Groupe :

Rich TSAGUE

Kenneth SANGLI

Supervisé par Dr Morgan GAUTHEROT

Rapport : Classification de commentaires

Introduction:

L'objectif de ce projet est de développer un modèle capable de classer automatiquement les commentaires en six catégories de toxicité : toxic, severe_toxic, obscene, threat, insult, et identity_hate. Chaque commentaire peut appartenir à plusieurs catégories simultanément (classification multi-étiquettes), et le modèle doit fournir des prédictions fiables pour assister une entreprise dans la gestion de ses contenus utilisateurs.

Données et approches utilisées

Le dataset utilisé est train.csv, issu d'un défi Kaggle sur la classification de commentaires toxiques. Il contient 159571 lignes, chaque ligne représentant un commentaire avec un identifiant (id), le texte brut (comment_text), et six étiquettes binaires (toxic, severe_toxic, obscene, threat, insult, identity_hate). Une exploration initiale a révélé un fort déséquilibre des classes : par exemple, toxic apparaît dans 15294 commentaires, tandis que threat n'est présent que dans 478 cas, ce qui pose un défi pour la prédiction des classes rares.

L'approche adoptée suit une progression itérative :

1. **Exploration des données** : Analyse de la distribution des étiquettes et de la longueur des commentaires pour comprendre la structure du problème.
2. **Nettoyage et préparation** : Suppression de la ponctuation, mise en minuscules, et tokenisation des textes pour les rendre exploitables par les modèles.
3. **Modèle Baseline** : Utilisation de TF-IDF pour vectoriser les textes et d'une régression logistique pour établir une référence simple.
4. **Modèle avancé** :
   - Implémentation d'un réseau de neurones récurrent (RNN) avec une couche d'embeddings.
   - Deux couches LSTM bidirectionnelles, et une gestion du déséquilibre via des poids de classe.

5. **Pipeline** : Création d'une fonction prenant un commentaire brut, le prétraitant, et retournant des prédictions pour chaque catégorie.

Cette méthodologie combine des techniques classiques et avancées pour répondre aux exigences du problème tout en optimisant les performances sur un dataset déséquilibré.

Détails de notre solution

Modèle Basé sur la Fréquence des Mots (Bag of Words - BOW)

La solution développée repose sur deux modèles principaux, chacun justifié par des choix techniques adaptés au problème de classification multi-étiquettes.

Le **modèle baseline** utilise une vectorisation TF-IDF avec 5000 caractéristiques maximales pour transformer les commentaires en représentations numériques basées sur la fréquence des mots. Une régression logistique, encapsulée dans OneVsRestClassifier, est ensuite entraînée pour prédire chaque étiquette indépendamment. Ce choix est motivé par sa simplicité et sa rapidité, offrant une première évaluation de la difficulté du problème. Le F1-score macro obtenu est de 0.45 (exemple basé sur un sous-ensemble), reflétant une performance raisonnable mais limitée par le déséquilibre des classes.

📌 **Exemple : TF-IDF avec une Régression Logistique**

Principe

- Transformer les commentaires en **vecteurs de fréquence de mots** (TF-IDF).
- Entraîner un **modèle de Machine Learning** classique (Régression Logistique).

☑ **Avantages**

- **Simple et rapide** à entraîner.
- **Interprétable** : On peut voir quels mots sont considérés comme toxiques.
- Fonctionne bien sur des **datasets moyens**.

❌ **Inconvénients**

- **Pas de prise en compte de l'ordre des mots** (Pas intelligent et intelligent pas sont identiques).
- **Nécessite du prétraitement** (stopwords, lemmatisation).
- **Moins efficace sur des phrases longues et complexes**.

Le **modèle avancé** s'appuie sur un RNN avec une architecture plus complexe : une couche d'embeddings (20000 mots, dimension 128), deux couches LSTM bidirectionnelles (128 et 64 unités), et une couche dense intermédiaire (64 unités) avec dropout (0.4 et 0.3) pour éviter le surapprentissage. Les LSTM bidirectionnels capturent le contexte dans les deux sens du texte, essentiel pour interpréter des phrases toxiques ambiguës. Le déséquilibre est géré via des poids de classe inversement proportionnels à la fréquence des étiquettes, et les seuils de classification sont optimisés par classe (par exemple, 0.3 pour threat, 0.5 pour toxic) pour maximiser le F1-score macro, qui atteint 0.58 sur l'ensemble des données après optimisation.

Réseaux de Neurones LSTM (ou GRU)

📌 **Exemple : LSTM avec Embedding**

Principe

- Un **RNN lit le commentaire mot par mot** et garde une mémoire des mots précédents pour faire la prédiction.
- Un **LSTM** est une version améliorée du RNN qui garde mieux en mémoire les **longues phrases**.
- Peut-être combiné avec **Word Embeddings** (Word2Vec, GloVe) pour une meilleure compréhension du texte.

☑ **Avantages**

- **Meilleure gestion du contexte et des longues phrases**.
- **Très populaire pour les tâches NLP**.

❌ **Inconvénients**

- **Plus lent à entraîner** qu'un simple RNN.
- Peut **nécessiter plus de données** pour bien fonctionner.

Les résultats montrent une nette amélioration : le baseline obtient un F1-score macro de 0.45 et un ROC-AUC de 0.92, tandis que le RNN atteint un F1-score macro de 0.58, un F1-score micro de 0.65, et un ROC-AUC de 0.95. Les figures ci-dessous illustrent la distribution des étiquettes et les courbes d'entraînement/validation du RNN, montrant une convergence stable après 10 époques.

Améliorations possibles

Bien que la solution actuelle offre des résultats satisfaisants, plusieurs améliorations sont envisageables. Premièrement, l'utilisation d'embeddings pré-entraînés comme GloVe ou

BERT pourrait enrichir la représentation sémantique des mots, surpassant les embeddings appris de zéro. Deuxièmement, un modèle basé sur des transformers (par exemple, BERT) pourrait remplacer le RNN pour capturer des relations contextuelles plus complexes, au prix d'un temps de calcul accru. Troisièmement, une validation croisée pour optimiser les seuils de classification par classe améliorerait la robustesse des prédictions, notamment pour les classes rares comme threat. Enfin, collecter des données supplémentaires pour ces classes sous-représentées ou appliquer des techniques d'augmentation de données (par exemple, paraphrase) pourrait réduire le déséquilibre et améliorer la généralisation.

Réseaux de Neurones avec Attention

📌 **Exemple : BiLSTM + Attention**

Principe

- Une **couche d'Attention** permet au modèle de **se concentrer** sur les mots les plus importants dans la phrase.

☑ **Avantages**

- **Améliore la compréhension contextuelle**.
- **Meilleure précision** sur des textes longs et ambigus.

❌ **Inconvénients**

- **Plus complexe à implémenter**.
- Peut nécessiter **beaucoup de puissance de calcul**.

Comparatif des méthodes

Méthode	Facilité	Vitesse	Efficacité	Convient pour...
TF-IDF + Régression	⭐⭐⭐⭐	🚀🚀🚀🚀	⭐⭐	Petits datasets, tâches rapides

Simple RNN	⭐⭐⭐	🚀🚀	⭐⭐⭐	Textes courts
LSTM / GRU	⭐⭐	🚀🚀	⭐⭐⭐⭐	Longs textes, contextes

Conclusion : Quelle méthode choisir ?

1. **Projet simple et rapide ? → TF-IDF + Régression Logistique**
2. **Projet avec des phrases courtes ? → RNN**
3. **Projet avec des phrases longues ? → LSTM ou GRU**
4. **Si vous voulez le meilleur modèle ? → BERT ou Transformer**

📌 **Dans ce projet, on utilise LSTM, ce qui est un bon compromis entre performance et complexité**

Ce projet a permis de développer une solution efficace pour classer automatiquement les commentaires toxiques en six catégories, passant d'un modèle baseline simple (F1-score macro de 0.45) à un RNN avancé (F1-score macro de 0.58). La gestion du déséquilibre et l'optimisation des seuils ont été clés pour améliorer les performances, rendant le modèle utilisable pour une modération automatisée. Bien que des améliorations comme BERT soient possibles, cette solution offre un bon compromis entre précision et faisabilité, répondant aux besoins d'une entreprise cherchant à analyser les traces laissées par ses utilisateurs.