INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

RAPPORT DE PROJET

Analyse de sentiment

Etudiant

MAHOUGNON AYENA

ENCADREMENT

M. BRUNO PORTIER

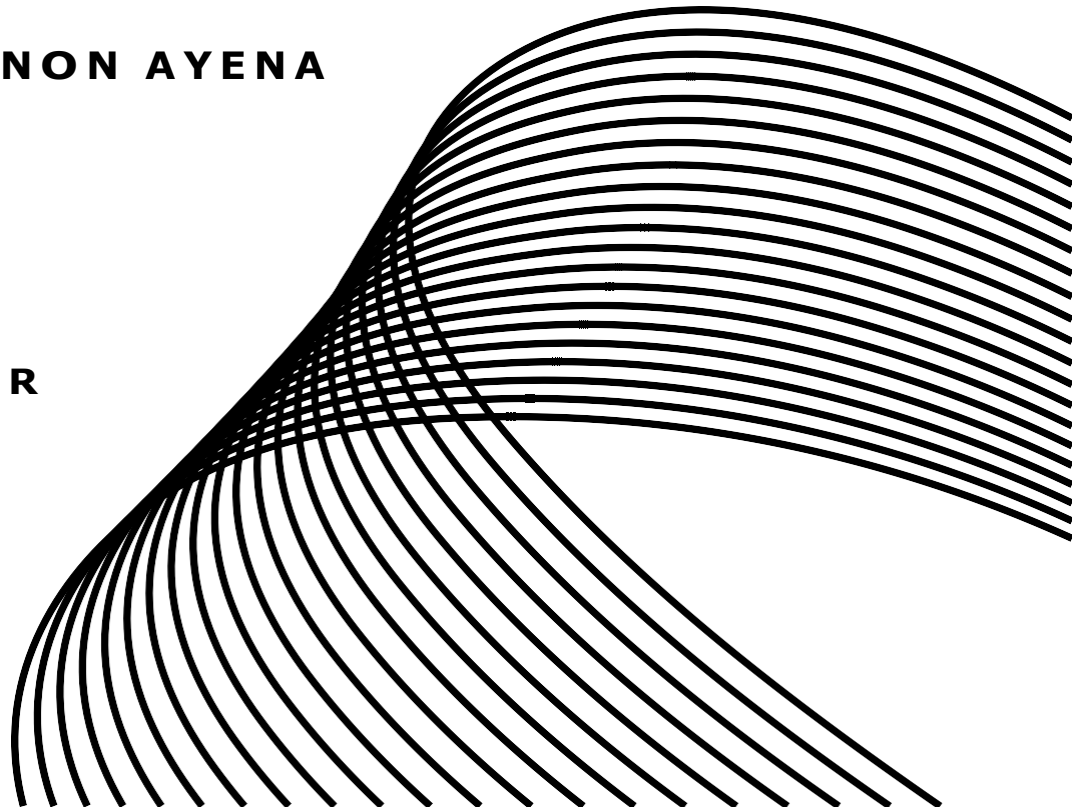


Table des matières

1 Introduction

Les conversations en ligne peuvent être très violentes, exprimer des idées et des opinions de manière libre et respectueuse devient systématiquement un défi. Le risque d'abus et de harcèlement numérique conduit souvent les individus à se taire, abandonnant ainsi leur quête de diversité d'opinions. Ceci a également pour conséquence d'affaiblir le débat public, qui se déroule de plus en plus sur les plateformes numériques. Ces plateformes luttent pour orchestrer des échanges fluides et respectueux, conduisant nombre d'entre elles à restreindre, voire supprimer les commentaires des utilisateurs.

Cependant, cette approche peut parfois ressembler à une forme de censure, mettant en danger le droit fondamental à la liberté d'expression. Si les plateformes ont la responsabilité de créer un environnement sûr pour leurs utilisateurs, il est essentiel qu'elles n'étouffent pas les voix dissidentes ou les points de vue alternatifs. Le défi est donc de trouver un équilibre entre la protection des utilisateurs et le respect de leur liberté d'expression, ce qui nécessite une nouvelle approche pour modérer et gérer les conversations en ligne.

C'est à cet effet que Kaggle a lancé un concours en collaboration avec l'équipe Conversation AI, une initiative de recherche fondée par Jigsaw et Google, qui consiste à développer un modèle de machine Learning capable de détecter différentes formes de toxicité comme les menaces, les obscénités, les insultes et la haine en ligne. Le concours peut être consulté sur kaggle(https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge)

C'est dans ce contexte que se situe notre projet. Nous nous baserons sur un ensemble de données constitué de commentaires issus de modifications sur les pages de discussion de Wikipédia. Le modèle final devrait permettre une interaction en ligne plus productive et respectueuse et ainsi aider les plateformes numériques à affiner la modération des contenus. Ce projet fera appel aux notions de NLP notamment l'analyse de sentiment qui vise à identifier et à extraire les sentiments subjectifs dans les sources de données textuelles.

Nous ferons dans un premier temps un état de l'art des techniques d'analyse de sentiments. Avant de poursuivre avec une analyse exploratoire des données, nous présenterons également notre environnement technique et les outils utilisés lors de ce projet. Ensuite, nous passerons à la phase de modélisation. En finalisation, nous mettrons en œuvre le déploiement web de notre modèle final.

2 Etat de l'art dans le domaine de l'analyse de sentiment

2.1 Qu'est-ce que le Natural Language Processing ?

2.1.1 Petite histoire

Le Natural Language Processing, ou Traitement Automatique du Langage, n'est pas une discipline nouvelle. Son origine remonte à la fin de la deuxième guerre mondiale, avec des recherches portant principalement sur la traduction automatique entre différentes langues. En 1954, un ordinateur réussit à traduire automatiquement 60 phrases du Russe à l'Anglais. La publication en 1957 du livre Syntactic structures par Noam Chomsky fut une révolution pour le domaine. Il y montra notamment qu'il existe des caractéristiques communes à tous les

langages et inventa un type de grammaire qui convertit le langage naturel en une forme compréhensible par des ordinateurs. A partir des années 80, l'augmentation de la capacité de traitement des ordinateurs, puis le développement d'Internet et de la communication textuelle numérisée (sms, emails, réseaux sociaux…), ainsi que plus récemment l'émergence d'infrastructures Big Data et d'algorithmes d'Intelligence Artificielle ont permis une explosion des capacités et des applications du Natural Language Processing.

2.1.2 Définition et angle d'approche

Le Traitement Automatique du Langage ou Natural Language Processing en Anglais, correspond à un cycle automatisé par l'informatique de lecture/correction/analyse de données textuelles pour en tirer différents types d'information. Une de ses déclinaisons fréquemment utilisées pour la recherche de données s'appelle le « Text Mining » (ensemble de méthodes, de techniques et d'outils pour exploiter les documents non structurés que sont les textes écrits). De plus, le Traitement Automatique du Langage est de nos jours souvent supporté par des algorithmes d'Intelligence Artificielle ou Machine Learning.

2.2 Qu'est-ce que l'analyse de sentiment ?

L'analyse de sentiments, également connue sous le nom de fouille d'opinions, est l'étude des opinions, des sentiments, des évaluations, des attitudes, des émotions et de la subjectivité envers différentes entités telles que les produits, les services, les organisations, les individus, les problèmes, les événements, les sujets et leurs attributs. Ce domaine est large et comprend plusieurs tâches sous différents noms tels que l'analyse de sentiments, l'extraction d'opinions, l'analyse de la subjectivité, l'analyse des affects, l'analyse des émotions et l'exploration des critiques. Les termes "analyse des sentiments" et " fouille d'opinions" ont été mentionnés pour la première fois en 2003 dans les travaux de Nasukawa et Yi et Dave, Lawrence et Pennock respectivement. Cependant, la recherche sur les sentiments et les opinions remonte à 2001 avec les travaux de Das et Chen, Morinaga et al., Pang, Lee et Vaithyanathan, Tong, Turney et Wiebe.
Bien que le traitement du langage naturel (NLP) ait une longue histoire, peu de recherches ont été menées sur les opinions et les sentiments des gens avant l'an 2000. Depuis lors, ce domaine est devenu un champ de recherche très actif.

L'analyse de sentiment fait appel à la notion centrale d'opinion.

2.2.1 Qu'est-ce qu'une opinion ?

Une opinion est un quintuple, $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$, où ei est le nom d'une entité, aij est un aspect de e_i, s_{ijkl} est le sentiment sur l'aspect a_{ij} de l'entité e_i, h_k est le détenteur de l'opinion, et t_l est le moment où l'opinion est exprimée par h_k. Le sentiment s_{ijkl} est positif, négatif ou neutre, ou exprimé avec différents niveaux de force/intensité, par exemple de 1 à 5 étoiles, comme utilisé par la plupart des sites Web.

Ces cinq composants sont essentiels. L'absence de l'un d'entre eux peut limiter la portée de l'analyse et en réduire la pertinence pour les applications pratiques.

Dans l'exemple « (1) I bought an iPhone a few days ago. (2) It was such a nice phone. (3) The touch screen was really cool. (4) The voice quality was clear too. (5) However, my mother was

mad with me as I did not tell her before I bought it. (6) She also thought the phone was too expensive, and wanted me to return it to the shop . . . » (Liu et Zhang, page 416)

(1) relate un fait objectif. (2), (3) et (4) expriment une opinion subjective plutôt positive ; (5) et (6) une opinion négative.

L'entité iPhone en général est le sujet de (2), (3), (4) et (6) sont relatifs respectivement aux aspects « touch screen », « voice quality » et « price » de l'iPhone ; « me » est le sujet de (5).

« I » (je) est le titulaire des opinions (2), (3) et (4) ; pour (5) et (6), il s'agit de « mother ».

Sur la base de cette définition, l'objectif de l'analyse de sentiments est défini comme la détermination de tous les quintuplés d'un texte donné.

Cette analyse peut être menée en fonction de plusieurs niveaux de granularité. Nous avons l'analyse au niveau du document (document level sentiment), l'individu statistique est le document, l'analyse au niveau des phrases (sentence level sentiment) et l'analyse au niveau des aspects (Entity and Aspect level).

2.2.2 Différents Niveaux de l'analyse de sentiments

Document level: L'objectif à ce niveau est de déterminer si un document d'opinion exprime un sentiment positif ou négatif. Par exemple, pour une critique de produit, le système évalue si l'opinion générale exprimée est positive ou négative. Cette tâche est souvent appelée "classification des sentiments au niveau du document". Cependant, elle ne s'applique qu'à des documents qui évaluent une seule entité et ne peut pas être utilisée pour des documents qui évaluent ou comparent plusieurs entités.

Sentence level: L'extraction d'opinion à niveau de phrase est un problème de classification consistant à classer chaque phrase dans un document d'opinion en tant qu'opinion positive, négative ou neutre. Ce processus est généralement effectué en deux étapes : la classification de la subjectivité pour distinguer les opinions (phrases subjectives) des faits (phrases objectives) et la classification du sentiment des phrases pour classer les phrases subjectives en tant que positives, négatives ou neutres. Les phrases plus complexes peuvent nécessiter des techniques avancées. Cependant, il est important de noter que la subjectivité n'est pas synonyme de sentiment, car certaines phrases objectives peuvent également impliquer des opinions. Par exemple, "Nous avons acheté la voiture le mois dernier et l'essuie-glace est tombé" peut impliquer une opinion négative.

Entity and Aspect level: L'analyse de la polarité d'un document ou d'une phrase d'opinion est utile, mais ne fournit pas une analyse détaillée des opinions. Elle ne permet pas de savoir précisément ce que les gens aiment et n'aiment pas. Cette classification ne permet pas d'identifier les sentiments et les cibles d'opinion dans un document textuel. Par conséquent, l'analyse des sentiments au niveau des aspects est une étape cruciale pour une analyse plus fine. L'analyse des sentiments au niveau des aspects définit une opinion comme un sentiment et une cible, et vise à découvrir les cibles spécifiques et à spécifier leur polarité de sentiment.

L'analyse des sentiments au niveau des aspects consiste en deux tâches principales : l'extraction d'aspects et la classification d'aspects. La première tâche, appelée extraction de cibles d'opinion (Liu, 2015), consiste à extraire les entités et leurs aspects d'un texte. Les entités peuvent être des produits, des services, des événements, etc. Les aspects sont les attributs et les composants

des entités, qui peuvent être exprimés implicitement ou explicitement. La deuxième tâche consiste à associer la polarité (positive, négative, neutre) à chaque aspect extrait.

2.3 Techniques d'extraction des données

Les techniques d'extraction de caractéristiques jouent un rôle crucial dans le traitement de textes. Elles permettent de convertir les données brutes en une forme numérique qui peut être utilisée pour diverses applications telles que la classification de documents, la détection de sentiments, la reconnaissance d'entités nommées, etc. Les techniques d'extraction de caractéristiques sont largement utilisées pour représenter un document sous forme de vecteur numérique. Cette représentation peut être utilisée pour faire des analyses statistiques, effectuer des comparaisons entre différents documents et les classer en groupes similaires. Les techniques d'extraction de caractéristiques peuvent être basées sur les fréquences des mots, la signification des mots, le contexte des mots, etc. Les algorithmes de traitement du langage naturel ont fait d'énormes progrès ces dernières années, permettant ainsi l'extraction de caractéristiques plus précises et plus utiles pour les tâches spécifiques.

Parmi ces techniques d'extraction de caractéristiques on a TF-IDF, 2.3.1 CountVectorizer et Word2Vec qui sont largement utilisées pour représenter les documents sous forme de vecteurs numériques en traitement de textes. Chacune de ces techniques a ses propres avantages et inconvénients, et le choix de la technique à utiliser dépendra des exigences spécifiques de chaque tâche.

2.3.1 CountVectorizer

Le CountVectorizer, ou vecteur de fréquence, est une méthode d'analyse textuelle qui transforme un ensemble de textes en une matrice de fréquences de mots. Chaque texte est représenté comme un vecteur dans un espace vectoriel de haute dimension, où chaque dimension correspond à un mot spécifique dans le corpus. Les entrées du vecteur représentent la fréquence de chaque mot dans le texte donné. Dans le contexte de l'analyse de sentiments, ces vecteurs peuvent servir de caractéristiques d'entrée pour des modèles de machine learning, tels que la régression logistique ou les machines à vecteurs de support, qui peuvent prédire le sentiment d'un texte en fonction de la fréquence de ses mots.

L'avantage du CountVectorizer est qu'il est facile à comprendre et à implémenter, et qu'il ne nécessite pas de prétraitement complexe ou d'entraînement intensif en ressources. Il peut également capturer des informations utiles sur la présence et l'abondance de mots spécifiques qui peuvent être associés à des sentiments positifs ou négatifs.

Cependant, le CountVectorizer présente également des limitations. Il ne tient pas compte de l'ordre des mots, de leur contexte ou de leur signification sémantique. Il peut également être sensible aux mots très fréquents qui ne portent pas beaucoup d'information sentimentale, ainsi qu'aux mots très rares qui peuvent causer du bruit. De plus, la haute dimensionnalité des vecteurs peut poser des défis en termes de calcul et de surapprentissage.

2.3.2 One-Hot Encoding

Le One-Hot Encoding est une technique de vectorisation largement utilisée dans le traitement du langage naturel et l'analyse de sentiment. Elle consiste à représenter chaque mot unique dans un texte en tant que vecteur dans un espace à n dimensions, où n est le nombre total de mots uniques dans le corpus de texte. Chaque mot est représenté par un vecteur où toutes les dimensions sont égales à zéro, à l'exception de celle correspondant à ce mot qui est égale à un.

L'avantage du One-Hot Encoding est sa simplicité et sa facilité d'interprétation. Il fournit une représentation binaire claire et distincte pour chaque mot, ce qui peut être particulièrement utile pour les modèles qui ne gèrent pas les données catégorielles.

Cependant, le One-Hot Encoding présente également des inconvénients significatifs lorsqu'il est appliqué à l'analyse de sentiment. Premièrement, il crée des vecteurs de grande dimension, ce qui peut être problématique pour les grands corpus de texte. Deuxièmement, il ne capture pas le contexte ni la relation sémantique entre les mots, ce qui est souvent crucial pour comprendre le sentiment. Enfin, cette technique traite tous les mots comme étant indépendants les uns des autres et attribue la même importance à chaque mot, ce qui peut ne pas être idéal dans l'analyse de sentiment où certains mots peuvent avoir plus de poids.

2.3.3 TF-IDF (Terme Fréquence - Inverse Document Frequency)

TF-IDF (Terme Fréquence - Inverse Document Frequency) est une technique d'extraction de caractéristiques qui considère à la fois la fréquence des mots dans un document et leur rareté à travers un ensemble de documents. Cette technique permet de mettre en évidence les mots importants dans un document en pondérant la fréquence d'un mot par sa rareté dans l'ensemble des documents.

La formule pour le TF-IDF est la suivante : TF-IDF(t, d) = TF(t, d) * IDF(t)

Où : t est un terme, d est un document, TF(t, d) est la fréquence du terme t dans le document d

IDF(t) est le logarithme inverse de la fréquence d'occurrence du terme t dans l'ensemble des documents

L'avantage de TF-IDF par rapport au CountVectorizer est qu'il tient compte de la signification sémantique des mots. Les mots couramment utilisés dans tous les documents auront une pondération plus faible dans TF-IDF, tandis que les mots rares et spécifiques à un document auront une pondération plus élevée. Cela permet de capturer de manière plus précise la signification du document.

L'implémentation de TF-IDF est également relativement simple, bien que légèrement plus complexe que celle de CountVectorizer. Il s'agit simplement de calculer la fréquence d'un mot dans un document et de la pondérer en fonction de sa rareté dans l'ensemble des documents.

2.3.4 Word2Vec

Word2Vec est une technique d'apprentissage profond qui permet de représenter les mots en tant que vecteurs de nombres continus. Les vecteurs ainsi formés capturent les propriétés

linguistiques telles que la proximité contextuelle, ce qui les rend particulièrement utiles pour les tâches telles que la classification de documents, la reconnaissance d'entités nommées, etc.

Word2Vec est une technique d'apprentissage en profondeur pour représenter des mots sous forme de vecteurs de nombres. Au lieu de considérer les mots comme des entiers distincts ou des vecteurs basés sur la fréquence d'occurrence comme avec CountVectorizer ou TF-IDF, Word2Vec utilise un réseau de neurones pour apprendre une représentation continue des mots.

L'avantage de Word2Vec par rapport aux autres techniques d'extraction de caractéristiques est qu'il capture les relations sémantiques entre les mots. Les mots similaires sont représentés par des vecteurs proches dans l'espace vectoriel, ce qui permet de résoudre des tâches telles que la similarité sémantique et la complétion de mots.

Word2Vec est souvent utilisé en combinaison avec d'autres techniques d'extraction de caractéristiques pour améliorer les performances de modèles de traitement de textes. Par exemple, on peut utiliser les vecteurs Word2Vec pour initialiser les poids d'un modèle de traitement de textes avant d'entraîner le modèle sur des données.

En combinant TF-IDF et Word2Vec, on peut bénéficier des avantages des deux méthodes. TF-IDF permet de donner du poids aux mots en fonction de leur importance dans le document, tandis que Word2Vec permet de capturer les relations sémantiques entre les mots. Cette combinaison peut donc fournir une représentation plus riche et plus significative des mots, ce qui peut améliorer les performances des modèles d'apprentissage automatique pour des tâches telles que la classification de textes ou l'analyse des sentiments.

2.4 Approches ou Types d'algorithmes d'analyse de sentiments

Aujourd'hui l'analyse de sentiments se concentre sur l'attribution d'une polarité à des expressions subjectives (les mots et les phrases qui expriment des opinions, des émotions, des sentiments, etc.) afin de décider de l'orientation d'un document (c.f. (Turney, 2002), (Wilson et al., 2004)) ou de la polarité positive/négative/neutre d'une opinion dans un document (c.f. (Hatzivassiloglou et al., 1997), (Yu et al., 2003), (Kim et al., 2004)). Des travaux allant au-delà ont mis l'accent sur la force d'une opinion d'expression où chaque proposition dans une phrase peut avoir un fond neutre, faible, moyen ou élevé (c.f. (Wilson et al., 2004)). Des catégories grammaticales ont été utilisées pour l'analyse de sentiments dans (Bethard et al., 2004) où des syntagmes adjectivaux comme trop riche ont été utilisés afin d'extraire des opinions véhiculant des sentiments. (Bethard et al., 2004) utilisent une évaluation basée sur la somme de scores des adjectifs et des adverbes classés manuellement, tandis que (Chklovski, 2006) utilise des méthodes fondées sur un modèle pour représenter des expressions adverbiales de degré telles que parfois, beaucoup, assez ou très fort. Ces diverses recherches nous facilitent l'élaboration de différentes méthodes pour la classification dans l'analyse de sentiment.

La classification des sentiments peut se faire généralement en approche d'apprentissage automatique (Machine Learning), approche basée sur le lexique et approche hybride. L'approche Machine Learning (ML) utilise les méthodes d'apprentissage supervisé et Non supervisé. L'approche basée sur le lexique repose sur un lexique des sentiments, une collection de termes de sentiments connus et précompilés. Il est divisé en une approche basée sur un dictionnaire et une approche basée sur un corpus qui utilisent des méthodes statistiques ou sémantiques pour trouver la polarité des sentiments. L'approche basée sur un dictionnaire qui dépend de la recherche

de mots de semences d'opinion, puis recherche le dictionnaire de leurs synonymes et antonymes. L'approche basée sur le corpus commence par une liste de départ de mots d'opinion, puis trouve d'autres mots d'opinion dans un grand corpus pour aider à trouver des mots d'opinion avec des orientations spécifiques au contexte. L'approche hybride combine les deux techniques et est très courante, les lexiques de sentiment jouant un rôle clé dans la majorité des méthodes.

2.4.1 Approche basée sur le lexique

L'approche basée sur le lexique est une méthode non supervisée d'analyse des sentiments dans les documents qui utilise un lexique pré-composé de termes de sentiments étiquetés pour déterminer la polarité générale d'un texte.

Ces termes sont utilisés pour exprimer des sentiments positifs ou négatifs. Les termes qui composent le lexique sont généralement des adjectifs et des adverbes, mais les noms et les verbes doivent également être pris en compte. L'objectif de l'utilisation d'un tel lexique est de déterminer le sentiment général d'un texte donné, en partant du principe que la polarité collective d'une phrase ou d'un document est la somme des polarités des phrases ou des mots individuels. Le document est classifié comme positif si la somme est positive, négatif si la somme est négative et neutre si la somme est égale à zéro

Il existe deux méthodes principales dans cette approche : basée sur le corpus et basée sur le dictionnaire. La méthode basée sur le corpus utilise la probabilité d'occurrence des mots de sentiments conjointement avec des mots positifs ou négatifs trouvés dans une grande quantité de textes, tandis que la méthode basée sur le dictionnaire utilise un dictionnaire existant de mots d'opinion avec leur force de sentiment (Ravi and Ravi, 2015). Le processus peut inclure l'utilisation de modificateurs de sentiments tels que les négations et les intensificateurs.

Nous pouvons illustrer l'approche basée sur le dictionnaire avec la phrase : « Cette journée est magnifique ». Dans ce contexte, nous considérons le dictionnaire suivant :

Beau : +2, magnifique : +3, terrible : -3…

En appliquant ce dictionnaire nous obtenons un total de +3 (Cette : 0, journée : 0, est : 0, magnifique : +3) indiquant une polarité positive.

2.4.2 Approche Machine Learning

S'appuyant sur des techniques d'apprentissage automatique (Machine Learning ou ML), cette deuxième approche utilise des outils de traitement du langage naturel (Natural Langage Processing ou NLP), qui est un sous-domaine de la linguistique computationnelle. Les méthodes statistiques ou basées sur l'apprentissage automatique créent un classifieur de sentiments utilisant un ensemble de caractéristiques sélectionnés dans le contenu textuel (sacs de mots, n-grammes, . . .) et appliquent des modèles de classification pour prédire la polarité. La tâche de sélection des fonctionnalités peut être résolue en utilisant soit des méthodes basées sur le lexique soit sur des statistiques. Basé sur le lexique, les méthodes nécessitent une annotation humaine, les autres sont automatiques.

L’avantage de l’approche par apprentissage automatique est que les valeurs de la polarité du lexique sont principalement calculées par estimation statistique. Cela minimise donc le travail humain pour la création de ressources et est avantageux en termes de couverture. Cependant, si les textes qui sont donnés en apprentissage ne sont pas représentatifs du texte auquel le modèle prédictif est appliqué, les prévisions peuvent s’avérer inexactes. Pour un état de l’art plus approfondi sur ces méthodes, voir les travaux de (Kumar et Prabhu, 2018).

2.4.3 Approche hybride

L’approche hybride comprend l’apprentissage automatique et la méthode basée sur le lexique (éventuellement au sein de règles linguistiques). Différents classifieurs de sentiments fondés sur un lexique ou sur les méthodes d’apprentissage sont utilisés en cascade, de sorte que lorsqu’un classifieur échoue, le suivant prend un tour pour classer, et ainsi de suite jusqu’à ce que le texte restant soit catégorisé. Une approche hybride peut consister à utiliser des techniques d’apprentissage automatique pour suggérer des nouveaux termes et construire des dictionnaires qui seront plus riches et plus spécifiques pour un terme dans son contexte. Les modèles hybrides donnent souvent de très bons résultats en termes de précision, citons : (Mudinas et al., n.d.) avec 89.64%, (Bahrainian et Dengel, n.d.) avec 89.13%, (Zhang et al., 2015) avec 85.40%, (Malandrakis et al., 2013) avec 85.80% (Shoukry et Rafea, n.d.) avec 80.90%. Par exemple (Gupta et Joshi, 2019) présente une analyse de sentiments sur Twitter grâce à une approche hybride dans laquelle le vecteur de caractéristiques basé sur SentiWordNet (SWN) sert d’entrée au modèle de classification SVM. Leurs résultats expérimentaux montrent que le vecteur de caractéristiques basé sur le contexte obtenu conduit à une amélioration de 2 à 6% du système d’analyse des sentiments sur Twitter.

Les différentes techniques et les algorithmes les plus utilisés pour la classification des sentiments sont illustrés sur la figure suivante :

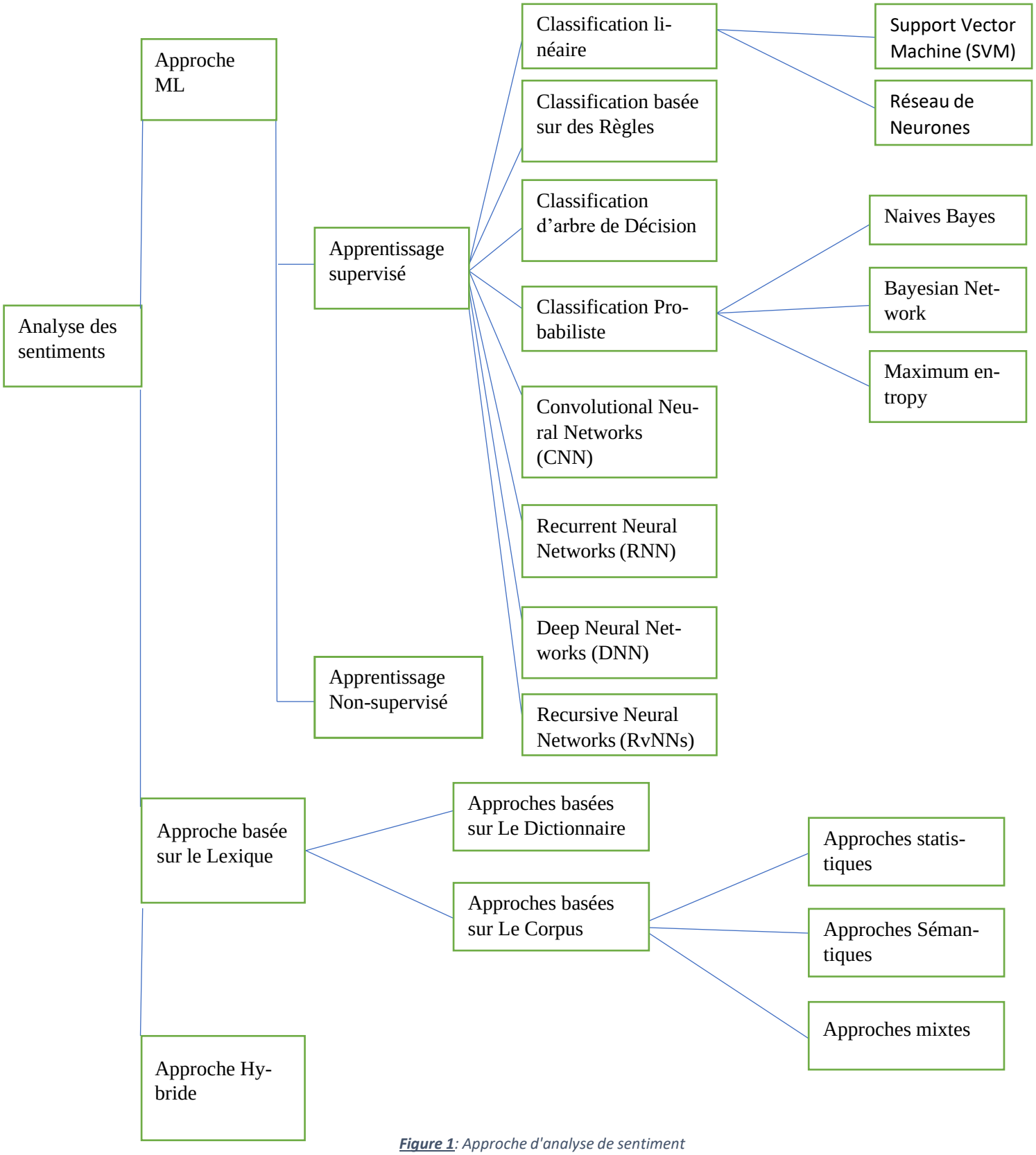


***Figure 1**: Approche d'analyse de sentiment*

2.5 Quelques méthodes d'apprentissage supervisé utilisées dans l'analyse de sentiments

Nous allons présenter quelques modèles de classification utilisés dans l'analyse de sentiments :

2.5.1 Machines à vecteurs de support

Les méthodes des SVM : Support Vector Machine a été introduite par Joachims [Joachims (1998), Joachims (1999)], puis utilisée par Drucker [Drucker et al. (1999)], Taira et Haruno [Taira & Haruno (1999)], et Yang et Liu [Yang & Liu (1999)]. La méthode des SVM géométriques peut être considérée comme la tentative de trouver, parmi toutes les surfaces σ_1, σ_2, ... d'un espace de dimensions |T| ce qui sépare les exemples d'apprentissage positifs des négatifs. L'ensemble d'apprentissage est donné par un ensemble de vecteurs associés à leur classe d'appartenance : (X_1, f_1), ..., (X_u, f_u), $X_j \in R^n$, $f_j \in \{+1, -1\}$ avec

– f_j représente la classe d'appartenance. Dans un problème à deux classes la première classe correspond à une réponse positive ($f_j = +1$) et la deuxième classe correspond à une réponse négative ($f_j = -1$)

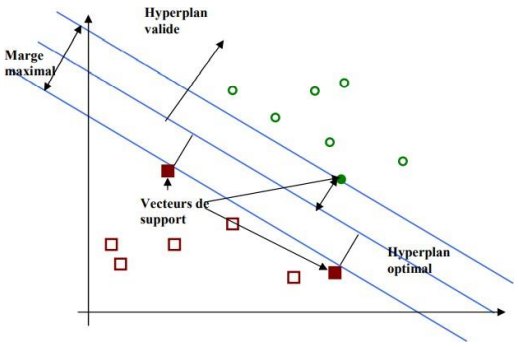
– X_j représente le vecteur du texte numéro j de l'ensemble d'apprentissage.

La méthode SVM sépare les vecteurs à classe positive des vecteurs à classe négative par un hyperplan défini par l'équation suivante : $W \otimes X + b = 0$, $W \in R^n$, $b \in R$.

En général, un tel hyperplan n'est pas unique. La méthode SVM détermine l'hyperplan optimal en maximisant la marge : la marge est la distance entre les vecteurs étiquetés positifs et les vecteurs étiquetés négatifs. L'ensemble d'apprentissage n'est pas nécessairement séparable linéairement, des variables d'écart ξ_j sont introduites pour tous les X_j. Ces ξ_j prennent en compte l'erreur de classification, et doivent satisfaire les inégalités suivantes :

– $W \otimes X + b \geq 1 - \xi_j$,

– $W \otimes X + b \leq 1 + \xi_j$,



Les points carrés et les points ronds représentent respectivement les réponse positives et négatives, les lignes représentent les surfaces de décision. La surface de décision σ_i montre le meilleur cas.

En prenant en compte ces contraintes, nous devons minimiser la fonction d'objectif suivante : $\frac{1}{2} \| W \|^2 + C \sum_{j=1}^{u} \xi_j$. Le premier terme de cette fonction correspond à la taille de la marge et le second terme représente l'erreur de classification, avec u représentant le nombre de vecteurs de l'ensemble d'apprentissage. Trouver la fonction objective précédente revient à résoudre le problème quadratique suivant : trouver la fonction de décision f telle que : f(X) = signe(g(X) dans laquelle la fonction g(X) est :

$$g(X) = \sum_{i=1}^{n} \lambda_i f_i X_i \otimes X + b$$

Avec :

Signe(x) représente la fonction suivante :

- Si X>0 alors Signe(x)=1
  Si <0 alors Signe(x)= -1

2.5.2 Classification naïve bayésienne (Naive Bayes ou NB)

La classification naïve bayésienne (Naive Bayes ou NB) (Li, 2010), (Li, 2014) : est une famille d'algorithmes utilisée pour construire un classifieur qui attribue des étiquettes aux instances de problèmes.

Un peu plus formellement, le problème de classification se traduit ainsi : "trouver la classe 'c'qui a la probabilité la plus grande étant donné le document 'd'fourni" :

$\hat{c} = \alpha r g max_c \, p(c/d)$

On ne sait pas calculer $p(c/d)$directement, on a donc besoin de simplifier un peu le problème. Ainsi, d'après le théorème de Bayes on a :

$p(c/d) = \frac{p(d/c)p(c)}{p(d)}$

L'objectif étant une maximisation sur la classe c, $p(d)$ n'influence pas le résultat. Le problème peut être simplifié :

$\hat{c} = \alpha r g max_c \, p(d/c)p(c)$

le document 'd'est représenté par un certain nombres de features (les mots qu'il contient), sans conserver l'ordre de ces mots (bag-of-words) et en considérant qu'ils sont indépendants. On a ainsi pour un document de N mots w_i
$p(d/c) = \prod_{i=1}^{N} p(w_i/c)$

Dans le cadre d'étude de texte, nous travaillons sur des probabilités faibles. Nous allons donc plutôt travailler à l'échelle logarithmique, ce qui ne change rien au problème de maximisation (la fonction log est monotone strictement croissante). Cela nous permet, en bonus, de travailler avec des sommes.

$\hat{c} = \alpha r g max_c \log (p(d/c)p(c)) = \alpha r g max_c log p(c) + \sum log(p(w_i/c))$

Dans le cadre d'une classification binaire de texte avec unigramme.

La question restante est donc : comment estimer $p(c)$ et $p(w_i/c)$ à partir de notre jeu de données d'entraînement. On va donc utiliser des fréquences.
La probabilité d'une classe est simplement la fréquence d'apparition de la classe dans le jeu de données d'entraînement :

$$p(\mathrm{c}) = \frac{Nc}{N_{totaldoc}}$$

Et la probabilité d'un mot dans une classe est simplement : la fréquence d'apparition de ce mot dans un type de document par rapport au nombre de mot total dans c.

$$p(w_i/c) = \frac{N_{w_i\ dans\ c}}{\sum_v N_{w_t\ dans\ c}}$$

On lisse cette probabilité pour les mots qui n'apparaitraient pas dans une classe, ce qui évite de rendre nulle notre fonction de vraisemblance si un mot est à zéro prob (lissage Laplacien) :

$$p(w_i/c) = \frac{N_{w_{-i}\ dans\ c} + 1}{\sum_v N_{w_{-t}\ dans\ c} + |V|}$$

2.5.3 Régression logistique

Les chercheurs Gamal *et al.* (2019) et Lin *et al.* (2015) ont présenté une étude empirique sur la classification de sentiments et la régression logistique qui est construite pour combiner différentes méthodes d'apprentissage machine et obtenir une performance exceptionnelle en matière de précision et de rappel.
Régression logistique : La régression logistique est un classifieur binaire. En utilisant la régression logistique, les données peuvent être classées en deux catégories, par exemple, positif ou négatif, oui ou non, courrier indésirable ou non, etc. Elle est également utilisée pour mesurer la probabilité d'un résultat binaire et résoudre des problèmes de classification. Elle fonctionne avec des variables dépendantes et indépendantes. Les variables indépendantes peuvent affecter les variables dépendantes.
L'équation (1) représente la régression linéaire, également appelée fonction logistique ou sigmoïde.

$$h_\theta(\mathrm{X}) = g(\theta^T.\mathrm{X}) \quad (1)$$

Pour la classification, nous représentons $g(\theta^T.x)$ par $g(z)$, ce qui transforme les valeurs réelles z en probabilités en utilisant la partie droite de l'équation (2), où z = θ^T.X

$$g(z) = \frac{1}{1+e^{-z}} \quad (2)$$

Le coût de la fonction peut être calculé avec l'équation (3) :

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} (y)^i log h_\theta(x)^i + (1-y^i)\log(1-h_\theta(x)^i) \quad (3)$$

où m est le nombre d'exemples d'entraînement, x(i) sont les entrées et y(i) sont les sorties. Cette équation est utilisée pour calculer le coût de la fonction en utilisant les m exemples d'entraînement. Pour calculer la descente de gradient :

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathrm{J}(\theta)$$

L'objectif de l'utilisation de la descente de gradient avec la dérivation de la fonction de coût est de minimiser la fonction de coût en choisissant les meilleures valeurs pour θ. Après la dérivation de J(θ), l'équation devient :

$\theta_j=\theta_j - \alpha \sum_{i=1}^{m} (hsz_\theta [(x)^i - (y)^i] x_j^i)$ (4)

Cette équation (4) est utilisée pour mettre à jour simultanément toutes les valeurs de θ_j afin de minimiser le coût de la fonction, en choisissant la meilleure valeur de 'α', qui est le taux d'apprentissage

La différence la plus importante entre le modèle de Naive Bayes et la régression logistique est que la régression logistique est un classificateur discriminant tandis que le modèle de Naive Bayes est un classificateur générateur.

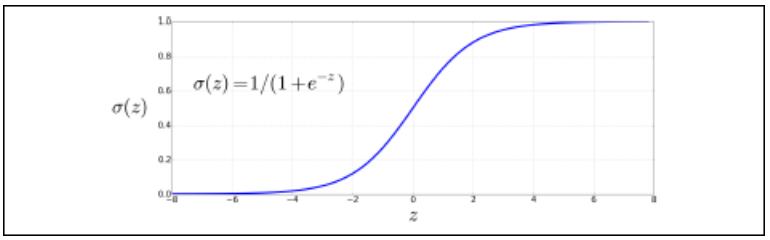


Figure 2: La courbe de la fonction sigmoïde

- ✓ Régression logistique multinomiale

La régression logistique multinomiale, également connue sous le nom de régression softmax est une généralisation du modèle binaire lorsque la variable dépendante possède plus de deux modalités. Elle utilise en effet la fonction softmax qui elle une extension de la fonction sigmoïde au cas des multi-classes.
Dans le modèle multinomial, une valeur de la variable dépendante est distinguée comme catégorie de référence. On compare alors la probabilité de succès dans les autres catégories de la variable avec la probabilité de succès de la catégorie de référence. Cet algorithme produit une matrice de coefficients de dimension C × J où Cest le nombre de classes attendues et J le nombre de caractéristiques (features). Les probabilités conditionnelles des C classes de résultats sont modélisées à l'aide de la fonction softmax suivante :

P (y=c/x; $\theta_1; \theta_2; \theta_3 \ldots; \theta_k$)= $\frac{\exp(\theta_c^T x)}{\sum_{j=1}^{k} exp(\theta_j^T x)}$ où $\theta_1; \theta_2; \theta_3 \ldots; \theta_k$ sont les paramètres.

2.5.4 Classification par la méthode des arbres de décision

Un classificateur de texte basé sur la méthode des arbres de décision est un arbre de noeuds internes qui sont marqués par des termes, les branches qui sortent des noeuds sont des tests sur les termes, et les feuilles sont marquées par catégories [Mitchell (1996)]. Ce classificateur classe un document du test dj en testant récursivement les poids des noeuds internes de vecteur $\vec{d}_y$, jusqu'à ce qu'une feuille soit atteinte. L'étiquette de ce noeud est alors attribuée à d_j. La plupart de ces classificateurs utilise une représentation du document binaire, et sont donc crée par des arbres binaires.

Il existe un certain nombre d'approches pour l'apprentissage de l'arbre de décision. Les plus populaires sont ID3 (utilisé par Fuhr [Fuhr et al. (1991)]), C4.5 (Cohen et Singer [Cohen & Singer (1999)], Joachims [Joachims (1998)]) et C5 (utilisé par Li et Jain [Li & Jain (1998)]).

Une méthode pour effectuer l'apprentissage d'un arbre de décision pour la catégorie ci consiste à vérifier si tous les exemples d'apprentissage ont la même étiquette (c_i ou ¯€), dans le cas contraire nous sélectionnons un terme t_k, et nous partitionnons l'ensemble d'apprentissage en classes de documents qui ont la même valeur pour t_k, et à la fin l'on crée les sous-arbres pour chacune de ces classes. Ce processus est répété récursivement sur les sous-arbres jusqu'à ce que chaque feuille de l'arbre généré de cette façon contienne des exemples d'apprentissage attribués à la même catégorie c_i, qui est alors choisie comme l'étiquette de la feuille. L'étape la plus importante est le choix du terme de tk pour effectuer la partition. Toutefois, une telle méthode de construction d'arbre peut faire l'objet de surapprentissage, comme certaines branches peuvent être trop spécifiques pour les données d'apprentissage. La plupart des méthodes d'apprentissage des arbres incluent une méthode pour la construction d'arbre et pour élaguer les branches trop spécifiques [Mitchell (1996)].

2.5.5 Les Réseaux de Neurones Récurrents (RNN)

Les Réseaux de Neurones Récurrents (RNN) sont des types spécifiques de réseaux de neurones artificiels (ANN) qui excellent dans la prise en compte de séquences et de motifs temporels dans des données. Ils sont largement utilisés dans de nombreux domaines d'applications liés au traitement du langage naturel (NLP), à la reconnaissance de la parole, et à la traduction automatique, entre autres. Leur efficacité a été prouvée dans la classification de texte, où ils sont couramment utilisés pour classer des textes en diverses catégories basées sur leur contenu.

Les RNN sont particulièrement adaptés aux données séquentielles car ils sont capables de maintenir en mémoire une information "historique" grâce à leur structure cyclique. Contrairement à un réseau de neurones traditionnel, où les informations circulent uniquement dans une direction (de l'entrée à la sortie), les RNN ont des boucles qui permettent aux informations de circuler dans les deux sens. Cette capacité à se souvenir des entrées précédentes donne aux RNN la capacité unique de comprendre le contexte et l'ordre des données, ce qui est essentiel pour la compréhension de la langue et la classification du texte.

Dans le contexte de la classification de texte, les RNN sont capables de traiter un document entier, un paragraphe, une phrase, ou même un seul mot, comme une séquence d'éléments interdépendants, et peuvent générer une prédiction basée sur cette séquence.

L'un des principaux avantages des RNN dans la classification de texte est leur capacité à traiter des textes de différentes longueurs. Contrairement aux autres techniques de machine learning qui nécessitent des vecteurs d'entrée de taille fixe, les RNN peuvent gérer des séquences d'entrée de longueurs différentes, ce qui est idéal pour le traitement du texte.

Cependant, les RNN traditionnels sont confrontés à un défi majeur : le problème des dépendances à long terme. C'est-à-dire que lorsqu'une séquence est trop longue, ils ont du mal à transporter l'information de ses premières parties vers ses parties ultérieures. Pour résoudre ce problème, des variantes plus sophistiquées de RNN, comme le Long Short-Term Memory (LSTM) et le Gated Recurrent Unit (GRU), ont été introduites.

En conclusion, les RNN et leurs variantes ont montré une grande promesse dans la tâche de classification de texte. Cependant, comme pour toute technologie, ils ont leurs limites et défis.

2.5.6 Les Réseaux de Neurones LSTM

Long Short-Term Memory (LSTM) est une architecture spéciale de réseau de neurones récurrent (RNN), introduite par Sepp Hochreiter et Jürgen Schmidhuber en 1997. Elle a été conçue pour résoudre le problème de disparition du gradient, une difficulté commune rencontrée lors de l'apprentissage de RNN. Au cours des deux dernières décennies, LSTM a fait l'objet d'un grand nombre de recherches et d'améliorations, et a trouvé de nombreuses applications dans diverses tâches d'apprentissage en séquence, notamment la classification de texte, la génération de texte, la traduction automatique, entre autres.

Les LSTM se distinguent des RNN traditionnels par leur utilisation de portes d'entrée, de sortie et d'oubli pour réguler le flux d'informations à travers la cellule mémoire. Ces portes permettent à l'information d'être conservée ou éliminée en fonction de son importance, ce qui facilite l'apprentissage des dépendances à long terme.

Plusieurs variantes de LSTM ont été proposées pour améliorer l'efficacité et la performance du modèle original. Par exemple, le Gated Recurrent Unit (GRU), introduit par Cho et al. en 2014, est une variante plus simple et plus efficace du LSTM, qui utilise seulement deux portes (réinitialisation et mise à jour) et fusionne la cellule d'état et la cellule cachée. D'autres améliorations comprennent le Peephole LSTM, où les portes peuvent "voir" l'état de la cellule mémoire, et le LSTM Bidirectionnel, qui permet au modèle de traiter les données dans les deux directions.

Les LSTM ont démontré des performances exceptionnelles dans diverses tâches de traitement du langage naturel (NLP). Par exemple, dans la classification de texte, ils ont été utilisés pour distinguer le spam des courriels légitimes, classer les sentiments dans les critiques de films, et bien d'autres.

Dans le domaine de la génération de texte, les LSTM ont été utilisés pour générer des poèmes, des scripts, et même des nouvelles complètes. Ils ont également été utilisés pour la traduction automatique, surpassant les modèles traditionnels basés sur les règles et les statistiques.

Bien que les LSTM soient très puissants pour traiter les données en séquence, ils ont aussi leurs limites. Par exemple, ils peuvent être assez gourmands en ressources et prendre beaucoup de temps à s'entraîner, surtout pour les grandes quantités de données. De plus, bien qu'ils soient capables de modéliser les dépendances à long terme, dans la pratique, ils peuvent encore avoir du mal avec les séquences vraiment longues.

3 Environnement Technique

Dans ce projet, nous avons utilisé une gamme d'outils et de technologies spécifiques pour concevoir, développer et déployer notre solution. Notre choix de langage de programmation était Python, un langage de haut niveau particulièrement adapté pour l'analyse de données et le développement d'applications d'apprentissage automatique, grâce à sa lisibilité, sa facilité d'apprentissage et l'accès à une vaste bibliothèque de packages.

Notre code a été écrit dans Visual Studio Code, un éditeur de code léger mais puissant qui offre des fonctionnalités avancées de codage et de débogage. Pour les calculs intensifs, tels que le traitement des réseaux de neurones, nous avons utilisé Google Colab, qui offre un environnement Jupyter Notebook dans le cloud avec accès à des ressources de calcul telles que les GPU et les TPU de Google.

Nous avons largement utilisé des packages Python spécifiques pour différentes parties du projet. Pour le prétraitement des données, nous avons utilisé pandas, numpy, re et string. Pour la visualisation des données, nous avons utilisé matplotlib et seaborn. Pour le prétraitement du texte, nous avons utilisé le lemmatiseur WordNet de nltk.stem. La bibliothèque scikit-learn a été utilisée pour le traitement des données, la mise en œuvre des modèles de machine learning et l'évaluation de leurs performances. De plus, nous avons utilisé la bibliothèque wordcloud pour visualiser les mots les plus fréquents dans les commentaires.
En termes de modélisation, nous avons expérimenté avec une gamme de modèles de machine learning, notamment la régression logistique, le Naive Bayes multinomial, la machine à vecteurs de support linéaire, l'arbre de décision, la forêt aléatoire et le Gradient Boosting Classifier, tous disponibles dans la bibliothèque scikit-learn.
Une fois notre modèle final prêt, nous l'avons déployé en utilisant Gradio, une bibliothèque Python qui permet de créer rapidement des interfaces utilisateur personnalisées pour les modèles de machine learning.
Enfin, pour garantir une collaboration efficace et le suivi des versions de notre code, nous avons utilisé Git. Pour la documentation du projet, nous avons utilisé Google Docs, qui nous a permis

de collaborer en temps réel et de maintenir toutes nos notes et documents de recherche dans un emplacement centralisé.

C'est l'environnement technique complet que nous avons utilisé pour réaliser ce projet, permettant une collaboration efficace et le développement d'une solution robuste pour notre problématique.

4 Collecte et exploration des données

Le challenge Kaggle "Toxic Comment Classification" est organisé par l'équipe Conversation AI et a pour objectif de développer un modèle de classification de commentaires toxiques qui peut aider à identifier automatiquement les commentaires offensants, haineux ou discriminatoires dans les conversations en ligne.

L'ensemble de données que nous utilisons se compose de commentaires issus des modifications de la page de discussion de Wikipédia.

Les modifications des pages de discussion de Wikipédia sont des contributions apportées par les utilisateurs sur ces pages spécifiques, dédiées à la conversation et à la collaboration. Elles sont principalement utilisées pour discuter des améliorations potentielles des articles de l'encyclopédie. Les modifications peuvent inclure l'ajout de nouveaux commentaires, la réponse à des commentaires existants, ou la modification de commentaires précédemment publiés. C'est un lieu essentiel d'échanges et de débats où les utilisateurs partagent leurs points de vue, posent des questions et cherchent un consensus sur des sujets controversés.

Ces commentaires ont été étiquetés par des humains pour leur comportement toxique. Les types de toxicité sont les suivants :

- **Toxic :** le commentaire contient du langage abusif ou offensant.
- **Severe Toxic :** le commentaire contient du langage extrêmement abusif ou offensant.
- **Obscene :** le commentaire contient du langage obscène ou vulgaire.
- **Threat :** le commentaire contient des menaces ou des incitations à la violence.
- **Insult :** le commentaire contient des insultes ou des propos offensants.
- **Identity Hate :** le commentaire contient des propos haineux ou discriminatoires envers une personne ou un groupe de personnes en raison de leur identité.

Chaque commentaire peut être étiqueté avec un ou plusieurs de ces labels, en fonction de la nature de la toxicité. Par exemple, un commentaire peut être étiqueté à la fois comme "toxic" et "insult". Il s’agit donc d’un problème de classification multilabel.

La classification multilabel est un type de problème d'apprentissage automatique où chaque exemple peut être associé à plusieurs classes ou étiquettes simultanément (ici 6 classes).

Le défi dans la résolution des problèmes de classification multilabel réside dans la gestion de la corrélation entre les différentes étiquettes. De plus, le nombre de combinaisons possibles d'étiquettes peut être très élevé, surtout lorsque le nombre total d'étiquettes est important comme dans notre cas, ce qui peut augmenter la complexité du problème.

Les données d'entrainement comprennent 159 571 commentaires réparties dans les classes évoquées ci-dessus. Les données de test comprennent 153 164 commentaires. Comme les données ont été utilisées à l'origine pour un concours Kaggle, l'ensemble de données test_labels contient des observations avec des étiquettes de valeur -1, ce qui indique qu'elles n'ont pas été utilisées pour la notation. Ces observations seront supprimées de notre ensemble de test.

Les 8 variables de la base de données sont : 'id', 'comment_text' 'toxic', 'severe_toxic', 'obscene ', 'threat', 'insult', 'identity_hate'. Id représente l'identifiant du commentaire, il est unique. Com ment_text représente le commentaire, il s'agit essentiellement de phrases courtes. Les variable s 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult' et 'identity_hate' sont des variables binaires q ui prennent la valeur 0 si le commentaire est jugé non toxique et la valeur 1 s'il appartient à la catégorie de toxicité considérée.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

Tableau 1: Aperçu des données

Les commentaires analysés sont rédigés en anglais. Par exemple, le commentaire suivant : "Pourquoi les modifications effectuées sous mon nom d'utilisateur Hardcore Metallica Fan ont-elles été annulées ? Ce n'était pas du vandalisme, juste de la fermeture sur certaines pages de discussion après que j'ai voté au New York Dolls FAC. Et s'il vous plaît, ne supprimez pas le modèle de la page de discussion puisque je suis à la retraite maintenant.89.205.38.27" est considéré comme non toxique. Il s'agit simplement d'un utilisateur qui exprime son désaccord face à l'annulation d'une modification qu'il a réalisée sur une encyclopédie, ce qui est un comportement tout à fait acceptable.

À l'inverse, le commentaire suivant : "Hé... qu'est-ce que c'est... un groupe exclusif de quelques TALIBANS de WP... qui sont doués pour détruire, des puristes autoproclamés qui GANGENT tous ceux qui leur posent des questions sur leur (non-)contribution ANTI-SOCIALE et DESTRUCTIVE à WP ?" est un exemple de commentaire toxique. Ce message, manifestement agressif, ne contribue en rien au débat constructif. Notre modèle sera conçu pour détecter automatiquement ce genre de commentaires afin de les modérer efficacement.

La figure suivante présente un nuage de mots qui met en évidence les termes les plus fréquemment rencontrés dans les commentaires toxiques. Un nuage de mots offre une illustration visuelle des données textuelles, les mots apparaissant le plus souvent étant mis en évidence par une taille plus importante. Comme on pouvait s'y attendre, parmi les mots qui se détachent figurent des termes offensants comme nègre, juif, pédé, gros, porc, idiot, etc. Il est probable que ces mots constituent des variables discriminantes dans notre modélisation.

Figure 3. Wordcloud des commentaires toxiques

L'analyse des données d'entraînement nous montre que la plupart des commentaires sont brefs, avec à peine 8,37% dépassant les 1000 mots.

Il est également à noter un déséquilibre significatif dans la distribution des données. Les commentaires toxiques ne représentent que 9,58% de l'ensemble des données. De plus, les commentaires obscènes constituent 5,29% du total, tandis que les insultes comptent pour 4,94%. Les commentaires extrêmement toxiques, haineux et menaçants sont encore moins nombreux, ne représentant respectivement que 1%, 0,88% et 0,33% du corpus total.

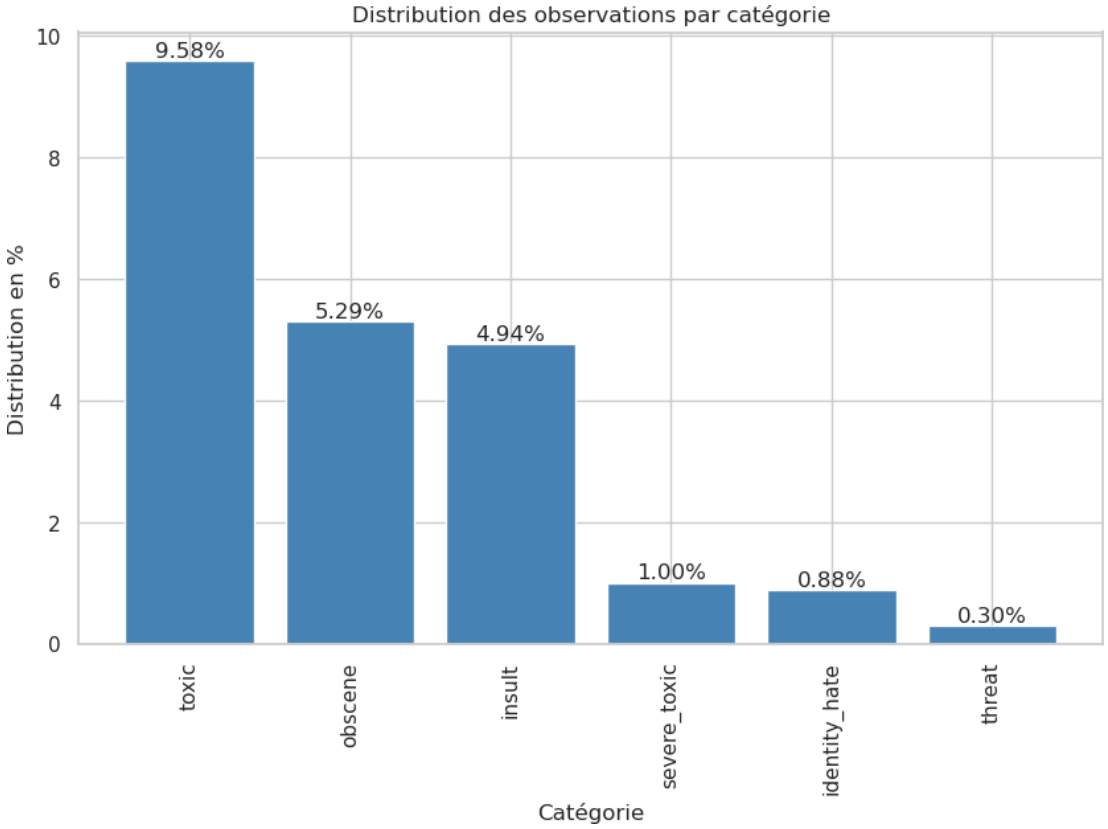


Figure 4. Distribution des observations par catégorie

Ceci soulève le problème des classes déséquilibrées, qui nécessite des stratégies spécifiques pour améliorer l'efficacité des modèles d'apprentissage automatique. Parmi ces stratégies, on trouve le SMOTE, l'undersampling, l'oversampling ou encore l'utilisation des poids de classe.

La figure 4 représente les corrélations entre différents types de commentaires toxiques. Le coefficient de corrélation varie de -1 à 1, où 1 indique une corrélation positive parfaite, -1 une corrélation négative parfaite et 0 aucune corrélation.

Il y a une corrélation plutôt forte (67,7%) entre les commentaires toxiques et les commentaires obscènes, ce qui signifie que lorsqu'un commentaire est jugé toxique, il est souvent jugé obscène aussi. Cela est assez logique. La corrélation entre les commentaires toxiques et les insultes est également forte (plus de 64%), indiquant que les commentaires qui sont jugés toxiques sont souvent jugés insultants. Les commentaires qui sont jugés obscènes sont très souvent jugés insultants aussi (corrélation de 74,1%). Les commentaires menaçants ont une corrélation relativement faible avec les autres catégories (entre 11% et 15%). Ces commentaires sont donc généralement distincts des autres types de commentaires toxiques. Les commentaires de haine ainsi que Les commentaires sévèrement toxiques ont une corrélation modérée avec les autres catégories.

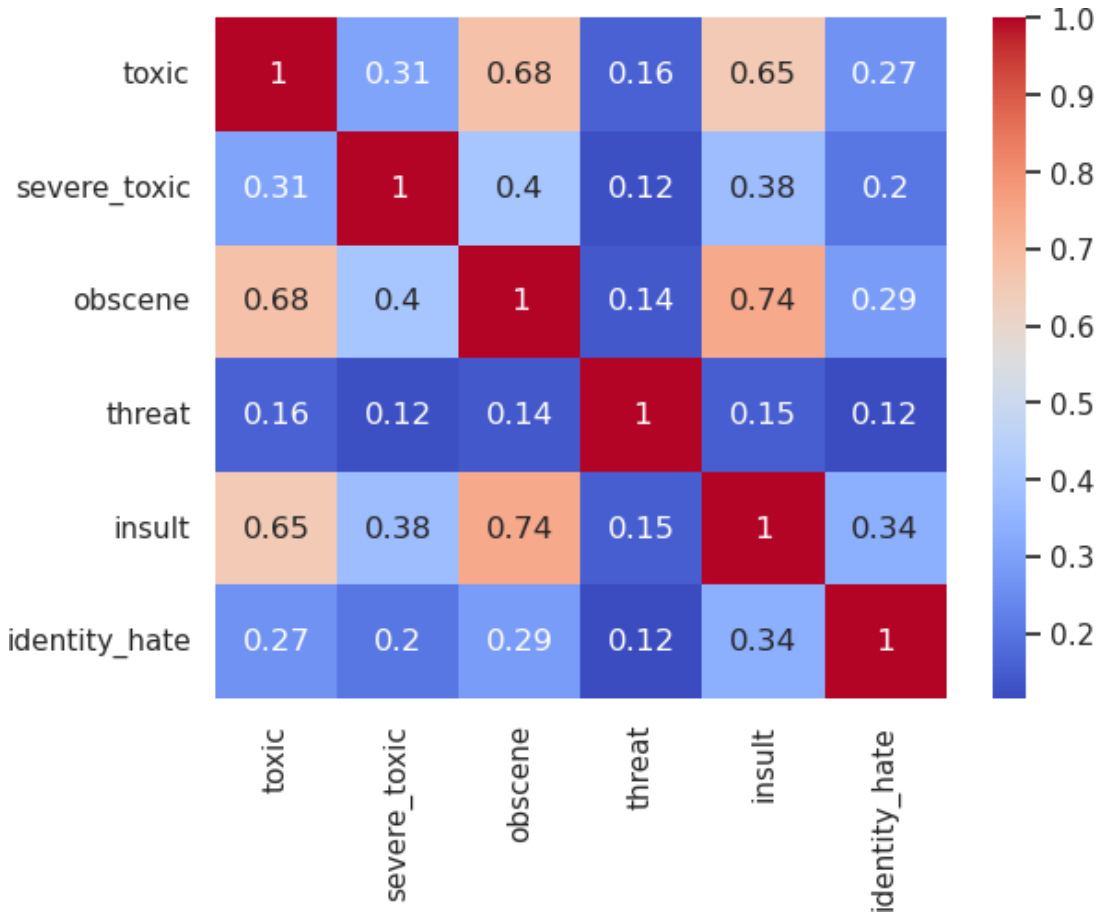


Figure 5. Matrice des corrélations des labels

Ces corrélations soulèvent la question de savoir s'il faut utiliser un modèle unique qui prédit toutes les classes de toxicité en même temps, ou des modèles indépendants pour chaque classe de toxicité. La compétition requiert l'utilisation de modèles indépendants mais nous pouvons

également explorer la piste du modèle unique qui pourrait tirer parti des corrélations pour améliorer les prédictions.

5 Prétraitement des données et Vectorisation du texte

Nous avons défini une fonction permettant de nettoyer et de normaliser le texte. Elle convertit tout le texte en minuscules, élimine la ponctuation, les chiffres et les caractères spéciaux, puis sépare le texte en mots individuels. Ensuite, elle supprime tous les caractères non-ASCII et réduit les mots à leur forme de base à l'aide de la lemmatisation. Enfin, elle supprime tous les mots courts de deux lettres ou moins. Le résultat est une liste de mots nettoyés et normalisés prêts pour la vectorisation.

Afin de choisir la technique de vectorisation la plus adaptée à notre jeu de données, nous effectuons une évaluation comparative (benchmarking) de trois différentes méthodes de vectorisation: 'Frequency Vector' (comptage des fréquences des mots), 'One Hot Encoding' (chaque mot est représenté par un vecteur de 0 et 1 indiquant sa présence ou absence) et 'TF-IDF' (Term Frequency-Inverse Document Frequency, qui mesure l'importance relative d'un mot dans le document par rapport à l'ensemble du corpus). Le but est d'évaluer leur performance sur un modèle de classification SVM pour chaque catégorie de commentaires toxiques. Le classifieur SVM constitue donc notre modèle de base car il gère très bien la haute dimensionnalité et le caractère creux des vecteurs issus de la vectorisation. C'est un classifieur usuel pour le traitement de texte.

Pour chaque méthode de vectorisation, le texte des commentaires est transformé en une matrice de vecteurs, puis utilise cette matrice pour entraîner et évaluer un modèle SVM, en utilisant l'AUC comme mesure de performance et une validation croisée stratifiée en 5 folds. Le temps de calcul est également mesuré pour chaque combinaison de méthode de vectorisation et de catégorie de commentaires.

Les résultats montrent que la méthode TF-IDF fournit généralement les meilleurs scores AUC pour chaque catégorie de commentaires (96,4% en moyenne), suivie de près par la méthode One Hot Encoding (93,3% en moyenne). Ces deux méthodes semblent surpasser la méthode de vectorisation par Frequency Vector.
En outre, la méthode TF-IDF est la plus rapide en termes de temps de calcul.

Method	AUC	Elapsed Time
Frequency Vector	0,907119	692,509456
One Hot Encoding	0,933110	489,908027
TF-IDF	0,963948	412,609730

Tableau 2: Benchmarking des méthodes de vectorisation

La méthode TF-IDF semble donc être la meilleure option en termes de performance du modèle et d'efficacité computationnelle.
Nous l'utilisons pour vectoriser notre texte : D'abord, nous prenons des mots individuels dans chaque commentaire. Le texte est nettoyé en supprimant les accents et les mots qui sont très communs en anglais, comme "the" ou "is". Enfin, nous n'utilisons que des mots qui apparaissent dans plus de 10 commentaires. Les commentaires de l'ensemble d'entrainement sont ensuite transformés en vecteurs numériques. L'ensemble de test est ensuite transformé, mais en utilisant les informations apprises de l'ensemble d'entraînement.

6 Modélisation et entraînement des modèles

6.1 Choix du modèle de classification

Afin de déterminer le classifieur le plus adapté à notre jeu de données, nous testons quatre différents modèles d'apprentissage automatique sur les commentaires. Les quatre modèles sont : Naive Bayes, régression logistique, Random Forest (Forets aléatoires) et le gradient boosting. Pour chaque modèle, nous utilisons la validation croisée pour diviser les données d'entraînement en cinq folds. Quatre de ces parties sont utilisées pour entraîner le modèle et la cinquième pour le tester.

Pour chaque modèle, la précision, le rappel, le score F1 et le score AUC sont mesurés. Ensuite, la moyenne de ces mesures est calculée.

L'analyse des résultats nous indique :

- Le modèle de régression logistique a obtenu les meilleurs résultats en termes de score AUC pour toutes les catégories (97,7%).
- Pour la catégorie 'toxic', le modèle Random Forest a le score F1 le plus élevé, ce qui signifie qu'il a un bon équilibre entre la précision et le rappel pour cette catégorie.
- Le modèle Gradient Boosting a eu une performance modérée en termes de score F1 pour la plupart des catégories.
- Le modèle bayésien semble avoir des performances généralement inférieures aux autres modèles, en particulier pour les catégories 'threat' et 'identity_hate' où le score F1 est très faible.

N°	Model	Precision	Recall	F1	AUC
0	GradientBoostingClassifier	0,671789	0,339269	0,44636	0,895526
1	LogisticRegression	0,764699	0,384206	0,490789	0,977412
2	MultinomialNB	0,599559	0,216655	0,296076	0,927569
3	RandomForestClassifier	0,699756	0,371189	0,438935	0,951222

Tableau 3: Benchmarking des classifieurs

Le modèle régression logistique semble être le meilleur choix pour obtenir une performance globalement bonne sur toutes les catégories.

6.2 Optimisation des performances

Avec le modèle sélectionné, nous pouvons essayer d'améliorer les performances en rééquilibrant les classes qui pour rappel sont très déséquilibrées.
Pour ce faire, nous utilisons un gridsearch pour trouver les meilleurs poids de classe pour chaque modèle de régression logistique.

Pour chaque type de toxicité, nous faisons varier le poids de la classe non-toxic de 5% à 95% par pas de 5 points (la catégorie de toxicité a pour poids la différence entre 1 et le poids de la classe non-toxic). Les performances sont évaluées par une validation croisée en termes d'AUC. La combinaison de poids présentant le meilleur AUC est retenu.

Best weights for label toxic are {'class_weight': {0: 0.05, 1: 0.95}} with AUC score: 0.8944369646738325
Best weights for label severe_toxic are {'class_weight': {0: 0.05, 1: 0.95}} with AUC score: 0.8483770782524036
Best weights for label obscene are {'class_weight': {0: 0.05, 1: 0.95}} with AUC score: 0.9226106203845411
Best weights for label threat are {'class_weight': {0: 0.05, 1: 0.95}} with AUC score: 0.6699023616803391
Best weights for label insult are {'class_weight': {0: 0.05, 1: 0.95}} with AUC score: 0.9095113371993795
Best weights for label identity_hate are {'class_weight': {0: 0.05, 1: 0.95}} with AUC score: 0.7812355413348103

Tableau 4:Poids optimaux par catégorie

Pour tous les types de toxicité - 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate' -, les meilleurs poids sont 5% pour la classe non toxique et 95% pour la classe toxique. Cela signifie que, pour obtenir les meilleures performances, nous devons donner un poids 20 fois plus important aux commentaires de classe positive (toxiques) par rapport aux commentaires de classe négative (non toxiques).

Ces poids optimaux donnent par validation croisée un AUC moyen (pour toutes les catégories de toxicité) de 97,6%, ce qui est légèrement inférieur à l'AUC moyen de 97,7% du modèle de régression logistique par défaut.
Nous pouvons donc conserver à cette étape le modèle par défaut.

Nous créons ensuite de nouvelles variables susceptibles de fournir des informations supplémentaires au modèle.

Dans ce cadre, nous construisons la variable comment_length qui représente la longueur des commentaires. Les commentaires très courts pourraient constituer des insultes d'où l'éventuelle utilité de cette variable. Nous ajoutons également une variable qui compte le nombre de points d'exclamation utilisés dans chaque commentaire. Les points d'exclamation peuvent être un indicateur de l'intensité émotionnelle du commentaire, ce qui pourrait être lié à la toxicité. Enfin, nous comptons dans une variable le nombre de mots entièrement en majuscules dans chaque commentaire. Les mots en majuscules sont souvent utilisés pour exprimer une forte émotion ou crier dans le langage de l'Internet, ce qui pourrait également être lié à la toxicité.

Ces nouvelles variables sont ajoutées aux variables issues de la vectorisation. L'AUC moyen est ensuite calculé sur l'ensemble des catégories de toxicité par validation croisée. Nous obtenons un AUC moyen de 97,1% ce qui est légèrement inférieur aux performances du modèle par défaut (97,7%). Ces nouvelles variables semblent non nécessaires. Le modèle par défaut peut être conservé.

Nous testons maintenant une nouvelle approche consistant à combiner la vectorisation TF-IDF à la technique Word2vec. Pour rappel, Word2vec est utilisé pour apprendre les vecteurs de mots à partir des commentaires de texte. Chaque mot est représenté comme un vecteur de 100 dimensions dans l'espace vectoriel. Ces vecteurs sont appris en essayant de prédire un mot donné à partir des mots voisins dans le contexte du mot, ou inversement, en essayant de prédire les mots de contexte à partir du mot donné.

TF-IDF est une mesure statistique qui évalue l'importance d'un mot dans un document par rapport à un corpus de documents. Les mots qui apparaissent fréquemment dans un document mais pas fréquemment dans le corpus reçoivent un score élevé.

Les deux approches sont combinées en calculant une moyenne pondérée des vecteurs de mots Word2Vec pour chaque commentaire. Le poids de chaque mot est déterminé par son score TF-IDF. De cette façon, chaque commentaire est représenté par un seul vecteur qui capture à la fois les informations sémantiques (de Word2Vec) et les informations statistiques (de TF-IDF) des mots dans le commentaire.
Cette approche fournit un AUC moyen par validation croisée de 50%, ce qui est caractéristique d'un modèle non-informatif. Cette approche n'est donc pas adaptée à notre jeu de données.

Le modèle de régression logistique est finalement retenu. Il sera entrainé sur le training set complet puis tester sur le test set.

6.3 Analyse des résultats

Sur L'ensemble d'entraînement, Les performances sont assez bonnes, avec des scores de rappel allant de 96,1% à 99,7%, des scores de précision de 96,0% à 99,6%, des scores F1 de 95,8% à 99,6% et des scores AUC de 98,1% à 99,4%. La catégorie "threat" (menace) est la mieux prédite avec le rappel le plus élevé (99,7%) et l'AUC la plus élevée (99,4%).

Train Results					
N°	Label	Recall	Precision	F1	AUC
0	toxic	0,961522	0,960509	0,958401	0,981249
1	severe_toxic	0,991095	0,989087	0,989482	0,991044
2	obscene	0,979182	0,978252	0,977616	0,990167
3	threat	0,997324	0,996764	0,996439	0,994365
4	insult	0,973015	0,970754	0,970423	0,985279
5	identity_hate	0,992586	0,991107	0,990695	0,988527

Tableau 5:Performances du modèle sur le train set

Sur L'ensemble de test, la performance du modèle semble également globalement assez bonne pour toutes les catégories, avec des scores de rappel allant de 93,5% à 99,6%, des scores de précision de 93,8% à 99,5%, des scores F1 de 93,7% à 99,6% et des scores AUC de 95,8% à 98,6%.

Test Results					
N°	Label	Recall	Precision	F1	AUC
0	toxic	0,935728	0,938648	0,937031	0,958490
1	severe_toxic	0,993123	0,992445	0,992762	0,985083
2	obscene	0,965957	0,963507	0,964267	0,971170
3	threat	0,996530	0,995339	0,995725	0,986095
4	insult	0,964175	0,960395	0,961200	0,966591
5	identity_hate	0,990465	0,988373	0,988378	0,979390

Tableau 6:Performances du modèle sur le test set

Les performances sur l'ensemble d'entraînement sont généralement plus élevées que sur l'ensemble de test, ce qui est attendu car le modèle s'est entraîné sur ces données.

Notons que les scores de rappel, précision et F1 sont pondérées par les poids des classes afin de juguler l'effet du déséquilibre prononcé des classes et d'obtenir une estimation réaliste des performances de notre modèle.

Du point de vue macro, nous avons 98,8% d'AUC moyen sur le train set contre 97,4% sur le test set. Le gap de performances est infime, le sur-apprentissage est donc évité.

Les mots les plus importants dans la prédiction des commentaires toxiques sont sans surprise des mots vulgaires et insultants : fuck, fucking, idiot, shit et stupid.

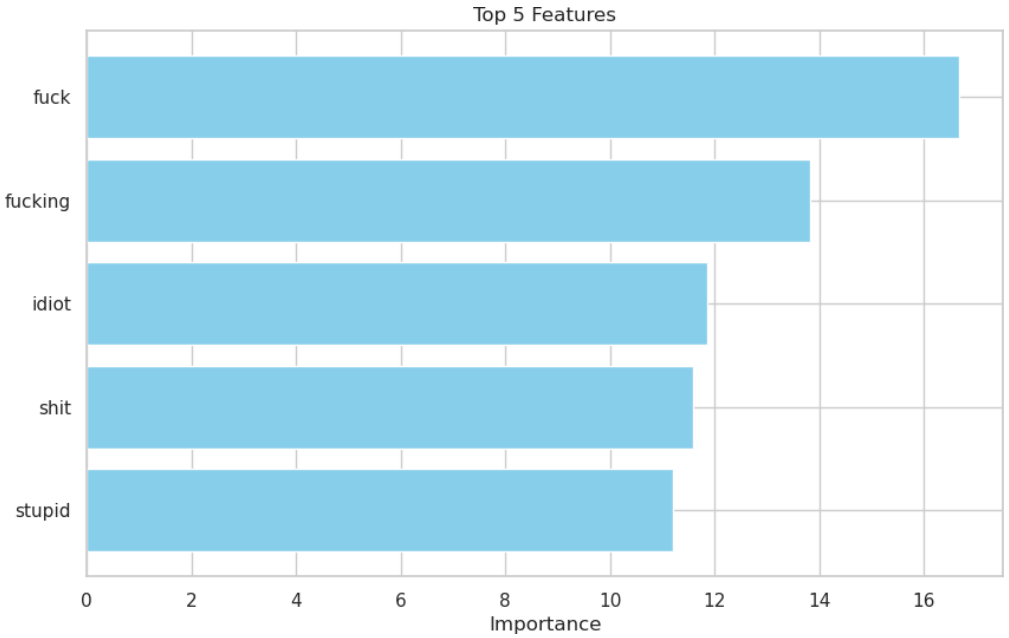


Figure 6. Importance des mots

Malgré ces bonnes performances macro, les modèles ont des performances mitigées quant à la détection des commentaires toxiques. Pour la catégorie des commentaires simplement toxiques par exemple, nous avons une sensibilité de 71%. Alors pourquoi 3 commentaires toxiques sur 10 sont mal classifiés par le modèle ?

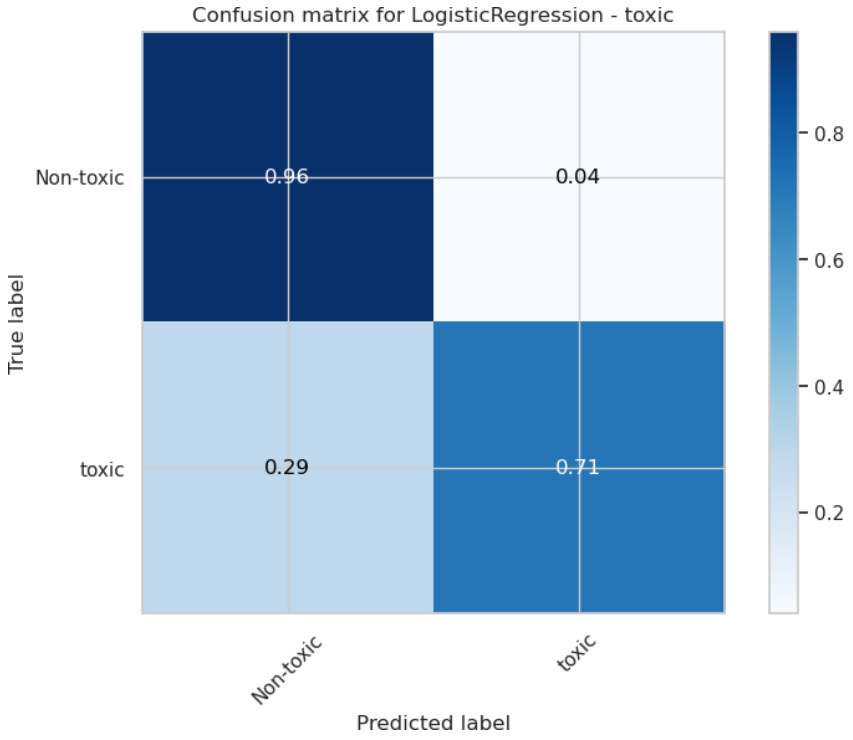


Figure 7. Matrice de confusion de la catégorie toxique

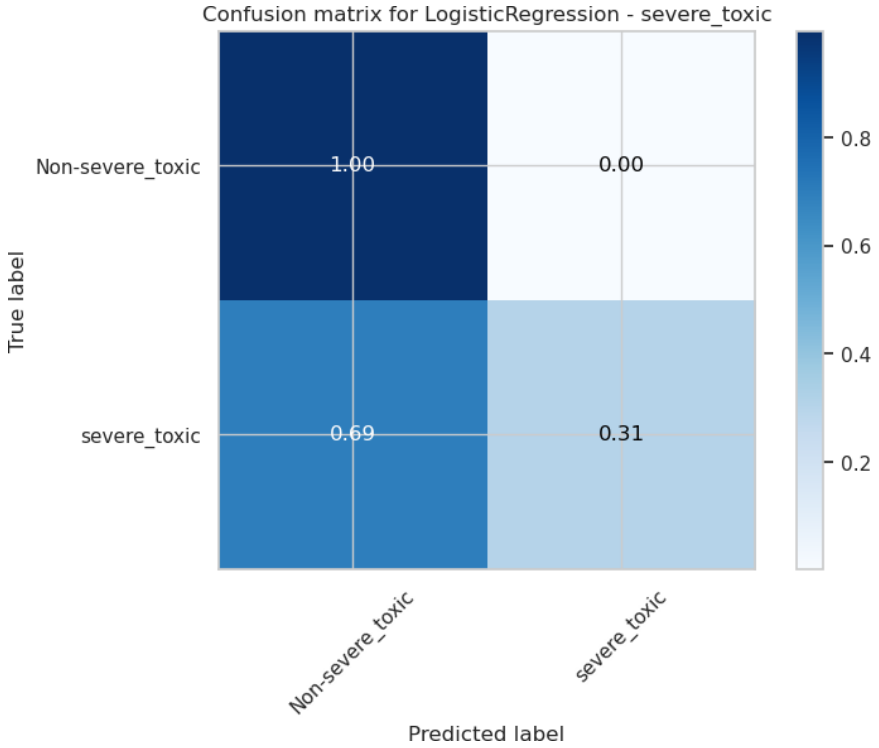


Figure 8. Matrice de confusion de la catégorie toxicité sévère

La figure 9 présente le nuage des mots des commentaires mal classifiés du test set. Nous notons le mot « *ucking » qui est le troisième le plus fréquent avec 253 occurrences.
Ce mot n'est cependant présent qu'à 5 reprises dans le train set, ce qui pourrait expliquer la difficulté du modèle à bien classifier les commentaires qui le contiennent.

Figure 9. Wordcloud des commentaires mal classés

Ces performances mitigées peuvent être expliquées par une répartition hétérogène des commentaires et le déséquilibre des classes.

6.4 Utilisation de l'apprentissage profond

Dans le but d'explorer d'autres méthodes de classification et d'améliorer potentiellement les performances de notre modèle, nous avons également mis en œuvre un modèle de réseau neuronal profond, plus précisément un réseau de neurones de type Long Short-Term Memory (LSTM). Contrairement aux méthodes traditionnelles d'extraction de caractéristiques telles que CountVectorizer, One-Hot Encoding, et TF-IDF, l'approche pour les LSTM utilise une technique d'extraction de caractéristiques différente appelée "embedding" ou "plongement de mots".

Le "Word Embedding" ou "plongement de mots" est une technique d'apprentissage profond où les mots sont représentés par des vecteurs denses dans un espace vectoriel de dimensions réduites par rapport à la taille du vocabulaire. Ces vecteurs sont appris à partir des données et sont capables de capturer la sémantique des mots, les relations sémantiques et syntaxiques entre eux, contrairement aux représentations one-hot qui traitent chaque mot comme une entité indépendante sans aucune relation avec les autres mots.

Nous avons suivi plusieurs étapes pour préparer nos données textuelles et entraîner notre modèle LSTM :

1. Tokenisation : Nous avons commencé par diviser le texte de chaque commentaire en tokens, ou mots individuels, créant ainsi une séquence de tokens pour chaque commentaire.

2. Construction du Vocabulaire : Ensuite, nous avons créé un vocabulaire à partir de l'ensemble de tous les tokens uniques présents dans notre jeu de données.
3. Conversion des Tokens en Identifiants : Chaque token dans les commentaires a été remplacé par son identifiant numérique unique correspondant dans le vocabulaire. Ces identifiants sont ensuite utilisés pour créer les vecteurs d'embedding.
4. Padding: Pour que les séquences aient toutes la même longueur, nécessaire pour les LSTM, nous avons ajouté des éléments de remplissage aux séquences plus courtes.
5. Création d'un Modèle LSTM : Nous avons créé un modèle LSTM avec une couche d'embedding suivie par plusieurs couches LSTM, et enfin une couche de sortie adaptée à notre tâche de classification. Cette architecture permet à notre modèle de traiter les commentaires comme des séquences de mots et d'apprendre des représentations vectorielles (embeddings) qui capturent la sémantique des mots en tenant compte de leur contexte dans le commentaire.

Après l'entraînement initial de notre modèle LSTM, nous avons obtenu une précision de classification de 76 %. En optimisant les hyperparamètres de notre modèle, comme la taille des embeddings et le nombre de neurones dans les couches LSTM, nous avons réussi à augmenter cette précision à 85 %.

Malgré l'efficacité des LSTM à mieux comprendre le contexte des commentaires, nous avons observé que les modèles de machine learning classiques tels que la régression logistique et SVM ont obtenu des performances supérieures, avec une précision supérieure à 95 %. De plus, les modèles de machine learning classiques sont plus simples à comprendre et à expliquer, évitant ainsi le problème des "boîtes noires" associé aux modèles d'apprentissage profond.

En outre, la complexité accrue des LSTM, leur temps d'entraînement plus long et leurs exigences en matière de ressources de calcul rendent ces modèles plus difficiles et coûteux à mettre en production. En tenant compte de ces facteurs, ainsi que des performances de classification, nous avons décidé de conserver notre modèle de machine learning classique pour la classification des commentaires toxiques.

6.5 Comparaison de résultats par rapport l’équipe gagnante

L’équipe gagnante a obtenu un score de 98,8% d’AUC moyen sur le test set. Pour obtenir ce résultat, l’équipe a combiné plusieurs modèles et techniques. Tout d'abord, ils ont utilisé des modèles d'incorporation de mots pré-entraînés comme FastText et GloVe, qui sont des représentations vectorielles des mots créées en étudiant de grandes quantités de texte sur internet. Ces incorporations ont permis au modèle de comprendre les relations sémantiques entre les mots dans le texte.

Ensuite, ils ont utilisé un modèle appelé Bi-GRU, ou Gated Recurrent Unit à deux directions, pour analyser les séquences de mots dans les phrases. Ce modèle est capable de lire une phrase de gauche à droite et de droite à gauche, ce qui lui permet de comprendre le contexte autour de chaque mot dans la phrase.

En plus de ces modèles, l'équipe gagnante a utilisé deux techniques supplémentaires pour améliorer leurs résultats. Ils ont utilisé une technique d'augmentation de données par traduction, qui implique la traduction des textes en différentes langues, puis leur re-traduction en anglais pour obtenir plus de données à entraîner. Ils ont également utilisé une technique appelée "pseudo-étiquetage", où ils ont utilisé leurs meilleures prédictions comme si elles étaient des données d'entraînement réelles.

C’est une approche plus complexe que la nôtre et qui nécessite une puissance de calcul plus importante. Avec une simple régression logistique, nous avons des performances très proches. Notre modèle peut donc être retenu si la facilité de mise en production et la puissance de calcul nécessaire sont un enjeu.

7 Déploiement du Modèle

Le déploiement du modèle que nous avons développé marquait une étape cruciale dans notre projet. En se concentrant sur l'identification de diverses formes de toxicité dans les commentaires, notamment la toxicité générale, la toxicité sévère, l'obscénité, la menace, l'insulte et la haine liée à l'identité, notre objectif était de créer un modèle qui pourrait prédire les probabilités d'appartenance à ces catégories. Gradio s'est présenté comme l'outil de choix pour créer une interface utilisateur attrayante et fonctionnelle qui pourrait interagir avec notre modèle.

En utilisant Gradio, nous avons pu concevoir une interface simple mais efficace. Elle comporte principalement deux éléments : une zone de texte où les utilisateurs peuvent entrer un commentaire, et une section d'affichage où les probabilités que le commentaire appartienne à chacune des catégories de toxicité sont présentées. Pour améliorer la lisibilité, nous avons utilisé des barres de progression, ce qui permet aux utilisateurs de visualiser rapidement l’intensité de la toxicité dans différentes catégories.

En plus de la mise en service locale, nous avons exploré l'intégration avec des services de cloud computing. Gradio offre la flexibilité de se configurer avec divers services de cloud, ce qui nous a permis de rendre notre interface accessible au public. Cela a également ouvert la porte à un public plus large.

L'utilisation de Gradio a permis une transition fluide de notre modèle de recherche et développement à une application web interactive qui peut être utilisée par d'autres. Cela a non seulement illustré les capacités de notre modèle de manière pratique, mais a également été un moyen efficace de rendre notre travail accessible à un public plus large.

Figure 10:Interface web de classification des commentaires toxiques

8 Conclusion

Ce projet de détection de la toxicité dans les commentaires en ligne a été très enrichissant. Nous avons exploré différentes techniques de prétraitement, de vectorisation et de modélisation pour développer un modèle de régression logistique capable d'identifier et de classifier les commentaires toxiques.

Au cours de ce projet, nous avons réalisé une analyse exploratoire approfondie des données, identifié les défis liés à la modération des commentaires en ligne et examiné les différentes approches pour résoudre ce problème complexe. Nous avons constaté que la modération automatisée des commentaires peut contribuer à créer un environnement en ligne plus sûr et respectueux, mais qu'elle doit être utilisée en complément de politiques de modération claires et d'une intervention humaine lorsque cela est nécessaire.

Notre modèle de régression logistique de base a montré de bonnes performances globales, avec des scores élevés en termes de précision, de rappel, de F1 et d'AUC. Cependant, nous avons également identifié des limitations dans la détection précise des commentaires toxiques, en particulier pour certaines catégories spécifiques.

Pour aller plus loin, des techniques avancées telles que l'utilisation d'incorporations de mots pré-entraînées et de modèles de réseaux neuronaux pourraient être explorées afin d'améliorer davantage les performances de détection.

Ce projet a été une formidable opportunité pour nous d'approfondir notre compréhension des différentes facettes de la modération des commentaires en ligne à travers le prisme du traitement automatique du langage naturel (NLP). Nous avons exploré et appliqué des concepts clés de la NLP tels que la vectorisation des mots, l'extraction de caractéristiques, la tokenisation, le prétraitement des données textuelles et l'application de divers modèles d'apprentissage automatique.
L'apprentissage et l'application de la NLP dans ce contexte nous a montré qu'elle peut être un outil puissant pour aider à rendre l'Internet un espace plus sûr et plus inclusif.

9 Références bibliographiques

Ouvrages et articles

- Liu, B. (2012) Sentiment Analysis and Opinion Mining (Synthesis Lectures on Human Language Technologies). Morgan & Claypool Publishers, Vermont, Australia.
- Erik Cambria, Dipankar Das, Sivaji Bandyopadhyay, Antonio Feraco: A Practical Guide to Sentiment Analysis, Springer, 2017
- Ricco Rakotomalala *,Opinion Mining et Analyse de Sentiments,Fouille d'opinions et analyses des sentiments*
- Benaissa Azzeddine Rachid, Harbaoui Azza and Ben Ghezala Henda (2018) ,*Sentiment Analysis Approaches Based On Granularity Levels,* University of Manouba, RIADI Laboratory, ENSI School, La Manouba, Tunisia
- Grzegorz Dziczkowski. Analyse des sentiments : système autonome d'exploration des opinions exprimées dans les critiques cinématographiques. Automatique / Robotique. École Nationale Supérieure des Mines de Paris, 2008. Français. ffNNT : 2008ENMP1637ff.
- Stefania Pecore. Analyse des sentiments et des émotions de commentaires complexes en langue française. Linguistique. Université de Bretagne Sud, 2019. Français. ffNNT : 2019LORIS522f
- Axel de Goursac (2018), Le Natural Language Processing, au cœur de l'interaction Humain-IA
- V. Srividhya, R. Anitha , Evaluating Preprocessing Techniques in Text Categorization, International Journal of Computer Science and Application Issue 2010
- Santiago González-Carvajal (2021), Eduardo C. Garrido-Merchán, Comparing BERT against traditional machine learning text classification, *Universidad Politécnica de Madrid,Spain,*
- Irfan Qutab, Sentiment Classification Using Multinomial Logistic Regression on Roman Urdu Text, International Journal of Innovations in Science and Technology. Vol 4, Issue 2, 2022, pp: 323- 335.

- https://enjoymachinelearning.com/blog/countvectorizer-vs-tfidfvectorizer/
- https://machinelearningmastery.com/gentle-introduction-bag-words-model/
- https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b3bf6373ff41a115197cb5b30e57830c16130c2c
- https://towardsdatascience.com/a-battle-against-amnesia-a-brief-history-and-introduction-of-recurrent-neural-networks-50496aae6740

- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

- https://deeplearning.cs.cmu.edu/S23/document/readings/LSTM.pdf