support Vector Machine (SVM) et Random Forest (RF) (Dramsch, 2020). Ces modèles se sont montrés efficaces, en termes de précision et de stabilité, comparés aux méthodes traditionnelles qui nécessitent l'intervention humaine pour concevoir explicitement des règles linguistiques et des caractéristiques permettant de déterminer la catégorie qui sera attribuée à chaque texte. À partir de l'année 2010, La classification de texte utilisant des modèles d'apprentissage profond a commencé à remplacer la classification de texte à base de méthodes traditionnelles et elle est devenue dominante dans les travaux de recherche. Ceci est dû au développement rapide qu'a connu ce domaine (Li, et al., 2022). Il existe trois types de classification : La classification binaire, la classification multi-classe et la classification multi-label. La classification binaire est utilisée dans les problèmes à deux classes (positif et négatif) et permet d'attribuer une seule étiquette (Akhmouch, Bouanani, Dias, & Moreno, 2021). La classification multi-classe est utilisée dans les problèmes de plus de deux classes et permet aussi d'attribuer une seule étiquette (Liu, Chen, Li, Zhao, & Wu, 2021). Contrairement à la classification binaire et multi-classe qui ne permettent d'attribuer qu'une seule étiquette, la classification multi-label est utilisée dans les problèmes de plus de deux classes et permet d'attribuer plusieurs étiquettes à la fois (Liu, Chen, Li, Zhao, & Wu, 2021). Dans ce travail, nous allons utiliser la classification binaire dans l'identification des commentaires toxiques.

Le processus de la classification de texte se divise en quatre principales étapes : Le prétraitement, l'extraction des caractéristiques, l'entrainement du modèle de classification et l'évaluation du modèle (Korde, 2012).

La première étape de ce processus appelée prétraitement consiste à normaliser les données brutes afin de les rendre exploitables par l'algorithme de classification (Korde, 2012). Par la suite, vient l'étape d'extraction de caractéristiques qui permet de convertir le texte nettoyé en représentations numériques faciles à être interprétées par l'algorithme d'apprentissage. Cette étape permet de réduire le nombre de caractéristiques en conservant celles pertinentes et en supprimant celles non pertinentes (Kowsari, et al., 2019). L'étape suivante est d'utiliser ces données pour entrainer le modèle de classification. L'étape finale consiste à évaluer les performances du modèle entraîné en l'utilisant dans la classification des nouveaux textes, appelés données de test. Cette évaluation permet d'estimer ses capacités à classer des données inconnues.

La classification de texte est très répandue dans plusieurs applications, comme les réponses aux questions (Kalchbrenner, Grefenstette, & Blunsom, 2014) et (Logeswaran, Lee, & Radev, 2018), l'analyse des sentiments (Mercha & Benbrahim, 2023) et (Revathy, et al., 2022), et la reconnaissance des actes de dialogue (Bothe, Weber, Magg, & Wermter, 2018).

2.3 Prétraitement et extraction des caractéristiques

2.3.1 Prétraitement du texte

La phase de prétraitement de données textuelles est une étape importante dans le processus de classification de texte. Elle consiste à normaliser les données brutes en éliminant le bruit afin de les rendre exploitable par l'algorithme de classification (Heitz, 2006). Elle consiste à nettoyer et préparer le texte avant de procéder à la classification en supprimant tous les termes qui ne sont pas informatifs et importants pour la classification. Ce processus permet d'enlever le bruit et de réduire la dimensionnalité du problème (Haddi, Liu, & Shi, 2013). Dans la littérature, plusieurs techniques de prétraitement ont été utilisées pour résoudre des problèmes de classification de texte, telles que la conversion des mots en minuscule, la tokenisation, la lemmatisation, etc. Certaines de ces techniques sont plus utilisées que d'autres. Le Tableau 2.1 donne un aperçu de celles qui sont les plus citées dans la littérature.

Tableau 2.1. Techniques de prétraitement les plus utilisées.

Technique	Description
Conversion des mots en minuscule	Convertir le texte en lettres minuscules
Suppression des mots vides (stop words)	Supprimer les mots tels que « it », « the » et « this » qui sont des mots peu informatifs ou non informatifs et qui n'apporte aucune valeur et ne représentent pas le contenu du texte (Pedregosa, et al., 2011).
Tokenisation	Diviser le texte en petits morceaux appelés jetons, ces morceaux peuvent être des mots ou des caractères (Vel, 2021).
Racinisation (stemming)	Supprimer les suffixes et ramener les mots à leurs racines (Vel, 2021).
Lemmatisation (lemmatization)	Ramener les termes à leur forme canonique significative en considérant le contexte (Vel, 2021).
Correction orthographique	Corriger les fautes d'orthographes.

2.3.2 Extraction des caractéristiques

Une fois le texte nettoyé, il faut procéder à l'extraction des caractéristiques. C'est une étape importante pour la classification de texte, qui permet de réduire la dimensionnalité du problème, de diminuer le temps d'entraînement et d'améliorer la performance du classificateur. Elle consiste à extraire les caractéristiques du texte (appelées aussi attributs) qui représentent les informations pertinentes et informatives et qui affectent la tâche de classification (Liang, Sun, Yunlei, & Gao, 2017). Par exemple, dans la tâche de détection de commentaires agressifs, la présence de mots comme « *stupid n**** » ou « *shut up* » fait référence à un discours haineux et agressif. Parmi les techniques d'extraction des caractéristiques les plus populaires, nous trouvons la méthode TF-IDF et la méthode de sac de mots (bag-of-words) (Jahan & Oussalah, 2021) et (Zhang, Jin, & Zhou, 2010).

La technique TF-IDF (Term Frequency-Inverse Document Frequency) consiste à évaluer l'importance d'un terme pour un document dans un corpus (Kumar & Subba, 2020). Cette importance augmente proportionnellement au nombre de fois où le terme apparaît dans le document mais elle est compensée par sa fréquence dans le corpus. Par exemple, un terme qui apparaît fréquemment à la fois dans le document et dans le corpus est moins informatif qu'un terme qui apparaît fréquemment dans le document et rarement dans le corpus. Cette méthode renvoie un tableau de vecteurs de caractéristiques pour chaque texte d'entrée. Ces vecteurs de caractéristiques seront utilisés par la suite pour entraîner le classificateur. Cette technique est composée de deux parties : Term Frequency (TF) et Inverse Document Frequency (IDF) (Zhang, Yoshida, & Tang, 2011).

TF représente la fréquence d'apparition d'un terme dans le document. Elle permet de calculer le nombre de fois qu'un terme t apparaît dans un document d.

$$tf(t,d) = \frac{nombre\ de\ fois\ d'apparitions\ de\ t\ dans\ d}{nombre\ total\ de\ termes\ dans\ d}$$

IDF mesure l'importance d'un terme dans le document et détermine à quel point il est courant ou rare dans l'ensemble du document.

$$idf(t,D) = \log_2\left(\frac{nombre\ total\ de\ documents}{nombre\ de\ documents\ dans\ D\ contenant\ le\ terme\ t}\right)$$

Avec :

- t : terme
- d : document
- D : corpus

Le TF-IDF est calculé comme suit :

$$TF - IDF(t,d,D) = TF * IDF$$

Un terme est dit pertinent dans la classification lorsque son score TF_IDF est élevé. Autrement dit, le terme doit apparaître fréquemment dans le document mais rarement dans l'ensemble de corpus (valeurs de TF et IDF élevées).

Quant à la technique sac de mots (Bag-of-Words ou BoW), c'est une représentation du texte qui décrit l'occurrence de chaque mot (jeton) dans le document sans tenir compte des détails grammaticaux. Elle consiste à convertir les jetons en un ensemble de caractéristiques pour entraîner le classificateur sans tenir compte du contexte. Elle permet de représenter tous les jetons apparus dans le document sous forme d'un vecteur de termes à longueur fixe, où la dimension de chaque terme est sa fréquence dans le corpus (Li, Mao, Xu, Li, & Zhang, 2020). Par exemple, un document contenant un vocabulaire de n termes sera représenté par un vecteur de n dimensions, où la $n^{ième}$ dimension du vecteur est la fréquence du $n^{ième}$ terme dans le document.

Prenons l'exemple des trois documents suivants pour voir les étapes utilisées par la technique du BoW pour convertir le texte en vecteurs (Qader, Ameen, & Ahmed, 2019):

- ✓ $Document_1$: 'this document is a scientific research'
- ✓ $Document_2$: 'I think scientific articles are interesting but I prefer to read scientific journals '

La première étape consiste à parcourir l'ensemble de chaque document et décomposer tout le texte en jetons. Ainsi le résultat sera comme suit :

- ✓ $Document_1$: 'document', 'is', 'research', 'scientific', 'this'
- ✓ $Document_2$: 'are', 'articles', 'but', 'interesting', 'journals', 'prefer', 'read', 'scientific', 'think', 'to'

Des sacs de mots sont créés permettant de représenter l'occurrence de chaque mot (jeton) dans le document. La valeur de chaque mot dans le document est la fréquence de son apparition dans ce dernier.

- ✓ BoW_1: {'document' :1, 'is' :1, 'research' :1, 'scientific' :1, 'this' :1}
- ✓ BoW_2 : {'are' :1, 'articles' :1, 'but' :1, 'interesting' :1, 'journals' :1, 'prefer' :1, 'read' :1, 'scientific' :2, 'think' :1, 'to' :1}

La deuxième étape consiste à créer un dictionnaire de mots de l'ensemble des documents en combinant les deux documents. Cette union est décrite comme suit :

$$BoW_3 = BoW_1 \cup BoW_2$$

Le troisième document qui résulte de l'union des deux documents sera comme suit :

- ✓ $Document_3$ = 'are', 'articles', 'but', 'document', 'interesting', 'is', 'journals', 'prefer', 'read', 'research', 'scientific', 'think', 'this', 'to'

Sa représentation BoW est la suivante :

- ✓ BoW_3= {'are' :1, 'articles' :1, 'but' :1, 'document' :1, 'interesting' :1, 'is' :1, 'journals' :1, 'prefer' :1, 'read' :1, 'research' :1, 'scientific' :3, 'think' :1, 'this' :1, 'to' :1}

Finalement, un vecteur qui représente le nombre d'occurrences de chaque mot dans les différents documents est construit pour chaque document en utilisant la représentation BoW_3.

- ✓ Document 1 : [0 0 0 1 0 1 0 0 0 1 1 0 1 0]
- ✓ Document 2 : [1 1 1 0 1 0 1 1 1 0 2 1 0 1]

Le nombre d'occurrences d'un terme dans un corpus est la fréquence de son apparition dans ce dernier. Il est utilisé généralement pour classer ou catégoriser un texte. La méthode BoW est utilisée dans plusieurs domaines, comme le classement des textes et des images, la détection des objets et la détermination des distances (Qader, Ameen, & Ahmed, 2019).

2.3.3 Revue de littérature

La recherche de (Muaad, et al., 2022) étudie l'impact du prétraitement et des méthodes d'extraction et de sélection de caractéristiques sur la classification multi-classe des textes arabes. Cinq ensembles de données en arabe ont été utilisés avec six classificateurs, tels que SVC et Naive Bayes, pour pouvoir évaluer l'efficacité de l'approche adoptée. Les techniques de prétraitement utilisées dans cette recherche sont : La tokenisation, la suppression des stop words et le stemming. Concernant l'extraction des caractéristiques, deux approches ont été expérimentées : BoW et TF-IDF. L'accuracy obtenue par les algorithmes de classification utilisés varie entre 73% et 96% pour les ensembles de données utilisés avec les deux techniques TF-IDF et BoW. Cette accuracy augmente d'une valeur comprise entre 1% et 2% lorsque les données subissent un prétraitement. Cependant, elle diminue avec la technique TF-IDF dans le cas de l'ensemble de données Covid-19 qui contient des textes courts. Elle diminue aussi avec l'algorithme de classification binaire BNB (Bernoulli Naive Bayes) qui ne fonctionne pas bien avec la classification multi-classe. Les chercheurs ont conclu que l'extraction des caractéristiques et le prétraitement des données affectent les performances de la classification et permet de les améliorer, et que le choix de l'algorithme de classification ainsi que la nature de l'ensemble de données peuvent affecter les performances positivement, comme ils peuvent l'affecter négativement.

La recherche de (Anoual & Zeroual, 2021) étudie l'impact des techniques de prétraitement les plus utilisées sur la classification des textes arabes. Ils ont exploré neuf sites Web de nouvelles et ont extrait seulement du texte arabe afin de construire un corpus équilibré en arabe. Les données comportent 300 000 articles qui sont classées en six catégories : politique, culture, économie, sport, santé et technologie. Les

chercheurs se sont basés sur la littérature dans leurs choix des techniques de prétraitement, des techniques d'extraction de caractéristiques et des algorithmes de classification. Ils ont utilisé trois algorithmes de classification (arbre de décision, SVM, NB), ainsi que plusieurs techniques de prétraitement : la suppression des mots vides, la racinisation et la lemmatisation. L'expérience a montré que l'utilisation des techniques de prétraitement individuellement ou en combinaison donne de bons résultats, mais les meilleurs résultats sont obtenus en combinant toutes les techniques ensemble.

Le travail de (Haddi, Liu, & Shi, 2013) montre que le prétraitement de texte permet d'améliorer les performances du système dans l'analyse des sentiments. Les chercheurs ont combiné plusieurs techniques de prétraitement afin d'enlever le bruit et les parties non informatives du texte. Ils ont utilisé plusieurs techniques : suppression des espaces blancs et de mots vides, expansion des abréviations, racinisation (Stemming) et gestion des négations. Pour extraire les caractéristiques, trois différentes façons ont été utilisées (TF-IDF, FF et FP). FF (feature frequency) est le nombre d'occurrences d'un terme dans le document et FP (feature presence) prend la valeur 1 si la caractéristique (l'attribut) est présente dans le document et 0 sinon. Les résultats obtenus avec le classificateur choisi (SVM) montrent que l'accuracy s'est considérablement améliorée avec le prétraitement du texte. La valeur obtenue est passée de 78,33% à 81,5% avec la technique TF-IDF, de 72,7% à 83% avec la technique FF, et de 82,7% à 83% avec FP. À partir des résultats obtenus dans ce travail, les chercheurs ont conclu que l'utilisation d'une bonne combinaison des méthodes de prétraitement et de représentation du texte affecte positivement les performances du système et permet de les améliorer.

2.4 Algorithmes d'apprentissage automatique supervisé

Il existe plusieurs algorithmes d'apprentissage automatiques supervisé. L'article de (Dawei, Alfred, Obit, & On, 2021) et celui de (Jahan & Oussalah, 2021) donnent une synthèse des algorithmes les plus utilisés dans la classification de texte. Dans cette partie, nous allons présenter certain d'entre eux.

2.4.1 L'algorithme SVM

L'algorithme Support Vector Machines (SVM) ou machine à vecteurs de support est un algorithme d'apprentissage automatique supervisé qui peut être utilisé dans les problèmes de classification et de régression (Ghosh, Dasgupta, & Swetapadma, 2019). Cet algorithme est devenu populaire et largement utilisé par les chercheurs. Il est très utilisé dans la classification de texte (Hasan, et al., 2022), dans les tâches d'analyse de sentiments (Satrya, Pratiwi, Fa'rifah, & Abawajy, 2022) et (Cahyanti, Adiwijaya, &

Faraby, 2020), il est utilisé aussi dans la détection des discours haineux (Nascimento, Cavalcanti, & Da Costa-Abreu, 2023).

Cette méthode d'apprentissage automatique permet d'obtenir de bonnes performances dans la classification de texte (Liu, Lv, Liu, & Shi, 2010). Elle consiste à construire un hyperplan dans un espace de n dimensions pour séparer deux classes en fonction des caractéristiques, de façon que ce dernier soit le plus éloigné possible des données de chaque classe. Autrement dit, le classificateur cherche à maximiser la marge qui est la distance entre la frontière de séparation et les échantillons les plus proches des deux classes (appelés vecteurs de support) afin de minimiser le bruit (Ghosh, Dasgupta, & Swetapadma, 2019). Pour identifier l'hyperplan, SVM utilise des fonctions appelées fonctions noyau. Ces dernières se divisent en deux catégories : fonctions de classification linéaire et d'autres de classification non-linéaire (Syed, et al., 2020). La fonction noyau de classification linéaire permet d'identifier l'hyperplan permettant de séparer linéairement les données et dont l'équation est $wx + b = 0$ où w est un vecteur et b est le biais. Ce dernier est dit optimal si pour tous les points de données de la classe positive (Class +1) $wx + b \geq 1$ et pour tous ceux de la classe négative (Class -1) $wx + b \leq -1$ (Hassanshahi, Jastrzebski, Malik, & Lahav, 2023).

Dans la classification non linéaire, les deux classes ne peuvent pas être séparées correctement par un hyperplan linéaire. Par conséquent, une étape supplémentaire est requise pour transformer la représentation des données afin que ces dernières puissent être manipulées linéairement. C'est le rôle de la fonction noyau qui permet de mapper l'espace entrée à un espace vectoriel de dimensions supérieures (Hassanshahi, Jastrzebski, Malik, & Lahav, 2023) pour ainsi pouvoir trouver un séparateur linéaire. Il existe différents types de noyaux : fonction du noyau RBF, fonction du noyau polynomial, fonction du noyau sigmoïde et la fonction du noyau linéaire qui est utilisée dans la classification linéaire (Syed, et al., 2020).

2.4.2 L'algorithme Logistic Regression

L'algorithme Logistic Regression ou régression logistique est un algorithme d'apprentissage automatique supervisé. Il existe deux types de régression logistique : multinomiale et binaire (Ma, Li, Wu, & Zhang, 2018) et (Edgar & Manz, 2017). La régression logistique multinomiale permet d'étudier plusieurs variables qui ne sont pas similaires et ne rentrent pas obligatoirement dans la même catégorie. Elle est utilisée dans les problèmes de classification où plus de deux résultats sont possibles (Gaikwad & Doke, 2023). Quant à la régression logistique binaire, qui est celle utilisée dans ce travail, elle permet de faire la prédiction de la

probabilité d'un résultat de classification binaire pouvant prendre deux valeurs (soit 0 ou 1, oui ou non) à l'aide d'un ensemble de données de variables indépendantes (Hanshika. V., Preethi., Sreemathy. S., & Ezhillin Freeda., 2023) et (Gaikwad & Doke, 2023). La Figure 2.6 donne un aperçu du modèle de régression logistique binaire. X1, X2 et X3 sont les variables d'entrées et les variables de sortie ne peuvent prendre que les valeurs 0 ou 1.

Supposons que Y est la variable dépendante que nous voulons prédire et que X est l'ensemble des variables indépendantes que nous allons utiliser pour faire la prédiction. La relation entre les variables dépendantes et indépendantes est modélisée sous la forme de l'équation linéaire suivante avec β0, β1, β2… βN comme coefficients de régression (Battineni, Sagaro, Chintalapudi, Amenta, & Tayebati, 2019):

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_N X_N$$

Cette équation est mappée à la fonction sigmoïde pour limiter la valeur du résultat Y à l'intérieur d'une plage comprise entre 0 et 1. L'équation de régression devient comme suit (Sinnott, Duan, & Sun, 2016):

$$Y = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_N X_N)}}$$

La régression logistique est une transformation d'une régression linéaire en utilisant la fonction sigmoïde (Sinnott, Duan, & Sun, 2016). La probabilité du résultat suit la loi de Bernoulli (Sinnott, Duan, & Sun, 2016) et (Pedregosa, et al., 2011) et elle est modélisée par une courbe sigmoïde bornée par 0 et 1. La formule de la régression logistique est définie comme suit (Belyadi & Haghighat, 2021):

$$Y = \frac{1}{1 + e^{-X}}$$

La loi de Bernoulli est une loi de probabilité d'une variable aléatoire discrète qui prend la valeur 1 (succès) avec une probabilité p ou 0 (échec) avec probabilité de 1-p. Elle est définie par l'équation suivante (Sinharay, 2010) et (Yelne, 2023):

$$P(X == x) = p^x (1 - p)^{1-x}$$

La régression logistique est une méthode qui est largement utilisée dans de nombreux domaines (banques,

finance, éducation et médecine) pour faire la classification et la prédiction (Gaikwad & Doke, 2023). C'est un modèle simple et efficace pour les problèmes de classification et qui présente de bonnes performances avec les classes qui sont linéairement séparables (Subasi, 2020). Il permet aussi d'obtenir de bonnes performances pour la détection du discours haineux sur les réseaux sociaux (Nascimento, Cavalcanti, & Da Costa-Abreu, 2023).

2.4.3 L'algorithme Naive Bayes

L'algorithme Naive Bayes est un classificateur basé sur le théorème de Bayes (Karabatak, 2015). Il fonctionne sur le principe de probabilités conditionnelles (Hindi, Aljulaidan, & AlSalman, 2020). Il permet de déterminer la probabilité d'un événement étant donné qu'un autre événement s'est produit, en utilisant la formule suivante (Chen, Huang, Tian, & Qu, 2009) :

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Où $P(A|B)$ est la probabilité postérieure, c'est à dire la probabilité de l'hypothèse A étant donné la preuve B. $P(B|A)$ est la probabilité à postériori, c'est à dire la probabilité de la preuve B étant donné l'hypothèse A . $P(A)$ est la probabilité à priori de A, c'est-à-dire la probabilité que l'hypothèse A soit vraie et $P(B)$ est la probabilité de la preuve B quelle que soit l'hypothèse A (Berrar, 2018).

Cet algorithme a été utilisé dans plusieurs domaines comme la médecine (Karabatak, 2015) et (Cui, et al., 2018), l'analyse de sentiments (Mathapati, Shahapurkar, & Hanabaratti, 2017) et (Martiti & Juliane, 2021) et aussi la classification de texte (Chen, Huang, Tian, & Qu, 2009) et (Hindi, Aljulaidan, & AlSalman, 2020).

2.5 Algorithmes d'apprentissage profond

Il existe plusieurs algorithmes d'apprentissage profond. L'article de (Jahan & Oussalah, 2021) donne un aperçu sur l'état de leurs utilisations dans des travaux antérieurs. Dans cette partie, nous allons présenter l'un de ces algorithmes, le réseau neuronal récurrent ainsi que deux de ses variantes qui sont très utilisées dans la détection des discours haineux, le LSTM (Long Short Term Memory) et le Bi-LSTM (Biderctionnal Long Short Term Memory). (Jahan & Oussalah, 2021) ont identifié 96 documents utilisant les architectures de réseaux de neurones pour détecter les discours de haine sur les réseaux sociaux. 23% de ces études utilisent LSTM, 8% utilise Bi-LSTM et 12% combine LSTM avec une autre architecture neuronale.

2.5.1 L'architecture RNN

Les réseaux de neurones récurrents (RNN) sont les architectures populaires très utilisées dans le traitement du langage naturel (NLP) (Zhao, et al., 2020). Ils ont une structure neuronale où les connexions entre les unités forment un graphe dirigé le long d'une séquence temporelle. Un RNN utilise son état interne pour traiter des séquences d'entrée. Il est doté d'une mémoire qui lui permet de conserver les informations et se souvenir de toutes les relations des mots qui précèdent le mot à prédire. Il est capable de créer des réseaux avec des boucles permettant aux informations de passer d'une couche à l'autre de sorte que chaque état est influencé par ses états précédents, ce qui lui permet d'avoir un comportement dynamique temporel (Sharma, Chatterjee, Kaur, & Vavilala, 2022). Cette capacité de capturer et de stocker des relations de longue dépendance favorise l'apprentissage dans le domaine temporel et l'exploitation de la nature séquentielle de l'entrée (Singh, Kuzhagaliyeva, & Sarathy, 2022). Ainsi, les réseaux de neurones récurrents sont utilisés dans le traitement de texte (Kumaraswamy, 2021), la reconnaissance vocale audiovisuelle (Feng, Guan, Li, Zhang, & Luo, 2017) et les séquences d'ADN (Lugo & Barreto-Hernández, 2021) et (Cheng, Huang, & Liou, 2012).

La structure d'un réseau neuronal est composée de trois types de couches. La couche d'entrée x, la couche cachée h et la couche de sortie o (Feng, Guan, Li, Zhang, & Luo, 2017). À l'instant actuel t, l'entrée actuelle x^t et l'état caché précédent $h^{(t-1)}$ sont ajoutés au RNN, permettant ainsi d'obtenir un état caché h^t. L'état h^t est utilisé pour produire la sortie o^t. À l'instant $(t+1)$, l'entrée $x^{(t+1)}$de la séquence d'entrée et l'état caché précédent $h^{(t)}$ passeront par le RNN et permettront d'obtenir ainsi l'état caché suivant $h^{(t+1)}$ qui sera utilisé pour produire la sortie $o^{(t+1)}$. Nous continuons ainsi jusqu'à mettre toutes les entrées de la séquence d'entrée dans le RNN. De cette façon, ce dernier est capable de garder en mémoire le contexte de l'entrée précédente pendant l'entraînement.

Les équations de base permettant de calculer la sortie du RNN sont définies comme suit :

$$h^{(t)} = \sigma\left(b_1 + Wh^{(t-1)} + Ux^{(t)}\right)$$

$$o^{(t)} = b_2 + Vh^{(t)}$$

Où $h^{(t)}$, $x^{(t)}$ et $o^{(t)}$ désignent respectivement l'état caché, le vecteur d'entrée et la valeur de la sortie à l'instant actuel t. $h^{(t-1)}$ représente l'état caché à l'instant précédent $(t-1)$ et U, W et V désignent

respectivement les poids de la couche d'entrée, la couche cachée et celle de sortie. Tandis que b_1 et b_2, désignent les biais et σ est la fonction d'activation.

Pour que le RNN puisse apprendre les dépendances à long terme, il faut rétro-propager le gradient loin dans le temps. La rétropropagation du gradient dans le temps (back propagation) est un algorithme qui utilise la descente de gradient pour calculer le gradient de l'erreur pour chaque neurone, à partir de la dernière couche vers la première couche. Cela consiste à évaluer une fonction de perte, l'erreur totale entre les échantillons d'entrée et la fonction de sortie. Cela permet de mettre à jour les paramètres de poids pour corriger les erreurs et trouver par la suite le minimum local permettant d'aboutir à une configuration stable du réseau de neurones (Cui, 2018). Cette propagation à travers de nombreux pas de temps peut devenir instable et créer des problèmes lors de l'entraînement, ce qui peut causer l'instabilité du réseau comme l'explosion ou la disparition de gradient. L'explosion de gradient est dû à l'accumulation de grands gradients d'erreurs, entraînant ainsi de très grandes mises à jour des poids du modèle, tandis que la disparition de gradient survient lorsque ce dernier est faible et qu'il ne cesse de s'atténuer à chaque pas de temps. Ces deux problèmes demeurent la difficulté principale lors de l'entraînement des réseaux neuronaux récurrents RNN (Feng, Guan, Li, Zhang, & Luo, 2017).

2.5.2 L'architecture LSTM

Le réseau neuronal LSTM (Long short term memory) est l'une des architectures de réseaux neuronaux récurrents les plus populaires (Dvornek & Li, 2023). Il est doté d'une cellule mémoire qui remplace la couche cachée des réseaux RNN permettant de capturer les dépendances à long terme entre les séquences de mots. Cela permet de résoudre le problème d'explosion ou de disparition de gradient rencontré avec les réseaux de neurones récurrents. La cellule mémoire est le point clé du LSTM et l'état de cette cellule permet aux informations de circuler le long de la chaîne. Elle est composée d'une porte d'entrée, d'une porte d'oubli, d'une porte de sortie et d'un neurone auto-récurrent.

Les portes sont composées d'une couche de réseau neuronal sigmoïde et d'une opération de multiplication ponctuelle (Jiang, 2022). Le LSTM utilise ces portes pour déterminer s'il doit conserver les informations pertinentes et oublier celles non pertinentes. Si la valeur de la porte est 1, il garde la valeur de la couche correspondante, et si la valeur de la porte est 0, elle réduit cette valeur à zéro (Feng, Guan, Li, Zhang, & Luo, 2017). La porte d'entrée détermine les informations qui seront ajoutées dans l'unité cellulaire, la porte d'oubli reçoit la valeur de la cellule mémoire h_{t-1} et le vecteur d'entrée x_t et génère

en sortie le nombre 1 pour indiquer à l'état de la cellule qu'il faut conserver les informations, ou le nombre 0 pour lui indiquer qu'il faut les rejeter, et ainsi la porte de sortie détermine quelles informations de l'unité cellulaire vont passer à la sortie (Jiang, 2022). Les mises à jour des états de cellules, les portes et la sortie sont définies par les équations suivantes (Bao, Yue, & Rao, 2017) :

$$i_t = \sigma\left(U_i x_t + W_i h_{(t-1)} + b_i\right)$$

$$f_t = \sigma\left(U_f x_t + W_f h_{(t-1)} + b_f\right)$$

$$o_t = \sigma\left(U_o x_t + W_o h_{(t-1)} + b_o\right)$$

$$\tilde{C}_t = \tanh\left(W_c x_t + U_c h_{(t-1)} + b_c\right)$$

$$C_t = i_t \tilde{C}_t + f_t C_{(t-1)}$$

$$h_t = o_t \odot \tanh C_t$$

Où :

- i_t, f_t et o_t sont respectivement les valeurs de la porte d'entrée, la porte d'oubli et celle de sortie.
- x_t est le vecteur d'entrée de la cellule mémoire à l'instant actuelle t.
- h_t et h_{t-1} sont respectivement les valeurs de la cellule mémoire à l'instant actuelle t et précédente $t-1$.
- C_t et C_{t-1} sont respectivement les états de la cellule mémoire à l'instant actuelle t et précédente $t-1$.
- $\tilde{C}_t$ est l'état candidat de la cellule mémoire à l'instant actuelle t.
- U_i, U_o, U_f, U_c, W_i, W_o, W_f et W_c sont les matrices de poids.
- b_i, b_o, b_f et b_c sont les biais.
- σ est la fonction sigmoïde.
- $tanh$ est la fonction tangente hyperbolique.
- $\odot$ est la multiplication par éléments.

Parmi les principaux domaines d'applications de l'architecture LSTM, il y a l'identification du génome (Lugo & Barreto-Hernández, 2021), la classification de texte (Jiang, 2022) et la reconnaissance vocale audiovisuelle (Feng, Guan, Li, Zhang, & Luo, 2017).

2.5.3 L'architecture Bi-LSTM

L'architecture Bi-LSTM (Bidirectional Long Short-Term Memory) est une architecture des réseaux neuronaux récurrents. Selon l'article de (Jahan & Oussalah, 2021), elle était moins utilisée que le CNN () et LSTM () dans la détection des discours haineux sur les réseaux sociaux, comme le montre la Figure 2.8. Cette architecture était très performante et a montré de bons résultats dans plusieurs travaux de classification de texte (Liang, Chen, Yi, & Fan, 2022) et (Guo, Zhang, & Xiao, 2022), d'analyse de sentiments (Xie, Chen, Gu, Liang, & Xu, 2019) et dans la détection des discours haineux (Baruah, Barbhuiya, & Dey, 2019).

Le Bi-LSTM est appelé LSTM bidirectionnel parce qu'il est constitué de deux réseaux LSTM indépendants qui fonctionnent dans les deux sens pour traiter une séquence de données. Le premier LSTM la traite en commençant par le début vers la fin et le deuxième la traite dans l'ordre inverse, de la fin vers le début (Yousaf & Nawaz, 2022). Il est capable de mieux comprendre la relation entre les séquences en conservant le contexte des données et en déterminant les mots suivants et précédents dans une phrase, permettant ainsi de capturer les dépendances à long terme entre les séquences de mots et prédire du texte par la suite (Xie, Chen, Gu, Liang, & Xu, 2019).

Le Bi-LSTM est constitué de deux couches cachées, une couche cachée avant h_t^f (forward) qui considère le vecteur d'entrée tel quel et une autre vers l'arrière h_t^b (backward) qui le considère dans l'ordre inverse. Le LSTM forward reçoit en entrée le vecteur caché précédent h_{t-1}^f et le vecteur d'entrée x_t et génère le vecteur caché à l'instant actuel h_t^f. Le LSTM backward calcule le vecteur caché inverse h_t^b en se basant sur le vecteur caché précédent opposé h_{t-1}^b et le vecteur d'entrée x_t. Les deux vecteurs cachés h_t^f et h_t^b sont fusionnés pour donner le vecteur caché de sortie final (Xie, Chen, Gu, Liang, & Xu, 2019). Le Bi-LSTM est représenté par les équations suivantes (Yousaf & Nawaz, 2022):

$$h_t^f = \tanh\left(U_h^f x_t + W_h^f h_{t-1}^f + b_h^f\right)$$

$$h_t^b = \tanh\left(U_h^b x_t + W_h^b h_{t+1}^b + b_h^b\right)$$

$$y_t = W_y^f h_t^f + W_h^b h_t^b + b_y$$

Où :

- x_t est le vecteur d'entrée à l'instant actuelle t.
- y_t est le vecteur de sortie à l'instant actuelle t.
- h_t^f et h_t^b sont respectivement l'état caché forward et celui backward à l'instant actuelle t.
- h_{t-1}^f et h_{t+1}^b sont respectivement l'état caché forward à l'instant $t-1$ et celui backward à l'instant actuelle $t+1$.
- U_h^f, U_h^b, W_h^f, W_h^b et W_y^f sont les matrices de poids.
- b_h^f, b_h^b et b_y sont les biais.
- $tanh$ est la fonction tangente hyperbolique.

Parmi les principaux domaines d'applications de l'architecture Bi-LSTM, il y a la reconnaissance d'entités nommées (Wang & Guan, 2020), la classification de texte (Jang, Kim, Harerimana, Kang, & Kim, 2020) et la détection des attaques et des intrusions ciblant la plateforme cloud (Xu & Zheng, 2022).

2.6 La validation croisée

Un modèle d'apprentissage automatique est entraîné sur des données étiquetées avant d'être utilisé sur des données de test. Mais, il faut le valider d'abord en évaluant ses performances sur des données inconnues pour voir s'il généralise bien et ne présente pas de sous-apprentissage, ni de sur-apprentissage. Le sur-apprentissage (overfitting) se produit lorsqu'un modèle s'adapte aux données d'entraînement et fournit de bonnes prédictions sur l'ensemble d'entraînement mais ne prédit pas bien sur l'ensemble de test (Badillo, et al., 2020). Tandis que le sous-apprentissage (underfitting) se produit lorsque les valeurs prédites par le modèle ne sont pas proches de celles obtenues avec l'ensemble d'entraînement (Badillo, et al., 2020).

La validation croisée est une méthode permettant d'évaluer les performances d'un modèle d'apprentissage automatique afin de s'assurer de l'exactitude des prédictions (García, Sánchez, & Marqués, 2019) et (Niu, Li, Wang, & Han, 2018). Elle est utilisée par les chercheurs pour éviter le sous-apprentissage et le sur-apprentissage (Baturynska & Martinsen, 2021) , et pour évaluer les performances du modèles (Niu, Li, Wang, & Han, 2018). Cette méthode consiste à diviser aléatoirement les données en K blocs de

tailles approximatives. Le modèle est entraîné par la suite sur les K-1 blocs et le dernier bloc est utilisé pour la validation. Cette procédure est répété K fois (Lameiro & Schreier, 2016). La validation croisée utilisée dans ce travail est celle de 5 blocs.

2.7 La technique GridSearch

Grid Search ou recherche par grille est une technique d'apprentissage automatique utilisé pour le réglage des hyperparamètres. Elle consiste à essayer différentes valeurs des hyperparamètres spécifiés, puis à choisir celle qui fournit les meilleurs résultats. Cette technique identifie d'abord les ensembles de valeurs des hyperparamètres choisis. Par la suite, elle effectue une recherche sur toutes les combinaisons de valeurs possibles pour trouver la combinaison idéale qui permet d'obtenir la meilleure performance. Autrement dit, la technique GridSearch est une méthode qui effectue plusieurs calculs sur des ensembles d'hyperparamètres spécifiés pour chaque algorithme d'apprentissage automatique, et renvoie l'ensemble d'hyperparamètres qui permet d'obtenir la meilleure performance du modèle (Contreras, Orellana-Alvear, Muñoz, Bendix, & Célleri, 2021), (Nobel, Kononova, Briaire, Frijns, & Bäck, 2022) et (Kumar, Sharma, & Varadwaj, 2011).

La méthode de recherche en grille est utilisée par plusieurs chercheurs pour déterminer les meilleures valeurs des hyperparamètres donnant la meilleure performance du modèle (Nobel, Kononova, Briaire, Frijns, & Bäck, 2022), (Contreras, Orellana-Alvear, Muñoz, Bendix, & Célleri, 2021), (Bayu, Arifin, & al Fatta, 2019) et (Kumar, Sharma, & Varadwaj, 2011).

2.8 La classification du contenu haineux

Un commentaire toxique est défini comme étant un commentaire irrespectueux ou offensant pouvant pousser les autres utilisateurs à quitter une discussion. Il peut prendre plusieurs formes comme une discrimination, une insulte, un harcèlement ou un discours haineux (Risch & Krestel, 2020). Tandis qu'un discours haineux, c'est tout discours contenant de la haine ou encourage la violence envers une personne ou un groupe de personnes sur la base des caractéristiques spécifiées telles que l'origine ethnique, le sexe, l'orientation sexuelle, la religion, l'âge ou la nationalité (Ayo, Folorunso, Ibharalu, & Osinuga, 2020).

Plusieurs chercheurs se sont intéressés à la classification des commentaires toxiques et des discours haineux. (Poojitha, Sai Charish, Kuamr Reddy, & Ayyasamy, 2023) propose un système de classification de commentaires toxiques appelé LSTM-CNN. Ce système est une combinaison des deux modèles

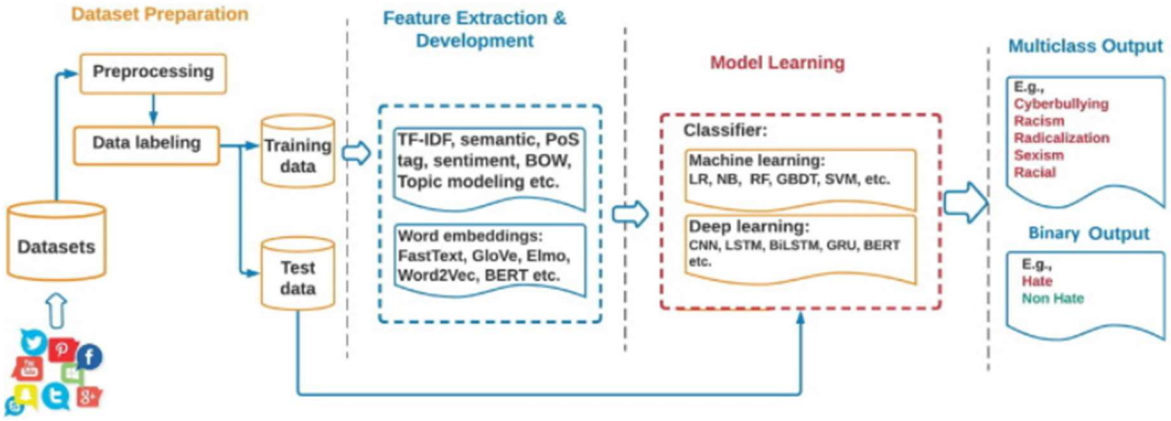
d'apprentissage profond LSTM (Long Short-Term Memory) et CNN (Convolutional Neural Network). Les chercheurs ont utilisé un ensemble de données collectées sur Youtube contenant des commentaires toxiques. Un rééchantillonnage a été appliqué aux données pour équilibrer la classe toxique et celle non-toxique. Pour préparer et nettoyer ces données, ils ont supprimé les mots vides, les signes de ponctuation et les nombres. Ils ont appliqué aussi la tokenisation, la racinisation (stemming) et la correction orthographique. Par la suite, les deux techniques d'extractions de caractéristiques TF-IDF et BoW ont été explorées par cette recherche avec d'autres différents modèles d'apprentissage automatique et profond tels que SVM (Support vector machine), RF (Random forest), LSTM (Long short-term memory) et CNN (Convolutional neural network). Les résultats obtenus montrent que le modèle proposé présente la meilleure performance par rapport aux autres modèles, il donne une accuracy de 97,7%, 96% est obtenue par LSTM et 94% par CNN et SVM. Les chercheurs ont conclu que le modèle proposé s'avère efficace à la fois avec les données équilibrées et celles non équilibrées, même si sa complexité de calcul est plus élevée que les modèles individuels. Ils ont conclu aussi que les performances des autres modèles avec l'ensemble de données équilibré sont meilleures que lorsque l'ensemble de données est déséquilibré. Les chercheurs ont conclu aussi que dans le cas d'un ensemble de données déséquilibré, un suréchantillonnage des données permet de donner de meilleurs résultats par rapport à un sous-échantillonnage et ceci est dû au nombre de caractéristiques qui diminue dans le deuxième cas.

L'article de (Nascimento, Cavalcanti, & Da Costa-Abreu, 2023) présente une analyse des différentes définitions du terme « discours haineux », un aperçu sur les recherches qui s'y sont intéressés entre l'année 2015 et 2022, les ensembles de données proposés dans la littérature, ainsi que les techniques utilisées pour l'extraction des caractéristiques et la classification. Les chercheurs ont conclu que la détection automatique des discours haineux est considérée dans la plupart des recherches comme un problème d'apprentissage supervisé et que les techniques d'extraction de caractéristiques qui sont utilisées pour résoudre ce problème sont les techniques simples comme BOW, n-grams ou TFIDF.

Dans la recherche de (Jahan & Oussalah, 2021), les auteurs se sont concentrés sur l'étude de trois principaux points : Les domaines d'application de la détection automatique des discours haineux, l'utilisation de l'apprentissage profond dans le même but et les ensembles de données disponibles. Dans leur article, ils donnent plusieurs définitions de discours haineux, les concepts qui y sont reliés ainsi qu'une vue d'ensemble sur les travaux élaborés en relation avec ce sujet. Ils donnent aussi un aperçu des méthodes utilisées et des différents ensembles de données accessibles au public. Selon (Jahan & Oussalah,

2021), le processus de détection de discours haineux se décline en plusieurs étapes, comme le montre la Figure 2.1 : la collection et préparation des données, l'extraction et représentation des caractéristiques, l'entraînement du modèle de classification et l'évaluation des performances du système.

Figure 2.1. Processus de détection automatique de discours haineux (Jahan & Oussalah, 2021).



(Jahan & Oussalah, 2021) ont passé en revue les travaux publiés entre 2000 et 2021 sur la détection automatique des discours haineux. Ils ont remarqué que la technique d'extraction de caractéristiques TF-IDF ainsi que l'algorithme d'apprentissage automatique SVM étaient les plus utilisées dans les recherches avant l'explosion de la technologie d'apprentissage profond et le grand succès qu'a connu cette dernière. Ils ont aussi trouvé que les techniques d'intégration de mots tels que Word2Vec, GloVe et FastText avec différents modèles d'apprentissage profond sont de plus en plus utilisés par les chercheurs. Ils donnent une comparaison entre des modèles d'apprentissage profond, dans la détection des discours haineux, ainsi que les forces et faiblesses de chacun. Après avoir analysé 69 ensembles de données, ils ont conclu que l'utilisation d'un ancien ensemble de données s'avère dans quelque cas difficile à cause de la suppression des données sur la plateforme source, ou de sa petite taille ainsi que la mauvaise annotation des données. Comme principaux défis qui peuvent nuire au progrès de ce domaine, ils ont identifié le manque de ressources pour la détection des discours haineux dans d'autres langues que l'anglais et la non-disponibilité d'un code régulièrement maintenu et accessible au public.

(Rasel, Sultana, Akhter, & Meesad, 2018) se sont concentrés dans leur recherche sur l'identification des commentaires agressifs sur les réseaux sociaux en les classant en trois catégories : offensant, discours haineux et neutre. Leur approche se déploie en plusieurs étapes. D'abord, ils ont construit un ensemble de données contenant 1000 commentaires collectés de Twitter. Cela étant, ces données ont été nettoyées

en appliquant plusieurs techniques de prétraitement : conversion en lettres minuscules, suppression des informations inutiles (nombres, symboles, signes de ponctuation, URLs, retweets et espaces blancs), tokenisation, correction orthographique, suppression des mots vides, racinisation (stemming) et suppression des contractions. Par la suite, ils ont extrait les caractéristiques à l'aide de trois différentes méthodes ; TF-IDF, n-gram et B0W. Les deux étapes suivantes consistaient à réduire d'abord les caractéristiques en éliminant les caractéristiques inutiles à l'aide de la technique LSA (Latent Semantic Analysis) et sélectionner par la suite les caractéristiques en utilisant la similarité cosinus. Trois algorithmes d'apprentissage automatiques ont été utilisés dans la phase d'entraînement : RF (Random Forest), SVM et LR (Logistic Regression). Chacun de ces modèles a été entraîné séparément et évalué avec la validation croisée pour comparer les performances des uns aux autres. L'approche proposée a permis d'obtenir de bonnes performances avec les trois classificateurs. L'accuracy obtenu a augmenté considérablement avec l'application du LSA et de la similarité cosinus. Ces deux techniques ont permis d'éliminer les caractéristiques redondantes et d'améliorer par la suite la performance. Le modèle le plus performant était RF (Random Forest) avec une accuracy de 93%. Par la suite vient LR (Logistic Regression) avec une accuracy de 86%. Avec cette approche, SVM était le moins performent et il a obtenu une accuracy de 72%. Les chercheurs ont conclu que la combinaison des techniques TF-IDF, LSA et similarité cosinus permet d'optimiser le vecteur de caractéristiques pour l'identification des commentaires agressifs sur les réseaux sociaux en réduisant le nombre de caractéristiques.

(Badjatiya, Gupta, Gupta, & Varma, 2017) ont utilisé des architectures différentes d'apprentissage profond pour identifier des tweets haineux, tels que LSTM (Long short-term memory) et CNN (Convolutional neural network), ainsi que trois techniques d'extraction de caractéristiques ; TF-IDF, BoW et n-gram. L'ensemble de données utilisé contient 16 000 tweets divisés en trois catégories : sexistes, racistes et neutres. Avant d'entraîner les modèles choisis, les intégrations des mots sont initialisées de deux façons : intégration aléatoire et intégration GLoVe. Les résultats obtenus montrent que les approches utilisant des modèles neuronaux sont plus performantes que celles qui utilisent des algorithmes de base. De plus, la combinaison des architectures de réseaux neuronaux avec la technique GBDT (Gradient Boosted Decision Tree) a donné les meilleurs résultats, comparée aux méthodes traditionnelles.

2.9 Conclusion

Dans ce chapitre, nous avons donné un aperçu sur l'apprentissage automatique. Nous avons présenté la classification de texte ainsi que les étapes liées à ce processus. Nous avons expliqué aussi quelques

algorithmes d'apprentissage automatique qui sont utilisés pour la classification supervisée. Enfin, nous avons présenté la classification du contenu toxique et nous avons donné un aperçu sur quelques travaux sur ce sujet. Dans le chapitre suivant, nous mettons l'accent sur la méthodologie suivie dans ce travail. Nous présentons les hypothèses à vérifier, les ensembles de données utilisés ainsi que la démarche adoptée dans la classification des discours haineux.

CHAPITRE 3
MÉTHODOLOGIE

Dans ce chapitre, nous présentons la méthodologie suivie dans ce travail. D'abord, nous donnons un aperçu sur les hypothèses à vérifier ainsi que sur les ensembles de données utilisés. Ensuite, nous décrivons les différentes étapes précisées dans la classification de texte. Et enfin, nous expliquons la démarche adoptée.

3.1 Hypothèses à vérifier

Vu que les techniques de prétraitement jouent un rôle important dans l'amélioration des performances du système de classification de texte, ce projet va se concentrer sur la vérification des hypothèses suivantes :

- L'utilisation de combinaisons de techniques de prétraitement du texte, influence-t-elle positivement ou négativement la performance d'un modèle d'apprentissage automatique de classification de texte ?
- Existe-il des techniques qui donnent de meilleurs résultats lorsqu'elles sont utilisées seules, que lorsqu'elles sont combinées avec d'autres techniques ?

3.2 Ensembles de données

Dans ce travail, quatre ensembles de données sont utilisés, comme le montre le Tableau 3.1. Tous les ensembles de données utilisées sont divisés en deux sous-ensembles, 80% des données sont utilisées pour l'entraînement du classificateur, tandis que 20% sont utilisées pour faire les tests.

Tableau 3.1. Ensembles de données utilisés.

Ensemble de données	Liens
JigsawToxic	https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data
Davidson	https://data.world/thomasrdavidson/hate-speech-and-offensive-language
COVID-HATE	http://claws.cc.gatech.edu/covid/#dataset
HatEaval	http://hatespeech.di.unito.it/hateval.html

3.2.1 Ensemble de données JigsawToxic

Cet ensemble de données est collecté de Wikipédia[29] par Jigsaw/Conversation AI[30] dans le cadre d'une compétition sur Kaggle [31] appelée Toxic comment classification challenge. Il contient 15 9571 commentaires en anglais classifiés manuellement. 16 225 commentaires ont été attribués à la classe toxique et les 143 346 commentaires restants sont classifiés comme non toxiques. Cet ensemble de données a été utilisé dans plusieurs recherches : (Badjatiya, Gupta, Gupta, & Varma, 2017); (Georgakopoulos, Tasoulis, Vrahatis, & Plagianakos, 2018). Le Tableau 3.2 montre la distribution des commentaires selon les catégories :

Tableau 3.2. Nombre de commentaires par catégories.

Classification toxique	Nombre de commentaires
Oui	16 225
Non	143 346
Total	**159 571**

29 https://www.wikipedia.org/

30 https://jigsaw.google.com/approach/

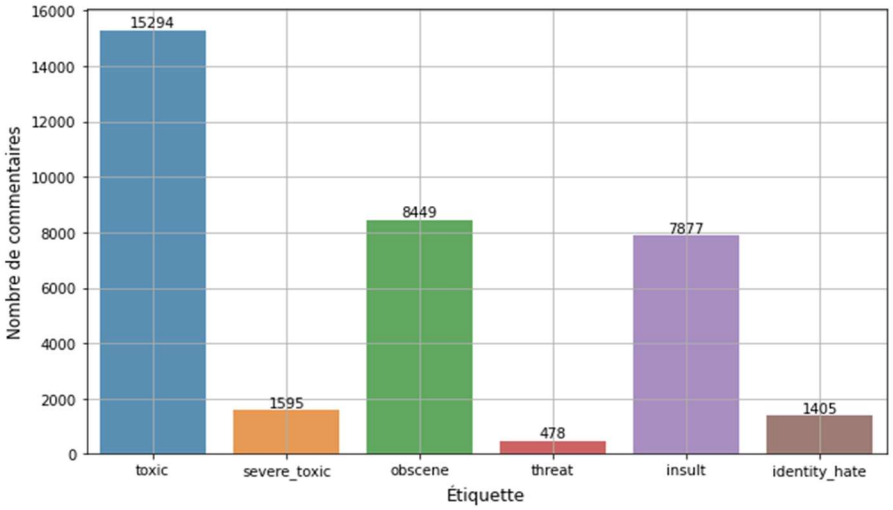
31 https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data

3.2.1.1 Exploration de l'ensemble de données

a) Nombre de commentaires par catégories

Comme le montre la Figure 3.1, les commentaires toxiques ont été classifiés en six catégories selon différents types de comportements toxiques et de discours haineux. Chaque commentaire peut avoir un label (tag) ou plus. 15294 des commentaires sont identifiés comme toxic (toxique), 1595 comme severe toxic (toxique grave), 8449 comme obscene (obscène), 478 comme threat (menace), 7877 comme insult (insulte) et 1405 comme commentaires identity hate (discrimination). La Figure 3.1 donne un aperçu sur la distribution des commentaires selon les catégories :

Figure 3.1. Nombre de commentaires par étiquette.



Le Tableau 3.3 donne un aperçu sur la distribution des commentaires en pourcentages selon les catégories.

Tableau 3.3. Nombre de commentaires par étiquette.

Classe	**Toxique**						**Non toxique**
Catégories	toxic	severe toxic	obscene	threat	insult	identity-hâte	
Nombre de commentaires	10,17%						89,83%
	10,6%	1,01%	5,591%	0,3005%	5,193%	0,8883%	

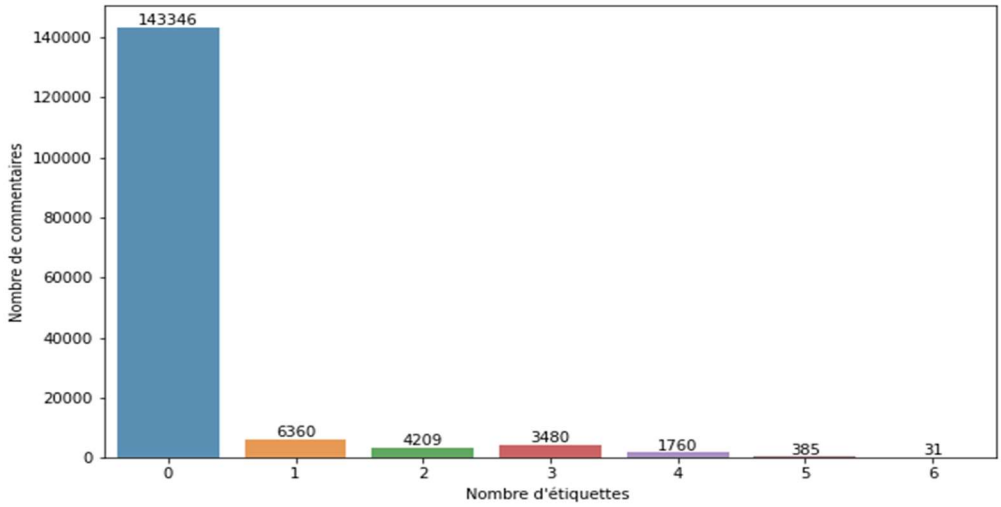
La plupart des commentaires appartiennent à la catégorie 'toxic', tandis que le nombre de commentaires appartenant aux catégories ' severely toxic', 'threats', et 'identity hate' représente un faible pourcentage par rapport au nombre total des commentaires toxiques. La somme des pourcentages des commentaires

de toutes les catégories toxiques dépasse le nombre total des commentaires toxiques qui est de 10,17%, puisque certains commentaires toxiques ont plus qu'une étiquette comme le montre la section suivante.

b) Distribution des catégories

La Figure 3.2 montre la distribution des commentaires selon le nombre d'étiquettes qu'ils ont.

Figure 3.2. Distribution des commentaires selon le nombre d'étiquettes.



La majorité des commentaires sont non toxiques, ils n'ont aucune étiquette et représentent un pourcentage de 89,83% du total des commentaires. Concernant les commentaires toxiques, chacun d'entre eux peut avoir une ou plusieurs étiquettes, comme le montre le Tableau 3.4.

Tableau 3.4. Distribution des commentaires selon le nombre d'étiquettes.

Nombre d'étiquettes	Aucune	1	2	3	4	5	6
Nombre de commentaires	143 346	6 360	4 209	3 480	1 760	385	31

Le nombre de commentaires qui ont une étiquette est 6360, ceux qui ont deux étiquettes sont 4209 commentaires, 3480 ont trois étiquettes, 1760 ont quatre étiquettes, 385 ont cinq étiquettes et 31 commentaires ont six étiquettes.

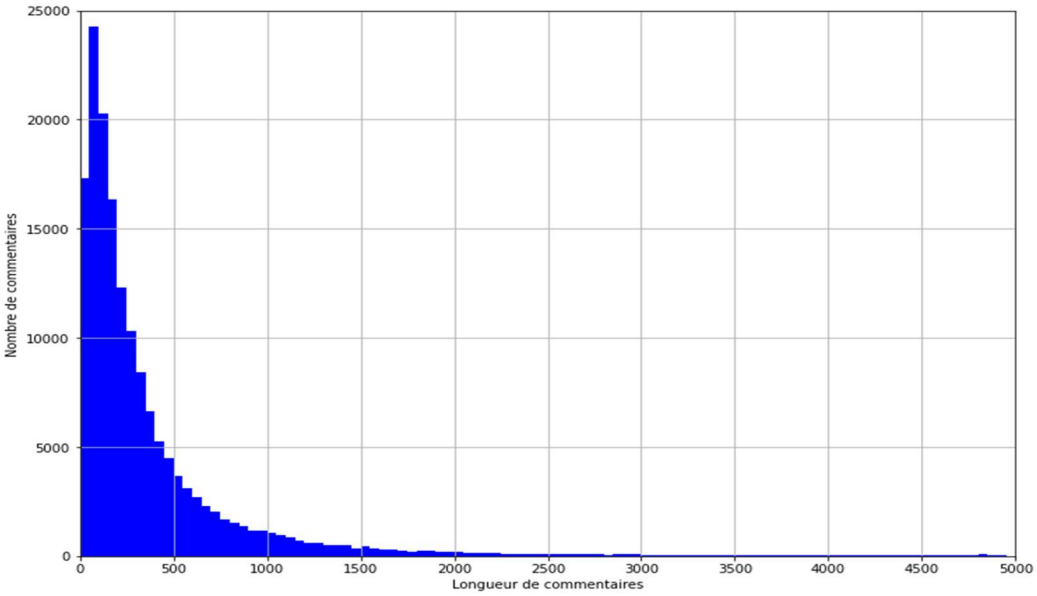
c) Détection des valeurs manquantes

L'ensemble de données ne contient aucune valeur nulle, comme le montre la Figure B.1 (voir ANNEXE B).

d) Distribution du nombre de caractères dans les commentaires

Il y a différents types de longueurs de commentaires, la majorité des commentaires ont une longueur inférieure à 500 caractères mais certains d'autres peuvent atteindre 5000 caractères. La Figure 3.3 montre la distribution du nombre de caractères dans les commentaires.

Figure 3.3. Nombre de commentaires selon la longueur.



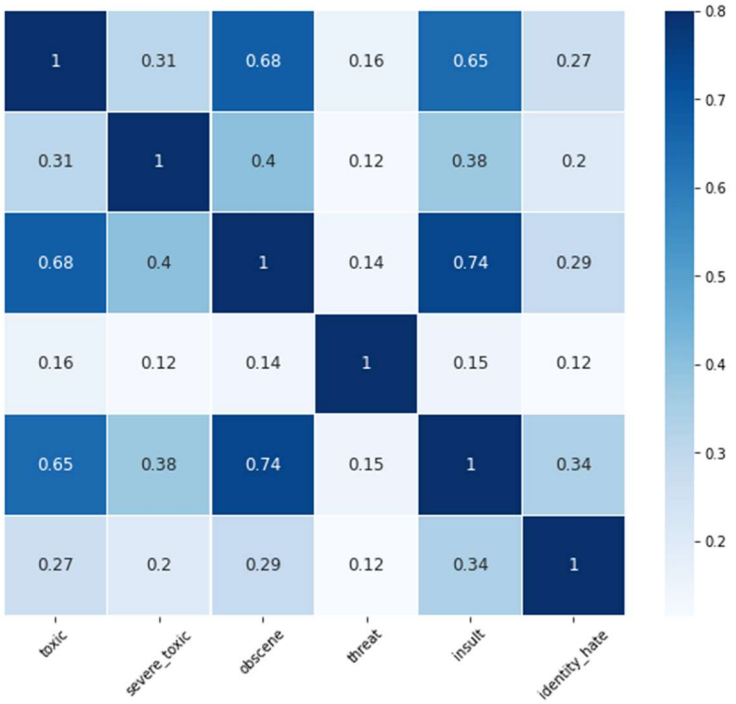
La Figure B.1 (voir ANNEXE B) montre l'exemple d'un commentaire dont la longueur est 4970 caractères et c'est le plus long commentaire dans l'ensemble de données utilisé. Les Figure B.2 et B.3 (voir ANNEXE B) présentent d'autres exemples de commentaires longs, l'un est de 4849 caractères de longueur et l'autre de 3469 caractères. Ces commentaires sont très longs parce qu'ils contiennent beaucoup de caractères spéciaux, d'où la nécessité de faire un prétraitement des données.

e) Matrice de corrélation

Une matrice de corrélation est un tableau qui montre les coefficients de corrélation entre les variables. Elle est utilisée pour tester le degré d'association entre les variables. La Figure 3.4 représente la matrice de corrélation entre les différentes catégories de la classe toxique. Le coefficient de corrélation peut avoir une valeur comprise entre -1 et +1. Plus la valeur absolue du coefficient est importante, plus la relation linéaire entre les variables est forte. Le signe du coefficient indique si les deux variables ont tendance à augmenter ou à diminuer ensemble. Si le coefficient est négatif, la première variable tend à augmenter

tandis que l'autre diminue et s'il est positif, les deux variables augmentent ensemble et diminuent ensemble.

Figure 3.4. Matrice de corrélation entre les catégories toxiques.

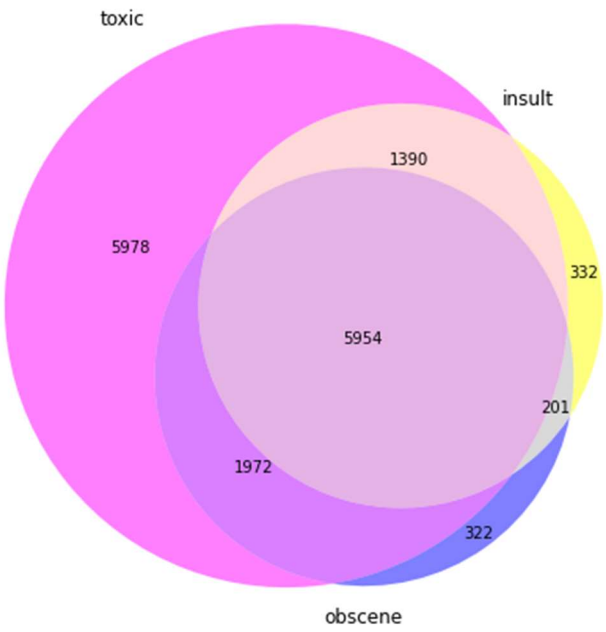


Le coefficient de corrélation est de 0.65 pour les labels 'toxic' et 'insult', de 0,74 pour les labels 'insult' et 'obscène' et de 0.68 pour les labels 'toxic' et 'obscene'. Cela signifie que ces labels sont fortement corrélés et si l'un d'entre eux augmente, les autres augmentent aussi et vis-versa.

f) Diagramme de Venn

Un diagramme de Venn représente des relations logiques entre des ensembles et des éléments de ces ensembles en se basant sur des cercles ou d'autres formes entrecroisées.

Figure 3.5. Diagramme de Venn.



Le diagramme de Venn de la Figure 3.5 montre qu'il existe une intersection entre les trois catégories 'insult', 'obscène' et 'toxic'. Tandis que le Tableau 3.5 montre les nombres de commentaires communs pour chaque couple de labels.

Tableau 3.5. Nombre de commentaires communs pour chaque étiquette.

Catégories	'toxic' & 'insult'	'obscene' & 'insult'	'toxic' & 'obscene'	'toxic' & 'obscene' & 'insult'
Nombre de commentaires communs	7 344	6 155	7 926	5 954

g) Cloud (Nuage de mots)

Un nuage de mots est une représentation visuelle de données textuelles dans laquelle l'importance des mots est visualisée au moyen de leur taille et de leur couleur. La Figure 3.6 montre les nuages de mots pour les six catégories des commentaires toxiques.

Figure 3.6. Nuages de mots pour les catégories toxiques.

Certains mots sont présents dans la plupart des catégories de commentaires toxiques.

3.2.1.2 Conversion en classification binaire

a) Problème

Tel que montré dans la section 3.1.1.1, chaque commentaire toxique est attribué au moins à l'une des six catégories : toxic (toxique), severe toxic (toxique grave), obscene (obscène), threat (menace), insult (insulte) et identity hate (discrimination). Afin de convertir ce problème de classification en classification binaire, toutes ces catégories seront combinées dans une seule catégorie toxique. Les commentaires de l'ensemble de données seront divisés en deux classes : Normal (non toxique) et toxique (appartient au moins à l'une des six classes). Les commentaires non toxiques comptent 143 346 commentaires et ceux

toxiques sont équivalents à 16225 commentaires. La distribution de données est donc fortement asymétrique et les deux classes doivent être équilibrées.

b) Échantillonnage

Un ensemble de données est dit déséquilibré, lorsque le ratio des données appartenant à une classe par rapport à l'ensemble des données est très faible ou très grand, ce qui peut influencer négativement sur les résultats de la classification et diminuer les performances du classificateur. L'échantillonnage est une technique utilisée pour résoudre ce problème. Cette stratégie permet de modifier l'ensemble de données afin de le rendre équilibré et améliorer par la suite les performances du classificateur. Il existe deux méthodes d'échantillonnage de données : Le sur-échantillonnage (Oversampling) et le sous-échantillonnage (Undersampling) (Mohammed, Rawashdeh, & Abdullah, 2020).

Le sur-échantillonnage est utilisé lorsque le ratio des données d'une classe dans l'ensemble des données est très faible par rapport aux autres classes. Cette technique consiste à augmenter la taille de la classe minoritaire afin d'équilibrer l'ensemble de données et avoir un nombre de données approximatif dans toutes les classes. Elle est utilisée lorsque l'ensemble de données contient un nombre suffisant de données. Pour augmenter la taille de la classe minoritaire, plusieurs techniques pourront être utilisées :

- Répétition : C'est une méthode qui consiste à répliquer aléatoirement des échantillons de la classe minoritaire.
- Bootstrap : C'est une méthode de rééchantillonnage qui permet de rééchantillonner de manière itérative un ensemble de données avec un remplacement aléatoire. Elle consiste à tirer à plusieurs reprises des échantillons aléatoires avec remise à partir de l'ensemble de données d'origine.
- SMOTE (Synthetic Minority Over-Sampling Technique) : C'est une technique de sur-échantillonnage qui permet de reproduire les données de la classe minoritaire pour les augmenter de façon aléatoire.

Le sous-échantillonnage est utilisé lorsque le ratio des données d'une classe dans l'ensemble des données est très grand par rapport aux autres classes. Cette technique consiste à supprimer des échantillons de la classe majoritaire afin d'avoir un nombre de données comparable dans toutes les classes. Elle est utilisée lorsque l'ensemble de données contient un nombre important de données. Les échantillons peuvent être

supprimés aléatoirement ou en utilisant l'algorithme NearMiss qui est une technique d'élimination des exemples de la classe majoritaire.

c) Échantillonnage de l'ensemble de données

Le nombre de données de la classe non toxique est plus grand que celui de la classe toxique. Pour résoudre ce problème, nous avons effectué un sous-échantillonnage aléatoire de la classe non toxique permettant ainsi d'obtenir un seuil acceptable d'équilibre. Nous avons réduit le nombre des données en sélectionnant au hasard des données parmi les données disponibles dans la classe toxique. Nous avons obtenu par la suite pour cette classe, un nombre de données comparable au nombre de données de la classe non toxique. Le Tableau 3.6 montre la nouvelle distribution des commentaires dans les deux classes.

Tableau 3.6. Nombre de commentaires par classe binaire.

Classe	Nombre de commentaires
Toxic	16 225
Normal	16 225
Total	**32 450**

Après l'échantillonnage, le nombre des commentaires toxiques est le même que le nombre des commentaires non toxiques. Il en résulte un ensemble de données équilibré.

3.2.2 Ensemble de données Davidson

L'ensemble de données Davidson (Davidson, Warmsley, Macy, & Weber, 2017) est un ensemble de données qui est accessible au public. Il contient des mots et des phrases identifiés par les internautes comme discours de haine et compilés par Hatebase.org[32] à l'aide de l'API Twitter. (Davidson, Warmsley, Macy, & Weber, 2017) ont exploré 84,4 millions de tweets de 33 458 utilisateurs de Twitter[33]. Par la suite, 25K tweets ont été échantillonnés aléatoirement et étiquetés manuellement par les utilisateurs de la

32 https://hatebase.org/

33 https://developer.twitter.com/en/docs/twitter-api

plateforme CrowdFlower (CF)[34]. Chaque tweet a été codé par trois annotateurs ou plus et classé dans l'une des catégories suivantes : "Haineux", "Offensant" ou "Aucun". Les tweets dont les accords des annotateurs sont faibles ont été éliminés. Il en résulte par la suite, 24 783 tweets dont 19190 tweets sont haineux, 1430 tweets sont offensants et 4163 tweets sont normaux. Le Tableau 3.7 montre une description de la distribution des classes :

Tableau 3.7. Nombre total des commentaires par catégories.

Discours haineux et offensants	**Nombre de commentaires**
Oui	20 620
Non	4 163
Total	24 783

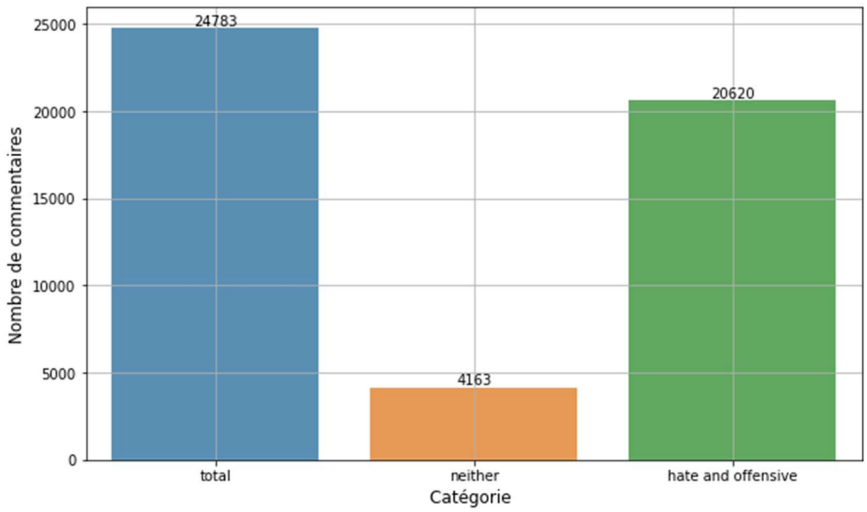
3.2.2.1 Exploration de l'ensemble de données

a) Nombre de commentaires par catégorie

Les commentaires ont été classifiés en trois catégories. Chaque commentaire peut appartenir à une seule catégorie. 19190 des commentaires sont identifiés comme langage offensant (offensive_language) et 1430 comme discours haineux (hate_speech). 4163 des commentaires sont neutres. La Figure 3.7 donne un aperçu de la distribution des commentaires selon les catégories :

34 https://visit.figure-eight.com/People-Powered-Data-Enrichment_T

Figure 3.7. Nombre des commentaires de chaque catégorie.



Le Tableau 3.8 donne un aperçu de la distribution des commentaires selon les catégories en pourcentages.

Tableau 3.8. Distribution des commentaires en pourcentages.

Classe	Haineux et offensant		**Normaux**
Catégories	hate_speech	offensive_language	
Nombre de commentaires	83,2%		16,8%
	5,77%	77,43%	

La plupart des commentaires appartiennent à la catégorie ‘offensive_language’, tandis que le nombre de commentaires appartenant à la catégorie 'hate_speech' représente un faible pourcentage par rapport au nombre total des commentaires agressifs.

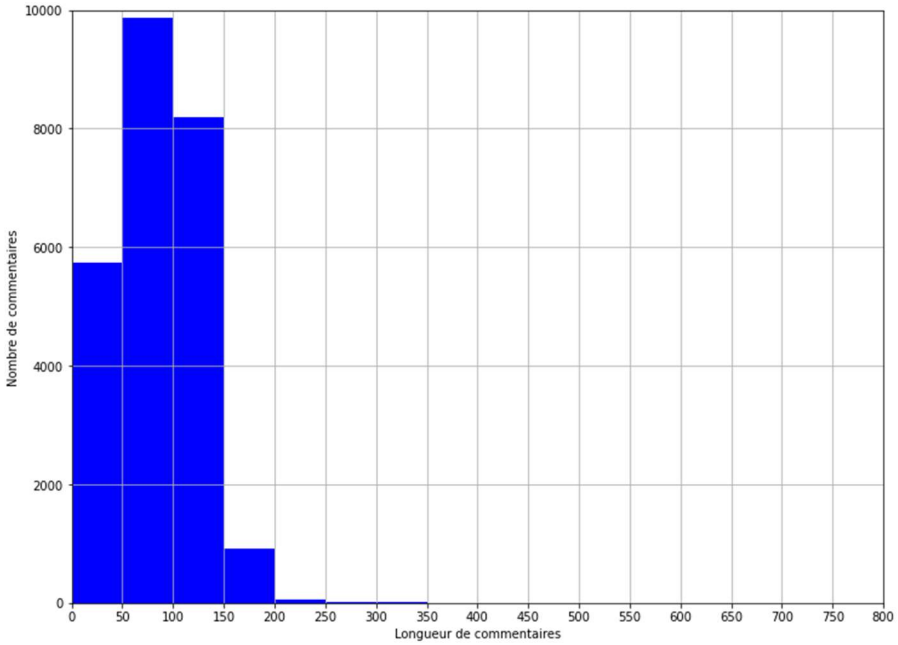
b) Détection des valeurs manquantes

L’ensemble de données ne contient aucune valeur nulle, comme le montre la Figure B.2 (voir ANNEXE B).

c) Distribution du nombre de caractères dans les commentaires

Il y a différents types de longueurs de commentaires. La majorité des commentaires ont une longueur inférieure à 150 caractères mais certains d’autres peuvent atteindre 350 caractères. La Figure 3.8 montre la distribution du nombre de caractères dans les commentaires.

Figure 3.8. Nombre de commentaires selon la longueur.



La Figure A.4 (voir ANNEXE A) présente un commentaire dont la longueur est 321 caractères et qui est le plus long dans l'ensemble de données. Les figures A.5 et A.6 représentent respectivement deux autres commentaires de longueur 281 et 267 caractères. Ces commentaires sont longs parce qu'ils contiennent des caractères spéciaux et aussi des chiffres, d'où la nécessité de faire un prétraitement des données.

d) Cloud (Nuage de mots)

La Figure 3.9 représente les nuages de mots pour les deux catégories des commentaires agressifs.

Figure 3.9. Nuages de mots pour les catégories toxiques.

Hate-speech

Offensive langage

Certains mots sont présents dans les deux catégories de commentaires.

3.2.2.2 Conversion en classification binaire

a) Problème

Tel que montré dans la section 3.1.2.1, chaque commentaire toxique peut être attribué à l'une des deux catégories : langage offensant (offensive_language) ou discours haineux (hate_speech). Afin de convertir le problème de classification en classification binaire, les deux catégories seront combinées en une seule catégorie toxique. Les commentaires seront divisés en deux classes : Normal (non toxique) qui contient 4136 commentaires et toxique (appartient à l'une des deux classes) qui contient 20 620 commentaires. Cela montre que la distribution de données est fortement asymétrique et que les deux classes doivent être équilibrées.

b) Échantillonnage de l'ensemble de données

Le nombre de données de la classe toxique est plus grand que celui de la classe non toxique. Comme expliqué dans la section 3.2.1.2 et pour résoudre ce problème, nous avons procédé à la fois, à un sur-échantillonnage aléatoire de la classe non toxique et un sous-échantillonnage aléatoire de la classe toxique. Nous avons réduit le nombre des données de la classe majoritaire en sélectionnant au hasard des données parmi les données disponibles dans cette classe. Par la suite, nous avons augmenté le nombre des données de la classe non toxique en faisant des copies des données choisis au hasard de cette classe plusieurs fois. Cette duplication des données, nous a permis d'obtenir un nombre de données de la classe non toxique comparable au nombre de données de la classe toxique et avoir un seuil acceptable d'équilibre par la suite. Le Tableau 3.9 montre la nouvelle distribution des commentaires dans les deux classes.

Tableau 3.9. Nombre de commentaires par classe binaire.

Classe	**Nombre de commentaires**
Toxic	16 500
Normal	16 652
Total	**33 152**

Après l'échantillonnage, le nombre des commentaires toxiques et celui des commentaires normaux sont comparables. Il en résulte un ensemble de données équilibré.

3.2.3 Ensemble de données COVID-HATE

L'ensemble de données COVID-HATE (He, et al., 2020) est un ensemble de données accessible au public et qui est formé de tweets haineux déclenchés par la propagation de la pandémie de Covid-19 et ciblant les communautés asiatiques. He Bing et al. (He, et al., 2020) ont exploré 206 millions de tweets contenant 127 millions de nœuds et 910 millions d'arêtes collectés en temps réel à l'aide de l'API de recherche de Twitter et l'API de diffusion de Twitter. Ils ont annoté manuellement les tweets par deux annotateurs pour les séparer en trois catégories : Hatespeech 'discours de haine', Counterhate 'contre-haine' et Neutral 'neutre'. Seuls les tweets étant sujets d'un accord entre les annotateurs ont été gardés, quant aux autres, ils ont été supprimés. Il en résulte 2290 tweets au total, dont 1344 tweets sont neutres, 429 tweets sont des discours de haine et 517 sont des tweets contre-haine. Le Tableau 3.10 montre une description de la distribution des classes :

Tableau 3.10. Nombre total des commentaires par catégories.

Catégorie	hate_speech	Counterhate	Neutral
Nombre de commentaires	429	517	1 344
Total	2 290		

3.2.3.1 Exploration de l'ensemble de données

a) Nombre de commentaires par catégorie

Les commentaires ont été classifiés en trois catégories comme le montre le Tableau 3.10. Chaque commentaire peut appartenir à une seule catégorie. 429 des commentaires sont identifiés comme discours de haine (hatespeech) et 517 comme contre-haine (counterhate). 1344 des commentaires sont neutres. La table ci-dessous donne un aperçu sur la distribution des commentaires selon les catégories en pourcentages.

Tableau 3.11. Nombre de commentaires par catégorie.

Classes	**Discours de haine et contre-haine**		**Neutre**
Catégories	***hate_speech***	***counterhate***	
#commentaires	41,31%		58,69%
	18,73%	22,58%	

La plupart des commentaires appartiennent à la catégorie ‘neutral’, tandis que le nombre de commentaires appartenant à la catégorie 'hate_speech' représente un pourcentage faible par rapport au nombre total des commentaires.

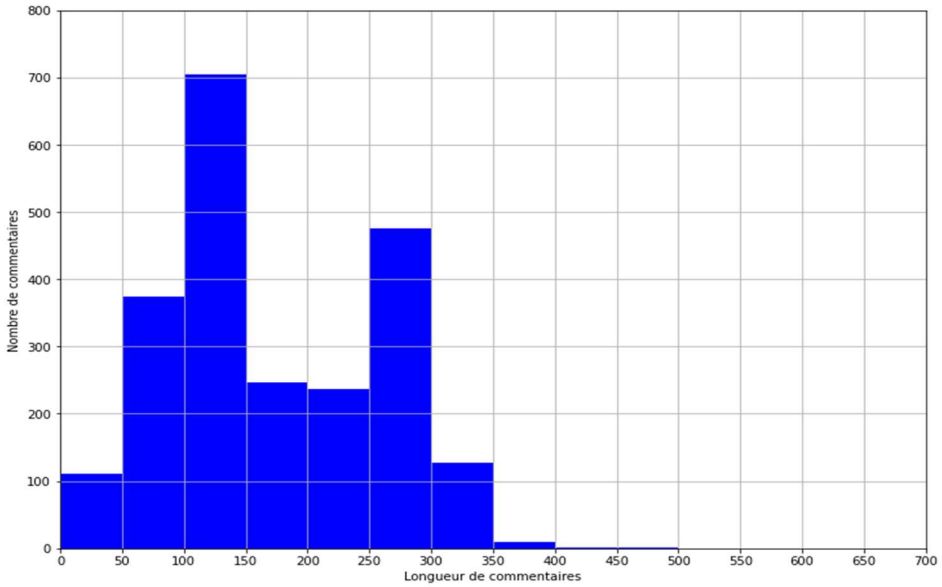
b) Détection des valeurs manquantes

L’ensemble de données ne contient aucune valeur nulle, comme le montre la Figure B.3 (voir ANNEXE B).

c) Distribution du nombre de caractères dans les commentaires

Il y a différents types de longueurs de commentaires, La majorité de celui-ci ont une longueur comprise entre 50 et 300 caractères mais certains d’autres peuvent dépasser 350 caractères. La Figure 3.10 montre la distribution du nombre de caractères dans les commentaires.

Figure 3.10. Nombre de commentaires selon la longueur.



La Figure A.7 (voir ANNEXE A) présente un commentaire dont la longueur est 151 caractères et qui est le plus long dans l'ensemble de données. Les figures A.8 et A.9 représentent respectivement deux autres commentaires de longueur 138 et 130 caractères. Ces commentaires sont longs parce qu'ils contiennent des noms d'utilisateurs, d'où la nécessité de faire un prétraitement des données.

d) Cloud (Nuage de mots)

La Figure 3.11 représente les nuages de mots pour les deux catégories des commentaires haineux.

Figure 3.11. Nuages de mots pour les catégories toxiques.

Hate-speech

Counter-hate

3.2.3.2 Conversion en classification binaire

a) Problème

Tel que montré dans la section 3.1.3.1, chaque commentaire peut être attribué à l'une des deux catégories : discours haineux (hatespeech) et contre-discours (counterhate). Afin de convertir le problème de classification en classification binaire, seules les deux catégories 'neutral' et 'hatespeech' seront gardées. Les commentaires seront divisés en deux classes : Neutral (neutre) qui contient 1344 commentaires et hatespeech (discours haineux) qui contient 429 commentaires. Cela montre que la distribution de données est fortement asymétrique et que les deux classes doivent être équilibrées.

b) Échantillonnage de l'ensemble de données

Le nombre de données de la classe 'neutral' est plus grand que celui de la classe 'hatespeech'. Comme expliqué dans la section 3.2.1.2, un sur-échantillonnage aléatoire de la classe 'hatespeech' est nécessaire pour résoudre ce problème. Pour ce faire, nous avons augmenté le nombre des données de cette classe en faisant des copies des données de la même classe plusieurs fois. La duplication des échantillons qui ont été choisis au hasard, nous a permis d'obtenir un seuil acceptable d'équilibre. Nous avons obtenu un

nombre de données de la classe hatespeech' comparable au nombre de données de la classe 'neutral'. Le Tableau 3.12 montre la nouvelle distribution des commentaires dans les deux classes.

Tableau 3.12. Nombre de commentaires par classe binaire.

Classe	Nombre de commentaires
Hatespeech	1287
Neutre	1 344
Total	**2 631**

Après l'échantillonnage, le nombre des commentaires haineux et celui des commentaires neutres sont approximativement identiques. Il en résulte un ensemble de données équilibrées.

3.2.4 Ensemble de données HatEval

L'ensemble de données HatEval (Basile, et al., 2019) est un ensemble de données qui est accessible au public. Il contient des mots et des phrases identifiés par les internautes comme discours de haine ciblant des groupes minoritaires (immigrants et femmes). Ces données ont été collectées sur Twitter et annotées par des contributeurs non formés sur la plateforme de crowdsourcing Figure Eight (F8)[35]. Selon l'office québécois de la langue française[36], le crowdsourcing appelé aussi approvisionnement par la foule ou externalisation ouverte est la pratique consistant à exploiter l'intelligence, les compétences et le savoir-faire des consommateurs ou des internautes dispersés dans le monde entier. Les contributeurs ont annoté les tweets comme agressifs 'AG' lorsqu'ils contiennent des discours de haine qui ciblent des groupes génériques ou une personne spécifique dont des femmes et des immigrants. L'ensemble de données est composé de trois sous-ensembles (d'entraînement, de développement et de test) que nous avons fusionnés pour obtenir un seul ensemble de données contenant 13000 tweets au total dont 7530 tweets neutres et 5470 tweets haineux. Le Tableau 3.13 montre une description de la distribution des classes :

35 http://www.figure-eight.com/

36 https://www.oqlf.gouv.qc.ca/office/mission.html

Tableau 3.13. Nombre total des commentaires par catégories.

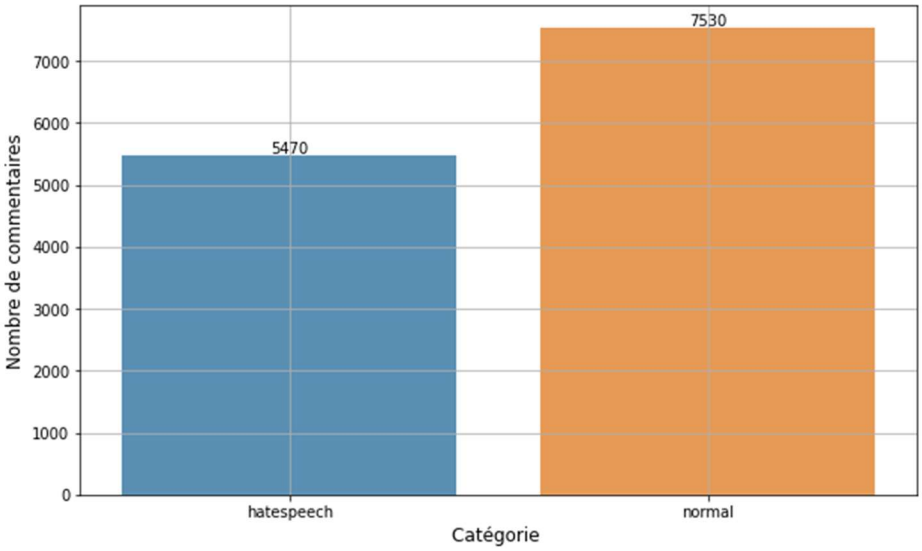
Discours haineux et agressifs	Nombre de commentaires
Oui	5 470
Non	7 530
Total	**13 000**

3.2.4.1 Exploration de l'ensemble de données

a) Nombre de commentaires par catégorie

Comme le montre la Figure 3.12, les commentaires ont été classifiés en deux catégories. Chaque commentaire peut appartenir à une seule catégorie. 5470 des commentaires sont identifiés comme discours haineux et agressifs (hatespeech) et 7530 comme neutre (neutral).

Figure3.12. Nombre de commentaires par catégorie.



La table 3.14 donne un aperçu de la distribution des commentaires selon les catégories en pourcentages.

Tableau 3.14. Nombre de commentaires par catégorie.

Classe	Haineux et agressifs	Normaux
Nombre de #commentaires	42,08%	57,92%

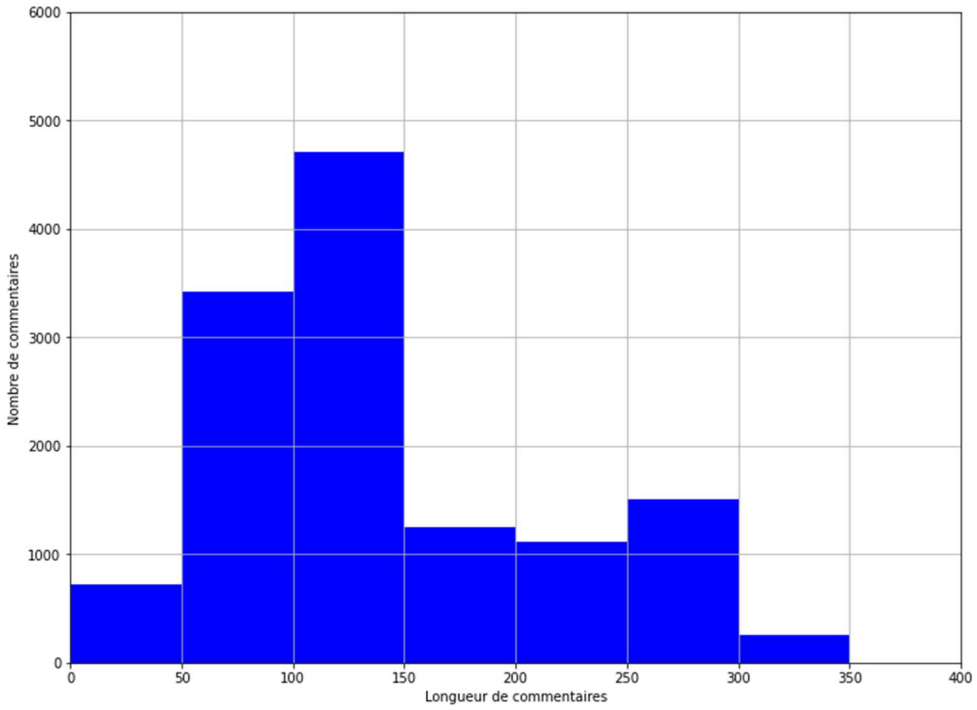
b) Détection des valeurs manquantes

L'ensemble de données ne contient aucune valeur nulle, comme le montre la Figure B.4 (voir ANNEXE B).

c) Distribution du nombre de caractères dans les commentaires

Il y a différents types de longueurs de commentaires, la majorité des commentaires ont une longueur inférieure à 150 caractères mais certains d'autres peuvent atteindre 350 caractères. La Figure 3.13 montre la distribution du nombre de caractères dans les commentaires.

Figure 3.13. Nombre de commentaires selon la longueur.



La Figure A.10 (voir ANNEXE A) présente un commentaire dont la longueur est 139 caractères et qui est le plus long dans l'ensemble de données. Les figures A.11 et A.12 représentent respectivement deux autres commentaires de longueur 136 et 130 caractères. Ces commentaires sont longs parce qu'ils contiennent des caractères spéciaux et des noms d'utilisateurs, d'où la nécessité de faire un prétraitement des données.

d) Cloud (Nuage de mots)

Un nuage de mots est une représentation visuelle de données textuelles dans laquelle l'importance des mots est visualisée au moyen de leur taille et de leur couleur. La Figure 3.14 représente les nuages de mots pour la classe 'hatespeech' contenant les discours haineux et agressifs.

Figure 3.14. Nuages de mots pour les catégories toxiques.

3.2.4.2 Conversion en classification binaire

Il s'agit ici d'un problème de classification binaire puisque l'ensemble de données contient juste deux classes. Les nombres de données dans les deux classes sont équivalents, 7530 données normales et 5470 discours de haine, cela signifie que cet ensemble de données n'est pas déséquilibré et il ne va pas subir par la suite un échantillonnage.

3.2.5 Tableau récapitulatif

Le Tableau 3.15 donne un aperçu global du nombre de commentaires des ensembles de données utilisés.

Tableau 3.15. Tableau récapitulatif des nombres de commentaires dans les ensembles de données.

Ensembles de données	Nombre de commentaires		
	Classe toxique	Classe non toxique	Total
JigsawToxic	16 225	16 225	**32 450**
Davidson	16 500	16 652	**33 152**
COVID-HATE	1287	1 344	**2 631**
HatEval	5 470	7 530	**13 000**

3.3 Étapes de processus de classification de texte

La classification de texte consiste à étiqueter un texte non classifié et spécifier sa catégorie en respectant certains critères et règles (Ma, Li, Wu, & Zhang, 2018). Ce processus passe par plusieurs étapes : Le prétraitement et la préparation des données, la représentation des données, l'entrainement du modèle de classification et la classification des données non étiquetées (Aboalnaser, 2019). Ces étapes peuvent être automatisées en utilisant un pipeline d'apprentissage automatique qui permet de rassembler une séquence d'étapes pour entraîner un modèle. C'est ainsi le cas dans plusieurs recherches comme la détection du langage offensant (Hajibabaee, et al., 2022), l'analyse de texte (Patel, Patole, & Metkar, 2022) et la classification d'émotions (Ahmed, Mukta, Al Mahmud, Hasan, & Gulzar Hussain, 2022).

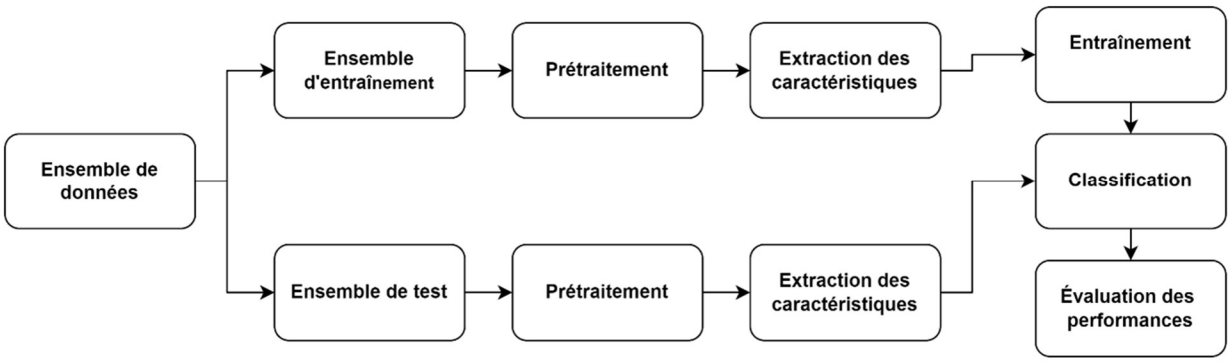
La classification est faite à l'aide du langage Python[37] qui est un langage de programmation favorisant la programmation orientée-objet. C'est un langage multiplateforme qui peut être utilisée avec Linux, Windows, MacOS, etc. Python est le choix le plus populaire par rapport à d'autres, grâce à sa lisibilité et sa facilité d'écriture (Ahmed, Kinjol, & Ananya, 2021). Il dispose de plusieurs bibliothèques permettant d'effectuer de nombreuses tâches comme l'apprentissage automatique, l'exploration de données, le calcul scientifique, le graphisme, etc.

L'apprentissage automatique est une branche de l'intelligence artificielle. C'est la science permettant de donner aux machines la capacité de l'auto-apprentissage et l'auto-évaluation afin de simuler

37 https://www.python.org/

l'apprentissage humain (Lv & Tang, 2011). Pour effectuer cette tâche, nous avons utilisé la bibliothèque NLTK [38] (Natural Language Toolkit) qui fournit des ressources lexicales et permet aussi de faire la tokenisation et la racinisation (stemming). Nous avons aussi utilisé la bibliothèque Sklearn[39] (scikit-learn) qui fournit des outils simples et efficaces pour l'analyse prédictive des données. Elle fournit plusieurs algorithmes d'apprentissage automatique, dont SVM (Support Vector Machines) qui est utilisé dans ce travail. Nous avons aussi utilisé la bibliothèque Keras qui permet d'implémenter les architectures neuronales. La Figure 3.15 montre les différentes étapes du processus de classification de texte :

Figure 3.15.Processus de classification de texte.



3.3.1 Prétraitement

La phase de prétraitement de données textuelles est une phase importante dans le processus de classification de texte. Elle consiste à éliminer le bruit, nettoyer et normaliser les données brutes afin de les rendre exploitable par l'algorithme de classification. Elle permet de supprimer tout ce qui est inutile dans le texte et qui n'apporte aucune information pertinente pour la classification de texte, afin d'optimiser les performances du classificateur et obtenir donc de meilleurs résultats. Selon Thomas Heitz (Heitz, 2006), le prétraitement de texte consiste à normaliser les données, à corriger les erreurs et les valeurs manquantes.

38 https://www.nltk.org/

39 https://scikit-learn.org/stable/

> *Les prétraitements des données textuelles consistent à normaliser les diverses manières d'écrire un même mot, à corriger les fautes d'orthographe évidentes ou les incohérences typographiques et à expliciter certaines informations lexicales exprimées implicitement dans les textes* (Heitz, 2006, p.499-500).

Selon Kumar et Harish (Keerthi Kumar & Harish, 2018), le nettoyage des données consiste à supprimer le maximum du bruit dans le texte. Ils ont considéré comme bruit : les URL, les noms d'utilisateurs, les hashtags, les signes de ponctuation et les mots vides. Ils ont aussi corrigé les négations abrégées et les mots étendus et ramené les mots à leurs racines en supprimant les suffixes les plus utilisés (stemming). Kowsher et al. (Kowsher, Tahabilder, Hossain Sarker, Islam Sanjid, & Prottasha, 2020) ont utilisé la lemmatisation comme technique de prétraitement pour du texte bengali. Elle consiste à ramener les termes à leur forme canonique significative en considérant le contexte. Dans la phase de prétraitement, Lubis et al (Lubis, Nasution, Sitompul, & Zamzami, 2022) ont supprimé les signes de ponctuations, les émoticônes et les mots considérés comme non importants. Ils ont aussi normalisé les tweets en les convertissant en lettres minuscules.

Dans ce travail, nous expérimentons plusieurs approches. Chacune d'entre elles englobe une ou plusieurs des techniques de prétraitement connues comme :

- Conversion des mots en minuscule ;
- Suppression des mots vides (stop words) , des mots fréquents et des mots rares ;
- Suppression des nombres et des signes non-alphabétiques ;
- Suppression des mentions utilisateurs, URLs et balises HTML ;
- Racinisation (stemming) et lemmatisation (lemmatization) ;
- Vérification orthographique ;
- Tokenisation ;

Chacune des approches expérimentées représente un pipeline contenant des blocs de prétraitement. Chaque bloc représente une ou plusieurs techniques de prétraitement. Ce qui diffère d'une approche à l'autre est parfois la nature des techniques utilisées, d'autres fois l'ordre dans lequel ces techniques sont utilisées. Autrement dit, les techniques sont fragmentées en diverses approches pour vérifier l'influence du choix et l'agencement de ces techniques sur la classification et identifier les facteurs qui doivent être vérifiés et respectés pour améliorer les performances du classificateur. Nous avons regroupé ces approches, selon leurs différentes variantes qui se ressemblent, en trois scénarios différents. Le scénario 1 correspond à l'approche 1, où le classificateur reçoit en entrée le texte brut après tokenisation. Le scénario 2 dans lequel nous avons regroupé les approches permettant de vérifier l'impact de la suppression du bruit sur les résultats de la classification, comme la suppression des mots vides, des

nombres, des caractères spéciaux, des mots rares et des mots fréquents, etc. Le scénario 3 regroupe les approches qui se focalise sur la correction orthographique et la normalisation du texte en utilisant la lemmatisation et la racinisation.

3.3.2 Extraction des caractéristiques

Après la phase de prétraitement, nous procédons à l'extraction d'attributs, appelée aussi sélection des caractéristiques. C'est un processus permettant de réduire le nombre de variables d'entrée en conservant les caractéristiques pertinentes et en supprimant celles non pertinentes. Pour cela, nous utilisons les méthodes du traitement du langage naturel pour vectoriser le texte nettoyé et le convertir en caractéristiques numériques. Ceux sont des vecteurs de caractéristiques qui seront utilisés pour entraîner le classificateur. Nous utilisons le modèle TfidfVectorizer de la bibliothèque Scikit-learn et le modèle Tokenizer de la bibliothèque Keras dans la vectorisation du texte.

Le modèle TfidfVectorizer permet de convertir le texte nettoyé en caractéristiques numériques. C'est une méthode utilisée dans le traitement du langage naturel pour vectoriser le texte (Kumar & Subba, 2020). Elle permet à l'aide du modèle N-grammes de représenter le texte sous forme de collections de jetons. Ces collections sont des séquences continues de n jetons dans un document où chaque séquence continue représente une caractéristique (Keerthi Kumar & Harish, 2018). Dans un unigramme (n=1), un mot représente une caractéristique, tandis que dans un bigramme (n=2), une caractéristique est représentée par deux mots et dans un trigramme (n=3), ce sont trois mots qui représentent une caractéristique. Cette méthode permet aussi de calculer le poids TF-IDF (term frequency–inverse document frequency) qui est utilisé pour évaluer l'importance d'un mot pour un document dans un corpus (Kumar & Subba, 2020). Cette importance augmente proportionnellement au nombre de fois où un mot apparaît dans le document, mais elle est compensée par la fréquence du mot dans le corpus. Cette méthode renvoie un tableau de vecteurs de caractéristiques pour chaque texte d'entrée. Ces vecteurs de caractéristiques seront utilisés par la suite pour entraîner les classificateurs SVM, LR et NB.

Le modèle Tokenizer est aussi une méthode permettant de vectoriser le texte en le transformant en une séquence numérique adaptée à une couche Keras Embedding d'un réseau neuronal. Ce modèle est utilisé avec les architectures d'apprentissage profond et permet de construire un vocabulaire en créant un mappage entre chaque mot unique du texte et son index unique qui est un entier. Par la suite, le texte est converti en une séquence d'entiers en remplaçant chaque mot par son index dans le vocabulaire. La

dernière étape consiste à transformer les longueurs des séquences obtenues en une longueur fixe en les complétant, ou en les tronquant, afin qu'elles aient toutes la même longueur. Nous obtenons ainsi des vecteurs de même longueur utilisés pour entraîner le Bi-LSTM.

3.3.3 Entraînement

Durant cette étape, nous allons entraîner les algorithmes SVM (Support Vector Machine), LR (Logistic Regression), NB (Naive Bayes) et Bi-LSTM (Bidirectional Long Short-Term Memory) sur les ensembles de données prétraitées. Ils vont prendre en entrée le texte après vectorisation pour obtenir par la suite, pour chaque approche, un modèle entraîné qui sera utilisé dans la prédiction des données des ensembles de test. Basée sur la littérature, SVM est parmi les méthodes d'apprentissage automatique qui permet d'obtenir de meilleures performances dans la classification de texte (Liu, Lv, Liu, & Shi, 2010). Il est largement utilisé par les chercheurs dans les tâches d'analyse de sentiments (Satrya, Pratiwi, Fa'rifah, & Abawajy, 2022) et (Cahyanti, Adiwijaya, & Faraby, 2020) et dans la classification de texte (Hasan, et al., 2022). Selon l'article de (Jahan & Oussalah, 2021), les algorithmes d'apprentissage automatique supervisé, tels que SVM (Support Vector Machine), LR (Logistic Regression) et NB (Naive Bayes) sont les plus utilisés dans la détection des discours haineux. CNN (Convolutional Neural Networks), LSTM (Long Short-Term Memory) et Bi-LSTM (Bidirectional Long Short-Term Memory) sont aussi les architectures neuronales les plus utilisées dans le même contexte.

Dans ce travail, nous utilisons SVM, NB, LR et Bi-LSTM pour un problème de classification binaire. Les commentaires ou les tweets sont classifiés en deux classes : toxique (offensif ou haineux) et neutre (non toxique).

3.3.4 Classification

Nous avons utilisé les modèles obtenus durant l'étape précédente pour prédire les données des ensembles de test. Cette étape est importante pour mesurer et évaluer les performances des modèles. Pour ce faire, nous avons utilisé les métriques suivantes : précision, rappel, accuracy et F1-mesure (Ahmed, Mukta, Al Mahmud, Hasan, & Gulzar Hussain, 2022), (Satrya, Pratiwi, Fa'rifah, & Abawajy, 2022). Ces métriques sont des indicateurs de performance et peuvent être calculées à partir de la matrice de confusion représenté dans le Tableau 3.16 (Satrya, Pratiwi, Fa'rifah, & Abawajy, 2022). Cette dernière permet d'évaluer comment un modèle a fait ses prédictions, autrement dit, elle met en valeur les prédictions correctes et celles incorrectes et elle composée de quatre valeurs : TP, TN, FP et FN.

Tableau 3.16. Matrice de confusion.

		Classes réelles	
		Positive (Normal)	Négative (Toxique)
Classes prédites	Positive (Normal)	TP	FP
	Négative (Toxique)	FN	TN

- TP : Vrai positif, c'est le nombre de commentaires positifs classés dans la classe positive.
- TN : Vrai négatif, c'est le nombre de commentaires négatifs classés dans la classe négative.
- FP : Faux positif, c'est le nombre de commentaires négatifs classés dans la classe positive.
- FN : Faux négatif, c'est le nombre de commentaires positifs classés dans la classe négative.

L'accuracy est le ratio entre le nombre de valeurs correctement prédites et le total des prédictions.

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

La précision est le ratio entre le nombre de valeurs positives correctement prédites et le total des prédictions positives.

$$Précision = \frac{TP}{TP + FP}$$

Le rappel (Recall) est le ratio entre le nombre de valeurs positives correctement prédites et le total des valeurs positives.

$$Rappel = \frac{TP}{TP + FN}$$

F1-mesure (F1-score) dépend à la fois de la précision et du rappel, c'est la moyenne harmonique des deux.

$$F1 - mesure = 2 * \frac{Précision * Rappel}{Précision + Rappel}$$

Une valeur élevée de F1-mesure signifie que la valeur des faux positifs et celle des faux négatifs sont faibles.

Après avoir calculé les valeurs des métriques de performance pour chaque modèle, nous allons comparer les différents résultats obtenus pour évaluer les performances du modèle avec chaque approche expérimentée. Ainsi, nous pourrons définir les combinaisons des techniques de traitement qui donne de meilleurs résultats, ainsi que les algorithmes utilisés.

3.4 Démarches suivies

Ce travail est divisé en deux principales étapes : La réalisation et la discussion des résultats. Durant la première étape, celle de la réalisation, plusieurs approches seront expérimentées avec quatre ensembles de données différents. Dans chaque approche, une ou plusieurs techniques de prétraitement seront utilisées avec les algorithmes d'apprentissage automatique supervisé SVM, LR et NB et avec l'architecture neuronale Bi-LSTM. En premier lieu, nous commençons par une explication de toutes les techniques utilisées ainsi que de toutes les approches expérimentées. Par la suite, nous présentons les différents résultats obtenus avec chaque approche.

La deuxième étape consiste à faire, pour chaque ensemble de données, une comparaison entre les différents résultats obtenus en termes de précision, d'accuracy, de rappel et de F1-mesure. Cette comparaison nous permettra de détecter les techniques de prétraitement qui améliorent les performances du système et celles qui les dégradent. Elle nous permettra aussi de connaitre les techniques de prétraitement qui donnent les meilleurs résultats lorsque utilisées seules, ainsi que les meilleures combinaisons possibles.

3.5 Conclusion

Dans ce chapitre, nous avons donné un aperçu sur la méthodologie adoptée dans ce travail ainsi que les hypothèses à vérifier. Nous avons présenté les quatre ensembles de données choisis. Il s'agit de : JigsawToxic, Davidson, COVID-HATE et HatEval. Nous avons expliqué les différentes étapes du processus de classification de texte, ainsi que la démarche suivie. Nous avons décrit le prétraitement des données, l'extraction des caractéristiques, l'entraînement du classificateur et la classification des données de test et les différentes métriques utilisées pour comparer les résultats des approches adoptées durant la phase du prétraitement. Dans le chapitre suivant, nous donnons d'abord un aperçu de l'environnement de travail. Par la suite, nous présentons les approches de prétraitement expérimentées. Et enfin nous présentons les algorithmes d'apprentissage automatique utilisés.

CHAPITRE 4
RÉALISATION

Dans ce chapitre, nous nous concentrons sur la partie réalisation du système. Nous présentons l'environnement de travail ainsi que la méthode de travail utilisés. Par la suite, nous donnons un aperçu sur les différentes techniques de prétraitement utilisées ainsi que les méthodes adoptées dans la phase d'entraînement et de classification.

4.1 Environnement de travail

4.1.1 Environnement logiciel

Nous avons utilisé le système d'exploitation Linux avec lequel nous avons installé Kali[40] Linux au sein de la machine virtuelle Oracle VM VirtualBox[41] 6.1. Kali Linux est une distribution Linux open source basée sur Debian et spécialisée dans la cybersécurité. Elle permet d'effectuer plusieurs tâches de sécurité de l'information, telles que les tests d'intrusion et l'ingénierie inverse. Nous avons aussi utilisé le langage Python version2.7, ainsi que l'application Jupyter Notebook[42] 6.1.3 pour créer et modifier le code.

4.1.2 Bibliothèques utilisées

Nous avons utilisé plusieurs bibliothèques dans ce travail. Le Tableau 4.1 donne un aperçu des bibliothèques utilisées.

40 https://www.kali.org/

41 https://www.virtualbox.org/

42 https://jupyter.org/

Tableau 4.1. Bibliothèques utilisées.

Bibliothèques	**Tâche effectuée**
Bibliothèque standard	re : fournir des opérations de correspondance d'expressions régulières
	pickle : sérialiser et désérialiser une structure d'objet Python
	time : fournir de nombreuses fonctions liées au temps
Pandas	Analyser et manipuler les données
Numpy	Faire le calcul scientifique
Sklearn	Model_selection : diviser l'ensemble de données en deux sous-ensembles, appliquer la validation croisée et la technique de recherche en grille
	SVM : appliquer l'algorithme Support Vector Machine
	Linear_model : appliquer l'algorithme Logistic Regression
	Naive_bayes : appliquer l'algorithme Naive Bayes
	Feature_extraction : appliquer la méthode d'extraction de caractéristiques
	metrics : calculer les métriques de performance du modèle
	pipeline : créer et utiliser un pipeline dans le processus de classification
Keras	Permet d'implémenter des modèles de réseaux neuronaux
Nltk	corpus : utiliser la liste des mots vides
	tokenize : diviser la chaîne de caractères en sous chaînes
	stem : supprimer les affixes morphologiques des mots afin de garder juste la racine
Matplotlib	pyplot : visualiser les données sous forme de graphiques
Seaborn	Visualiser les données sous forme de graphiques
Datetime_extractor	DateTimeExtrator : extraire les dates d'une chaîne de caractères
Spellchecker	Corriger les fautes d'orthographe
Openpyxl	Lire et écrire des fichiers Excel
Contractions	Corriger les contractions telle que « He's » avec « He is »
Unicodedata	Utiliser la base de données de caractères Unicode (UCD)

4.2 Méthode de travail

Dans cette section, nous expliquons les phases de prétraitement et d'entrainement. Nous avons adopté différentes approches dans lesquelles une ou plusieurs techniques de prétraitement sont utilisées avec les algorithmes de classification SVM (Support Vector Machine), LR (Logistic Regression), NB (Naive Bayes), ainsi que l'architecture neuronale Bi-LSTM (Bidirectionnal Long Short Term Memory). Dans ce qui suit, nous expliquons d'abord les différentes techniques de prétraitement expérimentées ainsi que les

approches adoptées. Par la suite, nous présentons la technique de recherche en grille GridSearchCV. Enfin, nous expliquons les phases d'extraction des caractéristiques et d'entrainement.

4.2.1 Prétraitement

La phase de prétraitement est une phase importante dans le processus de classification de texte. Elle consiste à éliminer le bruit, nettoyer et normaliser les données brutes afin de les rendre exploitable par un algorithme de classification. Elle permet de supprimer tout ce qui est inutile dans le texte et qui n'apporte aucune information pertinente pour la classification de texte, afin d'améliorer les performances du classificateur et obtenir par la suite de meilleurs résultats. Plusieurs techniques de prétraitement sont utilisées dans ce travail.

4.2.1.1 Techniques de prétraitement utilisées

a) Conversion des mots en minuscule

La conversion de tous les mots en minuscule permet d'éviter de considérer un mot écrit parfois en minuscule et parfois en majuscule comme deux mots différents. Elle permet aussi de réduire le nombre de mots que le dictionnaire doit contenir et d'avoir une cohérence de la sortie attendue. Voici un exemple de commentaire de l'ensemble de données JigsawToxic avant la conversion en minuscule : « *Hahaha, fuck you ESA, you worthless pieces of shit.* ». Après la conversion en minuscule, le commentaire devient comme suit : «*hahaha, fuck you esa, you worthless pieces of shit.*».

b) Suppression des mots vides (stop words)

Les mots vides (appelés stop words en anglais) sont des mots qui sont couramment utilisés mais qui n'ajoutent pas d'informations pertinentes au texte. Ce sont généralement des mots de liaison comme les prépositions, les conjonctions, les pronoms, etc. Le Tableau 4.1 représente la liste des mots vides supprimés dans le texte. Il contient 179 mots en anglais.

Tableau 4.2. Liste des mots vides (stop-words).

lt	Those	Your	haven't	From	mustn	yourself	Was
By	A	Has	To	While	They	other	This
But	Being	After	you're	Again	Am	at	In
needn't	You	hadn't	their	Who	were	do	S
Yours	About	Here	under	don't	Few	mustn't	Why
I	With	Down	above	Just	these	shouldn	As
wasn't	Very	Should	didn't	himself	having	it's	Its
Him	isn't	Can	mightn't	Are	weren't	myself	Not
Did	Haven	Ll	How	Only	same	weren	she's
Hadn	Been	Up	yourselves	Nor	Had	aren't	"shouldn't
During	O	Of	All	doesn't	"couldn't	does	Than
Aren	His	Below	through	wouldn't	themselves	t	Ain
Hasn	Isn	should've	"won't	Be	Too	out	My
you'd	"shan't	Or	She	ourselves	Me	shan	Re
hasn't	Didn	you'll	there	Such	whom	some	Doing
For	An	Our	He	you've	Own	we	Until
Any	Where	Will	Her	On	mightn	y	Against
Over	Before	Between	D	The	what	ve	And
More	Won	Into	once	Ours	No	is	Doesn
That	If	Have	that'll	couldn	M	off	Further
Each	When	Don	herself	Theirs	Both	which	Itself
Then	Wasn	Needn	so	Now	wouldn	because	Hers
Ma	Them	most'					

Cette technique consiste à transformer le texte en jetons pour pouvoir les comparer à la liste des mots vides et effacer ces derniers. D'abord, nous utilisons la méthode split() de python pour diviser le texte en une liste de chaînes par le séparateur choisi, qui est l'espace dans notre cas. Par la suite, nous vérifions la liste de chaînes pour effacer tous les mots vides. Enfin, nous utilisons la méthode join() pour concaténer les jetons de notre liste ensemble pour reformer des phrases. Cette méthode permet de supprimer tous les mots vides du texte ainsi que les espaces. Voici un exemple de commentaire de l'ensemble de données JigsawToxic avant la suppression des mots vides : «*if you have something to say to me say it now*». Après la suppression des mots vides, le commentaire devient comme suit : « *something say say*» .

c) Suppression des nombres et des signes non-alphabétiques

Cette technique consiste à supprimer les nombres, les signes de ponctuation et les signes non-alphabétiques appelés aussi caractères spéciaux. Les signes non-alphabétiques sont des caractères qui ne sont pas des nombres, ni des lettres. Ce sont les signes de ponctuation, les accents et les symboles tel

que: !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~. Voici un exemple de commentaire de l'ensemble de données JigsawToxic avant la suppression des nombres et des signes non-alphabétiques : «*15:19, 13 November 2014* ». Après la suppression des nombres et des signes non-alphabétiques, le commentaire devient comme suit : « *November* »

d) Suppression des tabulations, multiples espaces et sauts

Cette technique permet de supprimer les tabulations, les multiples espaces et les sauts. Elle devient inutile si nous avons appliqué la technique de suppression des mots vides puisque tous les types d'espaces sont supprimés lors de ce traitement. Voici un commentaire de l'ensemble de données JigsawToxic avant le prétraitement :

« *The second member of his surname is not spelled right... It starts with an F*

*You know, Carter-F*ck*».

Après le prétraitement, le retour à la ligne est supprimé et le commentaire devient comme suit : « *The second member of his surname is not spelled right... It starts with an F You know, Carter-F*ck*».

e) Suppression des mentions utilisateurs, URLs et balises HTML

Une mention d'utilisateur (@mention) est un pseudo se composant du nom d'utilisateur précédé par le symbole @. Elle permet de faire référence à un autre utilisateur du réseau social et diriger une publication publique vers lui, tandis qu'un URL est une référence qui permet à la fois de spécifier l'emplacement d'une ressource Web et de la récupérer. Quant aux balises HTML[43], elles correspondent à des éléments du code HTML d'une page Web sur Internet qui permettent de structurer son contenu. Les mentions utilisateurs, les URLs[44] et les balises HTML sont supprimés car ils sont inutiles dans la classification de texte et ne fournissent aucune information pertinente sur la nature du texte (contenu agressif ou normal). Voici un

43 HyperText Markup Language

44 Uniform Resource Locator

commentaire de l'ensemble de données COVID-HATE avant le prétraitement : « *Yaa, we won't call it Chinese. @firkey_ @pokershash #ChineseVirusCorona #ChineseVirus #worldlockdown… https://t.co/r2hO6dyFOc*». Après le prétraitement, le commentaire devient comme suit : « *Yaa, we won't call it Chinese. #ChineseVirusCorona #ChineseVirus #worldlockdown…* ».

f) Gestion des hashtags

Un hashtag est l'ensemble d'un ou plusieurs mots qui sont précédés du symbole dièse #. Il est utilisé par les utilisateurs de réseaux sociaux comme forme de balisage permettant le regroupement du contenu partageant un ou plusieurs sujets. Les mots qui constituent l'hashtag peuvent être des mots normaux, comme ils peuvent être des mots agressifs. C'est pourquoi, le symbole # sera supprimé et le mot sera gardé. Voici un commentaire de l'ensemble de données COVID-HATE avant le prétraitement : « *Yaa, we won't call it Chinese. @firkey_ @pokershash #ChineseVirusCorona #ChineseVirus #worldlockdown… https://t.co/r2hO6dyFOc*». Après le prétraitement, le commentaire devient comme suit : « *Yaa, we won't call it Chinese. @firkey_ @pokershash ChineseVirusCorona ChineseVirus worldlockdown… https://t.co/r2hO6dyFOc*».

g) Suppression des horodatages et des dates des messages

L'horodatage est l'association d'un évènement à une date avec une heure, par exemple : 2023-01-10 T 14:05 UTC. Les dates et les horodatages qui apparaissent dans les tweets et les commentaires ne fournissent aucune information pertinente sur la nature des données (agressives ou non), c'est pourquoi ils seront supprimés. Pour cela, nous allons d'abord supprimer toutes les chaînes de caractères qui représentent les mois. Par la suite, une suppression des nombres et de caractères spéciaux est nécessaire pour enlever les dates en chiffres. Voici un commentaire de l'ensemble de données COVID-HATE avant le prétraitement : « *15:19, 13 November 2014*». Ce commentaire est composé d'une date et une heure, le commentaire est supprimé en entier après le prétraitement et nous obtenons ce qui suit : « ».

h) Tokenisation

La tokenisation est le processus consistant à découper un texte en plus petits morceaux appelés jetons. Un jeton est une séquence de caractères comprise entre deux séparateurs (des espaces blancs ou tabulations). Il peut être un mot, un chiffre ou un signe de ponctuation. Le but de cette opération est d'extraire d'un texte brut les différents mots qui le composent afin de pouvoir les analyser séparément. Voici un commentaire de l'ensemble de données JigsawToxic avant la tokenisation : « *Is it gay if you bang*

an animal of the same secks?». Après la tokenisation, nous obtenons la liste suivante qui contient tous les jetons du commentaire : « [*'Is', 'it', 'gay', 'if', 'you', 'bang', 'an', 'animal', 'of', 'the', 'same', 'secks', '?'*] ».

i) Suppression des émojis et émoticônes

Les émojis sont des petites images utilisées pour exprimer des émotions ou des idées dans des messages texte. Par exemple, 😄, est un emoji qui représente un visage heureux. Ils sont divisés en huit catégories : smileys et personnes, animaux et nature, nourriture et boissons, activité, voyages et lieux, objets, symboles et drapeaux. Tandis que les émoticônes sont des représentations d'expressions faciales utilisant des caractères du clavier et des signes de ponctuation qui transmettent une expression émotionnelle dans un message texte, par exemple : ':)' est un émoticône qui représente un visage heureux. Un Unicode unique est attribué à chaque émoji et émoticône permettant ainsi de l'identifier. Les fichiers contenant les codes sont téléchargeables à partir des deux liens suivants :

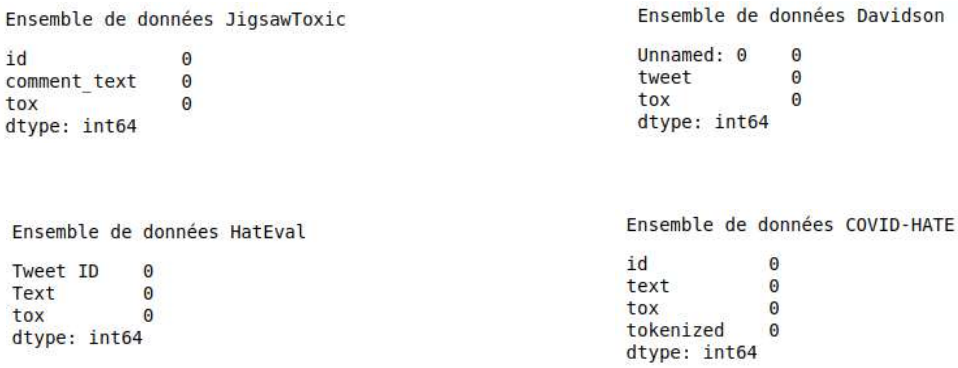
- Émojis : https://drive.google.com/open?id=1G1vIkkbqPBYPKHcQ8qy0G2zkoab2Qv4v
- Émoticônes : https://drive.google.com/open?id=1HDpafp97gCl9xZTQWMgP2kKK_NuhENlE

Les émojis et les émoticônes sont généralement utiles dans l'analyse d'opinion ou l'analyse de sentiments et ne fournissent pas d'informations pertinentes dans la classification de texte. Voici un commentaire de l'ensemble de données COVID-HATE avant la suppression des émojis : « *I want some chinamen hella bad I wish this coronavirus shit would die down a* 😂 *little* 😩*damn*». Après la suppression des émojis, nous obtenons ce qui suit : « *I want some chinamen hella bad I wish this coronavirus shit would die down a little damn* ».

j) Gestion des données manquantes

Les données manquantes sont un problème très répandu dans la fouille de données (data-mining). Les tweets collectés contenant des données manquantes seront supprimés en utilisant les deux fonctions isnull et dropna. La fonction 'isnull' est utilisée pour détecter les valeurs manquantes dans l'ensemble de données, tandis que la fonction 'dropna' permet de supprimer les lignes entières contenant une donnée manquante. Comme le montre la Figure 4.1, les ensembles de données utilisés ne contiennent aucune valeur manquante.

Figure 4.1. Valeurs manquantes dans les ensembles de données.



k) Gestion des accents

Généralement, les données textuelles utilisées ne doivent pas contenir des lettres accentuées, puisque l'utilisation des accents est limitée dans la langue anglaise. Cependant, il y a de bonnes chances de trouver un certain nombre d'accents dans le corpus, et ceci est dû soit aux fautes de frappes de l'utilisateur ou aux paramètres par défaut du clavier. Pour nettoyer le texte des accents, nous allons utiliser le module unidecode de Python qui permet de normaliser le texte en enlevant tous les accents dans une chaîne de caractères. Voici un commentaire de l'ensemble de données COVID-HATE avant la gestion des accents : « *a courageous, eye-opening exposé of the truth about leftwing fascism, and tactics of the liberals* ». Après la suppression des accents, nous obtenons ce qui suit: « *a courageous, eye-opening expose of the truth about leftwing fascism, and tactics of the liberals* ».

l) Gestion des contractions

Les contractions sont des formes abrégées de mots ou de combinaisons de mots. Elles sont créées en supprimant une ou plusieurs lettres et en les remplaçant par une apostrophe (par exemple : *it's, won't, wouldn't*.. etc.). Le remplacement de ces contractions par leurs mots d'origine contribue à la normalisation du texte. Pour effectuer cette tâche, nous allons utiliser la fonction « fix » du module Contractions de la bibliothèque Python. Elle permet l'expansion des contractions anglaises couramment utilisées et leur remplacement par leurs mots d'origines, par exemple : *I'm* devient *I am*, *it's* devient *it is* et *weren't* devient *were not*. Voici un commentaire de l'ensemble de données COVID-HATE avant la gestion des contractions : « *you don't need kids to live a fulfilling life, if anything they get in the way* ». Après la suppression des contractions, nous obtenons ce qui suit : « *you do not need kids to live a fulfilling life, if anything they get in the way* ».

m) Suppression des mots fréquents

Les mots très fréquents qui apparaissent le plus souvent dans l'ensemble de données sont généralement peu informatifs dans la classification de texte et ont un pouvoir discriminant faible en comparaison avec d'autres termes, c'est pourquoi ils peuvent être écartés du texte pour en réduire la dimensionnalité et améliorer la performance du système. La Figure 4.2 donne une idée sur les dix premiers mots fréquents pour chaque ensemble de données.

Figure 4.2. Les 10 mots fréquents sans prétraitement des données.

JigsawToxic		Davidson		COVID-HATE		HatEval	
to	37782	RT	8654	to	1054	to	5103
a	31679	the	8449	and	779	a	4275
I	29963	I	6719	a	724	and	3324
and	29919	to	5882	of	716	you	3224
you	28737	and	4434	is	667	of	2993
of	28372	you	4320	in	583	in	2507
is	23185	bitch	4216	Chinese	394	is	2284
that	18191	in	3600	you	394	for	1871
in	16464	is	3375	for	334	I	1829

La plupart des mots les plus fréquents pour chaque ensemble de données sont des mots vides (stop words). Ces mots n'ajoutent aucune information pertinente concernant la toxicité du texte. Dans un premier temps, nous allons procéder à la suppression de ces mots et voir l'impact de cela sur la liste des mots les plus fréquents pour chaque ensemble de données. La Figure 4.3 donne un aperçu sur les résultats obtenus.

Figure 4.3. Les 10 mots fréquents avec suppression des mots vides.

JigsawToxic		Davidson		COVID-HATE		HatEval	
,	65776	bitch	6522	.	2015	@	8248
!	48818	RT	6061	@	1766	.	6826
I	35515	bitches	2398	,	1242	:	5341
''	31029	like	2101	:	1072	,	5078
``	19611	hoes	1897	https	819	https	4253
?	13375	pussy	1705	!	691	!	3167
)	11185	hoe	1554	Chinese	486	I	2407
(	10098	ass	1228	coronavirus	447	?	1771
:	9953	got	1210	’	435	bitch	1686

La Figure 4.3 montre qu'après suppression des mots vides, la liste des dix mots les plus fréquents contient généralement des signes de ponctuation. Cependant, certaines listes contiennent des mots qui fournissent de l'information pertinente sur la toxicité du texte. C'est le cas du mot '*bitches*' dans l'ensemble de données Davidson et '*bitch*' dans HatEval. Nous pouvons en déduire par la suite, que la plupart des premiers mots les plus fréquents dans un ensemble de données sont soit les mots vides, les signes de ponctuation ou les mots toxiques qui sont largement utilisées par les utilisateurs des réseaux sociaux. Alors, pour éviter de supprimer des mots importants dans la classification de texte et diminuer ainsi les

performances du système, il faut commencer par supprimer les mots fréquents d'abord, par la suite nous pouvons supprimer les mots vides et les signes de ponctuation.

n) Suppression des mots rares

Les mots rares sont les mots de nature unique, ceux qui apparaissent moins fréquemment dans l'ensemble de données comme les noms propres, les noms de villes, les liens url et les hashtags. Ces mots sont généralement peu informatifs dans la classification de texte et ont un pouvoir discriminant faible en comparaison avec d'autres termes, c'est pourquoi ils peuvent être écartés du texte pour en réduire la dimensionnalité et améliorer la performance du système. La Figure 4.4 donne une idée sur les dix mots rares pour chaque ensemble de données.

Figure 4.4. Les 10 mots rares pour chaque ensemble de données.

JigsawToxic		Davidson		COVID-HATE		HatEval	
1.4m	1	@nikkiconner13	1	#Lockdown	1	her👇	1
articles.I	1	leader.	1	https://t.co/a6djq47XkD	1	Pizza-Worker	1
rewritten.	1	Heru	1	want?	1	cash?	1
U!!!	1	February	1	(1/2)	1	Gulasekaram,	1
pedia.	1	@wwwobert_:	1	supremacist	1	crap)	1
.SO	1	#Honda200	1	@KamVTV	1	https://t.co/1pPCNwi8yD	1
california,	1	Gotti	1	https://t.co/TKfZctggLt	1	@fox5sandiego	1
filming	1	Spittin	1	MSNBC,	1	Rush	1
Enron	1	http://t.co/8oxlQvQBrC"	1	💔	1	https://t.co/LBveD83yHG	1

La plupart des mots rares pour chaque ensemble de données sont des noms d'utilisateurs, des hashtags et des liens URL. Ces mots n'ajoutent aucune information pertinente concernant la toxicité du texte. Dans un premier temps, nous allons procéder à la suppression de ces mots et voir l'impact de ça sur la liste des mots rares pour chaque ensemble de données. La Figure 4.5 donne un aperçu sur les résultats obtenus.

Figure 4.5. Les 10 mots rares après 1ère phase du prétraitement.

JigsawToxic		Davidson		COVID-HATE		HatEval	
2035	1	REALLY?	1	clips,"	1	barbie	1
54.	1	21,	1	Residential	1	SJW's	1
Talk:Bob	1	"entertaining"	1	ppl,	1	pe	1
funny!!!!!!!!!!!!	1	rant	1	Whiny,	1	REALTIME	1
""destra"",	1	They've	1	flu?	1	Zac	1
Stupis	1	"lol"	1	plus	1	Reduction	1
evacuation,	1	gay”or	1	bot	1	TELANGANA....jai	1
Group)	1	stroked	1	list!	1	ðŸ˜,ðŸ˜‰.	1
citation...)...If	1	mandate	1	understanding	1	share..	1

La Figure 4.5 montre qu'après suppression des noms d'utilisateurs, des liens url et des hashtags, la liste des dix mots rares contient généralement des mots mal formés (mots contenant des signes de ponctuation). Avant de procéder à la suppression des mots rares, il est préférable de supprimer les signes

de ponctuation d'abord. Cela permet d'éviter de supprimer des mots qui sont importants dans la classification de texte, comme le cas du mot '*us….fu****' dans l'ensemble de données COVID-HATE. La Figure 4.6 donne un aperçu des résultats obtenus.

Figure 4.6. Les 10 mots rares après suppression des signes de ponctuation.

JigsawToxic		Davidson		COVID-HATE		HatEval	
Orgy	1	theeeese	1	NUTS	1	Corrected	1
inflamed	1	concluded	1	Support	1	sodomized	1
restitution	1	remarks	1	Cher	1	upskyrt	1
supremely	1	restricton	1	CandiceIyog	1	cuffed	1
nanodiamonds	1	eric	1	memo	1	designer	1
subhumans	1	bellybutton	1	popping	1	PoliticalIslam	1
Oneself	1	Chilli	1	bcz	1	JusticeForMollieTibbetts	1
Categorie	1	Boom	1	played	1	prospect	1
affirmed	1	crissi	1	puting	1	uphow	1

La Figure 4.6 montre qu'après suppression des signes de ponctuation, la liste des dix mots rares contient des mots ne donnant pas d'informations pertinentes dans la classification de texte. Alors, pour nettoyer le texte et supprimer le maximum de mots rares, il faudra supprimer en premier lieu les noms d'utilisateurs, les hashtags, les liens URL et les balises HTML. En deuxième lieu, il faut supprimer les signes de ponctuation et procéder par la suite à la suppression des dix premiers mots rares.

o) Vérification orthographique

C'est une étape importante du prétraitement du texte qui a un impact positif sur l'amélioration de la classification. Les fautes d'orthographes sont courantes dans les données textuelles et elles doivent être corrigées avant de procéder à la classification. Ces mots dont l'orthographe est incorrecte seront remplacés par les mots avec l'orthographe correcte en utilisant le package pyspellchecker. Ce package prend en charge plusieurs langues, dont l'anglais, l'espagnol, l'allemand, le français et le portugais. Voici un commentaire de l'ensemble de données Davidson avant la correction orthographique : « *weeknd dont save hoes b, idunno bout Drake It's to many hoes Tryna act like the Wifey type.* ». Après la correction orthographique, nous obtenons ce qui suit : « *weekend dont save hoes be dunno bout drake it's to many hoes tryna act like the wifey type* ».

p) Racinisation (stemming)

La racinisation (appelée stemming en anglais) est une technique de traitement de langage naturel permettant la conversion des mots en leurs racines en supprimant les suffixes les plus utilisés. Par exemple, le résultat de la racinisation (stem) pour les mots "*likes*", "*liked*", "*liking*" et "*likely*" est "*like*". Cette

technique est plus rapide, mais moins efficace que la lemmatisation car elle ne se base pas sur les dictionnaires et elle ne prend pas en compte des nombreuses exceptions des règles de dérivation. Ce qui conduit souvent à des significations incorrectes et à des fautes d'orthographe. Par exemple, le résultat de la racinisation (stem) pour les mots "*university* (*université en français*)" et "*universe* (*univers en français*)" est le même "*univers*", pourtant les deux mots sont différents et il ne faut pas les traiter de manière identique. Néanmoins, cette méthode reste efficace pour réduire la dimensionalité de l'ensemble de données. Elle utilise des algorithmes simples basées sur des règles de remplacement de chaînes de caractères. Dans ce travail, nous allons utiliser l'algorithme Porter Stemming qui utilise plusieurs règles pour supprimer les terminaisons morphologiques et flexionnelles les plus courantes, afin de ramener les mots à leurs racines. Voici un commentaire avant la racinisation : « *HE vandalized MINE using a sockpuppet accusation template. *I* merely returned the favor. Now blow it out your eye wall!* ». Après la racinisation, le commentaire devient comme suit : « *He vandal MINE use a sockpuppet accus template. *I* mere return the favor. Now blow it out your eye wall!*».

q) Lemmatisation (lemmatization)

La lemmatisation est une tâche qui s'appuie sur des outils de TALN[45] et nécessitant diverses ressources linguistiques (dictionnaires, règles de dérivation, etc.). Elle ramène les termes à leur forme canonique significative en considérant le contexte. Elle met tous les noms au singulier, les adjectifs au masculin singulier et tous les verbes conjugués à l'infinitif. Par exemple, le résultat de la lemmatisation (lemma) pour le mot "*books*" est "*book*" et pour le mot "*won*", c'est "*win*". Cette technique permet de traiter les différentes variantes issues d'une même forme canonique comme un mot unique. C'est une méthode efficace pour réduire considérablement la dimension du corpus.

Dans ce travail, nous allons utiliser WordNetLemmatizer.lemmatize(), un lemmatiseur de la bibliothèque NLTK[46], construit à l'aide de Wordnet qui est une base de données lexical de la langue anglaise, organisée par concept et par sens. La balise POS est l'un des paramètres de ce lemmatiseur. Elle a comme valeur par

45 Traitement Automatique de Langage Naturel

46 Natural Language Toolkit

défaut « n » pour les noms, mais elle peut avoir quatre autres valeurs différentes : « v » pour les verbes, « a » pour les adjectifs, « r » pour les adverbes et « s » pour les adjectifs satellites. Pour avoir un bon fonctionnement du lemmatiseur, il faut définir les informations de la balise POS, sinon il va considérer tous les jetons comme des noms, ce qui influencera négativement sur l'exactitude des résultats. Prenons l'exemple de commentaire suivant : « *HE vandalized MINE using a sockpuppet accusation template. *I* merely returned the favor. Now blow it out your eye wall!* » . Voici le résultat de la lemmatisation en gardant les paramètres par défaut : « *HE vandalized MINE using a sockpuppet accusation template. *I* merely returned the favor. Now blow it out your eye wall!* » . Pour définir les informations de la balise POS, il faut vérifier pour chaque jeton s'il est un adjectif, un adverbe ou un verbe, sinon il sera considéré comme un nom. Voici le commentaire après lemmatisation : « *HE vandalize MINE use a sockpuppet accusation template . * I * merely return the favor . Now blow it out your eye wall !* ».

La lemmatisation fonctionne mieux dans le 2ème cas et donne un résultat meilleur par rapport au 1er cas. Le Tableau 4.3 donne une comparaison entre quelques résultats d'autres exemples.

Tableau 4.3. Exemples de résultats de lemmatisation.

Jeton	**Balise POS**	**Résultats de lemmatisation**	
		Avec valeurs par défaut de la balise POS	**En définissant les valeurs de la balise POS**
Tweets	Nom «n»	Tweet	Tweet
Using	Verbe «v »	Using	Use
returned	Verbe «v »	Returned	Return
Am	Verbe «v »	Am	Be
Havely	Adverbe «r »	Havely	Havely
Careful	Adjectif «a »	Careful	Careful

r) Suppression des mots avec une longueur <= 2 caractères

Après avoir effectué toutes les étapes requises dans la phase de prétraitement du texte, ce dernier peut contenir des mots qui sont composés juste de deux caractères ou moins. Ces mots sont considérés comme du bruit puisqu'ils n'apportent pas de l'information pertinente concernant la toxicité du texte, d'où la nécessité de leur suppression. Ce qui permet de réduire la dimensionnalité et améliorer les performances du système. Voici un commentaire de l'ensemble de données HatEval avant l'application de la technique :

« *europe wants centers africa vet migrants critics say abdicating responsibilities co ptlsd jm w*». Après l'application de la technique, nous obtenons ce qui suit : « *europe wants centers africa vet migrants critics say abdicating responsibilities ptlsd* ».

4.2.1.2 Approches expérimentées

Dans toutes les approches expérimentées, nous avons utilisé le modèle TfidfVectorizer avec les algorithmes d'apprentissage supervisé choisis. Les paramètres lowercase et token_pattern de ce dernier ont pour valeurs par défaut : lowercase = True et token_pattern = '(?u)\b\w\w+\b'. Le premier paramètre permet de convertir le texte en lettres minuscules, quant au deuxième paramètre, il permet de faire la tokenisation du texte en acceptant les jetons constitués de 2 caractères alphanumériques ou plus et en rejetant les autres. Dans ce travail, nous faisons le prétraitement des données avant de faire passer le résultat au TfidfVectorizer. C'est pourquoi nous avons modifié les valeurs par défaut des paramètres lowercase = False et token_pattern = ' \S+', cela permet de ne pas convertir le texte en lettres minuscules et d'accepter tous les types de jetons, même ceux constitués d'un seul caractère ou des signes non-alphabétiques et de rejeter les espaces. Le paramètre ngram_range= (1, 1) permet d'avoir des jetons sous forme d'unigrammes uniquement.

Pour vectoriser le texte avant de le faire passer au modèle Bi-LSTM, nous avons utilisé le modèle Tokenizer. Les paramètres filters et lower de ce dernier ont pour valeurs par défaut : filters = '!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n' et lower = True. Le premier paramètre permet de supprimer les espaces, les caractères spéciaux et la ponctuation, et le deuxième permet de convertir le texte en minuscule. Puisque le prétraitement du texte se fait avant de le faire passer au Tokenizer, nous avons modifié ces deux paramètres filters = ' \r\t\n' et lower = False, cela permet de ne pas convertir le texte en lettres minuscules et d'accepter tous les types de jetons, même ceux constitués d'un seul caractère ou des signes non-alphabétiques et de rejeter les espaces.

Nous expérimentons trois scénarios contenant vingt-six approches au total. Le scénario 1 correspond à l'approche 1, où le classificateur reçoit en entrée le texte brut après tokenisation. Le scénario 2 dans lequel nous avons regroupé les approches de 2 à 17 et l'approche 25, permet de vérifier l'impact de la suppression du bruit sur les résultats de la classification, comme la suppression des mots vides, des nombres, des caractères spéciaux, des mots rares et des mots fréquents, etc. Les approches de 18 à 24 et l'approche 26 sont regroupées dans le scénario 3 qui se focalise sur la correction orthographique et la

normalisation du texte en utilisant la lemmatisation et la racinisation. Dans ces approches, nous prenons comme exemple le même commentaire de l'ensemble de données COVID-HATE. Voici le commentaire avant le prétraitement :

> *@CoviDcreations @clif_high #coronavirus Report has been published first time and results are extremely serious. Coronavirus has become the biggest challenge of 2020 in front of Scientists and health experts. Like, RT: https://t.co/lKia09KKE9 #CoronavirusOutbreak #wuhan #chinavirus #coronaviruschina*

a) Approche #1

Dans cette approche, les modèles TfidfVectorizer et Tokenizer reçoivent en entrée le texte brut sans prétraitement.

b) Approche #2

Dans cette approche, nous avons converti le texte en lettres minuscules. Ensuite, nous avons supprimé les mots vides (stop words) avant de procéder à la tokenisation du texte. Après le prétraitement, le commentaire devient comme suit :

> *@covidcreations @clif_high #coronavirus report published first time results extremely serious. coronavirus become biggest challenge 2020 front scientists health experts. like, rt: https://t.co/lkia09kke9 #coronavirusoutbreak #wuhan #chinavirus #coronaviruschina*

c) Approche #3

Dans cette approche, nous avons converti le texte en lettres minuscules. Ensuite, nous avons supprimé les mots vides (stop words), les nombres et les caractères spéciaux. Ce qui donne le résultat suivant :

> *covidcreations clif high coronavirus report published first time results extremely serious coronavirus become biggest challenge front scientists health experts like rt https t co lkia kke coronavirusoutbreak wuhan chinavirus coronaviruschina*

d) Approche #4

Dans cette approche, nous avons converti d'abord le texte en lettres minuscules. Par la suite, nous avons supprimé les nombres et les signes non-alphabétiques. Cela étant, nous avons procédé à la suppression des mots vides (stop words). Ce qui en résulte le commentaire suivant :

covidcreations clif high coronavirus report published first time results extremely serious coronavirus become biggest challenge front scientists health experts like rt https co lkia kke coronavirusoutbreak wuhan chinavirus coronaviruschina

e) Approche #5

Dans cette approche, nous avons converti d'abord le texte en lettres minuscules. Par la suite, nous avons supprimé les nombres et les signes non-alphabétiques. Ce qui donne le résultat suivant :

covidcreations clif high coronavirus report has been published first time and results are extremely serious coronavirus has become the biggest challenge of in front of scientists and health experts like rt https t co lkia kke coronavirusoutbreak wuhan chinavirus coronaviruschina

f) Approche #6

Dans cette approche, nous avons converti le texte en lettres minuscules. Par la suite, nous avons supprimé d'abord les nombres et les signes non-alphabétiques. Cela étant, nous avons enlevé les mentions utilisateurs, les hashtags, les URLs, les balises HTML, les horodatages des messages ainsi que les mots vides (stop words). Le commentaire devient comme suit après prétraitement :

covidcreations clif high coronavirus report published first time results extremely serious coronavirus become biggest challenge front scientists health experts like rt co lkia kke coronavirusoutbreak wuhan chinavirus coronaviruschina

g) Approche #7

Dans cette approche, nous avons d'abord converti le texte en lettres minuscules. Ensuite, nous avons supprimé les mentions utilisateurs, les hashtags, les URLs, les balises HTML, les horodatages des messages. Cela étant, nous avons enlevé les nombres et les caractères non alphabétiques. Les mots vides (stop words) sont supprimés en dernier lieu, ce qui donne le résultat suivent :

coronavirus report published first time results extremely serious coronavirus become biggest challenge front scientists health experts like rt coronavirusoutbreak wuhan chinavirus coronaviruschina

h) Approche #8

Dans cette approche, nous avons commencé par convertir le texte en lettres minuscules. Ensuite, nous nous sommes contentés de supprimer les mentions utilisateurs, les hashtags, les URLs, les balises HTML,

les horodatages des messages et par la suite les mots vides (stop words). Le commentaire devient comme suit :

> *coronavirus report published first time results extremely serious. coronavirus become biggest challenge 2020 front scientists health experts. like, rt: coronavirusoutbreak wuhan chinavirus coronaviruschina*

i) Approche #9

Dans cette approche, le texte est converti d'abord en lettres minuscules. Par la suite, nous avons supprimé les mots fréquents, ce qui donne ce qui suit :

> *@covidcreations @clif_high #coronavirus report has been published first time results are extremely serious. coronavirus has become the biggest challenge 2020 front scientists health experts. like, rt: https://t.co/lkia09kke9 #coronavirusoutbreak #wuhan #chinavirus #coronaviruschina*

j) Approche #10

Dans cette approche, nous avons d'abord converti le texte en lettres minuscules. Par la suite, nous avons supprimé les mots fréquents. En fin de compte, les mots vides (stop words) sont supprimés. Le commentaire devient comme suit :

> *@covidcreations @clif_high #coronavirus report published first time results extremely serious. coronavirus become biggest challenge 2020 front scientists health experts. like, rt: https://t.co/lkia09kke9 #coronavirusoutbreak #wuhan #chinavirus #coronaviruschina*

k) Approche #11

Dans cette approche, nous avons d'abord converti le texte en lettres minuscules. Par la suite, nous avons supprimé les mots vides (stop words). En fin de compte, les mots fréquents sont supprimés. Le commentaire devient comme suit :

> *@covidcreations @clif_high report published first time results extremely serious. become biggest challenge 2020 front scientists health experts. like, rt: https://t.co/lkia09kke9 #coronavirusoutbreak #wuhan #chinavirus #coronaviruschina*

l) Approche #12

Dans cette approche, le texte est converti en lettres minuscules. Ensuite, nous avons supprimé les mots fréquents. Cela étant, nous avons supprimé les mentions utilisateurs, les hashtags, les URLs, les balises HTML, les horodatages des messages et par la suite les mots vides (stop words). Le commentaire devient comme suit :

> *coronavirus report has been published first time results are extremely serious coronavirus has become the biggest challenge front scientists health experts like rt coronavirusoutbreak wuhan chinavirus coronaviruschina*

m) Approche #13

Dans cette approche, le texte est converti d'abord en lettres minuscules. Par la suite, nous avons supprimé les mots rares, ce qui donne ce qui suit :

> *@covidcreations @clif_high #coronavirus report has been published first time and results are extremely serious. coronavirus has become the biggest challenge of 2020 in front of scientists and health experts. like, rt: https://t.co/lkia09kke9 #coronavirusoutbreak #wuhan #chinavirus #coronaviruschina*

n) Approche #14

Dans cette approche, nous avons d'abord converti le texte en lettres minuscules. Ensuite, nous avons supprimé les mots rares. Cela étant, nous avons supprimé les mentions utilisateurs, les hashtags, les URLs, les balises HTML, les horodatages des messages et par la suite les nombres et les caractères spéciaux. Le commentaire devient comme suit :

> *@covidcreations @clif_high #coronavirus report published first time results extremely serious. coronavirus become biggest challenge 2020 front scientists health experts. like, rt: https://t.co/lkia09kke9 #coronavirusoutbreak #wuhan #chinavirus #coronaviruschina*

o) Approche #15

Dans cette approche, nous avons d'abord converti le texte en lettres minuscules. Ensuite, nous avons supprimé les mentions utilisateurs, les hashtags, les URLs, les balises HTML, les horodatages des messages et par la suite les nombres et les caractères spéciaux. Enfin de compte, nous avons enlevé les mots rares. Le commentaire devient comme suit :

coronavirus report has been published first time and results are extremely serious coronavirus has become the biggest challenge of in front of scientists and health experts like rt coronavirusoutbreak wuhan chinavirus coronaviruschina

p) Approche #16

Dans cette approche, le texte est converti en lettres minuscules. Ensuite, nous avons supprimé les mots rares. Cela étant, nous avons supprimé les mots vides (stop words). Le commentaire devient comme suit :

@covidcreations @clif_high #coronavirus report published first time results extremely serious. coronavirus become biggest challenge 2020 front scientists health experts. like, rt: https://t.co/lkia09kke9 #coronavirusoutbreak #wuhan #chinavirus #coronaviruschina

q) Approche #17

Dans cette approche, nous avons d'abord converti le texte en lettres minuscules. Ensuite, nous avons supprimé les émojis et les émoticônes. Cela étant, nous avons supprimé les mots vides (stop words). Le commentaire devient comme suit :

@covidcreations @clif_high #coronavirus report published first time results extremely serious. coronavirus become biggest challenge 2020 front scientists health experts. like, rt: https/t.co/lkia09kke9 #coronavirusoutbreak #wuhan #chinavirus #coronaviruschina

r) Approche #18

Dans cette approche, nous avons d'abord converti le texte en lettres minuscules, corrigé les contractions et enlevé les accents. Finalement, nous avons procédé à la correction des fautes d'orthographes. Ce qui donne le commentaire suivant :

@covidcreations @clif_high #coronavirus report has been published first time and results are extremely serious coronavirus has become the biggest challenge of 2020 in front of scientists and health experts like rty https://t.co/lkia09kke9 #coronavirusoutbreak whan #chinavirus #coronaviruschina

s) Approche #19

Dans cette approche, nous avons d'abord converti le texte en lettres minuscules. Par la suite, nous avons procédé à la correction des fautes d'orthographes. Cela étant, nous avons corrigé les contractions et enlevé les accents. Ce qui donne le commentaire suivant :

> *@covidcreations @clif_high #coronavirus report has been published first time and results are extremely serious coronavirus has become the biggest challenge of 2020 in front of scientists and health experts like rty https://t.co/lkia09kke9 #coronavirusoutbreak whan #chinavirus #coronaviruschina*

t) Approche #20

Dans cette approche, nous convertissons le texte en lettres minuscules. Par la suite, nous avons procédé à la racinisation (stemming), ce qui donne le commentaire suivant :

> *@covidcr @clif_high #coronaviru report ha been publish first time and result are extrem serious. coronaviru ha becom the biggest challeng of 2020 in front of scientist and health experts. like, rt: https://t.co/lkia09kke9 #coronaviruoutbreak #wuhan #chinaviru #coronaviruchina*

u) Approche #21

Dans cette approche, nous convertissons le texte en lettres minuscules. Par la suite, nous avons procédé à la lemmatisation, ce qui donne le commentaire suivant :

> *@ covidcreations @ clif_high # coronavirus report have be publish first time and result be extremely serious . coronavirus have become the biggest challenge of 2020 in front of scientist and health expert . like , rt : http : //t.co/lkia09kke9 # coronavirusoutbreak # wuhan # chinavirus # coronaviruschina*

v) Approche #22

Dans cette approche, nous avons d'abord converti le texte en lettres minuscules. Par la suite, nous avons procédé à la lemmatisation suivie de la racinisation (Stemming). Ce qui donne le commentaire suivant :

> *@ covidcr @ clif_high # coronaviru report have be publish first time and result be extrem seriou . coronaviru have becom the biggest challeng of 2020 in front of scientist and health expert . like , rt : http : //t.co/lkia09kke9 # coronaviruoutbreak # wuhan # chinaviru # coronaviruchina*

w) Approche #23

Dans cette approche, nous avons converti le texte en lettres minuscules. Par la suite, nous avons procédé à la racinisation (stemming) suivie de la lemmatisation. Ce qui donne le commentaire suivant :

> *@ covidcr @ clif_high # coronaviru report ha be publish first time and result be extrem serious . coronaviru ha becom the biggest challeng of 2020 in front of scientist and health expert . like , rt : http : //t.co/lkia09kke9 # coronaviruoutbreak # wuhan # chinaviru # coronaviruchina*

x) Approche #24

Dans cette approche, nous avons d'abord converti le texte en lettres minuscules. Par la suite, nous avons corrigé les contractions, enlevé les accents et corrigé les fautes d'orthographes. Finalement, nous avons procédé à la lemmatisation suivie de la racinisation (stemming). Ce qui donne le commentaire suivant :

> *@ covidcr @ clif_high # coronaviru report have be publish first time and result be extrem seriou coronaviru have becom the biggest challeng of 2020 in front of scientist and health expert like rti http : //t.co/lkia09kke9 # coronaviruoutbreak whan # chinaviru # coronaviruchina*

y) Approche #25

Dans cette approche, nous avons converti le texte en lettres minuscules. Ensuite, nous avons supprimé les mentions utilisateurs, les hashtags, les URLs, les balises HTML, les horodatages des messages et par la suite les nombres et les caractères spéciaux. Cela étant, nous avons supprimé les mots rares. Finalement, nous avons supprimé les jetons composés de deux caractères et moins. Ce qui donne ce qui suit :

> *coronavirus report has been published first time and results are extremely serious coronavirus has become the biggest challenge front scientists and health experts like coronavirusoutbreak wuhan chinavirus coronaviruschina*

z) Approche #26

Dans cette approche, nous avons d'abord converti le texte en lettres minuscules. Par la suite, nous avons corrigé les contractions, enlevé les accents et corrigé les fautes d'orthographes. Cela étant, nous avons procédé à la lemmatisation suivie de la racinisation (stemming). Finalement, nous avons supprimé les jetons composés de deux caractères et moins. Ce qui donne le commentaire suivant :

@ covidcr @ clif_high # coronaviru report have publish first time and result extrem seriou coronaviru have becom the biggest challeng 2020 front scientist and health expert like rti http : //t.co/lkia09kke9 # coronaviruoutbreak whan # chinaviru # coronaviruchina

4.2.2 GridSearchCV

La première étape était de trouver pour chaque ensemble de données, les meilleures valeurs des paramètres suivants : kernel et Gamma du classificateur SVM, solver, class-weight et C du classificateur LR et alpha du classificateur NB. Les autres paramètres des classificateurs vont garder leurs valeurs par défaut. Les paramètres kernel et Gamma du SVM permettent de spécifier respectivement le type de noyau et le paramètre de régularisation. Les paramètres solver, class-weight et C du LR permettent de spécifier respectivement l'algorithme d'optimisation, les poids associés aux classes et le paramètre de régularisation. Le paramètre alpha du NB représente le paramètre d'adoucissement. Pour pouvoir trouver les valeurs idéales de ces paramètres, nous avons utilisé la technique GridSearchCV qui est une technique de recherche à validation croisée sur une grille de paramètres spécifiés pour un estimateur. Le but de cette technique est d'obtenir les meilleures valeurs des paramètres spécifiés permettant d'avoir la meilleure performance du modèle. Dans notre cas, les paramètres que nous avons choisi pour la validation croisée utilisée avec la technique GridSearchCV sont comme suit : K-fold = 10 et scoring = 'accuracy'.

Le paramètre max-features du TfidfVectorizer et num_words du Tokenizer, sont fixés à 20 000. Ces paramètres permettent de ne prendre en compte que des principales caractéristiques maximales classées par fréquence des termes dans le corpus, ce qui permet par la suite, de réduire la dimensionalité de ce dernier. Le Tableau 4.4 montre les valeurs obtenues pour chaque ensemble de données.

Tableau 4.4. Valeurs obtenues avec GridSearchCV.

Classificateur		**Ensemble de données**			
		JigsawToxic	**Davidson**	**COVID-HATE**	**HatEval**
SVM	Kernel	Linear	Poly	Rbf	Rbf
	Gamma	Auto	Scale	Scale	Scale
LR	C	10	10	10	1
	Class-weight	Balanced	Balanced	None	None
	Solver	Saga	Saga	Liblinear	Liblinear
NB	Alpha	1	0.1	0.1	1

Les valeurs obtenues par la grille de recherche pour les paramètres choisis sont utilisées dans toutes les approches expérimentées par la suite.

En ce qui concerne le modèle neuronal Bi-LSTM, la technique GridSearchCV n'a pas été utilisée pour trouver les meilleures valeurs des hyperparamètres, en raison de la limitation en termes de ressources et par manque de temps. Nous nous sommes basés, pour le réglage de ces paramètres, sur la littérature, en adoptant les valeurs qui ont donné les meilleurs résultats.

4.2.3 Extraction des caractéristiques

Dans cette étape, la plupart des paramètres ont gardé leurs valeurs par défaut, sauf le paramètre max-features du TF-IDFVectorizer qui a été fixé à 20 000 pour tous les ensembles de données. Nous avons gardé la valeur par défaut des paramètres ngram_range= (1,1), max_df=1 et min_df=1. Ce qui signifie que les caractéristiques sont composées d'un seul mot et que tous les termes sont acceptés peu importe leurs fréquences d'apparition dans le document. Pour le model Tokenizer, nous avons fixé le paramètre num_words à 20 000, ce paramètre permet de ne prendre en compte que des principales caractéristiques maximales classées par fréquence des termes dans le corpus.

4.2.4 Entrainement

Dans la phase d'entraînement, nous avons utilisé la méthode de validation croisée avec un cv=5 pour éviter le sur-apprentissage. Nous avons entraîné séparément les algorithmes d'apprentissage supervisé SVM (Support Vector Machine), LR (Logistic Regression) et NB (Naive Bayes) avec les vecteurs de caractéristiques générés par la méthode TF-IDFVectorizer. Nous obtenons avec chaque algorithme et pour chaque ensemble de données plusieurs modèles entraînés. Chaque modèle correspond à l'une des approches expérimentées. Ces modèles sont utilisés pour prédire les données de test de chaque ensemble de données afin d'obtenir les métriques de performances pour chaque modèle (précision, rappel, accuracy et F1-mesure).

Concernant l'architecture neuronale, nous avons utilisé un modèle séquentiel de l'API Keras de TensorFlow. Il comprend une couche Embedding de dimension 400, une couche LSTM bidirectionnelle avec 64 unités, une couche GlobalMaxPool1D qui permet de créer un vecteur de caractéristique unidimensionnel et dans la sortie une couche Dense avec 2 unités et une fonction d'activation softmax. Le modèle est compilé en utilisant binary_crossentropy comme fonction de perte et l'accuracy comme métrique d'évaluation avec

l'optimiseur RMSprop. Nous avons choisi cet optimiseur à la suite des bons résultats qu'il a montré par rapport aux autres optimisateurs dans l'étude de (Zaman, Hasan, Sakline, Das, & Alam, 2021). L'architecture du Bi-LSTM utilisé est illustrée dans la Figure D.1 (voir ANNEXE D) .

Nous avons divisé aléatoirement l'ensemble d'entraînement en deux sous-ensembles, d'entraînement et de validation. Le modèle est entraîné séparément avec les quatre ensembles de données en utilisant les scénarios déjà définis dans les sections précédentes. L'entraînement se fait avec le sous-ensemble d'entraînement, tandis que l'évaluation des performances du modèle se fait à chaque époque (epoch) en utilisant le sous-ensemble de validation. Nous avons défini le nombre d'époques (epochs) à 100 et le paramètre batch_size à 128 pour tous les ensembles de données. Les autres paramètres ont gardé leurs valeurs par défaut. Pour éviter le sur-apprentissage, nous avons utilisé la fonction callback avec un EarlyStopping. Cette fonction permet de surveiller la progression du processus d'entraînement en analysant la métrique choisie, sauvegarder le modèle le plus performant en fonction de cette même métrique et arrêter le processus de l'entraînement lorsque cette métrique cesse de s'améliorer. Dans notre cas, nous avons choisi la valeur de perte de l'ensemble de validation (val_loss) comme métrique d'évaluation. L'entraînement se fait avec les séquences générées par le modèle Tokenizer. Les modèles obtenus pour chaque scénario sont utilisés pour prédire les données de test de chaque ensemble de données afin d'obtenir les métriques de performances pour chaque modèle (précision, rappel, accuracy et F1-mesure).

4.3 Conclusion

Dans ce chapitre, nous avons présenté l'environnement logiciel et les bibliothèques utilisées dans ce travail. Nous avons aussi présenté les différentes techniques expérimentées dans la phase de prétraitement, ainsi que celles utilisées dans la phase d'entraînement et dans la phase de classification. Dans le chapitre suivant, nous présentons une synthèse des résultats ainsi qu'une comparaison entre les scénarios expérimentés. Par la suite, nous donnons une explication des résultats obtenus avec les différentes approches utilisées.

CHAPITRE 5
RÉSULTATS ET DISCUSSION

Dans ce chapitre, nous dévoilons les résultats obtenus par les algorithmes utilisés pour les quatre ensembles de données. Nous présentons d'abord, les résultats obtenus avec chaque scénario et nous faisons une comparaison entre ces différents scénarios. Finalement, nous discutons les résultats obtenus avec les différentes approches utilisées.

5.1 Résultats

Nous avons utilisé les modèles obtenus de chaque approche pour prédire les données de test. Nous avons effectué ce travail pour chaque scénario et avec les quatre ensembles de données afin d'évaluer les performances des modèles. Les résultats obtenus avec les ensembles d'entraînement et ceux de test sont dévoilés dans les sous-sections suivantes.

5.1.1 Scénario #1

Nous dévoilons dans cette section, les résultats obtenus par les quatre algorithmes utilisés, SVM, LR, NB et Bi-LSTM, avec les quatre ensembles de données pour le scénario #1. Dans ce scénario, le classificateur reçoit en entrée le texte brut après tokenisation.

5.1.1.1 Ensemble de données JigsawToxic

Dans cette section, nous présentons les résultats de classification obtenus pour l'ensemble de données JigsawToxic par les algorithmes d'apprentissage automatique utilisés. Le Tableau 5.1 montre les résultats de classification obtenus par l'algorithme SVM, LR, NB et Bi-LSTM.

Tableau 5.1. Résultats de classifications avec le scénario #1 pour JigsawToxic.

	Algorithmes utilisés	Ensemble d'entraînement				Ensemble de test			
		Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Scénario #1	SVM	0,875	0,8552	0.8723	0,8901	0,8348	0,8155	0,8326	0,8504
	LR	0,8717	0,847	0,8683	0,8906	0,8339	0,7978	0,8287	0,8621
	NB	0,8686	0,8294	0,8631	0,8996	0,8342	0,755	0,821	0,8997
	Bi-LSTM	**Ensemble d'entraînement**		**Ensemble de validation**		**Ensemble de test**			
		Accuracy	Loss	Val_ accuarcy	Val_loss	Accuracy	Rappel	F1-mesure	Précision
		0,9397	0,1640	0,8965	0,2557	0,8854	0,8663	0,8839	0,9022

5.1.1.2 Ensemble de données Davidson

Dans cette section, nous présentons les résultats de classification obtenus pour l'ensemble de données Davidson par les algorithmes d'apprentissage automatique utilisés. Le Tableau 5.2 montre les résultats obtenus par les algorithmes SVM, LR, NB et Bi-LSTM.

Tableau 5.2. Résultats de classifications avec le scénario #1 pour Davidson.

	Algorithmes utilisés	Ensemble d'entraînement				Ensemble de test			
		Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Scénario #1	SVM	0,9782	0,9968	0,9785	0,9609	0,7521	0,9927	0,7983	0,6676
	LR	0,9612	0,9266	0,9597	0,9953	0,9315	0,8896	0,9278	0,9694
	NB	0,9311	0,957	0,9327	0,9097	0,891	0,9237	0,8933	0,8649
	Bi-LSTM	**Ensemble d'entraînement**		**Ensemble de validation**		**Ensemble de test**			
		Accuracy	Loss	Val_ accuarcy	Val_loss	Accuracy	Rappel	F1-mesure	Précision
		0,9988	0,0042	0,9806	0,0888	0,9839	0,9832	0,9837	0,9841

5.1.1.3 Ensemble de données COVID-HATE

Dans cette section, nous présentons les résultats de classification obtenus pour l'ensemble de données COVID-HATE par les algorithmes d'apprentissage automatique utilisés. Le Tableau 5.3 montre les résultats obtenus par les algorithmes SVM, LR, NB et Bi-LSTM.

Tableau 5.3. Résultats de classifications avec le scénario #1 pour COVID-HATE.

	Algorithmes utilisés	Ensemble d'entraînement				Ensemble de test			
		Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Scénario #1	SVM	0,9387	0,9228	0,936	0,9498	0,7609	0,6098	0,7188	0,875
	LR	0,8821	0,957	0,8876	0,8278	0,7192	0,7538	0,7289	0,7057
	NB	0,8778	0,9502	0,8832	0,8253	0,7306	0,7992	0,7482	0,7033
	Bi-LSTM	**Ensemble d'entraînement**		**Ensemble de validation**		**Ensemble de test**			
		Accuracy	Loss	Val_ accuarcy	Val_loss	Accuracy	Rappel	F1-mesure	Précision
		0,9894	0,0384	0,9051	0,2491	0,9298	0,9242	0,9295	0,9349

5.1.1.4 Ensemble de données HatEval

Dans cette section, nous présentons les résultats de classification obtenus pour l'ensemble de données HatEval par les algorithmes d'apprentissage automatique utilisés. Le Tableau 5.4 montre les résultats obtenus par les algorithmes SVM, LR, NB et Bi-LSTM.

Tableau 5.4. Résultats de classifications avec le scénario #1 pour HatEval.

	Algorithmes utilisés	Ensemble d'entraînement				Ensemble de test			
		Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Scénario #1	SVM	0,7271	0,5749	0,6371	0,7147	0,6854	0,4758	0,569	0,7077
	LR	0,7149	0,5677	0,6241	0,6931	0,6677	0,4678	0,5514	0,6713
	NB	0,6991	0,4452	0,5523	0,7274	0,6762	0,43	0,5369	0,7145
	Bi-LSTM	**Ensemble d'entraînement**		**Ensemble de validation**		**Ensemble de test**			
		Accuracy	Loss	Val_ Accuarcy	Val_loss	Accuracy	Rappel	F1-mesure	Précision
		0,8543	0,352	0,7288	0,5651	0,7088	0,4846	0,5924	0,7618

5.1.2 Scénario #2

Nous dévoilons dans cette section, les résultats obtenus par les quatre algorithmes utilisés, SVM, LR, NB et Bi-LSTM, avec les quatre ensembles de données pour le scénario #2. Ce scénario regroupe les approches de 2 à 17 et 25, et il permet de vérifier l'impact de la suppression du bruit sur les résultats de la

classification, comme la suppression des mots vides, des nombres, des caractères spéciaux, des mots rares et des mots fréquents.

5.1.2.1 Ensemble de données JigsawToxic

Dans cette section, nous présentons les résultats de classification obtenus pour l'ensemble de données JigsawToxic par les algorithmes d'apprentissage automatique utilisés, SVM, LR, NB et Bi-LSTM.

Le Tableau 5.5 montre les résultats obtenus par l'algorithme SVM, en termes de métriques de performances (précision, rappel, accuracy et F1-mesure).

Tableau 5.5. Résultats de classifications de SVM avec le scénario #2 pour JigsawToxic.

Approches expérimentées	**Ensemble d'entraînement**				**Ensemble de test**			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,8695	0,8405	0,8654	0,8917	0,8271	0,7978	0,823	0,8498
Approche #3	0,8981	0,8755	0,8956	0,9166	0,8658	0,844	0,8637	0,8843
Approche #4	0,897	0,8726	0,8942	0,9171	0,8666	0,8428	0,8642	0,8867
Approche #5	0,9029	0,8865	0,9011	0,9163	0,869	0,8574	0,8683	0,8795
Approche #6	0,8962	0,8715	0,8934	0,9165	0,8666	0,8428	0,8642	0,8867
Approche #7	0,8965	0,8716	0,8936	0,9169	0,8681	0,8455	0,8659	0,8873
Approche #8	0,8698	0,8399	0,8656	0,893	0,826	0,7963	0,8218	0,849
Approche #9	0,8744	0,8514	0,8713	0,8921	0,8345	0,8119	0,8317	0,8526
Approche #10	0,8676	0,8367	0,8632	0,8915	0,8276	0,7978	0,8234	0,8506
Approche #11	0,8591	0,8314	0,8549	0,8797	0,8202	0,7914	0,816	0,8421
Approche #12	0,8796	0,8588	0,8769	0,8958	0,8438	0,8253	0,8418	0,859
Approche #13	0,8794	0,8588	0,8767	0,8954	0,843	0,8256	0,8412	0,8574
Approche #14	0,8677	0,8364	0,8632	0,8919	0,828	0,7984	0,8239	0,851
Approche #15	0,9031	0,8868	0,9013	0,9163	0,8701	0,8584	0,8694	0,8807
Approche #16	0,8682	0,8372	0,8638	0,8921	0,8268	0,7975	0,8227	0,8495
Approche #17	0,8677	0,8378	0,8634	0,8907	0,8279	0,7963	0,8233	0,8523
Approche #25	0,9007	0,8818	0,8986	0,9162	0,8695	0,855	0,8684	0,8823

Le Tableau 5.6 montre les résultats obtenus par l'algorithme LR.

Tableau 5.6. Résultats de classifications de LR avec le scénario #2 pour JigsawToxic.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,8693	0,841	0,8653	0,891	0,8267	0,7773	0,8188	0,8649
Approche #3	0,8965	0,8742	0,894	0,9147	0,8712	0,8345	0,8671	0,9024
Approche #4	0,8944	0,8718	0,8918	0,9127	0,8684	0,8275	0,8637	0,9032
Approche #5	0,9046	0,8845	0,9025	0,9213	0,8766	0,8474	0,8737	0,9017
Approche #6	0,8965	0,8726	0,8938	0,9161	0,8707	0,8348	0,8668	0,9013
Approche #7	0,8963	0,8732	0,8937	0,9152	0,8726	0,8363	0,8686	0,9035
Approche #8	0,8693	0,841	0,8653	0,8911	0,8285	0,7797	0,8208	0,8664
Approche #9	0,8751	0,8491	0,8716	0,8953	0,8391	0,8018	0,8339	0,8687
Approche #10	0,8675	0,8389	0,8634	0,8893	0,8288	0,7797	0,8211	0,867
Approche #11	0,856	0,8327	0,8524	0,8731	0,8119	0,7623	0,8032	0,8488
Approche #12	0,9007	0,8803	0,8985	0,9174	0,874	0,8406	0,8704	0,9025
Approche #13	0,8806	0,8575	0,8776	0,8986	0,8357	0,8009	0,8308	0,8632
Approche #14	0,8681	0,841	0,8642	0,8888	0,831	0,7813	0,8232	0,8699
Approche #15	0,905	0,8847	0,9029	0,9219	0,876	0,8477	0,8732	0,9003
Approche #16	0,8673	0,8398	0,8633	0,8882	0,8304	0,781	0,8226	0,869
Approche #17	0,8682	0,8404	0,8642	0,8894	0,8307	0,7828	0,8232	0,868
Approche #25	0,9012	0,8812	0,899	0,9175	0,8724	0,8382	0,8687	0,9016

Le Tableau 5.7 montre les résultats obtenus par l'algorithme NB.

Tableau 5.7. Résultats de classifications de NB avec le scénario #2 pour JigsawToxic.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,8702	0,8628	0,869	0,8754	0,8445	0,8204	0,8417	0,864
Approche #3	0,8822	0,8858	0,8825	0,8792	0,8652	0,8571	0,8649	0,8729
Approche #4	0,8809	0,8807	0,8807	0,8808	0,8633	0,8553	0,8631	0,871
Approche #5	0,8818	0,8758	0,8809	0,886	0,8624	0,8287	0,8585	0,8905
Approche #6	0,8812	0,8811	0,881	0,8808	0,8633	0,8553	0,8631	0,871
Approche #7	0,8819	0,8834	0,8819	0,8805	0,8639	0,8565	0,8638	0,8712
Approche #8	0,8702	0,8634	0,8691	0,8749	0,8439	0,8186	0,8408	0,8643
Approche #9	0,8715	0,8571	0,8694	0,8821	0,8464	0,796	0,8392	0,8874
Approche #10	0,8702	0,8608	0,8687	0,8769	0,8435	0,8177	0,8403	0,8642
Approche #11	0,8631	0,8529	0,8615	0,8702	0,8391	0,8119	0,8356	0,8608
Approche #12	0,882	0,878	0,8813	0,8847	0,8633	0,8305	0,8596	0,8907
Approche #13	0,8715	0,8567	0,8693	0,8824	0,8445	0,792	0,8369	0,8873
Approche #14	0,87	0,8608	0,8685	0,8765	0,8447	0,8189	0,8416	0,8655
Approche #15	0,8815	0,876	0,8806	0,8853	0,8622	0,8284	0,8583	0,8905
Approche #16	0,8697	0,8606	0,8683	0,8762	0,8442	0,8171	0,8409	0,8661
Approche #17	0,8698	0,8626	0,8686	0,8748	0,8465	0,8214	0,8435	0,867
Approche #25	0,8849	0,8806	0,8842	0,888	0,8673	0,8431	0,8649	0,8879

Le Tableau 5.8 montre les résultats obtenus, en termes de métriques de performances (précision, rappel, accuracy et F1-mesure), ainsi que la valeur de perte (loss) pour l'ensemble d'entraînement et celui de validation, par l'algorithme Bi-LSTM.

Tableau 5.8. Résultats de classifications de Bi-LSTM avec le scénario #2 pour JigsawToxic.

Approches expérimentées	Ensemble d'entraînement		Ensemble de validation		Ensemble de test			
	Accuracy	Loss	Val_accuarcy	Val_loss	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,9520	0,1296	0,8919	0,2588	0,8861	0,9052	0,8890	0,8734
Approche #3	0,9526	0,1280	0,9058	0,2258	0,9139	0,9263	0,9155	0,9050
Approche #4	0,9534	0,1276	0,9088	0,2168	0,9122	0,8981	0,9115	0,9253
Approche #5	0,9488	0,1354	0,9099	0,2138	0,9052	0,9336	0,9085	0,8846
Approche #6	0,9550	0,1240	0,9083	0,2174	0,9125	0,9232	0,9140	0,9049
Approche #7	0,9533	0,1223	0,9083	0,2213	0,9123	0,9260	0,9141	0,9025
Approche #8	0,9407	0,1545	0,8855	0,2600	0,8867	0,8718	0,8858	0,9002
Approche #9	0,9533	0,1248	0,9001	0,2468	0,8875	0,8645	0,8856	0,9078
Approche #10	0,9448	0,1481	0,8839	0,2645	0,8847	0,8923	0,8864	0,8805
Approche #11	0,9517	0,1294	0,8772	0,2875	0,8750	0,8737	0,8757	0,8777
Approche #12	0,9602	0,1084	0,9101	0,2270	0,9105	0,9220	0,9121	0,9024
Approche #13	0,9535	0,1234	0,8968	0,2513	0,8900	0,8813	0,8897	0,8983
Approche #14	0,9430	0,1518	0,8844	0,2642	0,8892	0,8786	0,8888	0,8992
Approche #15	0,9616	0,1086	0,9178	0,2060	0,9142	0,9003	0,9135	0,9272
Approche #16	0,9432	0,1509	0,8842	0,2675	0,8824	0,8835	0,8833	0,8832
Approche #17	0,9446	0,1442	0,8873	0,2640	0,8869	0,8608	0,8846	0,9098
Approche #25	0,9517	0,1270	0,9137	0,2052	0,9080	0,8874	0,9067	0,9268

5.1.2.2 Ensemble de données Davidson

Dans cette section, nous présentons les résultats de classification obtenus pour l'ensemble de données Davidson par les algorithmes d'apprentissage automatique utilisés, SVM, LR, NB et Bi-LSTM.

Le Tableau 5.9 montre les résultats des métriques de performances (précision, rappel, accuracy et F1-mesure), obtenus par l'algorithme SVM.

Tableau 5.9. Résultats de classifications de SVM avec le scénario #2 pour Davidson.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,9772	0,9954	0,9776	0,9604	0,7628	0,9918	0,8052	0,6777
Approche #3	0,9779	0,9946	0,9782	0,9624	0,8064	0,9799	0,8334	0,7251
Approche #4	0,9778	0,9941	0,978	0,9625	0,8035	0,9799	0,8314	0,722
Approche #5	0,9785	0,9946	0,9788	0,9634	0,8317	0,9765	0,8516	0,755
Approche #6	0,9784	0,994	0,9786	0,9637	0,8214	0,9759	0,8438	0,7433
Approche #7	0,9784	0,9922	0,9786	0,9654	0,884	0,9561	0,8907	0,8337
Approche #8	0,9778	0,9943	0,9781	0,9623	0,8308	0,982	0,8516	0,7518
Approche #9	0,9772	0,9961	0,9776	0,9597	0,7697	0,9857	0,8089	0,6858
Approche #10	0,9766	0,9951	0,9769	0,9595	0,7729	0,9847	0,8109	0,6892
Approche #11	0,9747	0,9953	0,9752	0,9559	0,7475	0,9896	0,7949	0,6642
Approche #12	0,9773	0,9934	0,9776	0,9623	0,8457	0,9616	0,8604	0,7785
Approche #13	0,9604	0,9357	0,9593	0,9842	0,9284	0,8969	0,9253	0,9555
Approche #14	0,9772	0,9954	0,9776	0,9604	0,7629	0,9918	0,8053	0,6779
Approche #15	0,9787	0,9926	0,9789	0,9656	0,895	0,9481	0,8993	0,8553
Approche #16	0,9773	0,9954	0,9776	0,9605	0,7631	0,9918	0,8054	0,678
Approche #17	0,9774	0,9954	0,9778	0,9607	0,7637	0,9918	0,8058	0,6786
Approche #25	0,9781	0,9926	0,9784	0,9646	0,8863	0,9588	0,8929	0,8355

Le Tableau 5.10 montre les résultats obtenus par l'algorithme LR.

Tableau 5.10. Résultats de classifications de LR avec le scénario #2 pour Davidson.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,9621	0,9282	0,9606	0,9954	0,9299	0,8914	0,9263	0,964
Approche #3	0,9761	0,9558	0,9755	0,9961	0,9597	0,9436	0,9586	0,9742
Approche #4	0,9763	0,9563	0,9757	0,996	0,96	0,9442	0,9589	0,9742
Approche #5	0,9767	0,9569	0,9761	0,9961	0,9602	0,9436	0,9591	0,9751
Approche #6	0,9767	0,9562	0,9761	0,9968	0,96	0,9448	0,959	0,9736
Approche #7	0,9749	0,9539	0,9742	0,9955	0,9603	0,9436	0,9592	0,9754
Approche #8	0,9633	0,9307	0,962	0,9954	0,9342	0,8981	0,9311	0,9665
Approche #9	0,9584	0,9248	0,9568	0,9912	0,895	0,8789	0,8922	0,906
Approche #10	0,9599	0,9296	0,9585	0,9894	0,9041	0,9149	0,9041	0,8936
Approche #11	0,9584	0,9328	0,9572	0,9829	0,8744	0,8993	0,8762	0,8542
Approche #12	0,9687	0,9443	0,9678	0,9924	0,9269	0,9164	0,9253	0,9344
Approche #13	0,9635	0,9319	0,9622	0,9945	0,9294	0,8914	0,9259	0,9631
Approche #14	0,9623	0,9288	0,9609	0,9953	0,9297	0,892	0,9262	0,9631
Approche #15	0,9746	0,9531	0,974	0,9958	0,9593	0,9408	0,9581	0,9759
Approche #16	0,9624	0,929	0,961	0,9954	0,9297	0,892	0,9262	0,9631
Approche #17	0,9619	0,9282	0,9605	0,9951	0,9303	0,8923	0,9268	0,9641
Approche #25	0,9734	0,9517	0,9727	0,9947	0,9596	0,9402	0,9583	0,9772

Le Tableau 5.11 montre les résultats obtenus par l'algorithme NB.

Tableau 5.11. Résultats de classifications de NB avec le scénario #2 pour Davidson.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,938	0,9521	0,9387	0,9257	0,8914	0,9195	0,8933	0,8686
Approche #3	0,9413	0,9381	0,9409	0,9438	0,8926	0,9091	0,8933	0,878
Approche #4	0,9415	0,9368	0,941	0,9454	0,8946	0,9082	0,8949	0,8821
Approche #5	0,9403	0,9422	0,9403	0,9383	0,8953	0,9149	0,8963	0,8784
Approche #6	0,942	0,937	0,9416	0,9462	0,8944	0,9082	0,8948	0,8818
Approche #7	0,9393	0,9331	0,9387	0,9444	0,8943	0,9131	0,8952	0,878
Approche #8	0,9452	0,9436	0,9449	0,9463	0,8919	0,9219	0,894	0,8676
Approche #9	0,9268	0,9434	0,9278	0,9127	0,8791	0,8938	0,8796	0,8658
Approche #10	0,927	0,9352	0,9274	0,9199	0,8697	0,8813	0,8699	0,8588
Approche #11	0,9142	0,9185	0,9144	0,9103	0,8525	0,8542	0,8513	0,8485
Approche #12	0,9281	0,9195	0,9273	0,9352	0,8784	0,8917	0,8788	0,8663
Approche #13	0,9345	0,9558	0,9357	0,9164	0,8958	0,9246	0,8977	0,8722
Approche #14	0,9379	0,952	0,9386	0,9256	0,8914	0,9189	0,8932	0,869
Approche #15	0,9374	0,9365	0,9372	0,9379	0,8929	0,9167	0,8943	0,873
Approche #16	0,9379	0,9521	0,9386	0,9255	0,8914	0,9189	0,8932	0,869
Approche #17	0,9384	0,952	0,9391	0,9265	0,8914	0,9198	0,8933	0,8684
Approche #25	0,9399	0,9348	0,9394	0,9441	0,897	0,9179	0,8981	0,8791

Le Tableau 5.12 montre les résultats des métriques de performances (précision, rappel, accuracy et F1-mesure), ainsi que la valeur de perte (loss) pour l'ensemble d'entraînement et celui de validation, obtenus par l'algorithme Bi-LSTM.

Tableau 5.12. Résultats de classifications de Bi-LSTM avec le scénario #2 pour Davidson.

Approches expérimentées	Ensemble d'entraînement		Ensemble de validation		Ensemble de test			
	Accuracy	loss	Val_accuarcy	Val_loss	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,9987	0,0053	0,9817	0,0809	0,9842	0,9808	0,9839	0,9871
Approche #3	0,9935	0,0204	0,9789	0,0736	0,9747	0,9631	0,9741	0,9853
Approche #4	0,9937	0,0214	0,9796	0,0734	0,9769	0,9646	0,9764	0,9884
Approche #5	0,9957	0,0135	0,9829	0,0671	0,9816	0,9728	0,9812	0,9898
Approche #6	0,9957	0,0137	0,9751	0,0751	0,9784	0,9777	0,9782	0,9786
Approche #7	0,9976	0,0083	0,9791	0,0717	0,9831	0,9793	0,9829	0,9865
Approche #8	0,9969	0,01	0,9761	0,0885	0,976	0,9701	0,9756	0,9812
Approche #9	0,9981	0,0084	0,9776	0,0928	0,9793	0,9689	0,9789	0,9891
Approche #10	0,9957	0,0171	0,9703	0,0874	0,9753	0,9671	0,9748	0,9826
Approche #11	0,9981	0,0069	0,9739	0,1138	0,9783	0,9783	0,978	0,9777
Approche #12	0,9981	0,007	0,9713	0,1071	0,9786	0,9677	0,9781	0,9888
Approche #13	0,9983	0,0058	0,9794	0,081	0,9828	0,9829	0,9826	0,9823
Approche #14	0,9959	0,015	0,9766	0,0819	0,9762	0,9728	0,9758	0,9788
Approche #15	0,9972	0,0088	0,9806	0,0735	0,9854	0,9817	0,9852	0,9886
Approche #16	0,9969	0,0109	0,9781	0,0793	0,9796	0,9762	0,9793	0,9825
Approche #17	0,9952	0,0176	0,9746	0,0828	0,9775	0,9774	0,9773	0,9771
Approche #25	0,9963	0,0121	0,9766	0,0724	0,9798	0,9719	0,9794	0,987

5.1.2.3 Ensemble de données COVID-HATE

Dans cette section, nous présentons les résultats de classification obtenus pour l'ensemble de données COVID-HATE par les algorithmes d'apprentissage automatique utilisés, SVM, LR, NB et Bi-LSTM.

Le Tableau 5.13 montre les résultats des métriques de performances (précision, rappel, accuracy et F1-mesure), pour l'algorithme SVM.

Tableau 5.13. Résultats de classifications de SVM avec le scénario #2 pour COVID-HATE.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,9282	0,9276	0,9263	0,9251	0,7989	0,7386	0,7863	0,8405
Approche #3	0,9373	0,9355	0,9355	0,9356	0,8197	0,7727	0,8111	0,8536
Approche #4	0,9354	0,9384	0,9338	0,9294	0,8027	0,7538	0,7928	0,8361
Approche #5	0,9377	0,9296	0,9355	0,9416	0,7989	0,7235	0,7828	0,8527
Approche #6	0,9344	0,9364	0,9328	0,9293	0,8083	0,75	0,7968	0,8498
Approche #7	0,9349	0,9365	0,9333	0,9302	0,7913	0,7235	0,7764	0,8377
Approche #8	0,9301	0,9316	0,9284	0,9254	0,8008	0,7235	0,7844	0,8565
Approche #9	0,9401	0,9326	0,938	0,9438	0,7856	0,6894	0,7631	0,8545
Approche #10	0,9344	0,9286	0,9323	0,936	0,7761	0,6553	0,7457	0,865
Approche #11	0,9282	0,9237	0,926	0,9284	0,7704	0,6477	0,7387	0,8593
Approche #12	0,9354	0,9218	0,9328	0,9441	0,7875	0,678	0,7617	0,8689
Approche #13	0,9349	0,9267	0,9326	0,9386	0,7761	0,6705	0,75	0,851
Approche #14	0,9278	0,9276	0,9259	0,9242	0,7989	0,7386	0,7863	0,8405
Approche #15	0,9368	0,9316	0,9349	0,9384	0,7856	0,6818	0,7611	0,8612
Approche #16	0,9278	0,9276	0,9259	0,9242	0,7989	0,7386	0,7863	0,8405
Approche #17	0,9268	0,9276	0,925	0,9224	0,7913	0,7273	0,7773	0,8348
Approche #25	0,9335	0,9316	0,9316	0,9318	0,7932	0,7045	0,7734	0,8571

Le Tableau 5.14 montre les résultats des métriques de performances (précision, rappel, accuracy et F1-mesure), pour l'algorithme LR.

Tableau 5.14. Résultats de classifications de LR avec le scénario #2 pour COVID-HATE.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,8731	0,9599	0,8803	0,8129	0,7249	0,7576	0,7339	0,7117
Approche #3	0,8964	0,9668	0,9008	0,8435	0,7799	0,7955	0,7836	0,7721
Approche #4	0,8926	0,9658	0,8974	0,8383	0,7628	0,7879	0,7689	0,7509
Approche #5	0,903	0,9599	0,906	0,8581	0,7704	0,7689	0,7704	0,7719
Approche #6	0,8931	0,9658	0,8978	0,839	0,7628	0,7879	0,7689	0,7509
Approche #7	0,8997	0,9599	0,903	0,8528	0,723	0,7576	0,7326	0,7092
Approche #8	0,8835	0,9628	0,8894	0,8265	0,7268	0,7386	0,7303	0,7222
Approche #9	0,885	0,9579	0,8902	0,8315	0,7419	0,7614	0,7472	0,7336
Approche #10	0,8802	0,956	0,8859	0,8256	0,7078	0,7045	0,7072	0,7099
Approche #11	0,8745	0,954	0,881	0,8187	0,6812	0,6402	0,668	0,6983
Approche #12	0,8988	0,957	0,9019	0,853	0,7552	0,7614	0,7571	0,7528
Approche #13	0,8802	0,9618	0,8865	0,8222	0,7343	0,7576	0,7407	0,7246
Approche #14	0,874	0,9599	0,8811	0,8143	0,7249	0,7576	0,7339	0,7117
Approche #15	0,9007	0,955	0,9034	0,8571	0,7666	0,7689	0,7675	0,766
Approche #16	0,8721	0,9599	0,8795	0,8116	0,7249	0,7576	0,7339	0,7117
Approche #17	0,8736	0,9589	0,8806	0,8142	0,7192	0,75	0,7279	0,7071
Approche #25	0,8997	0,9599	0,903	0,8527	0,7343	0,7689	0,7436	0,7199

Le Tableau 5.15 montre les résultats des métriques de performances (précision, rappel, accuracy et F1-mesure), obtenus par l'algorithme NB.

Tableau 5.15. Résultats de classifications de NB avec le scénario #2 pour COVID-HATE.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,8645	0,9541	0,8726	0,8041	0,7306	0,7992	0,7482	0,7033
Approche #3	0,8864	0,9629	0,8919	0,8311	0,7495	0,8106	0,7643	0,723
Approche #4	0,8764	0,956	0,8828	0,8203	0,7362	0,822	0,7574	0,7023
Approche #5	0,8874	0,9541	0,8918	0,8375	0,7514	0,822	0,7681	0,7209
Approche #6	0,8741	0,956	0,8808	0,8169	0,7306	0,822	0,7535	0,6955
Approche #7	0,8645	0,9492	0,8721	0,8069	0,7192	0,8106	0,7431	0,6859
Approche #8	0,865	0,957	0,8733	0,8032	0,7457	0,8295	0,7657	0,711
Approche #9	0,8793	0,9541	0,8848	0,8251	0,7419	0,8182	0,7606	0,7105
Approche #10	0,8679	0,956	0,8756	0,8077	0,7343	0,8144	0,7544	0,7026
Approche #11	0,8655	0,9482	0,8727	0,8085	0,7192	0,7955	0,7394	0,6908
Approche #12	0,8731	0,9463	0,8789	0,8207	0,7476	0,822	0,7654	0,7162
Approche #13	0,8764	0,9521	0,8822	0,822	0,7419	0,822	0,7614	0,7092
Approche #14	0,8645	0,9541	0,8726	0,8041	0,7381	0,8258	0,7596	0,7032
Approche #15	0,8741	0,9453	0,8795	0,8225	0,7306	0,8068	0,75	0,7007
Approche #16	0,865	0,9541	0,873	0,8047	0,7381	0,8258	0,7596	0,7032
Approche #17	0,8631	0,9511	0,8711	0,8036	0,7362	0,8333	0,7599	0,6984
Approche #25	0,8741	0,9492	0,8799	0,8203	0,7211	0,8144	0,7452	0,6869

Le Tableau 5.16 montre les résultats des métriques de performances (précision, rappel, accuracy et F1-mesure), ainsi que la valeur de perte (loss) pour l'ensemble d'entraînement et celui de validation, pour l'algorithme Bi-LSTM.

Tableau 5.16. Résultats de classifications de Bi-LSTM avec le scénario #2 pour COVID-HATE.

Approches expérimentées	Ensemble d'entraînement		Ensemble de validation		Ensemble de test			
	Accuracy	loss	Val_accuarcy	Val_loss	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,9989	0,0091	0,9146	0,3007	0,9298	0,9356	0,9303	0,9251
Approche #3	0,9927	0,0299	0,9114	0,2406	0,9146	0,9394	0,9168	0,8953
Approche #4	0,9911	0,0367	0,9272	0,2248	0,9222	0,9205	0,9222	0,924
Approche #5	0,9793	0,0816	0,8892	0,2893	0,8843	0,875	0,8834	0,8919
Approche #6	0,9922	0,0383	0,9114	0,2689	0,9013	0,9242	0,9037	0,8841
Approche #7	0,9715	0,113	0,8797	0,2982	0,8729	0,8712	0,8729	0,8745
Approche #8	0,9894	0,0629	0,9335	0,2244	0,9127	0,875	0,9094	0,9467
Approche #9	0,9855	0,0424	0,9146	0,2575	0,9184	0,8864	0,9159	0,9474
Approche #10	0,9972	0,0154	0,9209	0,2819	0,9165	0,9356	0,9182	0,9015
Approche #11	0,9972	0,0108	0,9367	0,2479	0,9127	0,9205	0,9135	0,9067
Approche #12	0,9709	0,0972	0,8829	0,303	0,8861	0,9318	0,8913	0,8542
Approche #13	0,9983	0,0136	0,9051	0,248	0,9222	0,947	0,9242	0,9025
Approche #14	0,9978	0,0155	0,9177	0,2709	0,9222	0,9583	0,925	0,894
Approche #15	0,976	0,0904	0,8861	0,2684	0,8861	0,9167	0,8897	0,8643
Approche #16	0,9815	0,0832	0,8987	0,2732	0,8956	0,9205	0,8983	0,8773
Approche #17	0,9994	0,0074	0,9399	0,2038	0,9488	0,9394	0,9484	0,9575
Approche #25	0,9553	0,1346	0,8671	0,3404	0,8539	0,9356	0,8651	0,8046

5.1.2.4 Ensemble de données HatEval

Dans cette section, nous présentons les résultats de classification obtenus pour l'ensemble de données HatEval, par les algorithmes d'apprentissage automatique SVM, LR, NB et Bi-LSTM.

Le Tableau 5.17 montre les résultats des métriques de performances (précision, rappel, accuracy et F1-mesure), pour l'algorithme SVM.

Tableau 5.17. Résultats de classifications de SVM avec le scénario #2 pour HatEval.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,7269	0,5758	0,6373	0,7137	0,7069	0,5471	0,6198	0,7146
Approche #3	0,7361	0,6293	0,6652	0,7055	0,7158	0,6141	0,6535	0,6984
Approche #4	0,7363	0,6284	0,665	0,7063	0,7262	0,6211	0,6645	0,7143
Approche #5	0,745	0,6415	0,677	0,7169	0,7173	0,6053	0,6515	0,7053
Approche #6	0,7362	0,6249	0,6637	0,7077	0,7285	0,6123	0,6632	0,7232
Approche #7	0,7255	0,5887	0,6411	0,7042	0,7062	0,541	0,6165	0,7165
Approche #8	0,7253	0,5739	0,6351	0,7113	0,7046	0,5304	0,6105	0,7192
Approche #9	0,7326	0,5947	0,6495	0,7156	0,6996	0,5498	0,6151	0,698
Approche #10	0,7271	0,5755	0,6374	0,7143	0,7069	0,5471	0,6198	0,7146
Approche #11	0,7222	0,555	0,6247	0,7147	0,6919	0,4934	0,583	0,7125
Approche #12	0,7383	0,6108	0,6604	0,719	0,7023	0,5454	0,6153	0,7058
Approche #13	0,7346	0,5986	0,6527	0,7178	0,7	0,5392	0,6108	0,7043
Approche #14	0,7261	0,5749	0,6362	0,7124	0,7069	0,5471	0,6198	0,7146
Approche #15	0,7381	0,6106	0,6601	0,7186	0,7027	0,5383	0,6125	0,7105
Approche #16	0,7266	0,5749	0,6367	0,7136	0,7069	0,5471	0,6198	0,7146
Approche #17	0,7271	0,5751	0,6372	0,7145	0,7081	0,5445	0,6195	0,7186
Approche #25	0,7364	0,6127	0,6595	0,7144	0,6985	0,526	0,6036	0,7082

Le Tableau 5.18 montre les résultats des métriques de performances (précision, rappel, accuracy et F1-mesure), pour l'algorithme LR.

Tableau 5.18. Résultats de classifications de LR avec le scénario #2 pour HatEval.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,719	0,5779	0,6317	0,6967	0,6888	0,5507	0,6071	0,6764
Approche #3	0,7397	0,6378	0,6713	0,7086	0,7027	0,585	0,6321	0,6874
Approche #4	0,7373	0,6367	0,6689	0,7047	0,7015	0,5912	0,6336	0,6826
Approche #5	0,7418	0,6408	0,6742	0,7113	0,6965	0,5859	0,6277	0,6758
Approche #6	0,7365	0,6353	0,6678	0,7039	0,7015	0,5912	0,6336	0,6826
Approche #7	0,7254	0,5901	0,6417	0,7034	0,6912	0,5498	0,6085	0,6812
Approche #8	0,721	0,5783	0,6334	0,7003	0,6862	0,5383	0,5996	0,6766
Approche #9	0,7266	0,6007	0,6468	0,7008	0,6758	0,5392	0,5922	0,6567
Approche #10	0,7171	0,5749	0,6289	0,6942	0,6888	0,5507	0,6071	0,6764
Approche #11	0,7133	0,5486	0,6146	0,6989	0,6685	0,4943	0,5655	0,6608
Approche #12	0,7371	0,618	0,6622	0,7132	0,6823	0,548	0,601	0,6652
Approche #13	0,726	0,5958	0,6444	0,7018	0,6746	0,5357	0,5897	0,6559
Approche #14	0,7199	0,5797	0,6331	0,6974	0,6888	0,5507	0,6071	0,6764
Approche #15	0,7369	0,6185	0,6622	0,7126	0,6804	0,5401	0,596	0,6649
Approche #16	0,7186	0,5772	0,631	0,696	0,6892	0,5515	0,6078	0,6768
Approche #17	0,7181	0,5742	0,6294	0,6964	0,6881	0,5498	0,6061	0,6753
Approche #25	0,7325	0,6148	0,6571	0,7058	0,6835	0,548	0,6018	0,6674

Le Tableau 5.19 montre les résultats des métriques de performances (précision, rappel, accuracy et F1-mesure), pour l'algorithme NB.

Tableau 5.19. Résultats de classifications de NB avec le scénario #2 pour HatEval.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,707	0,5043	0,5893	0,7089	0,6908	0,5648	0,6146	0,674
Approche #3	0,7092	0,4849	0,5816	0,7266	0,7065	0,5278	0,6109	0,7252
Approche #4	0,7098	0,4997	0,5894	0,7187	0,7062	0,5551	0,6225	0,7087
Approche #5	0,704	0,436	0,551	0,7489	0,6988	0,4581	0,5705	0,7558
Approche #6	0,7107	0,5073	0,5937	0,7161	0,7035	0,563	0,6237	0,6991
Approche #7	0,705	0,4994	0,5852	0,7067	0,6965	0,5471	0,6115	0,6931
Approche #8	0,7058	0,4932	0,5829	0,7125	0,6904	0,5471	0,6067	0,6809
Approche #9	0,7059	0,4738	0,5732	0,7254	0,6846	0,4907	0,576	0,6971
Approche #10	0,7062	0,5038	0,5884	0,7074	0,6908	0,5648	0,6146	0,674
Approche #11	0,6987	0,4881	0,5746	0,6983	0,6815	0,5445	0,5988	0,6652
Approche #12	0,7032	0,4533	0,5599	0,7324	0,6931	0,4767	0,5755	0,7262
Approche #13	0,7013	0,4471	0,555	0,7318	0,6835	0,4476	0,5525	0,7216
Approche #14	0,7071	0,5052	0,5898	0,7085	0,6908	0,5648	0,6146	0,674
Approche #15	0,7004	0,4358	0,5477	0,7378	0,6877	0,4458	0,5548	0,7344
Approche #16	0,7072	0,5087	0,5916	0,7069	0,6908	0,5648	0,6146	0,674
Approche #17	0,7065	0,5084	0,5909	0,7054	0,6915	0,5639	0,6148	0,6758
Approche #25	0,7022	0,4669	0,5663	0,7199	0,6888	0,4872	0,5775	0,709

Le Tableau 5.20 montre les résultats des métriques de performances (précision, rappel, accuracy et F1-mesure), ainsi que la valeur de perte (loss) pour l'ensemble d'entraînement et celui de validation, obtenus par l'algorithme Bi-LSTM.

Tableau 5.20. Résultats de classifications de Bi-LSTM avec le scénario #2 pour HatEval.

Approches expérimentées	Ensemble d'entraînement		Ensemble de validation		Ensemble de test			
	Accuracy	loss	Val_accuarcy	Val_loss	Accuracy	Rappel	F1-mesure	Précision
Approche #2	0,8627	0,3586	0,741	0,5332	0,7462	0,7498	0,7206	0,6936
Approche #3	0,8118	0,442	0,7494	0,5232	0,7423	0,6088	0,6735	0,7535
Approche #4	0,8	0,4559	0,7359	0,536	0,7404	0,7568	0,7179	0,6828
Approche #5	0,7974	0,4538	0,734	0,5169	0,7285	0,5877	0,6539	0,737
Approche #6	0,8614	0,3289	0,7282	0,5456	0,7435	0,7568	0,7203	0,6872
Approche #7	0,852	0,3627	0,7333	0,5365	0,7385	0,6018	0,6676	0,7497
Approche #8	0,791	0,4846	0,7333	0,5432	0,7204	0,5419	0,6285	0,7482
Approche #9	0,8594	0,3777	0,7301	0,5493	0,7408	0,763	0,7199	0,6814
Approche #10	0,8083	0,4573	0,741	0,5351	0,7342	0,5965	0,6621	0,744
Approche #11	0,7619	0,5042	0,7141	0,5634	0,6892	0,4115	0,5362	0,7694
Approche #12	0,8545	0,3838	0,7397	0,5298	0,7496	0,7436	0,7217	0,701
Approche #13	0,8671	0,3421	0,75	0,5344	0,7385	0,63	0,6777	0,7333
Approche #14	0,7566	0,5406	0,6929	0,6259	0,7015	0,8379	0,7102	0,6163
Approche #15	0,8474	0,3771	0,7301	0,562	0,7446	0,7286	0,7135	0,6991
Approche #16	0,8633	0,3497	0,7256	0,5662	0,7269	0,7604	0,7085	0,6633
Approche #17	0,7967	0,4639	0,7359	0,5371	0,7223	0,5551	0,6357	0,7438
Approche #25	0,8581	0,3514	0,7487	0,5369	0,7381	0,7137	0,704	0,6947

5.1.3 Scénario #3

Nous dévoilons dans cette section, les résultats obtenus par les quatre algorithmes utilisés, SVM, LR, NB et Bi-LSTM, avec les quatre ensembles de données pour le scénario #3. Ce dernier englobe les approches de 18 à 24 et 26 et se focalise sur la correction orthographique et la normalisation du texte en utilisant la lemmatisation et la racinisation.

5.1.3.1 Ensemble de données JigsawToxic

Dans cette section, nous présentons les résultats de classification obtenus pour l'ensemble de données JigsawToxic par les algorithmes d'apprentissage automatique utilisés. Les tableaux 5.21, 5.22, 5.23 et 5.24 montrent respectivement les résultats des métriques de performances pour les algorithmes SVM, LR, NB et Bi-LSTM.

Tableau 5.21. Résultats de classifications de SVM avec le scénario #3 pour JigsawToxic.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,896	0,8767	0,8937	0,9114	0,8667	0,8455	0,8647	0,8848
Approche #19	0,8968	0,8785	0,8947	0,9115	0,8653	0,8425	0,8631	0,8847
Approche #20	0,8841	0,8656	0,8817	0,8984	0,8482	0,8339	0,847	0,8605
Approche #21	0,9023	0,8875	0,9007	0,9142	0,8737	0,8633	0,8731	0,8833
Approche #22	0,9045	0,8927	0,9032	0,914	0,8743	0,8639	0,8738	0,8839
Approche #23	0,9031	0,8893	0,9016	0,9142	0,8695	0,8565	0,8686	0,8811
Approche #24	0,9052	0,8913	0,9037	0,9164	0,8803	0,8666	0,8794	0,8926
Approche #26	0,9008	0,8833	0,8989	0,9151	0,8775	0,8608	0,8762	0,8922

Tableau 5.22. Résultats de classifications de LR avec le scénario #3 pour JigsawToxic.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,8955	0,8737	0,893	0,9132	0,8733	0,8422	0,8701	0,9
Approche #19	0,8963	0,8731	0,8937	0,9153	0,8755	0,8428	0,8721	0,9036
Approche #20	0,883	0,8605	0,8801	0,9006	0,8507	0,8229	0,8474	0,8734
Approche #21	0,905	0,8881	0,9032	0,9188	0,8807	0,8587	0,8788	0,9
Approche #22	0,9043	0,8889	0,9026	0,9168	0,8832	0,8623	0,8815	0,9015
Approche #23	0,904	0,8873	0,9022	0,9176	0,8781	0,8535	0,8758	0,8994
Approche #24	0,9046	0,8882	0,9029	0,9181	0,8869	0,8666	0,8853	0,9048
Approche #26	0,8995	0,8809	0,8974	0,9146	0,8827	0,8578	0,8805	0,9045

Tableau 5.23. Résultats de classifications de NB avec le scénario #3 pour JigsawToxic.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,8826	0,8717	0,8811	0,8907	0,8638	0,8229	0,8589	0,8982
Approche #19	0,8826	0,8716	0,881	0,8908	0,863	0,8226	0,8581	0,8969
Approche #20	0,8721	0,8526	0,8694	0,8869	0,8476	0,7926	0,8397	0,8928
Approche #21	0,8782	0,8633	0,8761	0,8893	0,8586	0,8113	0,8525	0,8981
Approche #22	0,8773	0,86	0,8749	0,8905	0,8595	0,8103	0,8531	0,9007
Approche #23	0,8769	0,8605	0,8747	0,8893	0,8595	0,8119	0,8534	0,8994
Approche #24	0,8801	0,8635	0,8779	0,8928	0,8639	0,8223	0,8589	0,899
Approche #26	0,8817	0,8713	0,8803	0,8895	0,87	0,8455	0,8675	0,8908

Tableau 5.24. Résultats de classifications de Bi-LSTM avec le scénario #3 pour JigsawToxic.

Approches expérimentées	Ensemble d'entraînement & de validation				Ensemble de test			
	Accuracy	loss	Val_accuarcy	Val_loss	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,9580	0,1182	0,9070	0,2209	0,9046	0,8942	0,9043	0,9146
Approche #19	0,9423	0,1549	0,9086	0,2203	0,9032	0,8951	0,9031	0,9112
Approche #20	0,9484	0,1376	0,9034	0,2389	0,8943	0,8697	0,8923	0,9162
Approche #21	0,9415	0,1537	0,9117	0,2125	0,9055	0,8966	0,9053	0,9142
Approche #22	0,9413	0,1548	0,9137	0,2086	0,9085	0,8914	0,9075	0,9242
Approche #23	0,9550	0,1239	0,9165	0,2141	0,9079	0,9159	0,9092	0,9026
Approche #24	0,9584	0,1197	0,9181	0,2094	0,9159	0,9119	0,9161	0,9203
Approche #26	0,9482	0,1350	0,9194	0,2117	0,9163	0,9073	0,9161	0,9251

5.1.3.2 Ensemble de données Davidson

Dans cette section, nous présentons les résultats de classification pour l'ensemble de données Davidson pour les algorithmes d'apprentissage automatique utilisés. Les tableaux 5.25, 5.26, 5.27 et 5.28 montrent respectivement les résultats obtenus par les algorithmes SVM, LR, NB et Bi-LSTM.

Tableau 5.25. Résultats de classifications de SVM avec le scénario #3 pour Davidson.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,9781	0,9951	0,9784	0,9623	0,8311	0,9793	0,8515	0,7532
Approche #19	0,9783	0,9954	0,9786	0,9624	0,8305	0,979	0,851	0,7526
Approche #20	0,9781	0,9958	0,9784	0,9617	0,7836	0,9881	0,8187	0,6988
Approche #21	0,9799	0,9906	0,9801	0,9698	0,8911	0,9497	0,8961	0,8482
Approche #22	0,9802	0,9902	0,9804	0,9707	0,8937	0,9494	0,8983	0,8524
Approche #23	0,98	0,9905	0,9801	0,97	0,8911	0,9487	0,896	0,8488
Approche #24	0,9796	0,9917	0,9798	0,9682	0,8978	0,9442	0,9013	0,8621
Approche #26	0,9788	0,9938	0,9791	0,9648	0,8634	0,9701	0,8753	0,7974

Tableau 5.26. Résultats de classifications de LR avec le scénario #3 pour Davidson.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,9694	0,9428	0,9685	0,9955	0,9477	0,9207	0,9456	0,972
Approche #19	0,9691	0,9432	0,9682	0,9946	0,9481	0,921	0,9461	0,9726
Approche #20	0,9642	0,9322	0,9629	0,9957	0,9327	0,8978	0,9296	0,9637
Approche #21	0,9752	0,9534	0,9746	0,9967	0,9569	0,9426	0,9558	0,9693
Approche #22	0,9756	0,9542	0,975	0,9966	0,9579	0,9436	0,9568	0,9705
Approche #23	0,9759	0,9542	0,9753	0,9974	0,9558	0,942	0,9547	0,9677
Approche #24	0,975	0,9527	0,9744	0,997	0,957	0,9414	0,9559	0,9707
Approche #26	0,9731	0,949	0,9724	0,9969	0,9549	0,9372	0,9536	0,9706

Tableau 5.27. Résultats de classifications de NB avec le scénario #3 pour Davidson.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,9405	0,9466	0,9407	0,9349	0,8914	0,9134	0,8927	0,8729
Approche #19	0,9409	0,9471	0,9411	0,9352	0,8919	0,9137	0,8931	0,8734
Approche #20	0,9344	0,9509	0,9353	0,9201	0,8928	0,9195	0,8945	0,8708
Approche #21	0,936	0,9356	0,9358	0,936	0,8929	0,9158	0,8943	0,8737
Approche #22	0,9344	0,9327	0,9341	0,9356	0,8884	0,9073	0,8894	0,8721
Approche #23	0,9346	0,9365	0,9345	0,9326	0,8857	0,9121	0,8875	0,8642
Approche #24	0,9335	0,9284	0,933	0,9376	0,8858	0,9051	0,8869	0,8693
Approche #26	0,9366	0,9254	0,9357	0,9463	0,8883	0,8993	0,8884	0,8776

Tableau 5.28. Résultats de classifications de Bi-LSTM avec le scénario #3 pour Davidson.

Approches expérimentées	Ensemble d'entraînement & de validation				Ensemble de test			
	Accuracy	Loss	Val_accuarcy	Val_loss	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,9952	0,0173	0,9791	0,0853	0,9793	0,9707	0,9789	0,9873
Approche #19	0,9984	0,0049	0,9829	0,0726	0,9816	0,9759	0,9813	0,9867
Approche #20	0,9981	0,0064	0,9806	0,082	0,9811	0,9735	0,9808	0,9882
Approche #21	0,9977	0,0083	0,9829	0,0585	0,9815	0,978	0,9812	0,9843
Approche #22	0,9992	0,0031	0,9844	0,0644	0,9833	0,9765	0,983	0,9895
Approche #23	0,9984	0,0049	0,9801	0,072	0,9807	0,971	0,9803	0,9897
Approche #24	0,9978	0,0074	0,9842	0,0634	0,9813	0,9728	0,9809	0,9891
Approche #26	0,9946	0,02	0,9736	0,087	0,9741	0,9756	0,9738	0,972

5.1.3.3 Ensemble de données COVID-HATE

Dans cette section, nous présentons les résultats de classification obtenus pour l'ensemble de données COVID-HATE par les algorithmes d'apprentissage automatique utilisés. Les tableaux 5.29, 5.30, 5.31 et 5.32 montrent respectivement les résultats obtenus par les algorithmes SVM, LR, NB et Bi-LSTM.

Tableau 5.29. Résultats de classifications de SVM avec le scénario #3 pour COVID-HATE.

Approches expérimentées	**Ensemble d'entraînement**				**Ensemble de test**			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,9335	0,9276	0,9313	0,9351	0,778	0,6667	0,7505	0,8585
Approche #19	0,9335	0,9296	0,9315	0,9335	0,7723	0,6667	0,7458	0,8462
Approche #20	0,9016	0,9306	0,9019	0,875	0,7666	0,7614	0,7657	0,7701
Approche #21	0,9415	0,9453	0,9403	0,9355	0,8121	0,7992	0,81	0,821
Approche #22	0,9392	0,9404	0,9377	0,9351	0,8102	0,7955	0,8077	0,8203
Approche #23	0,9387	0,9384	0,937	0,9359	0,8046	0,7765	0,7992	0,8233
Approche #24	0,9335	0,9345	0,9318	0,9293	0,7989	0,7424	0,7871	0,8376
Approche #26	0,9363	0,9345	0,9345	0,9345	0,8197	0,7614	0,8089	0,8627

Tableau 5.30. Résultats de classifications de LR avec le scénario #3 pour COVID-HATE.

Approches expérimentées	**Ensemble d'entraînement**				**Ensemble de test**			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,8874	0,9589	0,8924	0,8351	0,7457	0,7689	0,7519	0,7355
Approche #19	0,8921	0,9589	0,8964	0,8421	0,723	0,7538	0,7316	0,7107
Approche #20	0,8878	0,9667	0,8935	0,8307	0,7552	0,7803	0,7616	0,7437
Approche #21	0,8969	0,9599	0,9006	0,8486	0,7913	0,7955	0,7925	0,7895
Approche #22	0,8978	0,9619	0,9016	0,8488	0,7913	0,7992	0,7932	0,7873
Approche #23	0,8954	0,9609	0,8995	0,8459	0,7894	0,8144	0,7948	0,7762
Approche #24	0,904	0,957	0,9067	0,8618	0,7704	0,7841	0,7738	0,7638
Approche #26	0,9016	0,9589	0,9045	0,856	0,7609	0,7841	0,7667	0,75

Tableau 5.31. Résultats de classifications de NB avec le scénario #3 pour COVID-HATE.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,8812	0,9599	0,8871	0,8249	0,7324	0,8182	0,7539	0,699
Approche #19	0,8831	0,9609	0,8889	0,8271	0,7268	0,8144	0,7491	0,6935
Approche #20	0,8836	0,9531	0,8884	0,8319	0,7419	0,8182	0,7606	0,7105
Approche #21	0,8926	0,9541	0,8962	0,845	0,7609	0,8258	0,7758	0,7315
Approche #22	0,894	0,9492	0,897	0,8502	0,7704	0,8295	0,7835	0,7424
Approche #23	0,8859	0,9482	0,8899	0,8383	0,7704	0,8371	0,7851	0,7391
Approche #24	0,8902	0,9492	0,8937	0,8444	0,7609	0,8409	0,7789	0,7255
Approche #26	0,8869	0,9511	0,891	0,8383	0,7381	0,8182	0,7579	0,7059

Tableau 5.32. Résultats de classifications de Bi-LSTM avec le scénario #3 pour COVID-HATE.

Approches expérimentées	Ensemble d'entraînement & de validation				Ensemble de test			
	Accuracy	loss	Val_accuarcy	Val_loss	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,9978	0,0247	0,8956	0,2608	0,9184	0,9583	0,9217	0,8877
Approche #19	0,9966	0,0166	0,9209	0,2691	0,9336	0,928	0,9333	0,9387
Approche #20	0,9983	0,0155	0,9177	0,2589	0,9374	0,9508	0,9383	0,9262
Approche #21	0,9927	0,0338	0,9177	0,241	0,9203	0,9394	0,9219	0,9051
Approche #22	0,9938	0,0333	0,9304	0,2064	0,9241	0,9773	0,9281	0,8836
Approche #23	0,9916	0,04	0,8924	0,2554	0,9184	0,9545	0,9214	0,8905
Approche #24	0,9972	0,0151	0,9114	0,2209	0,9089	0,9318	0,9111	0,8913
Approche #26	0,9938	0,0198	0,9367	0,1615	0,9241	0,9129	0,9234	0,9341

5.1.3.4 Ensemble de données HatEval

Dans cette section, nous présentons les résultats de classification obtenus pour l'ensemble de données HatEval par les algorithmes d'apprentissage automatique utilisés. Les tableaux 5.33, 5.34, 5.35 et 5.36 montrent respectivement les résultats obtenus par les algorithmes SVM, LR, NB et Bi-LSTM.

Tableau 5.33. Résultats de classifications de SVM avec le scénario #3 pour HatEval.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,7388	0,6161	0,6628	0,7172	0,7015	0,5559	0,6192	0,6988
Approche #19	0,7383	0,6175	0,6629	0,7156	0,6988	0,5551	0,6167	0,6938
Approche #20	0,7368	0,6083	0,6583	0,7174	0,7031	0,5577	0,6212	0,701
Approche #21	0,7469	0,6498	0,6815	0,7165	0,7173	0,6396	0,6639	0,6901
Approche #22	0,7501	0,6531	0,6853	0,721	0,7158	0,6361	0,6615	0,6889
Approche #23	0,7478	0,648	0,6817	0,7191	0,7142	0,6361	0,6603	0,6863
Approche #24	0,7501	0,6597	0,6875	0,7179	0,6854	0,4758	0,569	0,7077
Approche #26	0,7456	0,6524	0,6813	0,713	0,72	0,6132	0,6566	0,7066

Tableau 5.34. Résultats de classifications de LR avec le scénario #3 pour HatEval.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,7312	0,6171	0,6568	0,7025	0,6896	0,5498	0,6073	0,6783
Approche #19	0,7321	0,6196	0,6585	0,7032	0,6885	0,5489	0,606	0,6764
Approche #20	0,7309	0,6138	0,6553	0,703	0,6788	0,5498	0,5991	0,6582
Approche #21	0,7473	0,6602	0,6853	0,7124	0,7038	0,6088	0,6422	0,6794
Approche #22	0,7467	0,6609	0,6851	0,7111	0,6996	0,6035	0,6369	0,6742
Approche #23	0,7422	0,6531	0,6786	0,7063	0,6981	0,6053	0,6364	0,6709
Approche #24	0,718	0,6715	0,6649	0,6586	0,6931	0,6511	0,6494	0,6477
Approche #26	0,738	0,6471	0,673	0,7013	0,6973	0,6	0,6338	0,6716

Tableau 5.35. Résultats de classifications de NB avec le scénario #3 pour HatEval.

Approches expérimentées	Ensemble d'entraînement				Ensemble de test			
	Accuracy	Rappel	F1-mesure	Précision	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,6951	0,4095	0,528	0,7439	0,6858	0,4467	0,5538	0,7284
Approche #19	0,6945	0,4072	0,5262	0,744	0,6865	0,4449	0,5534	0,7319
Approche #20	0,6958	0,4161	0,5327	0,7403	0,685	0,4537	0,5571	0,7213
Approche #21	0,6926	0,4009	0,5208	0,7434	0,6896	0,4502	0,5588	0,7363
Approche #22	0,691	0,3882	0,5114	0,7495	0,6858	0,4467	0,5538	0,7284
Approche #23	0,6909	0,3885	0,5116	0,749	0,6927	0,4485	0,5603	0,7463
Approche #24	0,696	0,4101	0,5292	0,7459	0,6827	0,4485	0,5524	0,7189
Approche #26	0,7017	0,4415	0,5523	0,7376	0,6938	0,4907	0,5832	0,7187

Tableau 5.36. Résultats de classifications de Bi-LSTM avec le scénario #3 pour HatEval.

Approches expérimentées	Ensemble d'entraînement & de validation				Ensemble de test			
	Accuracy	Loss	Val_accuarcy	Val_loss	Accuracy	Rappel	F1-mesure	Précision
Approche #18	0,8586	0,3643	0,7385	0,5352	0,7438	0,704	0,7058	0,7077
Approche #19	0,8381	0,3816	0,7327	0,545	0,7346	0,5524	0,6451	0,775
Approche #20	0,819	0,4061	0,7314	0,5387	0,7054	0,4476	0,5701	0,7852
Approche #21	0,8493	0,3624	0,7385	0,5218	0,7481	0,7225	0,7146	0,7069
Approche #22	0,8537	0,353	0,7474	0,5048	0,7496	0,7198	0,7151	0,7104
Approche #23	0,7857	0,4807	0,7346	0,5463	0,7281	0,7489	0,7063	0,6682
Approche #24	0,8136	0,4129	0,716	0,5599	0,7123	0,4599	0,5826	0,7945
Approche #26	0,8618	0,3369	0,7378	0,5229	0,7408	0,6546	0,688	0,7249

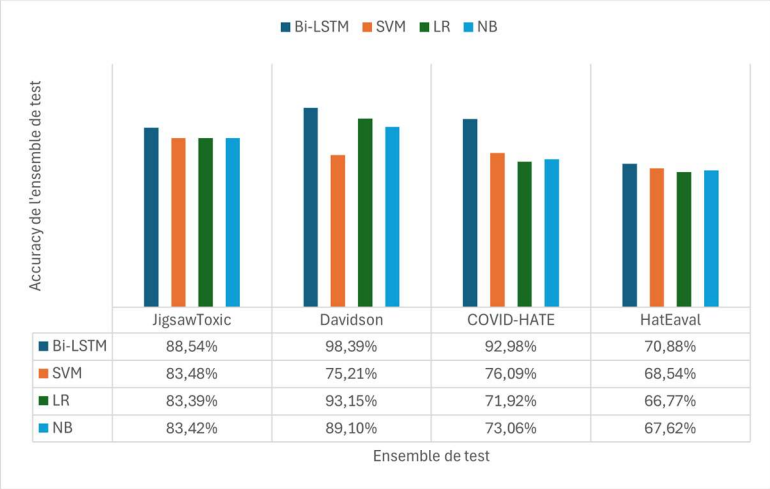
5.2 Discussion

Dans cette section, nous discutons et interprétons les résultats obtenus par les différents algorithmes et les différents scénarios expérimentés.

5.2.1 Comparaison des algorithmes utilisés

Comme prévu dans la littérature, les réseaux de neurones récurrents (RNN) fonctionnent bien dans la classification du contenu haineux. Selon les résultats obtenus de l'accuracy, du rappel, du F1-score et de la précision, le Bi-LSTM est plus performent avec tous les scénarios expérimentés, pour les quatre ensembles de données utilisés. Il donne des résultats meilleurs que ceux des autres algorithmes. Si nous prenons comme exemple, l'ensemble de données JigsawToxic, la meilleure valeur obtenue d'accuracy par l'algorithme SVM est de 83,48% avec le scénario #1. Dans ce scénario, nous avons passé au classificateur le texte brut après tokenisation et suppression des espaces. Pour le même scénario, l'algorithme LR donne 83,39% comme accuracy et NB donne 67,62%, tandis que le Bi-LSTM donne le meilleur résultat qui est de 88,54%. Si nous faisons la même comparaison pour chaque ensemble de données, nous constatons que le Bi-LSTM donne toujours de bons résultats avec tous les scénarios. La Figure 5.1 donne un aperçu des différents résultats obtenus avec le scénario #1.

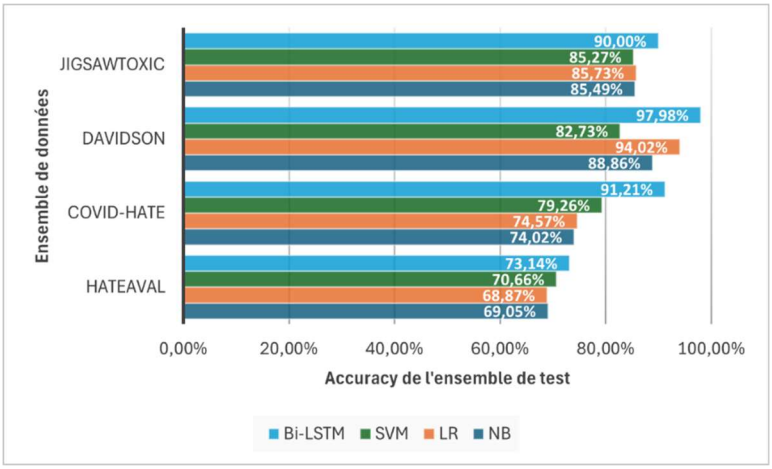
Figure 5.1. Accuarcy du scénario #1 de tous les algorithmes et ensembles de données.



	JigsawToxic	Davidson	COVID-HATE	HatEaval
Bi-LSTM	88,54%	98,39%	92,98%	70,88%
SVM	83,48%	75,21%	76,09%	68,54%
LR	83,39%	93,15%	71,92%	66,77%
NB	83,42%	89,10%	73,06%	67,62%

Pour mieux visualiser les performances des modèles utilisés, nous avons calculé l'accuracy moyenne obtenu par chaque algorithme sur l'ensemble des scénarios expérimentés. Pour calculer la moyenne des résultats obtenus pour l'accuracy par chaque algorithme, nous avons fait la somme de ces résultats et divisé cette somme par le nombre d'approches expérimentées (26 dans notre cas). Nous avons fait ce calcul pour les quatre ensembles de données. Les résultats obtenus montrent que le Bi-LSTM performe bien avec tous les ensembles de données. Il s'avère le meilleur et offre de bons résultats. Les valeurs moyennes d'accuracy obtenues par ce dernier sont les meilleures par rapport à celles obtenues par les autres algorithmes de classification. La Figure 5.2 qui donne un aperçu sur l'ensembles des valeurs obtenues par les algorithmes utilisés pour chaque ensemble de données.

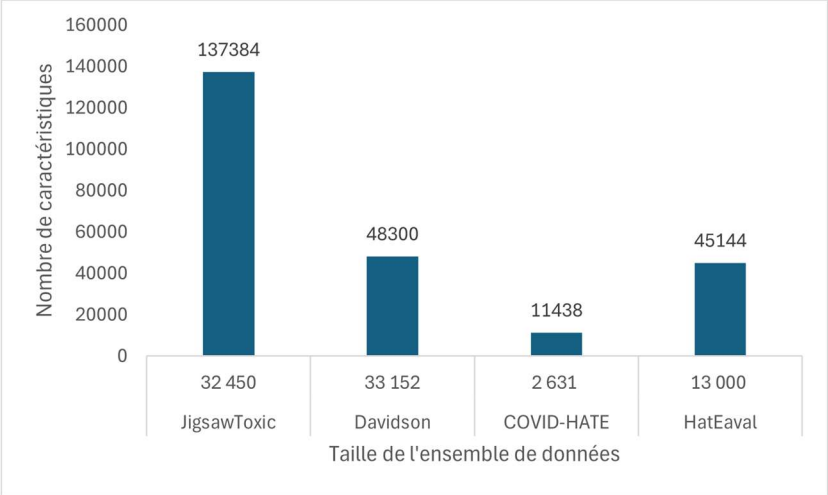
Figure 5.2. Accuracy moyenne des scénarios expérimentés pour chaque algorithme.



Comme nous pouvons le constater à partir de la Figure 5.2, la meilleure accuracy moyenne obtenue pour l'ensemble de données JigsawToxic est celle du Bi-LSTM qui est de 90%, suivie par celle de 85,73% qui est obtenue par LR. SVM a donné une accuracy moyenne de 85,27%, mais il était légèrement dépassé par NB qui a obtenu 85,49%. Le Bi-LSTM présente aussi des meilleures performances avec l'ensemble de données Davidson avec une accuracy moyenne de 97,98%, suivi de LR qui a donné 94,02%. SVM a donné une accuracy moyenne de 82,73%, mais il fonctionnait moins bien que NB qui a donné une valeur de 88,86%. Concernant l'ensemble de données COVID-HATE, SVM a obtenu un score de 79,26% mais ses performances étaient inférieures à celles de Bi-LSTM qui a donné 91,21% comme accuracy moyenne. NB a donné 74,02% pour le même ensemble de données et LR le dépassait légèrement avec un score de 74,57%. Les performances du Bi-LSTM ont visiblement diminué avec l'ensemble de données HatEval mais il surpasse toujours les autres algorithmes de classification avec une accuracy moyenne de 73,14%. SVM était légèrement meilleur que LR et NB et il a obtenu un score de 70,66% face à 68,87% et 69,05% qu'ont obtenu, respectivement LR et NB.

Les algorithmes SVM, LR et NB fonctionnent bien avec les ensembles de données utilisés, mais ils ont obtenu de meilleurs résultats avec les ensembles de données JigsawToxic, Davidson et HatEval qu'avec l'ensemble de données COVID-HATE. Cela peut être dû à la taille de ce dernier qui contient moins d'échantillons que les autres ensembles de données. Il contient 2 631 échantillons par rapport à 32 450, 33 152 et 13 000 échantillons que contiennent respectivement JigsawToxic, Davidson et HatEval. La taille d'un ensemble de données peut influencer sur les performances du classificateur qui dépendent du nombre de caractéristiques (attributs). Un nombre trop petit ou trop grand de caractéristiques peut causer la dégradation des performances du classificateur (Trivedi, Sharma, Soni, & Nair, 2015). La Figure 5.3 donne un aperçu sur le nombre des caractéristiques par vecteur de données sans pré-traitement de texte, pour les ensembles de données utilisés. Durant la phase d'entraînement, nous ne prenons en compte que des principales caractéristiques maximales classées par fréquence des termes dans le corpus. Le nombre maximal de caractéristiques utilisées est fixé à 20 000.

Figure 5.3. Nombre de caractéristiques par vecteur de données sans prétraitement de texte.



Le nombre de caractéristiques de l'ensemble de données Davidson, dont la taille est de 33 152 échantillons, est de 48 300 caractéristiques. Tandis que le nombre total de caractéristiques de l'ensemble de données COVID-HATE, dont la taille est de 2 631 échantillons, est de 11 438 caractéristiques. Nous pouvons constater que le nombre de caractéristiques dépend de la taille de l'ensemble de données et il diminue lorsque la taille de ce dernier est petite. Ce qui explique la dégradation des performances des algorithmes avec cet ensemble de données.

Le temps d'exécution aussi dépend de la taille de l'ensemble de données. Plus la taille est grande, plus que le temps nécessaire à l'exécution augmentera (Liang, Sun, Yunlei, & Gao, 2017). Le Tableau 5.4 donne un aperçu sur le temps d'exécution (en secondes) des algorithmes, pour chaque ensemble de données avec le scénario #1. Nous pouvons constater que NB présente le temps d'exécution minimal. Ce dernier est connu pour sa rapidité et ses bonnes performances dans la classification de texte (Dawar, Kumar, Negi, Pathan, & Layek, 2023).

Tableau 5.37. Temps d'exécution (en secondes) et accuracy moyenne des algorithmes pour chaque ensemble de données avec le scénario #1.

Algorithmes		NB	LR	SVM	Bi-LSTM
HatEaval	Accuracy moyenne	69,05%	68,87%	70,66%	73,14%
	Temps d'exécution	2,34	4,69	81,19	526,79
COVID-HATE	Accuracy moyenne	74,02%	74,57%	79,26%	91,21%
	Temps d'exécution	0,99	98,24	5,20	214,21
Davidson	Accuracy moyenne	88,86%	94,02%	82,73%	97,98%
	Temps d'exécution	2,97	603,87	959,01	1704,89
JigsawToxic	Accuracy moyenne	85,49%	85,73%	85,27%	90,00%
	Temps d'exécution	8,48	1560,67	622,63	2009,40

De point de vue accuracy, le Bi-LSTM fonctionne bien et présente les meilleures performances avec un temps d'exécution très long par rapport aux autres algorithmes utilisés. De point de vue temps d'exécution, NB s'avère le meilleur avec son temps d'exécution minimal. Même si ce dernier est moins performant que le Bi-LSTM, il donne de relatifs bons résultats dans la classification.

5.2.2 Comparaison des scénarios expérimentés

Dans cette section, nous comparons d'abord, les résultats obtenus lors des différents scénarios. Par la suite, nous prenons les résultats obtenus par la meilleure approche de chaque scénario et nous faisons une comparaison pour identifier les meilleures techniques de prétraitement dans la classification de texte.

5.2.2.1 Scénario #1

Dans ce scénario, nous avons passé au classificateur le texte brut après tokenisation et suppression des espaces. Selon les résultats obtenus et comme le montre la Figure 5.1 de la section 5.2.1, le Bi-LSTM fonctionne bien avec les quatre ensembles de données et donne les meilleurs résultats. Pour les deux ensembles de données JigsawToxic et COVID-HATE, SVM vient en deuxième position après le Bi-LSTM, suivie de NB et LR. Les résultats pour ces deux ensembles de données sont compris entre 92,98% et 71,92%. Pour l'ensemble de données Davidson, LR dépasse NB, qui à son tour dépasse SVM. Les résultats sont compris entre 98,39% et 75,21%. L'ensemble de données HatEval a obtenu des résultats moins bons que les autres ensembles de données, pour les quatre algorithmes. Ces résultats sont compris entre 70,88% et 66,77%, obtenues respectivement par le Bi-LSTM et LR. SVM dépasse légèrement NB, qui à son tour dépasse LR.

5.2.2.2 Scénario #2

Ce scénario englobe les approches 2 à 17, avec l'approche 25. Il permet de vérifier l'impact de la suppression du bruit sur les résultats de la classification, comme la suppression des mots vides, des nombres, des caractères spéciaux, des mots rares et des mots fréquents. Ce qui diffère une approche de l'autre est parfois la nature des techniques utilisées, d'autres fois l'agencement de ces techniques pour faire le prétraitement. Le Tableau 5.38 donne un aperçu des différentes techniques utilisées dans les approches expérimentées par ce scénario. Les chiffres dans le tableau (ex : bloc 1, bloc 2, etc.) présentent les agencements de ces techniques dans le pipeline de prétraitement. Dans toutes les approches, le texte subit au début une conversion en lettres minuscules et une tokenisation avec suppression des espaces avant de le faire passer au classificateur.

Tableau 5.38. Agencements des techniques de prétraitement dans les approches du scénario #2.

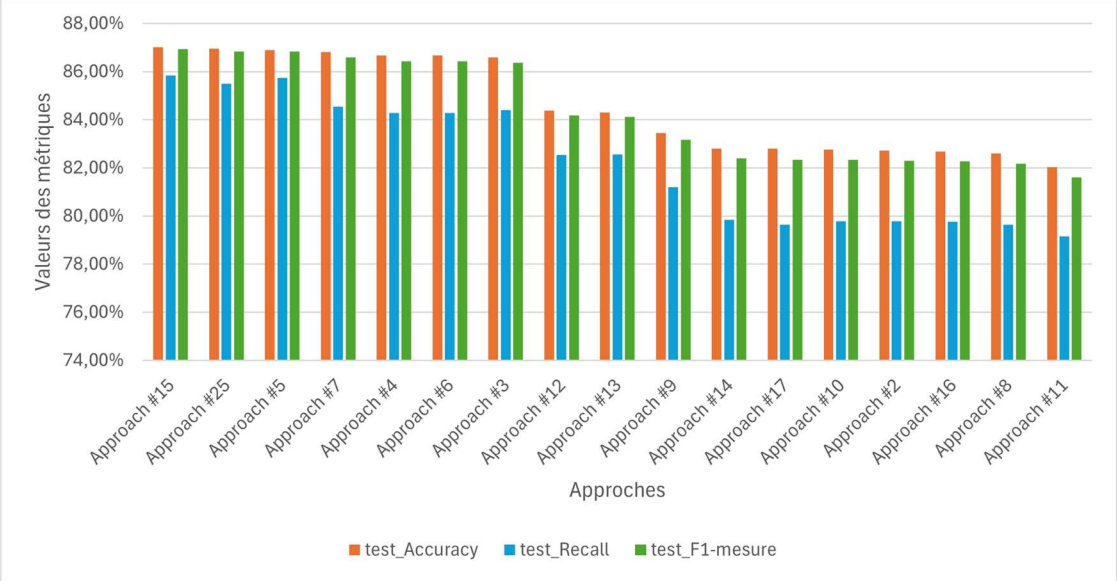
Approches	Conversion en lettres minuscules	Suppression de							Tokenisation (Rejet des espaces)
		Nombres et caractères spéciaux	Mentions utilisateurs, hashtags, URLs et balises HTML	Horodatages des messages	Stop words	Mots fréquents	Mots rares	Émojis et émoticônes	
Approche #2	Bloc 1				Bloc 2				Bloc 3
Approche #3	Bloc 1	Bloc 3			Bloc 2				Bloc 4
Approche #4	Bloc 1	Bloc 2			Bloc 3				Bloc 4
Approche #5	Bloc 1	Bloc 2							Bloc 3
Approche #6	Bloc 1	Bloc 2	Bloc 3	Bloc 4	Bloc 5				Bloc 6
Approche #7	Bloc 1	Bloc 4	Bloc 2	Bloc 3	Bloc 5				Bloc 6
Approche #8	Bloc 1		Bloc 2	Bloc 3	Bloc 4				Bloc 5
Approche #9	Bloc 1					Bloc 2			Bloc 3
Approche #10	Bloc 1				Bloc 3	Bloc 2			Bloc 4
Approche #11	Bloc 1				Bloc 2	Bloc 3			Bloc 4
Approche #12	Bloc 1		Bloc 3	Bloc 4	Bloc 5	Bloc 2			Bloc 6
Approche #13	Bloc 1						Bloc 2		Bloc 3
Approche #14	Bloc 1	Bloc 5	Bloc 3	Bloc 4			Bloc 2		Bloc 6
Approche #15	Bloc 1	Bloc 4	Bloc 2	Bloc 3			Bloc 5		Bloc 6
Approche #16	Bloc 1				Bloc 3		Bloc 2		Bloc 4
Approche #17	Bloc 1				Bloc 3			Bloc 2	Bloc 4
Approche #25	Bloc 1	Bloc 4	Bloc 2	Bloc 3			Bloc 5		Bloc 6 *

* Suppression des espaces et des jetons à deux caractères et moins

Nous prenons l'exemple des résultats obtenus pour l'ensemble de données JigsawToxic par l'algorithme SVM. La Figure 5.4 donne un aperçu sur les résultats obtenus pour les métriques accuracy, rappel et F1-mesure. Les approches sont triées selon les résultats obtenus d'accuracy, de la plus grande valeur à la plus petite. Nous constatons que la suppression à la fois, des émojis, des émoticônes et des mots vides donne une accuracy de 82,79% qui est meilleure que celle obtenue lorsque nous supprimons juste les mots vides (82,71%). Cette amélioration des performances est observée avec la majorité des algorithmes et des ensembles de données utilisées. La suppression des nombres et des caractères spéciaux après la suppression des mentions utilisateurs, des hashtags, des URLs, des balises HTML et des horodatages peut dans certains cas dégrader les performances du modèle. C'est le cas de l'ensemble de données COVID-HATE qui a subi une diminution de 1,7% et de l'ensemble de données HatEval qui a subi à son tour une diminution de 2,23% avec l'algorithme SVM.

L'approche #5 consistant à supprimer les nombres et les caractères spéciaux donne une accuarcy de 86,9% qui est meilleure que celle de 82,71%, obtenue avec l'approche #2, lorsque nous supprimons les mots vides. La plupart des algorithmes utilisés fonctionnent mieux avec la première approche qu'avec la deuxième. Dans certain cas, la combinaison de ces deux techniques permet d'obtenir des résultats meilleurs que lorsque chacune d'elles est utilisée seule. L'agencement de ces deux techniques n'influence pas beaucoup les résultats. L'ensemble de données COVID-HATE obtient 79,89% avec SVM lorsque l'une de ces techniques est utilisée seule, 80,27% lorsque les mots vides sont supprimés avant les nombres et les caractères spéciaux et 81,97% lorsque les mots vides sont supprimés après. L'ensemble de données HatEval obtient 71,73% avec SVM et après la suppression des nombres et caractères spéciaux, 70,69% avec la suppression des mots vides, 71,58% lorsque les mots vides sont supprimés avant les nombres et les caractères spéciaux et 72,62% lorsque les mots vides sont supprimés après.

Figure 5.4. Résultats obtenus par SVM pour l'ensemble de données JigsawToxic avec le scénario #2.



Commencer par la suppression des mots fréquents, par la suite supprimer les mentions utilisateurs, les hashtags, les URLs, les balises HTML et les horodatages et finir par supprimer les mots vides donne de meilleurs résultats dans la majorité des cas expérimentés, par rapport aux autres approches où les mots fréquents sont supprimés seuls ou avec les mots vides. Dans tous les cas expérimentés, la suppression des mots fréquents seuls ou combinée avec la suppression des mots vides donne des bons résultats. Cependant, l'agencement de ces deux techniques est important. Si nous supprimons les mots vides en premier, nous apercevons une dégradation des performances du modèle. Prenons l'exemple de

l'ensemble de données Davidson, il obtient 89,5% avec l'algorithme LR lorsque la technique de la suppression des mots fréquents est utilisée seule et 90,41% lorsqu'elle est suivie par la suppression des mots vides. L'accuracy obtenu devient 87,44%, si ces derniers sont supprimés en premier. L'ensemble de données JigsawToxic obtient respectivement 84,64% avec l'algorithme NB lorsque la technique de la suppression des mots fréquents est utilisée seule et 84,35% lorsqu'elle est combinée avec la suppression des mots vides. L'accuracy obtenu devient 83,91%, si ces derniers sont supprimés en premier.

Prenons l'exemple du commentaire suivant de l'ensemble de données COVID-HATE : « *@NaiobiT Ye cus the coronavirus hates all black ppl so it just goin past them like \"Hi bitch... Bye bitch\" Whoever said that...* ». Les dix mots les plus fréquents dans ce commentaire sont : [*('i', 4), ('you', 4), ('bitch', 2), ('and', 2), ('youre', 1), ('such', 1), ('an', 1), ('ugly', 1), ('fucking', 1), ('take', 1)*]. Après avoir supprimé les mots vides, nous obtenons ce qui suit [*('back', 1), ('words', 1), ('said', 1), ('wish', 1), ('slapped', 1), ('stood', 1), ('infront', 1), ('blahing', 1), ('eyes', 1), ('pussy', 1)*]. En supprimant les cinq mots les plus fréquents, le commentaire devient comme suit : *« such an ugly fucking take back no words said to wish slapped when were stood infront of me blahing your eyes out pussy* ». Si nous procédons à la suppression des mots vides d'abord et par la suite les mots fréquents, nous obtenons le résultat suivant : « *gly fucking take back words said wish slapped stood infront blahing eyes pussy* ». Nous remarquons que dans les deux cas, le terme « *bit**** », qui est un terme haineux (insulte), a été supprimé. Nous avons perdu un terme significatif et très important dans la classification. Comme mentionné dans la section 4.2.1.1, et à partir des deux figures 4.2 et 4.3, la plupart des mots fréquents dans le texte peuvent être soit des mots vides, des signes de ponctuation ou des termes haineux et des insultes. Cela montre qu'il faut éviter de supprimer les mots fréquents dans le texte et de se contenter de supprimer les mots vides, les nombres et les caractères spéciaux, pour ne pas perdre des termes qui sont pertinents dans la tâche de classification.

La combinaison des deux techniques de la suppression des mots rares et la suppression des mots vides n'améliore pas les performances du modèle dans la majorité des cas expérimentés et donne des résultats moins bons que lorsque la première est utilisée seule. L'ensemble de données COVID-HATE obtient avec l'algorithme LR, 73,43% si nous supprimons juste les mots rares et 72,49% si nous supprimons les mots vides aussi. La suppression d'abord des mentions utilisateurs, des hashtags, des URLs, des balises HTML, des horodatages des nombres, des caractères spéciaux et par la suite les mots rares donne de bons résultats que dans le cas inverse où les mots rares sont supprimés en premier. Le même ensemble de données obtient 76,66% dans le premier cas et 72,49% dans le deuxième. Comme mentionné dans la

section 4.2.1.1, et à partir des deux figures 4.4 et 4.5, la plupart des mots rares sont soit des noms d'utilisateurs, des hashtags, des liens URL ou des mots mal formés (mots contenant des signes de ponctuation ou des fautes). Avant de procéder à la suppression des mots rares, il est préférable de supprimer les signes de ponctuation d'abord. Cela permet d'éviter de supprimer des mots qui sont importants dans la classification de texte, comme le cas du mot « *flu….fu**** » dans le commentaire suivant de l'ensemble de données COVID-HATE :

> *Carona virus is more contagious than the flu..other than that it\'s the fucking flu..fuck carina virus..and fuck china... I\'m living my life..if my rights get infringed upon...I\'ll see them in court...live free...#America..#freedom...#fuckchina*

Si nous supprimons les signes de ponctuation d'abord, nous obtiendrons : « *flu fu**** » et nous aurons plus de chance que ce terme informatif ne sera pas considéré comme un mot rare et ne sera pas supprimé par la suite. Nous pouvons remarquer aussi que le terme « *#fu***china* » est un hashtag raciste qui pourra être utile dans la classification. C'est pourquoi il est préférable de se contenter de supprimer juste les signes # et garder les termes des hashtags. La suppression des jetons qui contiennent deux caractères et moins améliore les performances du modèle dans la plupart des cas expérimentés, mais dans d'autres cas, elle cause leur dégradation. Il est préférable alors, de procéder à la correction orthographique de ces termes avant de les supprimer.

La meilleure approche dans ce scénario avec l'ensemble de données JigsawToxic, en utilisant les deux algorithmes SVM et Bi-LSTM, est l'approche #15 qui consiste à convertir d'abord le texte en miniscule, supprimer les mentions utilisateurs, les hashtags, les URLs, les balises HTML, les horodatages, les nombres, les caractères spéciaux et par la suite les mots rares et faire la tokenisation (suppression des espaces). Cette approche représente la deuxième meilleure approche en utilisant l'algorithme LR avec le même ensemble de données. Elle est aussi parmi les trois meilleures approches pour les autres ensembles de données. L'ensemble de données JigsawToxic contient 3 221 commentaires neutres et 3269 commentaires haineux. Le SVM a mal classé 463 commentaires haineux et 380 commentaires neutres. Tandis que le Bi-LSTM a mal classé 326 haineux et 231 commentaires neutres. Le LR et le NB ont mal classé respectivement 498 commentaires haineux, 307 commentaires neutres, 561 commentaires haineux et 333 commentaires neutres. Les valeurs d'accuracy de cette approche varient entre 91,42% et 86,22% avec l'ensemble de données Jigsawtoxic, entre 98,54% et 89,29% avec Davidson, entre 88,61% et 73,06% avec COVID-HATE et entre 74,46% et 68,04% avec HatEval. Le rappel varie entre 90,03% et 82,84% avec

l'ensemble de données Jigsawtoxic, entre 98,17% et 91,67% avec Davidson, entre 91,67% et 68,18% avec COVID-HATE et entre 72,86% et 44,58% avec HatEval. Et finalement la F1-mesure varie entre 91,35% et 85,83% avec l'ensemble de données Jigsawtoxic, entre 98,52% et 89,43% avec Davidson, entre 88,97% et 75% avec COVID-HATE et entre 71,35% et 55,48% avec HatEval.

5.2.2.3 Scénario #3

Ce scénario englobe les approches de 18 à 24 et 26 et se focalise sur la correction orthographique et la normalisation du texte en utilisant la lemmatisation et la racinisation. La différence entre ces approches, dans ce scénario, est parfois la nature des techniques utilisées, d'autres fois l'agencement de ces techniques pour faire le prétraitement. Le Tableau 5.39 donne un aperçu des différentes techniques utilisées dans les approches expérimentées de ce scénario. Les chiffres dans le tableau (ex : bloc 1, bloc 2, etc.) présentent les agencements de ces techniques dans le pipeline de prétraitement. Dans toutes les approches expérimentées, le texte subit d'abord une conversion en lettres minuscules et une tokenisation avec suppression des espaces avant de passer le texte au classificateur.

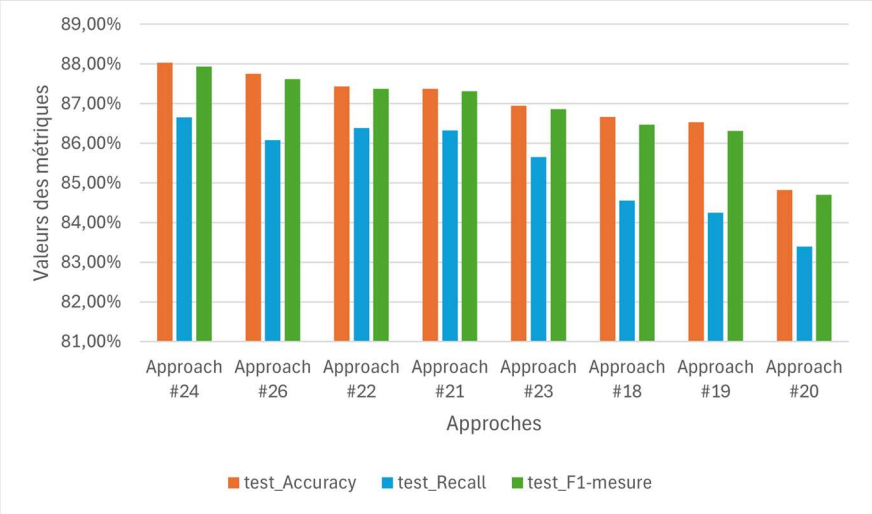
Tableau 5.39. Agencements des techniques de prétraitement dans les approches du scénario #3.

Approches	Conversion en lettres minuscules	Gestion de contractions	Gestion des accents	Correction orthographique	Lemmatisation	Stemming	Tokenisation (Rejet des espaces)
Approche #18	Bloc 1	Bloc 2	Bloc 3	Bloc 4			Bloc 5
Approche #19	Bloc 1	Bloc 3	Bloc 4	Bloc 2			Bloc 5
Approche #20	Bloc 1					Bloc 2	Bloc 3
Approche #21	Bloc 1				Bloc 2		Bloc 3
Approche #22	Bloc 1				Bloc 2	Bloc 3	Bloc 4
Approche #23	Bloc 1				Bloc 3	Bloc 2	Bloc 4
Approche #24	Bloc 1	Bloc 2	Bloc 3	Bloc 4	Bloc 5	Bloc 6	Bloc 7
Approche #26	Bloc 1	Bloc 2	Bloc 3	Bloc 4	Bloc 5	Bloc 6	Bloc 7*

* Suppression des espaces et des jetons à deux caractères et moins

Nous prenons l'exemple des résultats obtenus pour l'ensemble de données JigsawToxic par l'algorithme SVM. La Figure 5.5 donne un aperçu des résultats obtenus pour les métriques accuracy, rappel et F1-mesure. Les approches sont triées selon les résultats obtenus d'accuracy, de la plus grande valeur à la plus petite.

Figure 5.5. Résultats obtenus par SVM pour l'ensemble de données JigsawToxic avec le scénario #3.



Nous constatons que dans la majorité des cas expérimentés, procéder à la gestion des contractions d'abord, ensuite gérer les accents et enfin faire la correction orthographique, donne de meilleurs résultats que lorsque nous commençons par faire la correction orthographique. L'ensemble de données JigsawToxic donne avec cette approche (approche #18) en utilisant l'algorithme SVM, une accuracy de 86,67% qui est meilleure que celle obtenue lorsque nous commençons par la correction orthographique est qui est de 84,25%. Prenons l'exemple suivant de commentaire de l'ensemble de données Davidson : « *Ya'll take note. There ain't never tornadoes where colored folk live we r nt stupid pathtic kids u r u fat shite* ». Après avoir appliqué la correction orthographique, le commentaire devient : « *ya'll take note there ain't never tornadoes where colored folk live we r it stupid pathetic kids u r u fat shite* ». Nous remarquons qu'il y a des termes sous forme de contractions qui ne sont pas corrigés avec la correction orthographique comme « *ain't* » et « *u* ». En revanche, si nous faisons la gestion des contractions d'abords et par la suite nous procédons à la correction orthographique, nous obtenons ce qui suit : « *ya'll take note there are not never tornadoes where colored folk live we r it stupid pathetic kids you r you fat shite* ». Nous remarquons que le termes « *ain't* » et « *u* » deviennent respectivement « *are not* » et « *you* ». Cela montre que la combinaison de ces deux techniques permet de corriger plus de termes dans le texte.

Les résultats obtenus lorsque nous faisons la lemmatisation suivie du stemming sont meilleurs que ceux obtenus lorsque chacune de ces techniques est utilisée seule, où lorsque la lemmatisation est faite après le stemming. L'ensemble de données JigsawToxic donne une accuracy de 87,43% dans le premier cas avec SVM. Cependant, il obtient 86,95% lorsque le stemming est fait en premier. La lemmatisation et le

stemming utilisés seuls donnent respectivement 87,37% et 84,82% d'accuracy. Certes, la lemmatisation fonctionne bien avec la classification de texte et améliore les résultats quand elle est combinée avec d'autres techniques et surtout avec le stemming, mais il faut appliquer la lemmatisation et par la suite le stemming, parce la première tient compte du contexte. Prenons un exemple de l'ensemble de données COVID-HATE : « *At this point niggas are certified ass waters LMFAOOOO he prolly the one that started that coronavirus* 😂😂». La lemmatisation donne le résultat suivant : « *at thi point niggas be certifi as water lmfaoooo he prolli the one that start that coronaviru* 😂😂». Tandis que le stemming donne : « *at thi point nigga be certifi as water lmfaoooo he prolli the one that start that coronaviru* 😂😂». Le terme : « *niggas* » est un mot en pluriel utilisé dans les discours africain-américains. Avec le stemming le mot est converti au singulier, tandis que la lemmatisation a gardé le mot en pluriel.

L'approche #24 consistant à gérer les contractions d'abord, ensuite les accents et après faire la correction orthographique suivie de la lemmatisation et finir par le stemming donne les meilleurs résultats avec la plupart des algorithmes utilisés, pour les deux ensembles de données JigsawToxic et Davidson. En revanche, ce n'est pas le cas pour les deux ensembles de données COVID-HATE et HatEval. Les meilleurs résultats obtenus pour ces derniers sont ceux obtenus par l'approche #26. Cette approche ressemble à l'approche #24, elle utilise toutes les techniques de cette dernière mais elle permet en plus, de supprimer les termes de deux caractères et moins. Cette approche donne de bons résultats avec la plupart des ensembles de données. Les valeurs d'accuracy obtenues pour l'ensemble de données JigsawToxic par SVM et Bi-LSTM en utilisant cette approche sont respectivement 87,75% et 91,63%, et ceux du rappel sont 86,08% et 90,73%. L'ensemble de données COVID-HATE donne avec SVM une accuracy de 81,97% et un rappel de 76,14%. Si nous prenons l'exemple suivant de commentaire de l'ensemble de données Davidson : « *Ya'll take note. There ain't never tornadoes where colored folk live we r nt stupid pathtic kids u r u fat shite* ». Après avoir appliqué l'approche #24, le commentaire devient : « *ya 'll take note . there be not never tornado where color folk live we r nt stupid pathtic kid you r you fat shite* ». L'Approche #26 donne ce qui suit : « *'ll take note there not never tornado where color folk live stupid pathtic kid you you fat shite* ». Nous pouvons constater qu'avec la deuxième approche, tous les termes de deux caractères et moins sont supprimés. Cela permet de nettoyer le texte du bruit.

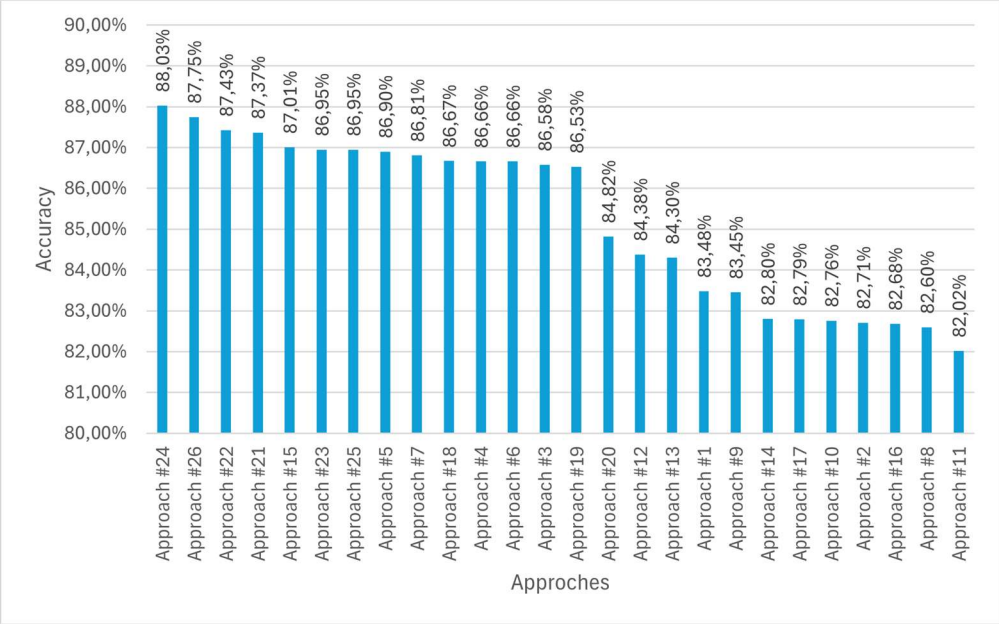
La meilleure approche identifiée dans ce scénario est l'approche #24, suivie de l'approche #26. La première consiste à convertir le texte d'abord en lettres minuscules, gérer les contractions et les accents, faire la correction orthographique, la lemmatisation suivie du stemming et par la suite la tokenisation. La

deuxième approche consiste à convertir le texte d'abord en lettres minuscules, gérer les contractions, ensuite les accents et après faire la correction orthographique suivie de la lemmatisation et du stemming et finir par la tokenisation en supprimant les espaces et les termes de deux caractères et moins. L'ensemble de données JigsawToxic contient 3 221 commentaires neutres et 3269 commentaires haineux. En utilisant l'approche #24, SVM a mal classé 436 commentaires haineux et 341 commentaires neutres. Tandis que Bi-LSTM a mal classé 288 haineux et 258 commentaires neutres. LR et NB ont mal classé respectivement 436 commentaires haineux, 298 commentaires neutres, 581 commentaires haineux et 302 commentaires neutres. En utilisant l'approche #26, SVM a mal classé 455 commentaires haineux et 340 commentaires neutres. Pour sa part, Bi-LSTM a mal classé 303 haineux et 240 commentaires neutres. Enfin, LR et NB ont mal classé respectivement 465 commentaires haineux, 296 commentaires neutres, 505 commentaires haineux et 339 commentaires neutres.

5.2.2.4 Comparaison des scénarios

Dans cette section, nous comparons les approches des trois scénarios expérimentés. Les résultats obtenus montrent que la plupart des approches expérimentées dans le scénario #3 donnent de meilleurs résultats que d'autres approches du scénario #2. Nous pouvons observer aussi, que passer le texte brut au classificateur (approche #1) peut être parfois plus pertinent et donne de meilleur accuracy que lorsque nous nettoyons le texte du bruit. La Figure 5.6 donne un aperçu des résultats obtenus d'accuracy de l'ensemble de données JigsawToxic pour l'algorithme SVM pour les trois scénarios. Les approches sont triées selon les résultats obtenus d'accuracy, de la plus grande valeur à la plus petite.

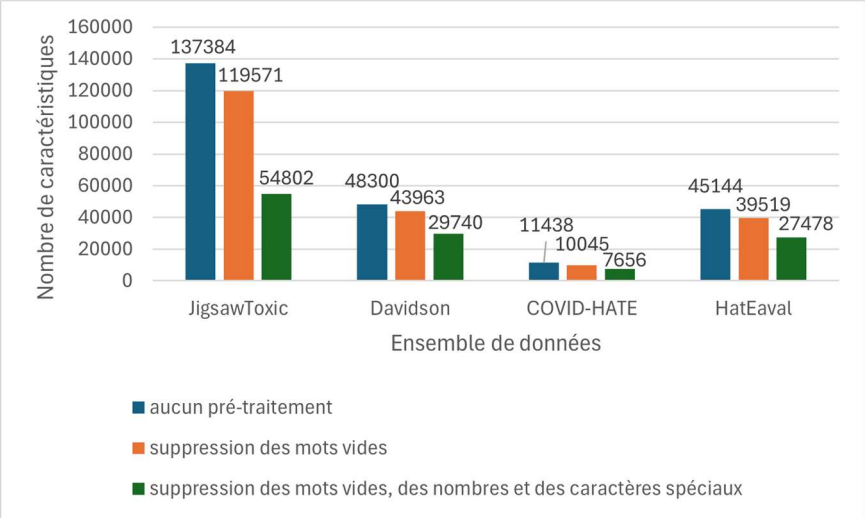
Figure 5.6. Comparaison des accurcay obtenues par SVM pour le scénario #1 et le scénario #2.



Généralement, les approches qui permettent de corriger l'orthographe et normaliser le texte donnent de meilleurs résultats que celles permettant de nettoyer le texte et supprimer le bruit. Ces résultats diffèrent parfois d'un ensemble de données à l'autre et cela peut être dû à la taille de l'ensemble de données, la quantité de bruit et le nombre de fautes d'orthographe qui s'y trouvent. La suppression du bruit est une tâche bénéfique dans l'amélioration des performances du modèle et qu'il faut adopter avec minutie, sinon elle peut toucher aussi des termes importants et pertinents dans la classification. Cela peut causer par la suite, la dégradation des performances du modèle. La normalisation et la correction orthographique du texte est aussi une tâche importante dans le prétraitement qui permet de récupérer des termes qui peuvent être importants dans la classification et qui sont ignorés s'ils ne sont pas corrigés. Comme nous l'avons mentionné dans les sections 5.2.2.2 et 5.2.2.3, le choix des techniques de prétraitement utilisées et leurs agencements peuvent influencer positivement ou négativement les résultats de la classification. Ceci peut influencer par la suite le nombre et la nature des caractéristiques dans l'ensemble de données qui sont considérés comme deux facteurs importants dans la tâche de classification (Liang, Sun, Yunlei, & Gao, 2017). Pour obtenir le nombre de caractéristiques par vecteur de données, nous avons utilisé la fonction shape() de python après vectorisation du texte non prétraité. La fonction shape() permet de renvoyer un tuple représentant la valeur de la dimension de l'objet Python. Nous avons procédé de la même façon pour obtenir le nombre de caractéristiques avec le texte prétraité. La Figure 5.7 donne un aperçu du nombre de caractéristiques par vecteur de données, avant et après le prétraitement du texte.

Durant la phase d'entraînement, nous ne prenons en compte que des principales caractéristiques maximales classées par fréquence des termes dans le corpus. Le nombre maximal de caractéristiques utilisées est fixé à 20 000.

Figure 5.7. Nombre de caractéristiques par vecteur de données avant et après le prétraitement du texte.



Si nous prenons pour exemple l'ensemble de données COVID-HATE qui contient seulement 2631 échantillons, nous pouvons constater à partir de la Figure 5.7 que le nombre de caractéristiques de l'ensemble de données est 11438 pour le scénario #1 (texte brut sans pré-traitement). Après avoir supprimer les mots vides (stop words), le nombre de caractéristiques devient 10 045 caractéristiques et si nous supprimons les caractères spéciaux aussi ainsi que les nombres, nous aurons 7656 caractéristiques. Nous pouvons remarquer que le nombre de caractéristiques dépend de la taille de l'ensemble de données. En supprimant du bruit, la taille de ce dernier est réduite et par la suite le nombre de caractéristiques diminue. Ce qui explique la dégradation des performances des algorithmes avec cet ensemble de données.

Les combinaisons de techniques de prétraitement qui ont donné les meilleurs résultats pour la plupart des ensembles de données et avec la majorité des algorithmes utilisés sont les suivants :

- Approche #15 : convertir le texte en lettres minuscules, supprimer les mentions utilisateurs, les hashtags, les URLs, les balises HTML, les horodatages, les nombres, les caractères spéciaux, supprimer par la suite les mots rares et faire la tokenisation en supprimant les espaces ;
- Approche #22 : convertir le texte en lettres minuscules, faire la lemmatisation suivie du stemming et faire la tokenisation en supprimant les espaces ;

- Approche #24 : convertir le texte en lettres minuscules, gérer les contractions et les accents, faire la correction orthographique suivie de la lemmatisation et du stemming et faire la tokenisation en supprimant les espaces ;
- Approche #26 : convertir le texte en lettres minuscules, gérer les contractions et les accents, faire la correction orthographique suivie de la lemmatisation et du stemming et faire la tokenisation en supprimant les espaces et les termes de deux caractères et moins.

Le Tableau 5.40 présente une comparaison des résultats obtenus par nos meilleurs approches avec ceux obtenus par quelques travaux dans la littérature. Ces travaux ont aussi étudié la détection et la classification du langage haineux et toxique. L'algorithme en commun utilisé dans toutes les approches est SVM. acc_J, acc_D, acc_C et acc_{H} désignent respectivement les valeurs de l'accuracy des ensembles de données JigsawToxic, Davidson, COVID-HATE et HatEval utilisés dans ce travail.

Tableau 5.40. Comparaison des résultats obtenus par SVM avec ceux obtenus dans la littérature.

Littérature	Techniques de prétraitement utilisées	Accuracy
(Hajibabaee, et al., 2022)	Conversion en lettres minuscules, suppression des URL, des mentions d'utilisateurs, des hashtags et des émojis et gestion des abréviations.	95%
(Poojitha, Sai Charish, Kuamr Reddy, & Ayyasamy, 2023)	Tokenisation, suppression des nombres, des signes de ponctuation et des mots vides, correction orthographique et stemming.	94%
(Rasel, Sultana, Akhter, & Meesad, 2018)	Conversion en lettres minuscules, suppression des nombres, des signes de ponctuation, des caractères spéciaux, des URL, des redondances (retweet) et des espaces blancs, correction orthographique, suppression des mots vides, stemming et suppression des contractions.	81%
(Rahul, Kajla, Hooda, & Saini, 2020)	Suppression des points, des virgules et des signes de ponctuation.	88.69%
(Dutta, Neog, & Baruah, 2024)	Suppression des signes de ponctuation et des émojis, tokenisation, POS, stemming et suppression des mots vides.	94%
(Sumanth, Samiuddin, Jamal, Domakonda, & Shivani, 2022)	Gestion des contractions, suppression des mots vides, des URL et des mentions utilisateurs, lemmatisation, conversion en lettres minuscules et suppression des espaces et des signes de ponctuation.	86,5%
Nos approches	Sans prétraitement	$acc_J = 83{,}48\%$ $acc_D = 75{,}21\%$ $acc_C = 76{,}09\%$ $acc_H = 68{,}54\%$
	Approche #15	$acc_J = 87{,}01\%$ $acc_D = 89{,}5\%$ $acc_C = 78{,}56\%$ $acc_H = 70{,}27\%$
	Approche #22	$acc_J = 87{,}43\%$ $acc_D = 89{,}37\%$ $acc_C = 81{,}02\%$ $acc_H = 71{,}58\%$
	Approche #24	$acc_J = 88{,}03\%$ $acc_D = 89{,}78\%$ $acc_C = 79{,}89\%$ $acc_H = 68{,}54\%$
	Approche #26	$acc_J = 87{,}75\%$ $acc_D = 86{,}34\%$ $acc_C = 81{,}97\%$ $acc_H = 72\%$

5.2.2.5 Conclusion

Dans ce chapitre, nous avons présenté d'abord, pour chaque scénario les résultats obtenus par les algorithmes utilisés avec les quatre ensembles de données différents. Nous avons fait par la suite, une comparaison entre ces différents scénarios. Finalement, nous avons discuté les résultats obtenus par les différentes approches pour identifier l'influence du prétraitement sur les résultats obtenus par la classification. Nous en avons déduit que le choix et l'agencement des techniques de prétraitement ainsi que la taille de l'ensemble de données d'entraînement peuvent avoir un impact positif, comme ils peuvent avoir un impact négatif, sur les résultats de la classification.

CONCLUSION

Dans ce travail, nous avons étudié les techniques de prétraitement qui permettent d'améliorer les performances d'un système de classification de discours haineux sur les réseaux sociaux. Nous avons exploré les agencements de techniques les plus pertinents dans un tel contexte, en expérimentant plusieurs approches qui combinent plusieurs techniques de prétraitement des données.

Plusieurs algorithmes d'apprentissage machine ont été utilisés (SVM, LR, NB et Bi-LSTM) avec quatre différents ensembles de données contenant du contenu haineux. Les meilleurs résultats sont ceux obtenus par le modèle neuronal, ce qui est conforme dans la littérature, avec les travaux de (Georgakopoulos, Tasoulis, Vrahatis, & Plagianakos, 2018) qui ont obtenu 91,2% comme accuracy avec le modèle CNN dans la classification des commentaires toxiques. Les expérimentations montrent que les résultats peuvent différer parfois d'un ensemble de données à l'autre dépendamment de la taille de l'ensemble de données et de la quantité de bruit et de fautes d'orthographe qui s'y trouvent. Nous avons observé que la combinaison de la lemmatisation et de la racinisation (stemming) améliore les performances du classificateur. Cependant, Il est préférable de procéder à la lemmatisation avant de faire la racinisation (stemming) puisque la première technique tient compte du contexte, à l'opposé de la deuxième. Nous avons constaté aussi que dans certains cas, l'utilisation de la tokenisation seule peut donner de meilleurs résultats que lorsque le texte subit un prétraitement. La combinaison de la correction orthographique et la normalisation du texte s'avère plus bénéfique que la suppression du bruit qui peut toucher des termes importants et pertinents dans la classification. La même remarque a été observée avec la suppression des mots fréquents et des mots rares, qui peut aussi toucher des termes importants dans la classification et causer par la suite, la dégradation des performances du modèle.

Les scénarios d'ingénierie de données expérimentés dans ce travail sont intéressants. Ils nous ont permis de voir de près l'influence du prétraitement sur la classification. Ils nous en aussi permis d'identifier les facteurs qui doivent être vérifiés et respectés pour améliorer et optimiser un algorithme de classification et faire en sorte que la classification réagisse positivement au prétraitement. Le choix et l'agencement des techniques de prétraitement ainsi que la taille et la nature de l'ensemble de données d'entraînement représentent des facteurs primordiaux permettant d'améliorer les performances d'un classificateur, ou au contraire, de les dégrader.

ANNEXE A

EXEMPLES DE COMMENTAIRES

Figure A.1. Exemple de commentaire le plus long de l'ensemble de données JigsawToxic.

```
# print longest comment
index = train["comment_text"][np.argmax(lengths)]
print('comment length:', len(index))
print(train["comment_text"][np.argmax(lengths)])
```

```
comment length: 4970
YOU!
You blocked me worm wh
y!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Figure A.2. Deuxième exemple d'un commentaire long de l'ensemble de données JigsawToxic.

```
comment length: 4849
why don't you suck my ass until your lips bleed?

CONGRATS!  NOW YOU'VE LOST A GOOD ARTICLE AS WELL AS A READER!  I WILL NEVER COME BACK TO WIKISHIT AGAI
N!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Figure A.3. Troisième exemple d'un commentaire long de l'ensemble de données JigsawToxic.

```
comment length: 3469
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! !!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! !!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! !!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1lol imma donkey
LUNCHABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Figure A.4. Exemple du commentaire le plus long de l'ensemble de données Davidson.

```
# print longest comment
print('comment length:', max)
print(train["tweet"][np.argmax(lengths)])
```

```
comment length: 321
RT @TrxllLegend: One good girl is worth a thousand bitches

&#128112; = &#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#1
28109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;
&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#1281
09;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#1
28109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;
&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#128109;&#1281
09;&#8230;
```

Figure A.5. Deuxième exemple de commentaire long de l'ensemble de données Davidson.

```
# print other longest comment
print('comment length:', len1)
print(train["tweet"][lengths.index(len1)])
print('comment length:', len2)
print(train["tweet"][lengths.index(len2)])
```

```
comment length: 281
No summer school? &#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#1285
15;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#1
28515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;
&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#1285
15;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#128515;&#1
28515;&#128515;&#128515;&#128515; eat a dick school. Im done with your bitch ass !!!!!!
```

Figure A.6. Troisième exemple de commentaire long de l'ensemble de données Davidson.

```
comment length: 267
&#8220;@Untouchable_T: Never seen so many perfect bitches til I made a Twitter &#128564; but &#128056;&#9749;&#650
39;&#8221;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128
175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#
128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#12817
5;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#12
8175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;&#128175;
```

Figure A.7. Exemple de commentaires le plus long de l'ensemble de données COVID-HATE.

```
# print longest comment
print('comment length:', max)
print(train["Text"][np.argmax(lengths)])
```

```
comment length: 151
@GSawision @MonicaFibonacci @sameold2cents1 @leighgt @JohnMorelli_BC @JohnAll63815927 @LeeeLauren @AdamHarryCassi1
@hotdogterminatr @2dratbil @WorldPe56580574 @gbobke @truebluecanadi1 @baletown_crawl @BrenBlake2 @cheryl1492 @canu
kcookie @JustinSchafer1 @RickHi2315 @NotToday1701 @johnsta04633925 @Moonshi84280504 @DawsonMdhoust @Danbalkwill1 @
IvanhansenIvan @Kromski1 @lassiter1550 @KieserBill @_LoveMTB_ @FuzzySockzz @RKDickie1 @Victoria59L @myearwood65 @A
lbertaCanadien @LinnieRants @canadaknife @PDK4863 @PLHCrawford @arnie27027288 @saskcowmomma3 @canadaunited1 @paddy
hullabaloo @whelanmj @JustaHu64027144 @boysha @bkruhlak @Canadian_Chris_ @Blanchecartefo1 @zeldacanuck @MzDeplorab
le CORONAVIRUS BIOWEAPON MYSTERY SOLVED: It's all in the name!  https://t.co/ERjbdfoiER  #pharmagreed #VaxxOFF Not
one rich famous person or a dirty greedy politician has been infected or died, why the worry? just another #BillGa
tes trick to kill #deathbydesign #greedkills
```

Figure A.8. Deuxième exemple de commentaire long de l'ensemble de données COVID-HATE.

```
# print other longest comment
print('comment length:', len1)
print(train["Text"][lengths.index(len1)])
print('comment length:', len2)
print(train["Text"][lengths.index(len2)])
```

```
comment length: 138
@Big_crusher1000 @Lynda63986855 @Shaun_Girk @DorisMele @BILLHALES88 @traveler002 @kit_bramat @spinson7746 @davidf4
444 @MemeMercenary @fedagentmark @vicksiern @jan_aurora @ChrisPBaconLT @ScottRickhoff @Quin4Trump @TheWickerhead @
Jamie32377541 @stand4honor @establishmentno @RosaWal90673631 @Rosemar06585176 @robcarlson20 @jjpalmer2015 @ISafeye
t @scampbell123451 @RemiDSS @ALEXNEWMAN_JOU @WMRDC @libertytarian @ddwiese @PAMsLOvE @RealDeanCain @BrianHanes4 @k
br_kag @Lots_Of_Fun_69 @seecyn5858 @Brooke_Kelly87 @Duwayn55629746 @cmccbyfaith @MacShiver @WagonKnoggin @PatriotB
rwnEyes @mamoobonnie @bmickeydanger @kay89266490 @usvetram @redwins3_first @gatewaypundit @PageSix To all Commie D
emocraps:  STFU!  Our POTUS is handling Coronavirus just fine.  Now, thank God none of you nor Ovomit is in charg
e!  With your De-population agenda, you would be USELESS!
```

Figure A.9, Troisième exemple de commentaire long de l'ensemble de données COVID-HATE.

```
comment length: 130
@SolomonYue @SpeakerPelosi @tedcruz @marcorubio @UN @RedCross @lionsclubs @AmChamHK @ABAesq @hrw @UN_HRC @pressfre
edom @10DowningStreet @INTERPOL_HQ @GoldmanSachs @MorganStanley @jpmorgan @MerrillLynch @IMFNews @wto @wef @WhiteH
ouse @IndexCensorship @BorisJohnson @HouseDemocrats @WWF @sgfintechfest @PMOIndia @CanadianPM @JPN_PMO @NZNational
Party @iingwen @WorldBank @AmerBanker @WHO @DubaiPressClub @cityusucbc @ABC @WSJ @FoxNews @Reuters @RoyalFamily @U
KHouseofLords @HouseofCommons @DominicRaab @UniofOxford @UN_Women @KenRoth @chillilucas_hk https://t.co/sT7HEsBWIt
production line better move back, quickly. corona is not ordinary flu. it also attack kidneys and sex organ with a
high infecting rate. #ChinaVirus
```

Figure A.10. . Exemple de commentaires le plus long de l'ensemble de données HatEval.

```
# print longest comment
print('comment length:', max)
print(copy_data["text"][np.argmax(lengths)])
```

```
comment length: 139
@AliceEvansGruff @cmd51375 @Privacy_Painter @caterita2008 @pat_hardy_ @avaliv_________ @raindovemodel @KSantorri @
LizzyBDizzy101 @ggbrokensilence @mae_quez @Jacmalta151 @TakeThatMorals @ClickitH @marczak_rob @elena_audry @Vellin
iV @snowmancalgary @Lizzie_Borden4 @CIAdaughter @Wiki_Vic @HeltonGreen @Unicorn4Glitter @MeganMia5 @LoveWorksDotCo
m @aogfx @RealAJBenza @infinity_1616 @Stephanielaz918 @RealpeopleSar @mynameisphaedra @AllenLynching @OrMyLast @Cr
ystal_Ball1 @JoeRipper4 @wtfimontwitr @Bourdain @AsiaArgento @rosemcgowan @LeahMcSweeney @Genxpunk69 @jordanbpeter
son "Stop Name Calling" Grow a pair of balls or dont say anything stupid if you know the truth how come you havent
released a statement of evidence at all instead of being a arrogant pest. Remember you believe women cant rape men
also meaning you think they cant rape women either.
```

Figure A.11. Deuxième exemple de commentaire long de l'ensemble de données HatEval.

```
# print other longest comment
print('comment length:', len1)
print(copy_data["text"][lengths.index(len1)])
print('comment length:', len2)
print(copy_data["text"][lengths.index(len2)])
```

```
comment length: 136
@FrancesannMaga @harperjeff30 @speakupsal @redbudacres @gary_w723 @MYSTERYSAIL @bellwetherdude @arkiegal411 @GOPMi
llenials @Ptrtldy1L @CynthiaRunnels1 @SunshineLK10 @brenda_lummus @Jetsta812 @TWGroupsWarrior @crimsonfaith88 @Smi
thsCoffeepot @CONNOR4TRUMP @DebbieTheMOTS @atillathehun412 @144000bound @ANONmonkywrench @TransformativeV @blacktu
lip966 @1Nvrdul @RR2A11 @protrumpuk @wolferkitten @patni1111 @Trina1732674 @chemist552 @joyreaper @hankcioffi @cos
tman77 @BobC34019033 @bobhelps @JeffFromIowa @Ranger3079 @horse1157 Are you asking me if my statement on #IllegalA
liens was a generalized statement @FrancesannMaga ? If your question is to me, yes, it was generalized,, I did not
mean to profile ALL to be the same, but I think MANY illegals work the fields for plantations & are exploited doin
g so
```

Figure A.12. Troisième exemple de commentaire long de l'ensemble de données HatEval.

```
comment length: 130
@SolomonYue @SpeakerPelosi @tedcruz @marcorubio @UN @RedCross @lionsclubs @AmChamHK @ABAesq @hrw @UN_HRC @pressfre
edom @10DowningStreet @INTERPOL_HQ @GoldmanSachs @MorganStanley @jpmorgan @MerrillLynch @IMFNews @wto @wef @WhiteH
ouse @IndexCensorship @BorisJohnson @HouseDemocrats @WWF @sgfintechfest @PMOIndia @CanadianPM @JPN_PMO @NZNational
Party @iingwen @WorldBank @AmerBanker @WHO @DubaiPressClub @cityusucbc @ABC @WSJ @FoxNews @Reuters @RoyalFamily @U
KHouseofLords @HouseofCommons @DominicRaab @UniofOxford @UN_Women @KenRoth @chillilucas_hk https://t.co/sT7HEsBWIt
production line better move back, quickly. corona is not ordinary flu. it also attack kidneys and sex organ with a
high infecting rate. #ChinaVirus
```

ANNEXE B

VALEURS MANQUANTES DANS LES ENSEMBLES DE DONNÉES

Figure B.1. Nombre de valeurs manquantes dans l'ensemble de données long JigsawToxic .

```
# Detet the missing values or NaN values in DataFrame
print('Number of missing comments in comment text column:')
print(train['comment_text'].isnull().sum())
```

```
Number of missing comments in comment text column:
0
```

Figure B.2. Nombre de valeurs manquantes dans l'ensemble de données Davidson.

```
# Detect the missing values or NaN values in DataFrame
print('Number of missing comments in comment text column:')
print(train['tweet'].isnull().sum())
```

```
Number of missing comments in comment text column:
0
```

Figure B.3. Nombre de valeurs manquantes dans l'ensemble de données COVID-HATE.

```
# Detect the missing values or NaN values in DataFrame
print('Number of missing comments in comment text column:')
print(train['comment_text'].isnull().sum())
```

```
Number of missing comments in comment text column:
0
```

Figure B.4. Nombre de valeurs manquantes dans l'ensemble de données HatEval.

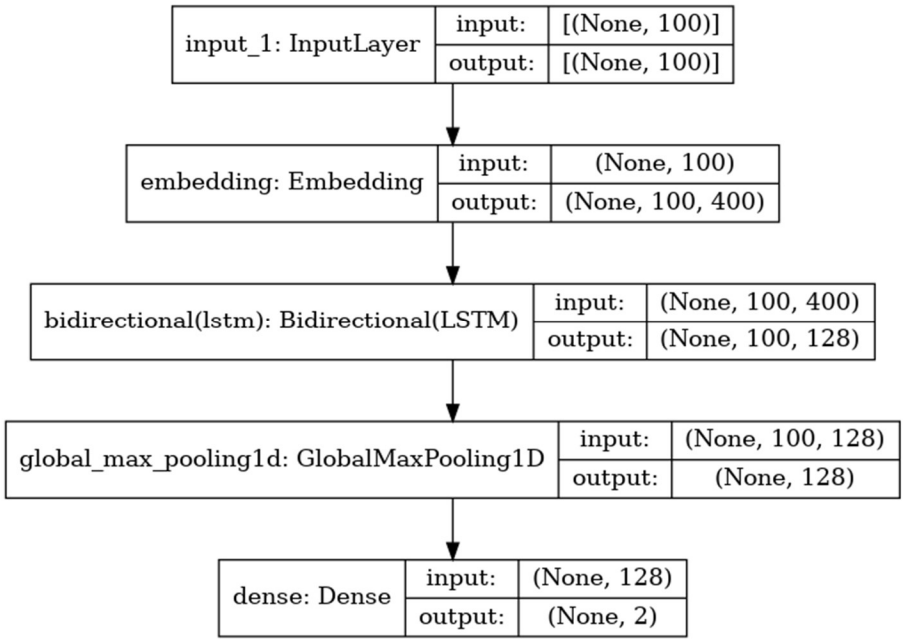
```
# Detect the missing values or NaN values in DataFrame
print('Number of missing comments in comment text column:')
print(copy_data['text'].isnull().sum())
```

```
Number of missing comments in comment text column:
0
```

ANNEXE C

ARCHITECTURE DU MODÈLE BI-LSTM UTILISÉ

Figure C.1. Architecture du modèle Bi-LSTM utilisé.



BIBLIOGRAPHIE

Aboalnaser, S. A. (2019). Machine Learning Algorithms in Arabic Text Classification: A Review. *2019 12th International Conference on Developments in eSystems Engineering (DeSE)*, (pp. 290-295). doi:10.1109/DeSE.2019.00061

Afzaal, H., Farooque, A. A., Esau, T. J., Schumann, A. W., Zaman, Q. U., Abbas, F., & Bos, M. (2023). Chapter 10 - Artificial neural modeling for precision agricultural water management practices. Dans Q. Zaman (Éd.), *Precision Agriculture* (pp. 169-186). Academic Press. doi:https://doi.org/10.1016/B978-0-443-18953-1.00005-2

Ahmed, T., Mukta, S. F., Al Mahmud, T., Hasan, S. A., & Gulzar Hussain, M. (2022). Bangla Text Emotion Classification using LR, MNB and MLP with TF-IDF & CountVectorizer. *2022 26th International Computer Science and Engineering Conference (ICSEC)*, (pp. 275-280). doi:10.1109/ICSEC56337.2022.10049341

Ahmed, Z., Kinjol, F. J., & Ananya, I. J. (2021). Comparative Analysis of Six Programming Languages Based on Readability, Writability, and Reliability. *2021 24th International Conference on Computer and Information Technology (ICCIT)*, (pp. 1-6). doi:10.1109/ICCIT54785.2021.9689813

Akhmouch, H., Bouanani, H., Dias, G., & Moreno, J. G. (2021, June). Stratégie Multitâche pour la Classification Multiclasse (A Multitask Strategy for Multiclass Classification). In P. Denis, N. Grabar, A. Fraisse, R. Cardon, B. Jacquemin, E. Kergosien, & A. Balvet (Ed.), *Actes de la 28e Conférence sur le Traitement Automatique des Langues Naturelles. Volume 1 : conférence principale* (pp. 227–236). Lille: ATALA. Retrieved from https://aclanthology.org/2021.jeptalnrecital-taln.22

Anoual, E. k., & Zeroual, I. (2021, February). The effects of Pre-Processing Techniques on Arabic Text Classification. *International Journal of Advanced Trends in Computer Science and Engineering, 10*, 41-48. doi:10.30534/ijatcse/2021/061012021

Ayo, F. E., Folorunso, O., Ibharalu, F. T., & Osinuga, I. A. (2020). Machine learning techniques for hate speech classification of twitter data: State-of-the-art, future challenges and research directions. *Computer Science Review, 38*, 100311. doi:https://doi.org/10.1016/j.cosrev.2020.100311

Badillo, S., Banfai, B., Birzele, F., Davydov, I., Hutchinson, L., Kam-Thong, T., . . . Zhang, J. D. (2020, March). An Introduction to Machine Learning. *Clinical Pharmacology & Therapeutics, 107*. doi:10.1002/cpt.1796

Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017). Deep Learning for Hate Speech Detection in Tweets. *Proceedings of the 26th International Conference on World Wide Web Companion* (pp. 759–760). Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee. doi:10.1145/3041021.3054223

Bao, W., Yue, J., & Rao, Y. (2017, July). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE, 12*, 1-24. doi:10.1371/journal.pone.0180944

Baruah, A., Barbhuiya, F., & Dey, K. (2019, June). ABARUAH at SemEval-2019 Task 5 : Bi-directional LSTM for Hate Speech Detection. Dans J. May, E. Shutova, A. Herbelot, X. Zhu, M. Apidianaki, & S. M. Mohammad (Éd.), *Proceedings of the 13th International Workshop on Semantic Evaluation* (pp. 371–376). Minneapolis: Association for Computational Linguistics. doi:10.18653/v1/S19-2065

Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Rangel Pardo, F. M., . . . Sanguinetti, M. (2019, June). SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter. *Proceedings of the 13th International Workshop on Semantic Evaluation* (pp. 54–63). Minneapolis: Association for Computational Linguistics. doi:10.18653/v1/S19-2007

Battineni, G., Sagaro, G., Chintalapudi, N., Amenta, F., & Tayebati, S. K. (2019, December). Comparative Machine-Learning Approach: A Follow- Up Study on Type 2 Diabetes Predictions by Cross-Validation Methods. *Machines, 7*, 74. doi:10.3390/machines7040074

Baturynska, I., & Martinsen, K. (2021, January). Prediction of geometry deviations in additive manufactured parts: comparison of linear regression with machine learning algorithms. *Journal of Intelligent Manufacturing, 32*. doi:10.1007/s10845-020-01567-0

Bayu, T., Arifin, O., & al Fatta, H. (2019, July). Optimization of Hyper Parameter Bandwidth on Naïve Bayes Kernel Density Estimation for the Breast Cancer Classification., (pp. 226-231). doi:10.1109/ICOIACT46704.2019.8938497

Belyadi, H., & Haghighat, A. (2021). Chapter 5 - Supervised learning. Dans H. Belyadi, & A. Haghighat (Éd.), *Machine Learning Guide for Oil and Gas Using Python* (pp. 169-295). Gulf Professional Publishing. doi:https://doi.org/10.1016/B978-0-12-821929-4.00004-4

Berrar, D. (2018, January). Bayes' Theorem and Naive Bayes Classifier. doi:10.1016/B978-0-12-809633-8.20473-1

Bothe, C., Weber, C., Magg, S., & Wermter, S. (2018, May). A Context-based Approach for Dialogue Act Recognition using Simple Recurrent Neural Networks. Dans N. Calzolari, K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, . . . T. Tokunaga (Éd.), *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).* Miyazaki: European Language Resources Association (ELRA). Récupéré sur https://aclanthology.org/L18-1307

Cahyanti, F. E., Adiwijaya, & Faraby, S. A. (2020). On The Feature Extraction For Sentiment Analysis of Movie Reviews Based on SVM. *2020 8th International Conference on Information and Communication Technology (ICoICT)*, (pp. 1-5). doi:10.1109/ICoICT49345.2020.9166397

Chen, J., Huang, H., Tian, S., & Qu, Y. (2009). Feature selection for text classification with Naïve Bayes. *Expert Systems with Applications, 36*, 5432-5435. doi:https://doi.org/10.1016/j.eswa.2008.06.054

Cheng, W.-C., Huang, J.-C., & Liou, C.-Y. (2012). Segmentation of DNA using simple recurrent neural network. *Knowledge-Based Systems, 26*, 271-280. doi:https://doi.org/10.1016/j.knosys.2011.09.001

Contreras, P., Orellana-Alvear, J., Muñoz, P., Bendix, J., & Célleri, R. (2021, February). Influence of Random Forest Hyperparameterization on Short-Term Runoff Forecasting in an Andean Mountain Catchment. *Atmosphere, 12*, 238. doi:10.3390/atmos12020238

Crowston, K., & Bolici, F. (2019). Impacts of Machine Learning on Work. *Hawaii International Conference on System Sciences.* Récupéré sur https://api.semanticscholar.org/CorpusID:53658963

Cui, N. (2018, April). Applying Gradient Descent in Convolutional Neural Networks. *Journal of Physics: Conference Series, 1004*, 012027. doi:10.1088/1742-6596/1004/1/012027

Cui, S., Zhao, L., Wang, Y., Dong, Q., Ma, J., Wang, Y., . . . Ma, X. (2018). Using Naive Bayes Classifier to predict osteonecrosis of the femoral head with cannulated screw fixation. *Injury, 49*, 1865-1870. doi:https://doi.org/10.1016/j.injury.2018.07.025

Davidson, T., Warmsley, D., Macy, M. W., & Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. *CoRR, abs/1703.04009*. Récupéré sur http://arxiv.org/abs/1703.04009

Dawar, I., Kumar, N., Negi, S., Pathan, S., & Layek, S. (2023). Text Categorization using Supervised Machine Learning Techniques. *2023 Sixth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU)*, (pp. 185-190). doi:10.1109/WiDS-PSU57071.2023.00046

Dawei, W., Alfred, R., Obit, J., & On, C. (2021, January). A Literature Review on Text Classification and Sentiment Analysis Approaches. doi:10.1007/978-981-33-4069-5_26

Domingos, P. (2012, October). A few useful things to know about machine learning. *Commun. ACM, 55*, 78–87. doi:10.1145/2347736.2347755

Dramsch, J. S. (2020). Chapter One - 70 years of machine learning in geoscience in review. Dans B. Moseley, & L. Krischer (Éd.), *Machine Learning in Geosciences* (Vol. 61, pp. 1-55). Elsevier. doi:https://doi.org/10.1016/bs.agph.2020.08.002

Dutta, S., Neog, M., & Baruah, N. (2024). Assamese Toxic Comment Detection On Social Media Using Machine Learning Methods. *2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE)*, (pp. 1-8). doi:10.1109/ic-ETITE58242.2024.10493331

Dvornek, N. C., & Li, X. (2023). Chapter 13 - Deep learning with connectomes. Dans M. D. Schirmer, T. Arichi, & A. W. Chung (Éd.), *Connectome Analysis* (pp. 289-308). Academic Press. doi:https://doi.org/10.1016/B978-0-323-85280-7.00013-0

Edgar, T. W., & Manz, D. O. (2017). Chapter 4 - Exploratory Study. Dans T. W. Edgar, & D. O. Manz (Éd.), *Research Methods for Cyber Security* (pp. 95-130). Syngress. doi:https://doi.org/10.1016/B978-0-12-805349-2.00004-2

Feng, W., Guan, N., Li, Y., Zhang, X., & Luo, Z. (2017, May). Audio visual speech recognition with multimodal recurrent neural networks., (pp. 681-688). doi:10.1109/IJCNN.2017.7965918

Gaikwad, M., & Doke, A. (2023, May). Comparative Analysis of Statistical Optimizers for Logistic Regression. *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)*, (pp. 1193-1197). doi:10.1109/ICICCS56967.2023.10142817

García, V., Sánchez, J., & Marqués, A. (2019, November). Synergetic Application of Multi-Criteria Decision-Making Models to Credit Granting Decision Problems. *Applied Sciences, 9*, 1-15. doi:10.3390/app9235052

Georgakopoulos, S. V., Tasoulis, S. K., Vrahatis, A. G., & Plagianakos, V. P. (2018). Convolutional Neural Networks for Toxic Comment Classification. *Proceedings of the 10th Hellenic Conference on Artificial Intelligence.* New York, NY, USA: Association for Computing Machinery. doi:10.1145/3200947.3208069

Ghosh, S., Dasgupta, A., & Swetapadma, A. (2019). A Study on Support Vector Machine based Linear and Non-Linear Pattern Classification. *2019 International Conference on Intelligent Sustainable Systems (ICISS)*, (pp. 24-28). doi:10.1109/ISS1.2019.8908018

Guo, H., Zhang, J., & Xiao, L. (2022). A news text classification method based on the BiLSTM-Attention. *2022 International Conference on Data Analytics, Computing and Artificial Intelligence (ICDACAI)*, (pp. 468-472). doi:10.1109/ICDACAI57211.2022.00099

Haddi, E., Liu, X., & Shi, Y. (2013). The Role of Text Pre-processing in Sentiment Analysis. *Procedia Computer Science, 17*, 26-32. doi:https://doi.org/10.1016/j.procs.2013.05.005

Hajibabaee, P., Malekzadeh, M., Ahmadi, M., Heidari, M., Esmaeilzadeh, A., Abdolazimi, R., & Jones, J. H. (2022). Offensive Language Detection on Social Media Based on Text Classification. *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, (pp. 92-98). doi:10.1109/CCWC54503.2022.9720804

Hanshika. V., G., Preethi., S., Sreemathy. S., P., & Ezhillin Freeda., S. (2023). Flood Prediction Using Logistic Regression. *2023 International Conference on Circuit Power and Computing Technologies (ICCPCT)*, (pp. 1174-1179). doi:10.1109/ICCPCT58313.2023.10245832

Hasan, S. A., Hussain, M. G., Protim, J., Rahman, M. M., Fahim, N., Chowdhury, M. Z., & Pritom, A. I. (2022). Classification of Multi-Labeled Text Articles with Reuters Dataset using SVM. *2022 International Conference on Science and Technology (ICOSTECH)*, (pp. 1-5). doi:10.1109/ICOSTECH54296.2022.9829153

Hassanshahi, M. H., Jastrzebski, M., Malik, S., & Lahav, O. (2023). A quantum-enhanced support vector machine for galaxy classification. *A quantum-enhanced support vector machine for galaxy classification*.

He, B., Ziems, C., Soni, S., Ramakrishnan, N., Yang, D., & Kumar, S. (2020, May). Racism is a Virus: Anti-Asian Hate and Counterspeech in Social Media during the COVID-19 Crisis. *arXiv e-prints*, arXiv:2005.12423. doi:10.48550/arXiv.2005.12423

Heitz, T. (2006). Modélisation du prétraitement des textes. *JADT'06 (International Conference on Statistical Analysis of Textual Data)*, *1*, pp. 499-506. Besançon, France. Récupéré sur https://hal.inria.fr/inria-00119608

Hindi, K. M., Aljulaidan, R. R., & AlSalman, H. (2020). Lazy fine-tuning algorithms for naïve Bayesian text classification. *Applied Soft Computing, 96*, 106652. doi:https://doi.org/10.1016/j.asoc.2020.106652

Jahan, M. S., & Oussalah, M. (2021, May). A systematic review of Hate Speech automatic detection using Natural Language Processing. *A systematic review of Hate Speech automatic detection using Natural Language Processing*.

Jalal, N., Mehmood, A., Choi, G. S., & Ashraf, I. (2022). A novel improved random forest for text classification using feature ranking and optimal number of trees. *Journal of King Saud University - Computer and Information Sciences, 34*, 2733-2742. doi:https://doi.org/10.1016/j.jksuci.2022.03.012

Jang, B., Kim, M., Harerimana, G., Kang, S., & Kim, J. W. (2020). Bi-LSTM Model to Increase Accuracy in Text Classification: Combining Word2vec CNN and Attention Mechanism. *Applied Sciences*. Récupéré sur https://api.semanticscholar.org/CorpusID:225242195

Jha, A., & Mamidi, R. (2017, August). When does a compliment become sexist? Analysis and classification of ambivalent sexism using Twitter data. doi:10.18653/v1/W17-2902

Jiang, Q.-Y. (2022). Text classification method based on LSTM. *Proceedings of the 6th EAI International Conference on IoT in Urban Space, Urb-IoT 2021, 20-21 December 2021, Shenzhen, People's Republic of China*. Récupéré sur https://api.semanticscholar.org/CorpusID:249525767

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014, April). A Convolutional Neural Network for Modelling Sentences. *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference, 1*. doi:10.3115/v1/P14-1062

Karabatak, M. (2015). A new classifier for breast cancer detection based on Naïve Bayesian. *Measurement, 72*, 32-36. doi:https://doi.org/10.1016/j.measurement.2015.04.028

Keerthi Kumar, H. M., & Harish, B. S. (2018, November). Classification of Short Text Using Various Preprocessing Techniques: An Empirical Evaluation: Proceedings of the 5th ICACNI 2017, Volume 3. doi:10.1007/978-981-10-8633-5_3

Kinnunen, T., & Li, H. (2010, January). An Overview of Text-Independent Speaker Recognition: from Features to Supervectors. *Speech Communication, 52*, 12-40. doi:10.1016/j.specom.2009.08.009

Klang, E., Barash, Y., Margalit, R. Y., Soffer, S., Shimon, O., Albshesh, A., . . . Kopylov, U. (2020). Deep learning algorithms for automated detection of Crohn's disease ulcers by video capsule endoscopy. *Gastrointestinal Endoscopy, 91*, 606-613.e2. doi:https://doi.org/10.1016/j.gie.2019.11.012

Korde, V. (2012, March). Text Classification and Classifiers:A Survey. *International Journal of Artificial Intelligence & Applications, 3*, 85-99. doi:10.5121/ijaia.2012.3208

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., Brown, D., . . . Barnes. (2019, April). Text Classification Algorithms: A Survey. *Information (Switzerland), 10*. doi:10.3390/info10040150

Kowsher, M., Tahabilder, A., Hossain Sarker, M. M., Islam Sanjid, M. Z., & Prottasha, N. J. (2020). Lemmatization Algorithm Development for Bangla Natural Language Processing. *2020 Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, (pp. 1-8). doi:10.1109/ICIEVicIVPR48672.2020.9306652

Kumar, A. C., John, J. A., Raja, M., & Vijaya, P. (2023). Chapter 4 - Genetic factor analysis for an early diagnosis of autism through machine learning. Dans A. K. Tyagi, & A. Abraham (Éd.), *Data Science for Genomics* (pp. 69-84). Academic Press. doi:https://doi.org/10.1016/B978-0-323-98352-5.00001-X

Kumar, R., Sharma, A., & Varadwaj, P. (2011, July). A prediction model for oral bioavailability of drugs using physicochemical properties by support vector machine. *Journal of natural science, biology, and medicine, 2*, 168-73. doi:10.4103/0976-9668.92325

Kumar, V., & Subba, B. (2020). A TfidfVectorizer and SVM based sentiment analysis framework for text data corpus. *2020 National Conference on Communications (NCC)*, (pp. 1-6). doi:10.1109/NCC48643.2020.9056085

Kumaraswamy, B. (2021). 6 - Neural networks for data classification. Dans D. Binu, & B. R. Rajakumar (Éd.), *Artificial Intelligence in Data Mining* (pp. 109-131). Academic Press. doi:https://doi.org/10.1016/B978-0-12-820601-0.00011-2

Lameiro, C., & Schreier, P. J. (2016). Cross-validation techniques for determining the number of correlated components between two data sets when the number of samples is very small. *2016 50th Asilomar Conference on Signals, Systems and Computers*, (pp. 601-605). doi:10.1109/ACSSC.2016.7869113

Le, A., Liu, B., & Huang, H. (2009, June). Integration of computer-aided diagnosis/detection (CAD) results in a PACS environment using CAD-PACS toolkit and DICOM SR. *International journal of computer assisted radiology and surgery, 4*, 317-29. doi:10.1007/s11548-009-0297-y

Li, P., Mao, K., Xu, Y., Li, Q., & Zhang, J. (2020). Bag-of-Concepts representation for document classification based on automatic knowledge acquisition from probabilistic knowledge base. *Knowledge-Based Systems, 193*, 105436. doi:https://doi.org/10.1016/j.knosys.2019.105436

Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., . . . He, L. (2022, April). A Survey on Text Classification: From Traditional to Deep Learning. *ACM Trans. Intell. Syst. Technol., 13*. doi:10.1145/3495162

Liang, H., Sun, X., Yunlei, S., & Gao, Y. (2017, December). Text feature extraction based on deep learning: a review. *EURASIP Journal on Wireless Communications and Networking, 2017*. doi:10.1186/s13638-017-0993-1

Liang, Z., Chen, P., Yi, Y., & Fan, Y. (2022). News Text Classification Method for Edge Computing Based on BiLSTM-Attention. *2022 6th Asian Conference on Artificial Intelligence Technology (ACAIT)*, (pp. 1-6). doi:10.1109/ACAIT56212.2022.10137822

Liu, H., Chen, G., Li, P., Zhao, P., & Wu, X. (2021). Multi-label text classification via joint learning from label embedding and label correlation. *Neurocomputing, 460*, 385-398. doi:https://doi.org/10.1016/j.neucom.2021.07.031

Liu, Z., Lv, X., Liu, K., & Shi, S. (2010). Study on SVM Compared with the other Text Classification Methods. *2010 Second International Workshop on Education Technology and Computer Science*, *1*, pp. 219-222. doi:10.1109/ETCS.2010.248

Logeswaran, L., Lee, H., & Radev, D. (2018). Sentence Ordering and Coherence Modeling Using Recurrent Neural Networks. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence.* New Orleans, Louisiana, USA: AAAI Press.

Lubis, A. R., Nasution, M. K., Sitompul, O. S., & Zamzami, E. M. (2022). Spelling Checking with Deep Learning Model in Analysis of Tweet Data for Word Classification Process. *2022 9th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, (pp. 343-348). doi:10.23919/EECSI56542.2022.9946476

Lugo, L., & Barreto-Hernández, E. (2021). A Recurrent Neural Network approach for whole genome bacteria identification. *Applied Artificial Intelligence, 35*, 642-656. doi:10.1080/08839514.2021.1922842

Lv, H., & Tang, H. (2011). Machine Learning Methods and Their Application Research. *2011 2nd International Symposium on Intelligence Information Processing and Trusted Computing*, (pp. 108-110). doi:10.1109/IPTC.2011.34

Ma, Y., Li, Y., Wu, X., & Zhang, X. (2018). Chinese Text Classification Review. *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, (pp. 737-739). doi:10.1109/ITME.2018.00167

Mahmoodi, D., Marvi, H., Taghizadeh, M., Soleimani, A., Razzazi, F., & Mahmoodi, M. (2011, July). *Age Estimation Based on Speech Features and Support Vector Machine.* doi:10.1109/CEEC.2011.5995826

Mao, W., & Wang, F.-Y. (2012). Chapter 8 - Cultural Modeling for Behavior Analysis and Prediction. Dans W. Mao, & F.-Y. Wang (Éd.), *New Advances in Intelligence and Security Informatics* (pp. 91-102). Boston: Academic Press. doi:https://doi.org/10.1016/B978-0-12-397200-2.00008-7

Martiti, & Juliane, C. (2021, January). Implementation of Naive Bayes Algorithm on Sentiment Analysis Application. doi:10.2991/aer.k.211106.030

Mathapati, P., Shahapurkar, A., & Hanabaratti, K. (2017, July). Sentiment Analysis using Naïve bayes Algorithm. *International Journal of Computer Sciences and Engineering, 5*, 75-77. doi:10.26438/ijcse/v5i7.7577

Mercha, E. M., & Benbrahim, H. (2023). Machine learning and deep learning for sentiment analysis across languages: A survey. *Neurocomputing, 531*, 195-216. doi:https://doi.org/10.1016/j.neucom.2023.02.015

Misra, S., & Li, H. (2020). Chapter 9 - Noninvasive fracture characterization based on the classification of sonic wave travel times. Dans S. Misra, H. Li, & J. He (Éd.), *Machine Learning for Subsurface Characterization* (pp. 243-287). Gulf Professional Publishing. doi:https://doi.org/10.1016/B978-0-12-817736-5.00009-0

Mohammed, R., Rawashdeh, J., & Abdullah, M. (2020). Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results. *2020 11th International Conference on Information and Communication Systems (ICICS)*, (pp. 243-248). doi:10.1109/ICICS49469.2020.239556

Muaad, A. Y., Davanagere, H., Guru, D., Jv, B. B., Chola, C., Gumaei, A., & Al-antari Aisslab, M. A. (2022, April). Arabic Document Classification: Performance Investigation of Preprocessing and Representation Techniques. *Mathematical Problems in Engineering, 2022*, 1-16. doi:10.1155/2022/3720358

Nascimento, F., Cavalcanti, G., & Da Costa-Abreu, M. (2023, June). Exploring Automatic Hate Speech Detection on Social Media: A Focus on Content-Based Analysis. *SAGE Open, 13*. doi:10.1177/21582440231181311

Niu, M., Li, Y., Wang, C., & Han, K. (2018, July). RFAmyloid: A Web Server for Predicting Amyloid Proteins. *International journal of molecular sciences, 19*. doi:10.3390/ijms19072071

Nobel, J., Kononova, A., Briaire, J., Frijns, J., & Bäck, T. (2022, November). Optimizing Stimulus Energy for Cochlear Implants with a Machine Learning Model of the Auditory Nerve. *Optimizing Stimulus Energy for Cochlear Implants with a Machine Learning Model of the Auditory Nerve*. doi:10.48550/arXiv.2211.07285

Oh, S., Jang, K., Kim, J., & Moon, I. (2022). Online state of charge estimation of lithium-ion battery using surrogate model based on electrochemical model. Dans L. Montastruc, & S. Negny (Éd.), *32nd European Symposium on Computer Aided Process Engineering* (Vol. 51, pp. 1447-1452). Elsevier. doi:https://doi.org/10.1016/B978-0-323-95879-0.50242-3

Patel, S., Patole, R., & Metkar, S. (2022). Text Analysis And Classification Using Topic Modeling. *2022 International Conference on Signal and Information Processing (IConSIP)*, (pp. 1-6). doi:10.1109/ICoNSIP49665.2022.10007479

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Peng, J., Jury, E., Dönnes, P., & Ciurtin, C. (2021, September). Machine Learning Techniques for Personalised Medicine Approaches in Immune-Mediated Chronic Inflammatory Diseases: Applications and Challenges. *Frontiers in Pharmacology, 12*. doi:10.3389/fphar.2021.720694

Poojitha, K., Sai Charish, A., Kuamr Reddy, M. A., & Ayyasamy, S. (2023, April). Classification of social media Toxic comments using Machine learning models. *arXiv e-prints*, arXiv:2304.06934. doi:10.48550/arXiv.2304.06934

Qader, W. A., Ameen, M. M., & Ahmed, B. I. (2019). An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges. *2019 International Engineering Conference (IEC)*, (pp. 200-204). doi:10.1109/IEC47844.2019.8950616

Rahul, Kajla, H., Hooda, J., & Saini, G. (2020). Classification of Online Toxic Comments Using Machine Learning Algorithms. *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, (pp. 1119-1123). doi:10.1109/ICICCS48265.2020.9120939

Rasel, R. I., Sultana, N., Akhter, S., & Meesad, P. (2018, September). Detection of Cyber-Aggressive Comments on Social Media Networks: A Machine Learning and Text mining approach., (pp. 37-41). doi:10.1145/3278293.3278303

Revathy, G., Alghamdi, S. A., Alahmari, S. M., Yonbawi, S. R., Kumar, A., & Haq, M. A. (2022). Sentiment analysis using machine learning: Progress in the machine intelligence for data science. *Sustainable Energy Technologies and Assessments, 53*, 102557. doi:https://doi.org/10.1016/j.seta.2022.102557

Risch, J., & Krestel, R. (2020, January). Toxic Comment Detection in Online Discussions. doi:10.1007/978-981-15-1216-2_4

Robinson, K. G., & Akins, R. E. (2021). Chapter 24 - Machine learning in epigenetic diseases. Dans T. O. Tollefsbol (Éd.), *Medical Epigenetics (Second Edition)* (éd. Second Edition, Vol. 29, pp. 513-525). Academic Press. doi:https://doi.org/10.1016/B978-0-12-823928-5.00038-4

Satrya, R. N., Pratiwi, O. N., Fa'rifah, R. Y., & Abawajy, J. (2022). Cryptocurrency Sentiment Analysis on the Twitter Platform Using Support Vector Machine (SVM) Algorithm. *2022 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS)*, (pp. 1-5). doi:10.1109/ICADEIS56544.2022.10037413

Sharma, D. K., Chatterjee, M., Kaur, G., & Vavilala, S. (2022). 3 - Deep learning applications for disease diagnosis. Dans D. Gupta, U. Kose, A. Khanna, & V. E. Balas (Éd.), *Deep Learning for Medical Applications with Unique Data* (pp. 31-51). Academic Press. doi:https://doi.org/10.1016/B978-0-12-824145-5.00005-8

Shrivastava, D., Sanyal, S., Maji, A. K., & Kandar, D. (2020). Chapter 17 - Bone cancer detection using machine learning techniques. Dans S. Paul, & D. Bhatia (Éd.), *Smart Healthcare for Disease Diagnosis and Prevention* (pp. 175-183). Academic Press. doi:https://doi.org/10.1016/B978-0-12-817913-0.00017-1

SijiGeorgeC, G., & B.Sumathi. (2020). Grid Search Tuning of Hyperparameters in Random Forest Classifier for Customer Feedback Sentiment Prediction. *International Journal of Advanced Computer Science and Applications, 11*. Récupéré sur https://api.semanticscholar.org/CorpusID:222433720

Singh, E., Kuzhagaliyeva, N., & Sarathy, S. M. (2022). Chapter 9 - Using deep learning to diagnose preignition in turbocharged spark-ignited engines. Dans J. Badra, P. Pal, Y. Pei, & S. Som (Éd.), *Artificial Intelligence and Data Driven Optimization of Internal Combustion Engines* (pp. 213-237). Elsevier. doi:https://doi.org/10.1016/B978-0-323-88457-0.00005-9

Sinharay, S. (2010). Discrete Probability Distributions. Dans P. Peterson, E. Baker, & B. McGaw (Éd.), *International Encyclopedia of Education (Third Edition)* (éd. Third Edition, pp. 132-134). Oxford: Elsevier. doi:https://doi.org/10.1016/B978-0-08-044894-7.01721-8

Sinnott, R. O., Duan, H., & Sun, Y. (2016). Chapter 15 - A Case Study in Big Data Analytics: Exploring Twitter Sentiment Analysis and the Weather. Dans R. Buyya, R. N. Calheiros, & A. V. Dastjerdi (Éd.), *Big Data* (pp. 357-388). Morgan Kaufmann. doi:https://doi.org/10.1016/B978-0-12-805394-2.00015-5

Srivastava, R., Kumar, S., & Kumar, B. (2023). 7 - Classification model of machine learning for medical data analysis. Dans T. Goswami, & G. R. Sinha (Éd.), *Statistical Modeling in Machine Learning* (pp. 111-132). Academic Press. doi:https://doi.org/10.1016/B978-0-323-91776-6.00017-8

Subasi, A. (2020). Chapter 3 - Machine learning techniques. Dans A. Subasi (Éd.), *Practical Machine Learning for Data Analysis Using Python* (pp. 91-202). Academic Press. doi:https://doi.org/10.1016/B978-0-12-821379-7.00003-5

Subasi, A. (2020). Chapter 5 - Other classification examples. Dans A. Subasi (Éd.), *Practical Machine Learning for Data Analysis Using Python* (pp. 323-390). Academic Press. doi:https://doi.org/10.1016/B978-0-12-821379-7.00005-9

Sumanth, P., Samiuddin, S., Jamal, K., Domakonda, S., & Shivani, P. (2022). Toxic Speech Classification using Machine Learning Algorithms. *2022 International Conference on Electronic Systems and Intelligent Computing (ICESIC)*, (pp. 257-263). doi:10.1109/ICESIC53714.2022.9783475

Syed, K., Sleeman, W., Nalluri, J., Kapoor, R., Hagan, M., Palta, J., & Ghosh, P. (2020, January). Artificial intelligence methods in computer-aided diagnostic tools and decision support analytics for clinical informatics. doi:10.1016/B978-0-12-817133-2.00002-1

Syeda, H. B., Syed, M., Sexton, K. W., Syed, S., Begum, S., Syed, F., . . . Yu Jr, F. (2021, January 11). Role of Machine Learning Techniques to Tackle the COVID-19 Crisis: Systematic Review. *JMIR Med Inform, 9*, e23811. doi:10.2196/23811

Trivedi, M., Sharma, S., Soni, N. A., & Nair, S. (2015). Comparison of Text Classification Algorithms. *International journal of engineering research and technology, 4*. Récupéré sur https://api.semanticscholar.org/CorpusID:61323088

Turing, A. M. (1995). Lecture to the London Mathematical Society on 20 February 1947. 1986. *M.D. computing : computers in medical practice, 12 5*, 390-7. Récupéré sur https://api.semanticscholar.org/CorpusID:8597454

Umoh, U. A., Eyoh, I. J., Murugesan, V. S., & Nyoho, E. E. (2022). Chapter 14 - Fuzzy-machine learning models for the prediction of fire outbreaks: A comparative analysis. Dans R. Pandey, S. K. Khatri, N. kumar Singh, & P. Verma (Éd.), *Artificial Intelligence and Machine Learning for EDGE Computing* (pp. 207-233). Academic Press. doi:https://doi.org/10.1016/B978-0-12-824054-0.00025-3

Vel, S. (2021, March). Pre-Processing techniques of Text Mining using Computational Linguistics and Python Libraries., (pp. 879-884). doi:10.1109/ICAIS50930.2021.9395924

Wang, Z., & Guan, H. (2020). Research on Named Entity Recognition of Doctor-Patient Question Answering Community Based on BiLSTM-CRF Model. *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, (pp. 1641-1644). doi:10.1109/BIBM49941.2020.9313535

Weber, A. (2009). *Manual on Hate Speech.* Conseil de l'Europe. Récupéré sur https://books.google.ca/books?id=d74nxD9oRkUC

Xie, J., Chen, B., Gu, X., Liang, F., & Xu, X. (2019). Self-Attention-Based BiLSTM Model for Short Text Fine-Grained Sentiment Classification. *IEEE Access, 7*, 180558-180570. doi:10.1109/ACCESS.2019.2957510

Xu, S., & Zheng, S. (2022). DDoS Attack Detection for Cloud Control System Based on BiLSTM Algorithm. *2022 China Automation Congress (CAC)*, (pp. 406-411). doi:10.1109/CAC57257.2022.10056023

Yelne, P. (2023). *Machine Learning & AI.* Codegyan. Récupéré sur https://books.google.ca/books?id=quXAEAAAQBAJ

Yousaf, K., & Nawaz, T. (2022, January). A Deep Learning-Based Approach for Inappropriate Content Detection and Classification of YouTube Videos. *IEEE Access, 10*, 1-1. doi:10.1109/ACCESS.2022.3147519

Zaman, S., Hasan, M., Sakline, R., Das, D., & Alam, M. (2021, December). A Comparative Analysis of Optimizers in Recurrent Neural Networks for Text Classification. *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)* (pp. 1-6). Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/CSDE53843.2021.9718394

Zhang, W., Yoshida, T., & Tang, X. (2011). A comparative study of TF*IDF, LSI and multi-words for text classification. *Expert Systems with Applications, 38*, 2758-2765. doi:https://doi.org/10.1016/j.eswa.2010.08.066

Zhang, Y., Jin, R., & Zhou, Z.-H. (2010, December). Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics, 1*, 43-52. doi:10.1007/s13042-010-0001-0

Zhao, W., Zhang, G., Yuan, G., Liu, J., Shan, H., & Zhang, S. (2020). The Study on the Text Classification for Financial News Based on Partial Information. *IEEE Access, 8*, 100426-100437. doi:10.1109/ACCESS.2020.2997969