Titouan Glangetas et Romain Dujardin M1 Isen spé IA

Rapport sur la Classification de Commentaires Toxiques

Introduction

Pour une entreprise permettant aux utilisateurs d'écrire des commentaires, la détection des commentaires toxiques est cruciale, à la fois pour préserver la qualité des échanges et protéger les utilisateurs. Dans ce cadre, notre étude s'est concentrée sur la classification automatique des commentaires toxiques en utilisant des techniques évolutives en traitement du langage naturel (NLP) :

1. Un modèle de base utilisant la vectorisation TF-IDF associé à une régression logistique,
2. Une première approche de deep learning basée sur des réseaux de neurones récurrents (RNN) avec LSTM unidirectionnel,
3. Un modèle final optimisé intégrant une vectorisation intégrée et un LSTM bidirectionnel couplé à plusieurs couches denses.

Chaque étape permet d'apporter des améliorations pour mieux capturer le sens des commentaires et traiter efficacement la classification multi-label.

Partie 1 : Modèle 1 – Approche Basique avec TF-IDF

1.1 Présentation du Modèle

Le Modèle 1 utilise la méthode TF-IDF (Term Frequency-Inverse Document Frequency) pour transformer les commentaires en vecteurs numériques. TF-IDF combine deux mesures :

- **TF (Term Frequency)** : fréquence d'apparition d'un mot dans un document.
- **IDF (Inverse Document Frequency)** : évalue l'importance d'un mot à travers l'ensemble des documents en réduisant l'importance des termes très fréquents et en augmentant celle des termes rares.

Chaque commentaire est nettoyé (minuscules et suppression des caractères spéciaux), puis vectorisé en limitant à 10 000 caractéristiques via TfidfVectorizer. Ensuite, la classification est effectuée selon une approche multi-label (toxic, severe_toxic, obscene, threat, insult, identity_hate), en utilisant un classificateur OneVsRest avec régression logistique pour chaque classe.

1.2 Points Forts et Limitations

Avantages du Modèle 1 (TF-IDF + Régression Logistique) :

- Simple à implémenter et rapide en temps de calcul.
- Interprétable grâce à la transparence des scores TF-IDF attribués aux mots.

Limitations :

- Ne prend pas en compte l'ordre des mots ni leur contexte (approche « sac de mots »), perdant ainsi les nuances des commentaires.
- Incapacité à gérer efficacement les interactions complexes entre les mots ou les classes, à cause de la nature linéaire du modèle.
- Performances inégales selon les classes : bonnes sur les catégories fréquentes (toxic, obscene) mais faibles sur les classes rares (« severe_toxic », « threat », « identity_hate »).

Résultats clés :

- Précision moyenne : ~0.87
- F1-score global : ~0.68 (faible sur certaines classes sensibles).

	precision	recall	f1-score	support
toxic	0.91	0.62	0.74	3056
severe_toxic	0.59	0.26	0.36	321
obscene	0.92	0.62	0.74	1715
threat	0.45	0.12	0.19	74
insult	0.84	0.53	0.65	1614
identity_hate	0.68	0.15	0.25	294
micro avg	0.88	0.56	0.68	7074
macro avg	0.73	0.38	0.49	7074
weighted avg	0.87	0.56	0.68	7074
samples avg	0.06	0.05	0.05	7074

Partie 2 : Modèle 2 – Approche Deep Learning avec RNN et LSTM Unidirectionnel

Présentation du Modèle

Le Modèle 2 utilise une approche deep learning basée sur les réseaux de neurones récurrents (RNN) avec des couches LSTM unidirectionnelles, afin de mieux capturer le contexte et l'ordre des mots.

Étapes clés du prétraitement :

- Nettoyage du texte (minuscules, suppression caractères spéciaux).
- Tokenisation et padding (vocabulaire limité à 20 000 mots, séquences de longueur fixe à 100 mots).

Architecture :

- Une couche d'embedding pour apprendre les représentations sémantiques (dimension 128).
- Deux couches LSTM unidirectionnelles pour mémoriser l'ordre et les dépendances temporelles des mots.
- Une couche Dense (64 neurones, activation ReLU) et du Dropout pour réduire le surapprentissage.
- Couche de sortie multi-label avec activation sigmoid (6 classes de toxicité).

Entraînement :

- Fonction de perte binary_crossentropy, optimiseur adam.
- EarlyStopping pour éviter le surapprentissage.

Ce modèle améliore la prise en compte du contexte par rapport à l'approche TF-IDF, mais reste limité car l'information contextuelle est capturée uniquement dans un sens.

Avantages du Modèle 2 par rapport au Modèle 1 :

- Capture du contexte grâce aux couches LSTM qui prennent en compte l'ordre des mots.
- Capacité à modéliser les dépendances contextuelles et non linéaires entre mots, améliorant ainsi la détection des nuances dans les phrases toxiques.

Limitations du Modèle 2 :

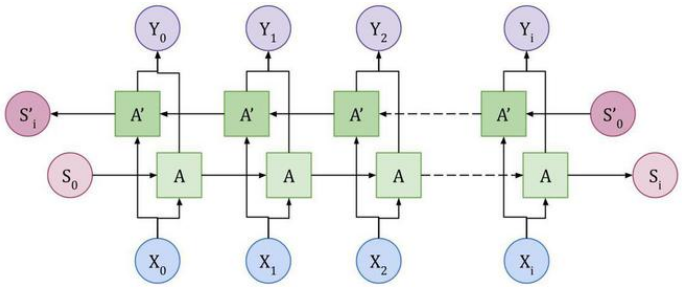
- L'utilisation d'un LSTM unidirectionnel limite la prise en compte du contexte global (informations situées après un mot non exploitées).
- Temps d'entraînement plus élevé et nécessité de ressources plus importantes (GPU), particulièrement avec de grands jeux de données.

Résultats du modèle 2 sur le jeu de validation :

- Amélioration du F1-score pour certaines classes par rapport au modèle 1 (par exemple, une meilleure capture de *toxic* et *obscene*), mais toujours des lacunes notables pour *severe_toxic*, *threat* et *identity_hate*.

	precision	recall	f1-score	support
toxic	0.84	0.76	0.80	3056
severe_toxic	0.52	0.04	0.08	321
obscene	0.85	0.75	0.80	1715
threat	0.00	0.00	0.00	74
insult	0.75	0.61	0.67	1614
identity_hate	0.00	0.00	0.00	294
micro avg	0.82	0.65	0.73	7074
macro avg	0.49	0.36	0.39	7074
weighted avg	0.76	0.65	0.69	7074
samples avg	0.07	0.06	0.06	7074

Partie 3 : Modèle 3 – Approche Optimisée avec LSTM Bidirectionnel et TextVectorization



Présentation du Modèle

Le Modèle 3 améliore significativement le prétraitement en utilisant la couche TextVectorization de Keras. Cette méthode intègre en une seule étape le nettoyage, la tokenisation et la limitation du vocabulaire, simplifiant ainsi le pipeline.

1.Prétraitement et Vectorisation

- **TextVectorization :** nettoyage, tokenisation et padding intégrés, simplifiant le processus.
- **Vocabulaire étendu à 200 000 mots :** permet une meilleure capture des nuances lexicales, améliorant ainsi la qualité du modèle.

2.Création du dataset avec Tensorflow

Le dataset est créé avec l'API **tf.data.Dataset** de TensorFlow, permettant une gestion efficace des données grâce à :

- **Caching** : données prétraitées stockées en mémoire.
- **Shuffling et Batching** : garantissent des lots de données variés pour réduire le surapprentissage.
- **Prefetching** : améliore l'efficacité de l'entraînement en anticipant les besoins du modèle.

Les données sont divisées en trois ensembles : entraînement, validation et test, afin d'évaluer efficacement les performances du modèle.

3. Architecture du Modèle

- **Embedding :** Transformation des mots en vecteurs denses pour capturer les similarités sémantiques.
- **LSTM Bidirectionnel** : lecture des séquences dans les deux sens pour saisir pleinement le contexte.
- Couches Denses (128, 256, puis 128 neurones) avec activation ReLU, améliorant l'extraction des caractéristiques et la classification multi-label.
- Couche de sortie multi-label (6) avec activation sigmoid pour prédire indépendamment chaque type de toxicité.
- **Évaluation détaillée** : précision, rappel, F1-score et accuracy.
- Sauvegarde au format H5 pour garantir la reproductibilité et faciliter des analyses ultérieures.

Exemple de résultats obtenus (via Keras et sklearn) :

```
Recall par label: [0.98043053 0.86792453 0.97472924 0.76470588 0.96835443 0.8705036 ]
F1-score par label: [0.98267408 0.84662577 0.97649186 0.80412371 0.97204574 0.89962825]
F1-score macro (moyenne des F1-scores): 0.9135982355653286
F1-score micro (calculé globalement): 0.9667812142038946
```

Partie 4 : Axes d'Amélioration et Limites

Limites techniques :

- Besoin élevé en ressources(GPU), notamment pour le LSTM bidirectionnel, entraînant des temps longs d'entraînement (3h30/epochs en local).
- Difficultés avec les données déséquilibrées (classes minoritaires comme « severe_toxic » ou « identity_hate » sont mal représentées et difficiles à détecter).

Axes d'amélioration proposés :

- Augmentation des données (paraphrase, synonymie, génération automatique) pour améliorer la robustesse du modèle.
- Utilisation de modèles Transformer (BERT, RoBERTa) pour une meilleure compréhension contextuelle, grâce à leurs bloque MHA.
- Optimisation de l'entraînement (calcul distribué ou GPU plus puissants) afin de raccourcir les temps d'entraînement et faciliter les expérimentations.

Conclusion

Ce projet présente trois modèles évolutifs pour classer automatiquement les commentaires toxiques :

1. **Modèle TF-IDF + Régression Logistique :**

    - Simple, rapide, et interprétable.
    - Limité par l'absence de prise en compte du contexte.

2. **Modèle LSTM Unidirectionnel :**

    - Meilleure gestion du contexte grâce au deep learning.
    - Capture partielle des nuances contextuelles.
    - Ressources computationnelles importantes nécessaires.

3. **Modèle final (LSTM Bidirectionnel + TextVectorization)** :

    - Capture complète du contexte (lecture bidirectionnelle).
    - Pipeline de prétraitement simplifié et optimisé.
    - Amélioration significative de la précision et du F1-score global.

Limites persistantes :

- Difficulté à gérer précisement les classes minoritaires.
- Ressources computationnelles élevées.

Perspectives d'amélioration :

Réel :

- Augmenter les données pour mieux gérer les catégories rares.
- Optimiser l'entraînement avec des ressources plus puissantes.

Hypothétiques :

- Intégrer des modèles avancés basés sur les Transformers (BERT, RoBERTa).
- Intégrer une solution MLGAN, similaire à un modele GAN, qui permettrais de générer des fausses données sur les classes les plus faible pour entrainer notre modele principal

En somme, ce travail a permis d'explorer différentes approches pour la classification de commentaires toxiques, mettant en évidence l'importance de capturer le contexte du langage naturel. L'évolution vers des architectures plus avancées comme les Transformers représente une direction prometteuse pour affiner encore davantage la précision des prédictions.