NLP – Modération de Chat

Nakache Eithan
Ziane Camil
Hadj-Said Samy
Perez Jason
Six Briac
Bellamy Baptiste

Sujet

Modération en temps réel des chats sur une plateforme de contenu, par l'intermediaire de modèle NLP

Table des matières

1. Introduction

Ce projet se propose de développer des modèles du traitement du language naturel dédiés à la modération en temps réel des chats sur une plateforme de contenu. L'objectif principal est d'assurer la modération des messages des utilisateurs afin de prévenir la diffusion de contenus inappropriés, tels que les insultes, les spams et autres formes de communications indésirables.

Dataset

Le dataset choisit pour ce projet est le **Jigsaw Toxic Comment Classification**. Ce dataset recense un grand nombre de commentaires anglais, provenant de Wikipedia, labellisés par des humains en fonction de leur toxicité. Ce dernier provient d'une compétition Kaggle et peut être obtenu à partir du lien suivant : jigswaw toxic comment classification challenge.

Objectif

Plusieurs modèles de machine learning et de deep learning seront développés et entraînés dans le but de prédire la pertinence et l'acceptabilité des messages. Une fois ces modèles développés, nous procéderons à une évaluation rigoureuse de leurs performances par comparaison mutuelle. L'ultime étape de ce projet consistera en la création d'une preuve de concept sous la forme d'une interface web. Cette interface permettra de modérer en direct les messages échangés sur un chat de live streaming, démontrant ainsi l'applicabilité pratique de nos recherches.

2. Présentation du jeu de données

Objectifs et Financement du Dataset

Le dataset de classification des commentaires toxiques de JIGSAW a été créé pour promouvoir la recherche sur la détection de la toxicité dans les commentaires en ligne. Il vise à identifier des comportements indésirables tels que les commentaires toxiques. Ce corpus a été collecté dans le but de développer des technologies et des méthodes capables de modérer automatiquement ces types de contenu nuisible sur les plateformes en ligne.

Le projet a été financé par Jigsaw (anciennement connu sous le nom de Google Ideas) et Google, dans le cadre d'un concours organisé sur la plateforme Kaggle. Ce partenariat a non seulement mis à disposition les ressources nécessaires, mais a également encouragé la communauté globale des data scientists à résoudre ce problème urgent de modération des contenus toxiques sur Internet.

Contexte et Caractéristiques des Données du Dataset

Les commentaires inclus dans le dataset proviennent des pages de discussion de Wikipedia. Ces discussions sont menées en anglais par des contributeurs qui échangent sur les améliorations à apporter aux articles et sur les modifications nécessaires. Ces échanges sont caractéristiques des interactions collaboratives typiques sur Wikipedia, où les utilisateurs débattent de la véracité, de la neutralité et de la complétude des articles.

Le format du texte est écrit et prend la forme de communications en ligne non formelles mais structurées. Cela signifie que les commentaires, bien que rédigés dans un cadre informel, suivent une certaine structure logique et sont orientés vers des objectifs spécifiques de collaboration et de modification de contenu.

Démographie des auteurs

Les informations démographiques spécifiques sur les auteurs des commentaires ne sont pas fournies.

Processus de collecte

Le dataset comprend environ 160 000 commentaires pour l'entraînement et 60 000 commentaires pour le test. Ces données ont été extraites dans le but de représenter divers comportements toxiques, bien que la méthode exacte d'échantillonnage n'ait pas été spécifiée.

Étant issues de plateformes ouvertes, les questions de consentement sont gérées dans le cadre des normes de Wikipedia concernant la publication de commentaires publics. Toutefois, les détails spécifiques concernant le consentement des auteurs ne sont pas divulgués.

En ce qui concerne le prétraitement, les données ont été anonymisées et les informations personnellement identifiables ont été supprimées pour protéger la vie privée des utilisateurs.

Processus d'annotation

Le dataset est structuré autour de plusieurs catégories d'annotations qui permettent de définir la nature de la toxicité des commentaires. Celles-ci incluent : **Toxique**, **Très toxique**, **Obscène**, **Menace**, **Insulte**, **Haine identitaire**. Ces catégories ont été choisies pour couvrir un large éventail de comportements toxiques potentiellement rencontrés dans les commentaires en ligne.

La méthode d'annotation repose sur l'intervention de multiples annotateurs pour chaque commentaire. Cette approche vise à maximiser la fiabilité des annotations. Le recours à plusieurs annotateurs permet de réduire les biais individuels et d'améliorer la précision générale des données annotées, assurant ainsi que les modèles de machine learning entraînés avec ce dataset peuvent fonctionner de manière efficace et équitable.

Distribution

Le dataset est disponible à des fins de recherche non commerciales. Les utilisateurs doivent généralement accepter des conditions d'utilisation qui limitent l'utilisation commerciale et la redistribution.

3. Statistiques descriptives du jeu de données

Division du dataset

Le dataset a été divisé en trois parties : entraînement, validation et test. Voici la répartition du nombre de commentaires dans chacune de ces parties :

Catégorie	Nombre de commentaires
Train	127,656
Validation	31,915
Test	63,978

TABLE 3.1 – Répartition du nombre de commentaires

Répartition des labels

Les commentaires toxiques sont minoritaires dans l'ensemble des données. En effet il y a **10.2%** de commentaire globalement non-toxique. Cela peut poser des problèmes lors de l'entraînement des modèles, car les classes minoritaires peuvent être sous-représentées et donc mal apprises. Il y a aussi une répartition inégale au sein des labels de toxicité On peut remarquer que la somme des pourcentages n'est pas égale à 100% car un commentaire peut avoir plusieurs labels. On est donc dans un problème de classification multi-labels.

Label	Pourcentage
toxic	94.3%
severe_toxic	9.8%
obscene	51.9%
threat	3.1%
insult	48.2%
identity-hate	8.6%

TABLE 3.2 – Répartition des labels sur les commentaires globalement toxiques

Corrélation entre les labels

On peut remarqué que les labels de toxicité sont fortement corrélés entre eux. On peut dès à présent anticiper une difficulté du modèle à distinge un commentaire toxique d'un commentaire obscene (**74%** de corrélation). Cela représente un point à prendre en compte lors de la conception du modèle. La matrice de corrélation est inclus dans l'appendice A.

Longueur des commentaires

La distribution de la longueur des commentaires est très variée. En effet l'écart type est bien plus élevé que la moyenne. La longueur moyenne des commentaires est de **395** caractères, avec un écart-type de **593** caractères. Cela peut poser des problèmes lors de la conception du modèle. Il est donc important de prétraiter les données pour normaliser la longueur des commentaires.

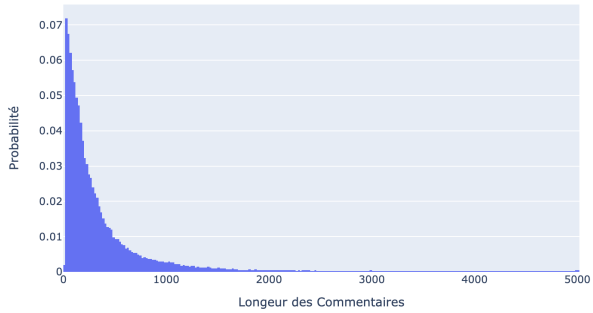

FIGURE 3.1 – Distribution de la longueur des commentaires

Distribution des mots

Dans le jeu de données, il y a un total de environs 180 000 mots uniques. Sans surprises, les mots les plus fréquents sont les mots de liaison et les mots vides. Mais en vue de la source du dataset les mots **articles**, **wikipedia** et **pages** apparaissent aussi très souvent dans les commentaires (top 26, 30, 31 respectivement). En effet les utilisateurs peuvent citer des sources pour appuyer leurs propos. Dans les commentaires toxiques, on trouve des insultes, des mots vulgaires et des mots discriminatoires. On peut visualiser ces derniers en utilisant WordCloud. Cela représente une représentation visuelle des mots les plus utilisés. En annexe A, on peut voir une représentation WordCloud générée à partir des commentaires du jeu de données filtré selon le type de toxicité.

4. Tokenizer

4.1 Pré-traitement du jeu de données

Le pré-traitement des données est une étape essentielle dans le processus d'analyse de texte. Il vise à nettoyer, transformer et préparer les données brutes afin de les rendre exploitables pour l'entraînement des modèles. Dans cette section, nous appliquerons différentes techniques de pré-traitement sur notre jeu de données afin de le rendre apte à être utilisé dans nos modèles prédictifs.

4.1.1 Tokenisation à base d'expressions régulières (RegexTokenizer)

La tokenisation est le processus de division du texte en unités plus petites appelées "tokens". La tokenisation à base d'expressions régulières utilise des règles basées sur des motifs d'expressions régulières pour diviser le texte en tokens.

Implémentation de la tokenisation à l'aide de la classe RegexTokenizer

Nous commençons par importer la classe RegexTokenizer et appliquer la tokenisation sur notre jeu de données.

Avant tokenisation	**Après tokenisation**
"Hello, World !"	['Hello', ',', 'World', '!']
"I love NLTK"	['I', 'love', 'NLTK']

TABLE 4.1 – Exemple de tokenisation à base d'expressions régulières

4.1.2 Tokenisation byte-pair encoding (TikToken)

Le byte-pair encoding (BPE) est une méthode de tokenisation qui découpe le texte en sous-unités de texte appelées "tokens" en utilisant un algorithme de compression de données.

Implémentation de la tokenisation à l'aide de la bibliothèque TikToken

Nous appliquons la tokenisation BPE sur notre jeu de données en utilisant la bibliothèque TikToken.

Avant tokenisation	Après tokenisation
“Hello, World !”	[’Hello’, ’,’, ’World’, ’ !’]
“I love TikToken”	[’I’, ’love’, ’Ti’, ’k’, ’To’, ’ken’]

TABLE 4.2 – Exemple de tokenisation BPE

4.1.3 Comparaison avec d’autres tokenizers

Nous avons également comparé la performance de notre tokenizer avec d’autres options disponibles dans la bibliothèque MinBPE.

Tokenizer	Temps d’entrainement (minutes)
RegexTokenizer	182
TikToken	0 (déja entrainé)
BasicTokenizer	69

TABLE 4.3 – Comparaison des temps d’entrainement des tokenizers

Nous constatons que le RegexTokenizer et le TikToken sont significativement plus rapides que le BasicTokenizer. TikToken est le plus rapide parmi tous les tokenizers testés.

4.1.4 Méthodes de normalisation du texte

La normalisation du texte est une étape cruciale du pré-traitement des données textuelles. Elle vise à uniformiser le texte en le mettant en minuscules, en supprimant les stop words et en lemmatisant les mots.

Suppression des stop words

Les stop words sont des mots courants qui n’apportent pas beaucoup de valeur sémantique au texte(exemple : ’a’ , ’the’ ,’for’,...). Nous les supprimerons de notre jeu de données.

Lemmatisation

La lemmatisation consiste à réduire les mots fléchis ou dérivés à leur forme de base ou racine. Cela permet de normaliser le texte et de réduire la dimensionnalité de l’espace des features.

Mise en minuscules

La mise en minuscules permet d'uniformiser le texte en convertissant toutes les lettres en minuscules. Cela permet d'éviter les doublons dus à la casse.

Identification des verbes et des noms

Nous avons modifié la méthode de lemmatisation pour identifier les verbes et les noms dans une phrase. Parfois, les verbes étaient considérés comme des noms et ne pouvaient pas être lemmatisés avec le lemmatiseur par défaut.

Les caractères spéciaux

En examinant attentivement notre base de données, nous avons constaté la présence de nombreux tokens représentant des URL, des adresses e-mail et d'autres éléments inutiles pour nos calculs d'entraînement des modèles. Par conséquent, nous avons décidé de les supprimer afin de réduire au maximum le bruit dans nos données et ainsi optimiser nos modèles.

Remplacement des emojis

Certains emojis peuvent etre important pour l'indication d'une toxicité ou non c'est pourquoi nous avons pris la décision de remplacer certains emojis par un mot le représentant au mieux afin de permettre aux models de mieux s'adapter.

Effacement des ponctuations

Parfois, certaines ponctuations occupent trop d'espace sans être utiles au modèle. Ainsi, nous pouvons supprimer tous les caractères qui ne sont pas dans la table ASCII ainsi que les signes de ponctuation qui pourraient compromettre l'entraînement des modèles.

Effacement des duplication

En constatant que plusieurs messages vulgaires étaient dissimulés parmi nos tokens en raison de la duplication des premières et/ou dernières lettres, nous avons été contraints de corriger ces tokens pour leur attribuer une plus grande valeur.

Pré traitement	Remove special caractère	Remove Stop W	Replace emojis	Lowercase	Lemmatization	Remove punctuation	Remove duplication
BPE Tokenizer 2	VRAI	FAUX	FAUX	VRAI	FAUX	VRAI	VRAI
BPE Tokenizer 1	VRAI	FAUX	FAUX	VRAI	FAUX	FAUX	VRAI
Word Tokenizer 4	VRAI	VRAI	VRAI	VRAI	VRAI	VRAI	VRAI
Word Tokenizer 3	VRAI	VRAI	VRAI	VRAI	VRAI	FAUX	FAUX
Word Tokenizer 2	VRAI	FAUX	FAUX	VRAI	FAUX	VRAI	FAUX
Word Tokenizer 1	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX
GPT Tokenizer 4	VRAI	VRAI	VRAI	VRAI	VRAI	VRAI	VRAI
GPT Tokenizer 3	VRAI	VRAI	VRAI	VRAI	VRAI	FAUX	FAUX
GPT Tokenizer 2	VRAI	FAUX	FAUX	VRAI	FAUX	FAUX	FAUX
GPT Tokenizer 1	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX
Baseline	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX

TABLE 4.4 – Pré traitement comparatif

Tableau représentatif des différentes fonctions de pré-traitement

4.1.5 Pré-traitement supplémentaire du jeu de données

En plus des techniques de pré-traitement déjà mentionnées, nous avons également effectué les actions suivantes pour rendre notre jeu de données plus adapté à l'entraînement de nos modèles :

Réflexion sur la suppression des symboles opérateurs et des chiffres

Dans le cadre du prétraitement des données, nous avons envisagé initialement de supprimer les symboles opérateurs et les chiffres présents dans nos textes. Cependant, après réflexion, nous avons réalisé que ces symboles pouvaient avoir une valeur sémantique importante pour déterminer si une phrase est toxique ou non. Ainsi, au lieu de les supprimer, nous avons mis en place un processus sophistiqué consistant à utiliser une série d'expressions régulières (regex) pour remplacer ces symboles par leur signification correspondante. Par exemple, nous avons remplacé des symboles comme " :/" par leur signification afin d'améliorer la compréhension du texte par nos modèles. Cette approche nous permet d'assurer des interactions plus contextuelles et pertinentes avec les utilisateurs, tout en garantissant une analyse précise et efficace du langage. Chacune de ces expressions régulières a été soigneusement personnalisée par notre équipe pour répondre aux besoins spécifiques de notre projet, permettant ainsi à notre IA de mieux comprendre le langage naturel et d'interagir de manière plus fluide avec les utilisateurs.

Élément toxique	Fréquence
8======d	penis
:)))))	happy
:(	sad
:)	happy
:D	laught
8==========>	penis

TABLE 4.5 – Exemple d'éléments toxiques contenant des symboles opérateurs et des chiffres que nous avons remplacé par leurs signification

Suppression des balises HTML, des URL et des adresses e-mail

Nous avons supprimé les balises HTML, les URL et les adresses e-mail de notre jeu de données. Ces éléments étaient fréquents dans notre base de données en raison d'erreurs de récupération de données lors de la collecte de commentaires sur Wikipedia.

Correction des erreurs orthographiques et des fautes de frappe

Nous avons utilisé un algorithme de correction des erreurs orthographiques et des fautes de frappe pour améliorer la qualité de nos données textuelles. Cet algorithme remplace les mots incorrectement orthographiés par les mots les plus proches dans notre dictionnaire de référence. Par exemple les caractères spéciaux, tels que 'ı' utilisé à la place de 'i', seront corrigés. Voici des exemples de phrases avant et après l'application de notre algorithme de correction :

Avant correction	Après correction
lıes	lies
penisd i'\m	penis i'm
fùck8	fuck

TABLE 4.6 – Exemple de phrase avant et après correction

Non-application de la correction d'orthographe

Initialement, nous avions envisagé d'appliquer un algorithme de correction d'orthographe pour améliorer la qualité de nos données textuelles. Cependant, après avoir

réalisé que cette étape prenait trop de temps et pouvait introduire des erreurs, notamment en modifiant des noms propres en mots n'ayant aucun rapport, nous avons décidé de ne pas l'appliquer. En effet, cela pourrait introduire des biais importants, notamment si un prénom ressemble à une insulte, ce qui pourrait transformer une phrase anodine en une phrase insultante. Par conséquent, nous avons choisi de ne pas effectuer cette étape de correction d'orthographe pour garantir l'intégrité et la qualité de nos données textuelles.

Exemple d'une phrase après toutes les étapes de normalisation

Étape	**Texte normalisé**
Texte initial	motherrrrrrrrrrrFuckkkkkkkk!!!!aaaaaaa***** stringssss ***!!!!!!!bitchhhhhh :) 8=========
Après remplacement des emojis	motherrrrrrrrrrrrFuckkkkkkkk!!!!aaaaaaa***** stringssss ***!!!!!!!bitchhhhhh happy 8=========
Après suppression des caractères spéciaux	motherFuck a strings bitch happy penis
Après tokenisation	'motherFuck', 'a', 'strings', 'bitch', 'happy', 'penis'
Après conversion en minuscules	'motherfuck', 'a', 'strings', 'bitch', 'happy', 'penis'
Après lemmatisation	'motherfuck', 'a', 'string', 'bitch', 'happy', 'penis'
Après suppression des stopwords	'motherfuck', 'string', 'bitch', 'happy', 'penis'
Texte final après re-tokenisation	mother fuck string bitch happy penis

TABLE 4.7 – Étapes de normalisation du texte

5. Naive Bayes

5.1 Pré-traitement des données

Nous avons d'abord effectué un pré-traitement des données en utilisant différentes techniques :

5.1.1 Tokenisation des commentaires

Nous avons utilisé la tokenisation pour diviser les commentaires en tokens individuels.

Tokenisation à l'aide de nôtre API `tokenize_apy`

La tokenisation des commentaires a été réalisée en utilisant l'API `tokenize_apy`.

Équilibrage du jeu de données

Nous avons équilibré le jeu de données en échantillonnant un nombre égal de commentaires toxiques et non toxiques.

5.2 Analyse exploratoire des données

Voici les statistiques descriptives sur le jeu de données équilibré :

5.2.1 Statistiques descriptives

— Nombre total de documents : 25 962
— Nombre total de tokens : 127 656
— Nombre de classes à prédire : 2 (toxique ou non toxique)
— Les tokens les plus fréquents incluent "grand", "ter", "burn", etc.

5.3 Entraînement du modèle

Nous avons entraîné le modèle Bayésien naïf en suivant les étapes suivantes :

5.3.1 Préparation des données

— Calcul de la fréquence des mots et sélection des stop words.
— Entraînement du modèle en utilisant `CountVectorizer` et `MultinomialNB`.

5.3.2 Évaluation des performances du modèle

Les "Feature Dimensions" indiquent la taille de la matrice de caractéristiques utilisée pour chaque méthode. Le premier nombre représente le nombre d'échantillons (lignes) dans l'ensemble de données, et le deuxième nombre représente le nombre de caractéristiques (colonnes) générées par le vectoriseur.

TABLE 5.1 – Rapports de classification combinés pour différentes méthodes

Méthode	Dimensions des Features	Classe 0		Classe 1		Précision	Macro Moy.	Moy. Pondérée
		Précision	Rappel	Précision	Rappel			
comment_text_baseline	(63978, 52493)	0.98	0.85	0.39	0.86	0.85	(0.68, 0.86, 0.72)	(0.92, 0.85, 0.88)
comment_text_bpe_tokenize_full_normalization	(63978, 22188)	0.98	0.86	0.40	0.87	0.86	(0.69, 0.86, 0.73)	(0.93, 0.86, 0.88)
comment_text_word_tokenize_no_normalization	(63978, 52478)	0.98	0.85	0.39	0.86	0.85	(0.68, 0.85, 0.72)	(0.92, 0.85, 0.88)
comment_text_gpt_tokenize_no_normalization	(63978, 44632)	0.97	0.90	0.45	0.74	0.89	(0.71, 0.82, 0.75)	(0.92, 0.89, 0.90)
comment_text_word_tokenize_normalization	(63978, 42934)	0.98	0.85	0.39	0.87	0.85	(0.69, 0.86, 0.73)	(0.93, 0.85, 0.88)
comment_text_gpt_tokenize_normalization	(63978, 24578)	0.98	0.87	0.40	0.80	0.86	(0.69, 0.84, 0.73)	(0.92, 0.86, 0.88)
comment_text_word_tokenize_full_normalization	(63978, 41908)	0.98	0.85	0.39	0.88	0.85	(0.69, 0.86, 0.73)	(0.93, 0.85, 0.88)
comment_text_gpt_tokenize_full_normalization	(63978, 22750)	0.98	0.86	0.40	0.85	0.86	(0.69, 0.85, 0.73)	(0.92, 0.86, 0.88)
comment_text_word_tokenize_simple_normalization	(63978, 48335)	0.98	0.85	0.39	0.86	0.85	(0.68, 0.86, 0.72)	(0.92, 0.85, 0.88)
comment_text_gpt_tokenize_simple_normalization	(63978, 29389)	0.98	0.86	0.39	0.82	0.86	(0.69, 0.84, 0.72)	(0.92, 0.86, 0.88)
comment_text_bpe_tokenize_simple_dup_normalization	(63978, 27126)	0.98	0.86	0.39	0.85	0.86	(0.69, 0.85, 0.73)	(0.92, 0.86, 0.88)
comment_text_bpe_tokenize_no_dup_no_punc_normalization	(63978, 27290)	0.98	0.86	0.39	0.85	0.86	(0.69, 0.85, 0.73)	(0.92, 0.86, 0.88)

5.3.3 Validation croisée

Nous avons également effectué une validation croisée en utilisant la métrique `f1_macro`.

Métrique	**Score**
f1_macro (moyenne)	0.8438
f1_macro (écart type)	0.0055

TABLE 5.2 – Résultats de la validation croisée

5.4 Analyse des résultats

5.4.1 Calcul des probabilités

— Probabilités a priori pour chaque classe : $\{0 : 0.89831, 1 : 0.10168\}$
— Vocabulaire contient 33397 mots uniques

5.4.2 Calcul des vraisemblances

Nous avons calculé les probabilités conditionnelles (vraisemblances) pour chaque mot dans chaque classe.

5.4.3 Prédiction

Nous avons utilisé le modèle entraîné pour prédire la toxicité des commentaires de test. Par exemple, le commentaire "chretian attack" a été prédit comme toxique avec une probabilité de 0.000168.

5.4.4 Matrice de confusion

La matrice de confusion montre que le modèle prédit correctement les commentaires toxiques et non toxiques dans une large mesure.

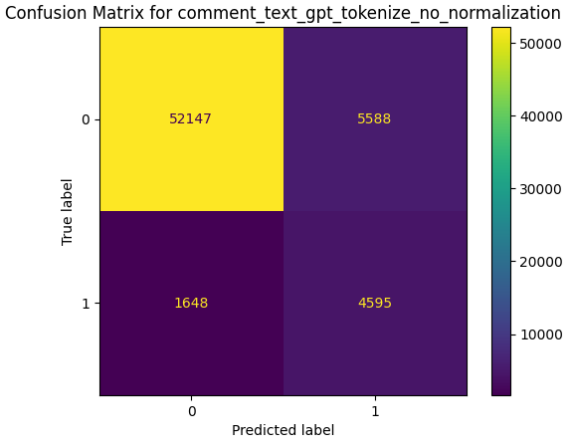

FIGURE 5.1 – Matrice de confusion du modèle Bayésien naïf

5.4.5 Etudes Annexe sur les données

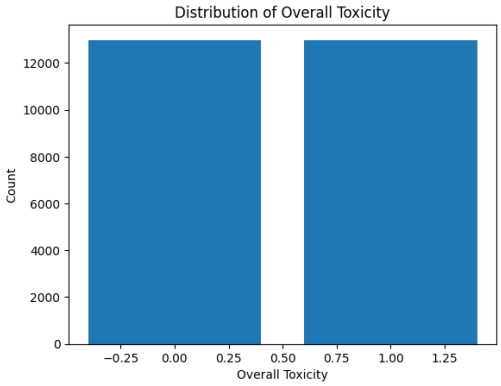

FIGURE 5.2 – Distribution de la toxicité des commentaires

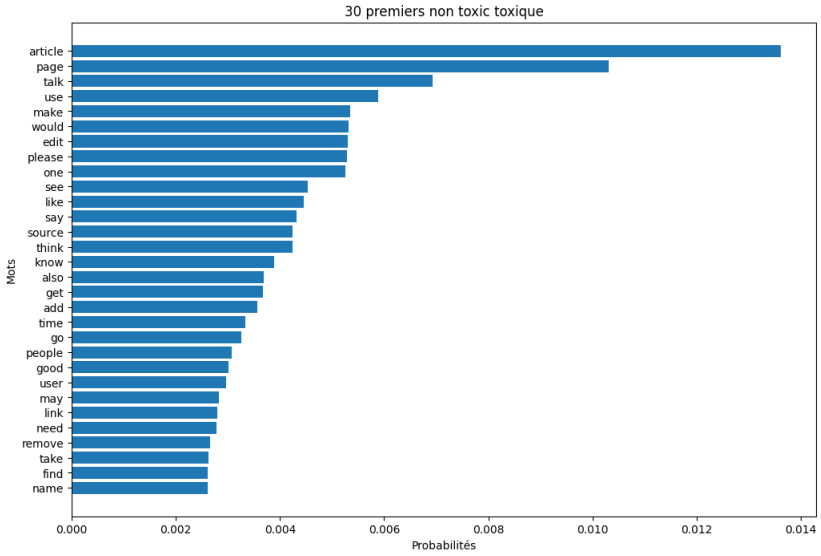

FIGURE 5.3 – Top 30 mots non toxiques

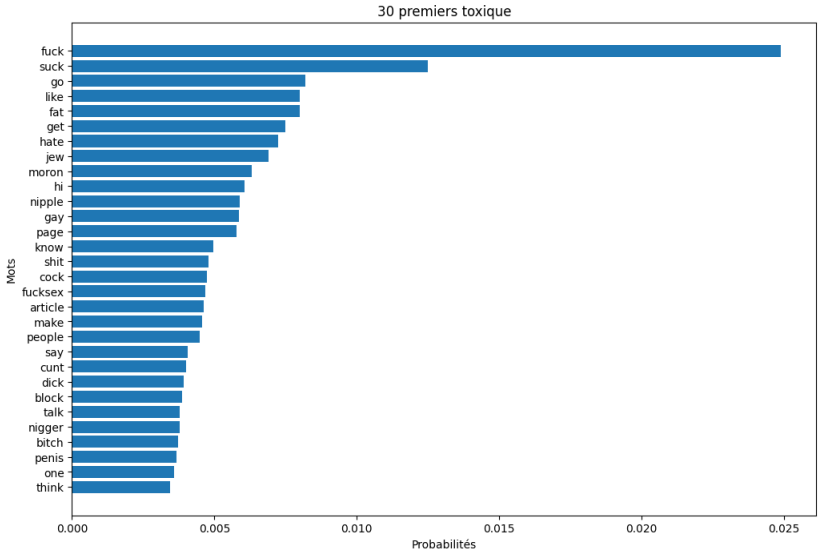

FIGURE 5.4 – Top 30 mots toxiques

6. N-gram

6.1 Pré-traitement des données

Nous avons d'abord effectué un pré-traitement des données en utilisant différentes techniques pour améliorer la qualité des entrées pour le modèle N-gram.

6.1.1 Tokenisation des commentaires

La tokenisation divise les commentaires en unités de base appelées tokens, facilitant l'analyse et la modélisation des données textuelles.

Tokenisation à l'aide de notre API `tokenize_apy`

Nous avons utilisé l'API `tokenize_apy` pour effectuer la tokenisation des commentaires. Cette API nous permet de transformer les phrases en une séquence de mots individuels ou tokens, essentielle pour la création des n-grams.

Équilibrage du jeu de données

Pour éviter les biais dans notre modèle, nous avons équilibré le jeu de données en échantillonnant un nombre égal de commentaires toxiques et non toxiques. Cette étape est cruciale pour s'assurer que le modèle apprend de manière équitable des deux types de données.

6.2 Analyse exploratoire des données

Avant de procéder à l'entraînement du modèle, nous avons effectué une analyse exploratoire pour comprendre les caractéristiques de notre jeu de données.

6.2.1 Statistiques descriptives

Voici les statistiques descriptives sur le jeu de données équilibré :

— Nombre total de documents : 223 549
— Nombre total de tokens : Variable selon les n-grams
— Les n-grams les plus fréquents incluent "nigger nigger", "on my", "be blocked", etc.

6.3 Entraînement du modèle

Nous avons entraîné notre modèle N-gram en suivant une série d'étapes méthodiques.

6.3.1 Préparation des données

— Utilisation de `CountVectorizer` pour créer des bigrams, trigrams et 4-grams.
— Entraînement du modèle en utilisant les fréquences des n-grams extraits des commentaires.

6.3.2 Création du modèle de bigrams et trigrams

Nous avons utilisé 'CountVectorizer' pour créer des modèles de bigrams (séquences de deux mots) et trigrams (séquences de trois mots) à partir des phrases.

6.3.3 Fréquence des n-grams

Nous avons calculé la fréquence des n-grams les plus courants dans le jeu de données pour avoir une meilleure compréhension de leur distribution.

Bigram	**Fréquence**
nigger nigger	3411
on my	3067
be blocked	2761
fuck you	2681

TABLE 6.1 – Fréquence des bigrams les plus courants

Trigram	**Fréquence**
once upon a	26
on my talk	20
be blocked from	15

TABLE 6.2 – Fréquence des trigrams les plus courants

6.3.4 Fonction de prédiction

Nous avons défini des fonctions de prédiction pour compléter les phrases basées sur les n-grams les plus fréquents.

Prédiction avec les bigrams

— Exemple de prédiction bigram : "Do you like playing" → "the"

Prédiction avec les trigrams

— Exemple de prédiction trigram : "once upon a" → "time"

Complétion de phrase

Nous avons également créé une fonction `complete_the_phrase` qui prend en entrée une phrase de départ et prédit la suite.

— Exemple d'utilisation : "once upon" → "a time when I was just a few days ago I'm not"

6.4 Évaluation des performances du modèle

Pour évaluer les performances de notre modèle, nous avons calculé plusieurs métriques de performance, incluant la précision, le rappel et le F1-score pour différentes configurations de n-grams. On test sur les une cinquentaine de phrase le modèle configuré pour obtenir ces resultas.

N-gram	Précision	Rappel	F1-score
Bigram	0.0	0.0	0.0
Trigram	0.0	0.0	0.0
4-gram	0.0	0.0	0.0

TABLE 6.3 – Performances du modèle N-gram avec différentes configurations

6.5 Analyse des résultats

6.5.1 Interprétation des fréquences des n-grams

Les n-grams les plus fréquents identifiés révèlent des schémas de langage courants et des expressions fréquentes dans le corpus. Ces informations sont cruciales pour améliorer la compréhension et la génération de texte.

6.5.2 Prédiction et complétion de phrases

Les modèles de bigrams et trigrams ont été utilisés pour prédire le mot suivant dans une phrase donnée, et pour compléter des phrases entières. Cela démontre l'efficacité des n-grams pour la génération de texte basée sur des séquences de mots précédents.

6.6 Conclusion

Le modèle N-gram, bien que simple, offre une capacité intéressante pour prédire et générer du texte basé sur les séquences de mots précédents. Cependant, il est limité par la taille du contexte qu'il peut utiliser (les n-grams). Des modèles plus avancés, comme les modèles de langage basés sur les réseaux de neurones, peuvent utiliser un contexte plus large et capturer des relations plus complexes dans les données textuelles.

A. Appendix

A.1 Corrélation entre les labels

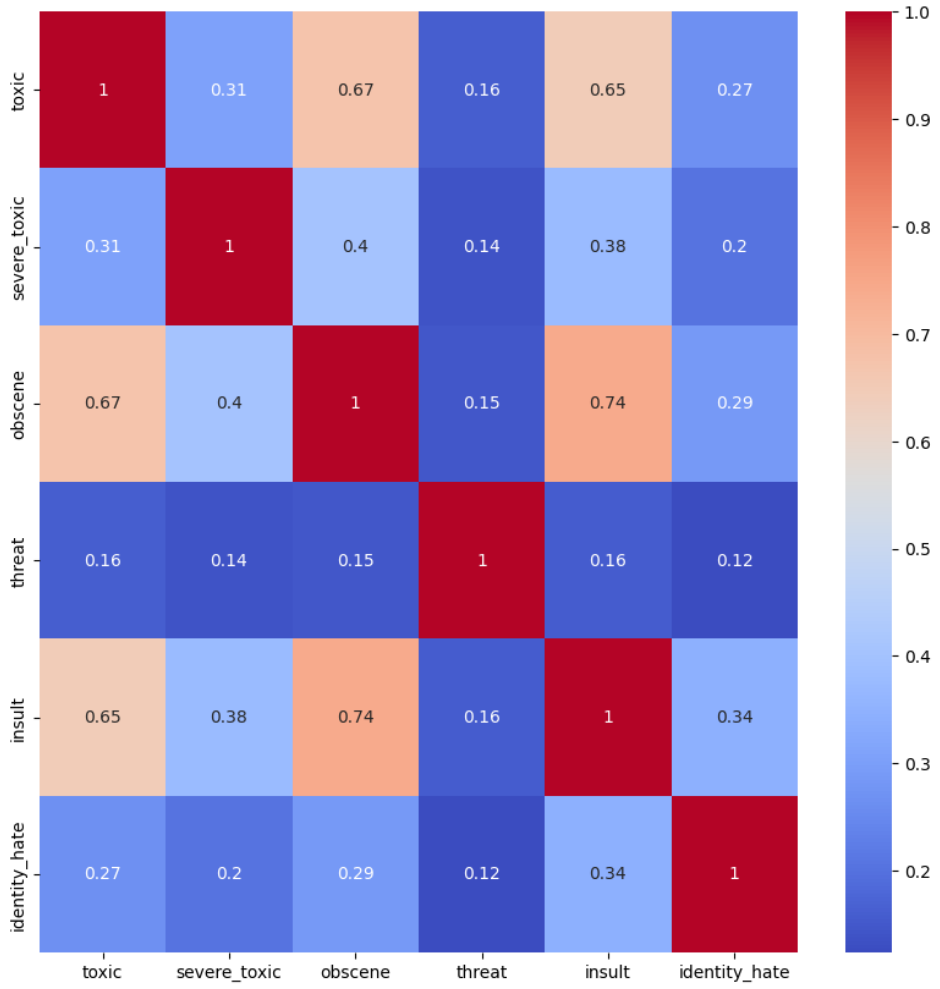

FIGURE A.1 – Matrice de corrélation des labels du dataset

A.2 Distribution des mots

(a) Word Cloud – Toxique

(b) Word Cloud – Haine identitaire

(c) Word Cloud – Obscene

(d) Word Cloud – Insult

FIGURE A.2 – Word Clouds