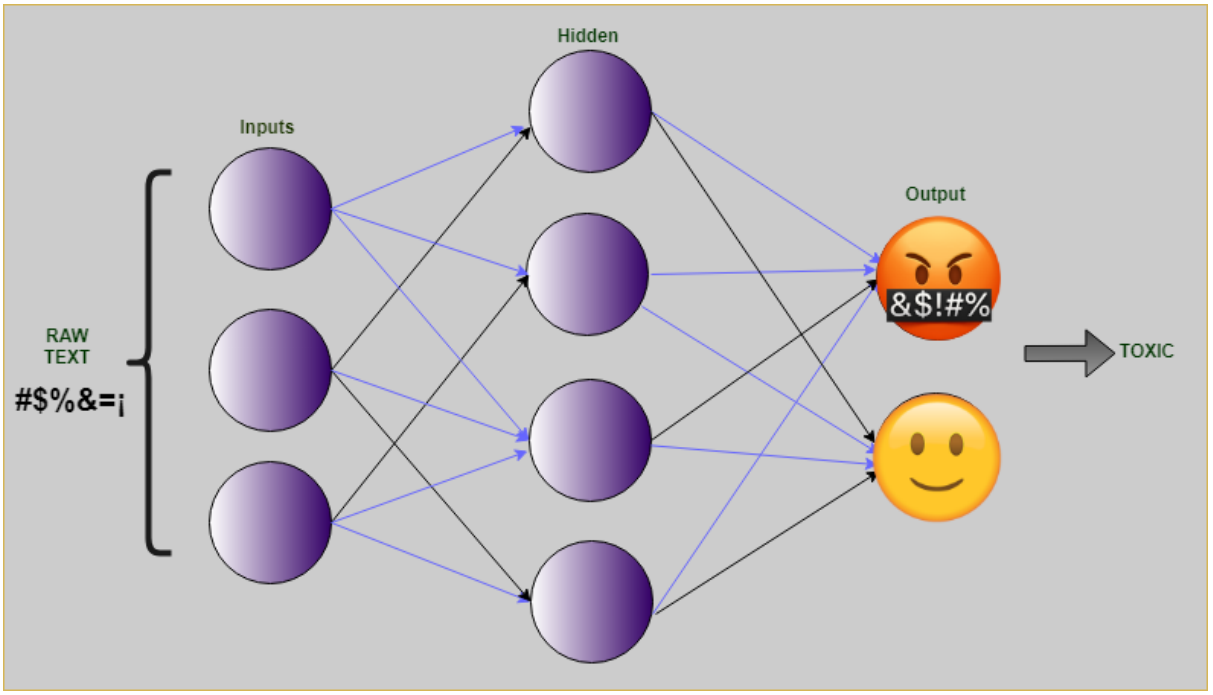
Giraud Pierre
Manouvriez Louis
Schneider jean

Junia-ISEN M1 2022-2023
Natural Language Processing
15/02/2023

Toxic Comment Classification



Repository du projet : https://github.com/Giraud-Pierre/Toxic_Comment_classification

I. Présentation du projet:

Discuter de choses qui nous passionnent sur Internet peut être compliqué. Les commentaires et réponses injurieuses pullulent et les plateformes de partage ont du mal à mettre en place des dispositifs de filtration des commentaires pour permettre aux utilisateurs de choisir le type de commentaires auxquels ils veulent avoir accès et de communiquer dans un bon environnement. Jigsaw et Google ont déjà développé un certain nombre de modèles avec Perspective API, dont certains pour détecter les commentaires toxiques, mais les modèles font encore des erreurs

Ce projet s'inscrit dans ce cadre. Il s'agit d'une reprise d'un concours kaggle dont le but était de développer un modèle prédictif permettant de prédire si un commentaire est toxique ou non. Le concours a mis à disposition un grand nombre de commentaires Wikipédia en anglais en les classant selon 6 classes mutuellement non-exclusive (un commentaire peut appartenir à 1, plusieurs ou aucune des 6 classes) : 'toxique', 'sévèrement toxique', 'obscène', 'menaçant, 'insultant et 'haine raciale'.

II. Analyse des données:

Les commentaires qui nous sont mis à disposition sont répartis en 2 parties :

- Des données d'entraînement qui vont servir à l'entraînement des modèles
- Des données de validation qui vont permettre de mesurer l'efficacité des modèles

A. Données d'entraînement :

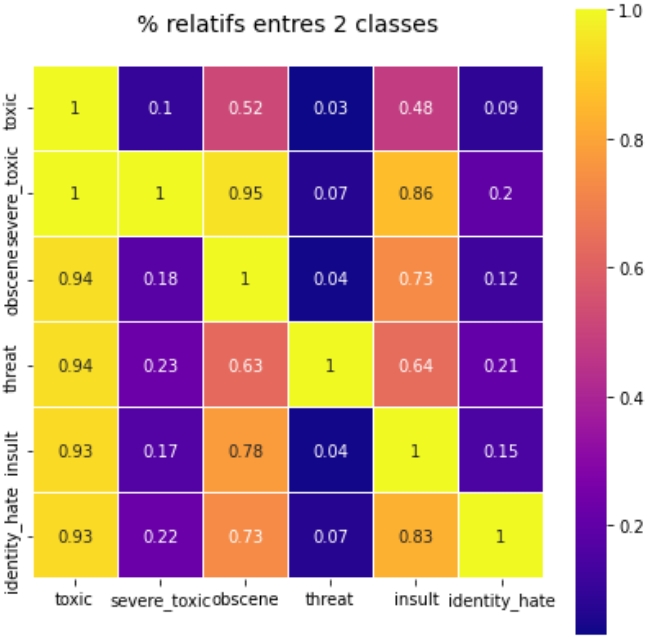
Les données d'entraînement sont comprises dans un seul fichier csv (de type tableur) qui regroupe à la fois les commentaires et leurs appartenances à chacune des 6 classes ou non (1 si le commentaire appartient à la classe, 0 sinon). Il contient 159 571 commentaires dont la majorité est non toxique et dont voilà la répartition des commentaires toxiques :

```
{'toxic': '9.58% des commentaires totaux',
 'severe_toxic': '1.0% des commentaires totaux',
 'obscene': '5.29% des commentaires totaux',
 'threat': '0.3% des commentaires totaux',
 'insult': '4.94% des commentaires totaux',
 'identity_hate': '0.88% des commentaires totaux'}
```

On peut remarquer que certaines classes sont vraiment peu représentées (sévèrement toxique, menaçant et haine raciale) , ce qui va potentiellement poser problème pour les modèles concernant ces classes.

En regardant les commentaires communs à 2 classes, on remarque qu'il y a des classes de commentaires qui sont souvent associées:

- Presque tous les commentaires appartenant à une autre classe de toxicité appartiennent aussi à la classe 'toxique'
- 95% des commentaires sévèrement toxiques sont obscènes et 85% sont des insultes
- Insulte et obscénité sont très souvent liées (72% des commentaires obscènes sont insultant et 73% des commentaires insultant sont obscènes
- La majorité des commentaires menaçant et de haine raciale sont aussi obscène et insultant



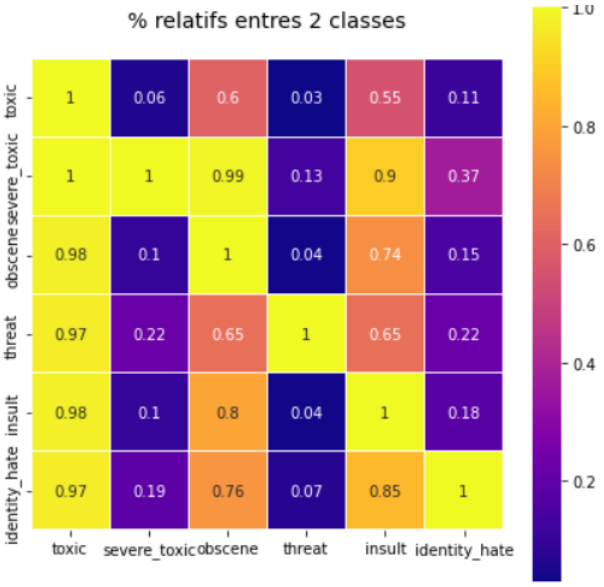
(pour l'ensemble de cette répartition, voir directement sur le code dans la partie 'Étude des jeux de données')

B. Données de validation :

Les données de validation sont séparées en 2 fichiers csv. Un fichier contient les commentaires et un autre contient les classes auxquelles ces commentaires appartiennent.

Certaines de ces données ont été exclues et toutes leurs classes ont été mises à -1. Il convient donc au préalable de les enlever des données avant de commencer à travailler dessus.

Une fois cela fait, les données de validation regroupent 63 978 commentaires et leur répartition est sensiblement la même que pour les données d'entraînement.




```
{'toxic': '9.52% des commentaires totaux',
 'severe_toxic': '0.57% des commentaires totaux',
 'obscene': '5.77% des commentaires totaux',
 'threat': '0.33% des commentaires totaux',
 'insult': '5.36% des commentaires totaux',
 'identity_hate': '1.11% des commentaires totaux'}
```

III. Comment utiliser notre projet :

- Copier le notebook 'Classification_commentaires_toxiques.ipynb' de la branche main dans votre drive et l'ouvrir avec google colab
- Créer un raccourci du dossier ci-dessous dans un dossier 'NLP_4A' dans votre drive:
  - https://drive.google.com/drive/folders/1W5GG4tJkbGfFiyt8-k9VFJXM99Zaoeuc?usp=sharing
- Exécuter la partie '***importation des packages***' qui va importer les bibliothèques nécessaires au projet
- Si vous voulez réaliser la partie '***études des jeux de données***' pour observer la répartition des commentaires dans les données, effectuez au préalable la partie '***importation des données***', sinon passez directement à la partie 'préparation des données'
- Dans la '***préparation des données***',
  - Les 2 premières étapes '***Définition de la fonction prepare_string()***' et '***nettoyages des commentaires***' peuvent être passées. Ces étapes prenant un certain temps, les données nettoyées ont été fournies dans le dossier fourni précédemment.
  - Il faut lancer la partie '***chargement des données une fois traitées***' et les parties suivantes pour rendre ces données utilisables pour les modèles suivants

- Dans la partie '***étude de différents modèles***', chaque modèle est exécutable indépendamment si vous souhaitez voir comment il fonctionne. Lors du lancement, le programme sauvegarde le meilleur modèle pour chaque type de modèle dans un dossier 'Model' sur votre drive.
    - **Note 1 :** Certains modèles prenant un certain temps à s'entraîner, des modèles préalablement entraînés sont fournis dans le dossier 'Model' du dossier 'Projet' fourni précédemment.
    - **Note 2 :** Les sauvegardes de modèle pèsent relativement lourd (plusieurs centaines de MO) et peuvent prendre une place conséquente sur votre google drive. **Pensez à supprimer les sauvegardes stockées sur votre drive quand vous avez fini d'utiliser le projet.**
- La partie '**pipeline'** sert à utiliser les modèles entraînés ci-dessus.
    - Il faut choisir le modèle que vous souhaitez utiliser
    - copier coller le contenu de la cellule 'création du modèle' du modèle approprié dans la case prévue à cet effet au début de la partie pipeline
    - Indiquez le chemin sur votre drive contenant la sauvegarde du modèle correspondant (typiquement soit dans '`/content/drive/MyDrive/Model/{nom du fichier}`' soit dans '`/content/drive/MyDrive/NLP_4A/Projet/Model/{nom du fichier}`' si vous avez suivi les étapes correctement jusqu'ici)
    - Utiliser la fonction '*Predict'* pour prédire l'appartenance aux classes d'un array ou d'une liste de commentaires.
    - Utiliser la fonction '*Predict_F1score'* si vous avez déjà labellisé vos commentaires pour calculer le F1_score de la prédiction faite par le modèle.

IV. Choix des modèles:

Nous avons créé plusieurs modèles pour essayer de résoudre ce problème en augmentant l'efficacité. Nous avons choisi comme mesure de l'efficacité de nos modèles le f1_score qui permet de maximiser à la fois la sensibilité et la spécificité (precision et recall) et est plus efficace que l'accuracy sur des classes à répartition inéquitable.

Nous avons tout d'abord essayé de résoudre le problème avec un arbre de décision de type random forest. Il s'agit d'une méthode de machine learning de type apprentissage supervisé.

Cependant, nous avons rencontré des problèmes de RAM sur Google Colab et nous avons dû diminuer le nombre de données. Cela a encore posé des problèmes, car certaines classes étant peu représentées, il a fallu choisir les données à conserver pour ne pas se retrouver avec des classes auxquelles n'appartiennent aucun commentaire du jeu de données.

Au final, même si les résultats semblent satisfaisants, ils ne sont pas très fiables à notre sens, car ils ne prennent pas en compte toutes les données. Nous sommes donc passés sur des réseaux de neurones récurrents et notamment sur du LSTM. Nous avons réalisé un modèle très simple avec une couche d'embedding et une couche de LSTM à 64 neurones cependant nous avons très vite été confrontés à de l'overfitting avec un f1_score sur les données de validation qui reste autour de 60%.

Cela peut être dû au nombre de paramètres très important lié à la matrice d'embedding. Nous avons donc pris le parti de diminuer ce nombre de paramètres en limitant la taille du bag of word aux 20 000 mots les plus utilisés dans les commentaires et en implémentant une matrice de poids, pour mieux représenter les classes peu représentées. Cependant, le problème d'overfitting n'a pas été résolu.

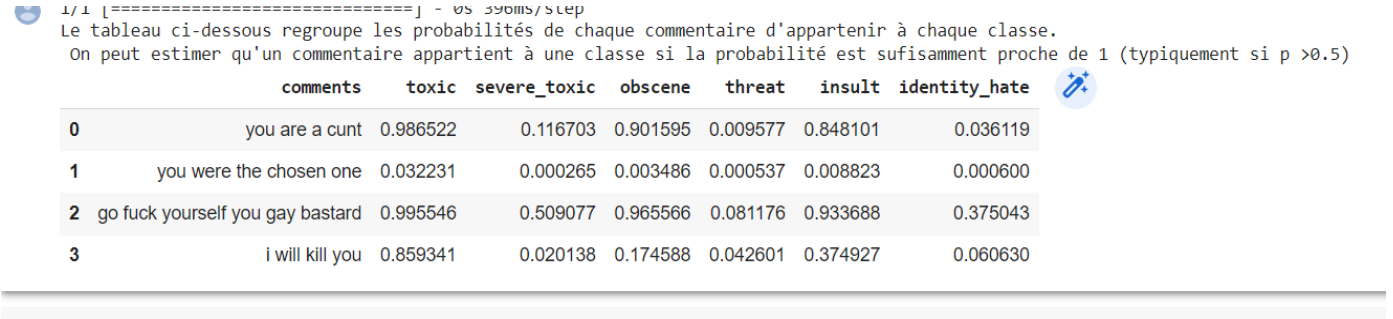
Nous avons donc essayé de réaliser nous-même une matrice d'embedding de type GloVe pour sauter la couche d'embedding et diminuer l'overfitting, mais cela n'a pas amélioré sensiblement le modèle.

Pour tenter encore et toujours de réduire cet overfitting, nous avons essayé d'introduire des dropouts dans le LSTM, ce qui a normalement pour effet de diminuer l'overfitting. Cependant, même avec des dropouts de 50% sur les sorties et les récurrences du LSTM, cela n'a pas sensiblement améliorer le résultat.

Le dernier modèle que nous avons utilisé utilise un LSTM bidirectionnel, mais cela n'a pas non plus sensiblement amélioré le problème.

V. Pistes d'amélioration

Malgré nos différents modèles, nous n'avons pas réussi à dépasser 64% de f1_score, même s'ils restent au final relativement performants pour catégoriser des commentaires.


1/1 [==============================] - 0s 396ms/step
Le tableau ci-dessous regroupe les probabilités de chaque commentaire d'appartenir à chaque classe.
On peut estimer qu'un commentaire appartient à une classe si la probabilité est sufisamment proche de 1 (typiquement si p >0.5)

	comments	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	you are a cunt	0.986522	0.116703	0.901595	0.009577	0.848101	0.036119
1	you were the chosen one	0.032231	0.000265	0.003486	0.000537	0.008823	0.000600
2	go fuck yourself you gay bastard	0.995546	0.509077	0.965566	0.081176	0.933688	0.375043
3	i will kill you	0.859341	0.020138	0.174588	0.042601	0.374927	0.060630

Voici une prédiction réalisée avec quelques commentaires trouvés sur le net. Nous pouvons remarquer qu'il a un peu de mal à distinguer les commentaires menaçants et il arrive presque à reconnaître le 3e commentaire comme 'identity hate'. Cela peut s'expliquer par la faible proportion de ces 2 types de commentaires dans les données fournies. Cependant, le système semble plutôt performant.

Si nous avions eu plus de temps, nous aurions pu essayer de modifier différents paramètres comme par exemple diminuer le learning rate et le batch size. Nous aurions pu aussi introduire différentes couches cachées comme des couches denses ou une couche maxpool pour réduire les dimensions en sorties du LSTM.

Enfin, nous aurions aussi pu tenter d'utiliser une bag of word limité à 20 000 mots avec la matrice GloVe, mais cela aurait demandé de recoder une partie des fonction de tokenize (notamment fit_on_text et text_to_sequences) pour obtenir des résultats intermédiaires pour réaliser la matrice GloVe et nous n'avons pas vraiment eu le temps de l'implémenter.