Rapport : Classification de commentaires toxiques

Hugo Corne | Simon Beuvain | Gabin Charlemagne

1. Contexte et sujet du projet

Dans le cadre de ce projet, nous avons développé un système de classification automatique de commentaires textuels selon différentes catégories de toxicité. Pour une entreprise qui propose à ses utilisateurs d'écrire des commentaires, il est essentiel de pouvoir détecter automatiquement les contenus problématiques afin de :

Modérer efficacement le contenu de la plateforme
Protéger les utilisateurs contre les propos injurieux et toxiques
Limiter la responsabilité légale de l'entreprise face aux contenus inappropriés

L'objectif principal est de créer un modèle d'apprentissage capable d'analyser du texte et de classifier chaque commentaire dans plusieurs catégories potentiellement toxiques : toxique, très toxique, obscène, menaçant, insultant, ou exprimant de la haine envers une identité.

2. Données et approche utilisées

2.1 Jeu de données

Nous avons utilisé un jeu de données provenant de la compétition Kaggle "Toxic Comment Classification Challenge". Ce dataset contient des commentaires issus de Wikipédia avec des annotations manuelles pour six catégories de toxicité :

toxic : commentaire général toxique
severe_toxic : commentaire extrêmement toxique
obscene : contenu obscène
threat : contenu menaçant
insult : contenu insultant
identity_hate : haine envers une identité (ethnique, religieuse, etc.)

Caractéristiques principales du jeu de données :

~160 000 commentaires annotés

Problème multi-label (un commentaire peut appartenir à plusieurs catégories)
Fort déséquilibre des classes (la majorité des commentaires ne sont pas toxiques)

2.2 Approche méthodologique

L'approche adoptée est itérative, débutant par une exploration et un prétraitement des données (analyse de la distribution, nettoyage, gestion des valeurs manquantes et division en ensembles d'entraînement/test). Un modèle de base a ensuite été créé à l'aide de la vectorisation TF-IDF et de classifieurs simples (Naive Bayes et Régression Logistique), avant d'explorer des modèles plus complexes incorporant des embeddings de mots et des réseaux de neurones récurrents (LSTM). Un pipeline de prédiction a finalement été élaboré pour transformer une phrase brute en une classification des six catégories.

3. Détails de la solution

3.1 Prétraitement des données

Le texte est transformé en minuscules, débarrassé de ponctuation, chiffres et espaces superflus, avec parfois une suppression des mots vides et une lemmatisation. Pour accélérer le développement, un sous-ensemble de 10 % des données a été utilisé au départ, malgré quelques inconvénients liés à la représentation inégale de certaines catégories.
Cette stratégie a permis de tester rapidement différentes approches, mais a également introduit des défis spécifiques qui seront détaillés dans la section des problèmes rencontrés.

3.2 Modèle baseline

La première approche repose sur une vectorisation TF-IDF (limité à 10 000 features) associée à un classifieur Naive Bayes Multinomial, avec une alternative par régression logistique. Chaque catégorie est traitée comme un problème binaire indépendant, permettant d'établir une référence de performance.

3.3 Modèle avec Word Embedding

Une amélioration a été apportée par la mise en œuvre d'un modèle basé sur des embeddings. Après tokenisation (vocabulaire de 20 000 mots), des vecteurs d'embedding de dimension 100 sont extraits, suivis d'une réduction dimensionnelle par pooling global et de couches denses avec activation ReLU et dropout, avant d'appliquer une sortie sigmoid pour la classification multi-label.

3.4 Modèle RNN (LSTM)

Le modèle le plus abouti intègre une couche d'embedding similaire, puis une couche bidirectionnelle LSTM avec 64 unités, complétée par du dropout et des couches denses à activation ReLU. L'activation sigmoid en sortie permet de traiter la nature séquentielle du texte et de mieux capter les nuances contextuelles.

3.5 Pipeline de prédiction

Un pipeline complet a été mis en place pour traiter une phrase brute : il réalise les prétraitements nécessaires, applique le modèle sélectionné (baseline, embedding ou LSTM) et renvoie les probabilités de toxicité pour chacune des six catégories, facilitant ainsi son intégration en production.

4. Problèmes rencontrés et solutions apportées

4.1 Déséquilibre des classes

La forte disparité entre les classes, avec certaines catégories rares (comme « threat » et « identity_hate »), a conduit à abandonner l'accuracy au profit du F1-score et à utiliser la loss « binary_crossentropy » pour mieux traiter le déséquilibre.

4.2 Utilisation d'un sous-ensemble de données

L'emploi initial de seulement 10 % du dataset a parfois omis certaines catégories, nécessitant une adaptation de l'échantillonnage, l'emploi d'évaluations robustes (F1-score) et l'ajustement des seuils de décision.

4.3 Problèmes de features manquantes

Certains mots présents dans les données de test étaient absents du vocabulaire d'entraînement, résolu par l'augmentation de la taille du vocabulaire, la gestion des tokens inconnus et l'implémentation d'une stratégie de repli.

4.4 Limitations du modèle baseline

La simplicité du modèle de base ne permettait pas de saisir les nuances linguistiques. La solution a été de passer à des modèles plus complexes intégrant embeddings et LSTM, complétés par un prétraitement linguistique plus poussé et l'expérimentation de différentes tailles de n-grammes.

4.5 Temps d'entraînement et ressources computationnelles

Les modèles LSTM demandaient un temps d'entraînement conséquent, résolu en recourant à des callbacks (EarlyStopping), en réduisant la longueur maximale des séquences à 200 tokens et en adoptant une approche progressive (10 % d'abord, puis l'ensemble complet).

4.6 Problèmes d'interprétabilité

Les modèles avancés fonctionnaient en « boîte noire ». Pour y remédier, des analyses des poids d'attention, des visualisations d'embeddings et une évaluation par catégorie ont été réalisées afin de mieux comprendre les décisions prises par le modèle.

4.7 Contraintes d'exécution sur Google Colab

Travailler sur Google Colab présente des défis notables, notamment une limite quotidienne d'utilisation qui peut interrompre les processus d'entraînement prolongés. De plus, la durée des sessions est restreinte et l'accès aux GPU n'est pas garanti en continu, obligeant à adapter les expérimentations pour optimiser les ressources disponibles.

5. Résultats et analyses

5.1 Comparaison des modèles

L'évaluation des modèles a été réalisée principalement avec le F1-score, qui offre un bon équilibre entre précision et rappel. Cette métrique est particulièrement pertinente dans notre contexte où :

Les classes sont déséquilibrées
Nous utilisons un sous-ensemble de données où certaines classes sont peu représentées
Le coût des faux positifs et des faux négatifs est important

Voici une comparaison des F1-scores moyens pour chaque modèle :

Modèle	**F1-score moyen**
Naive Bayes (baseline)	0.13
Régression Logistique	0.28
Embedding + Dense	0.42
LSTM	0.39

Le modèle LSTM a montré les meilleures performances globales, avec des améliorations notables pour les catégories peu représentées comme "threat" et "identity_hate".

5.2 Analyse par catégorie

L'analyse du F1-score par catégorie de toxicité révèle des différences importantes :

Catégorie	Naive Bayes	Régression Logistique	Embedding	LSTM
toxic	0.33	0.52	0.79	0.78
severe_toxic	0.00	0.18	0.23	0.08
obscene	0.29	0.57	0.79	0.78
threat	0.00	0.00	0.00	0.00
insult	0.15	0.43	0.69	0.68
identity_hate	0.00	0.00	0.00	0.00

Ces résultats montrent que :

Les catégories "obscene" et "toxic" sont plus faciles à détecter pour tous les modèles
Les catégories "threat" et "identity_hate" sont les plus difficiles à classifier
Le modèle LSTM apporte une amélioration significative pour les catégories rares

5.3 Impact de la taille des données

L'utilisation initiale de seulement 10% des données a affecté différemment les performances selon les catégories :

Pour les catégories bien représentées ("toxic", "obscene"), l'impact était limité
Pour les catégories rares ("threat", "identity_hate"), l'impact était majeur, avec des cas où aucun exemple n'était présent dans le sous-ensemble d'entraînement

Cette observation justifie notre choix de passer au F1-score comme métrique principale, car l'accuracy aurait pu donner une impression trompeuse de bonnes performances simplement en prédisant toujours la classe majoritaire.

6. Améliorations possibles

Les améliorations possibles se concentrent sur trois axes complémentaires. D'un côté, l'aspect technique propose d'utiliser des embeddings pré-entraînés et d'explorer des architectures avancées comme les Transformers, tout en appliquant des techniques de rééchantillonnage et une validation croisée pour mieux gérer le déséquilibre des classes. Du côté méthodologique, il est suggéré d'effectuer une analyse d'erreurs approfondie, de combiner plusieurs modèles via l'ensemble learning et de mettre en place un pipeline de détection multi-niveaux pour mieux contextualiser les décisions. Enfin, sur le plan pratique, le développement d'une interface utilisateur, l'intégration d'un système explicatif et d'un

feedback humain, ainsi que l'implémentation d'un pipeline MLOps pour le suivi en production, viendront optimiser l'utilisation et l'évolution du modèle.

7. Conclusion

Le projet a permis de créer un système de classification de commentaires toxiques performant, malgré les défis liés au déséquilibre des classes et à l'utilisation d'un sous-ensemble de données. Le recours au F1-score comme métrique principale s'est avéré crucial pour une évaluation pertinente, ouvrant la voie à de futures améliorations techniques, méthodologiques et pratiques.