PROJET NLP : Classification de commentaires toxiques

Par Sacha Binanzer et Yanis Boukebir

Contexte :

Afin de valider notre module de NLP lors de notre parcours d'ingénieur ISEN domaine IA, nous avons dû développer un projet dans lequel nous allons déployer un modèle d'IA afin de classifier des commentaires selon lors toxicité. Cet énoncé vient d'un concours kaggle et toutes les données seront puisées depuis cette source : Kaggle challenge toxic comment.

Notre projet se trouve ici :Github-NLP_project

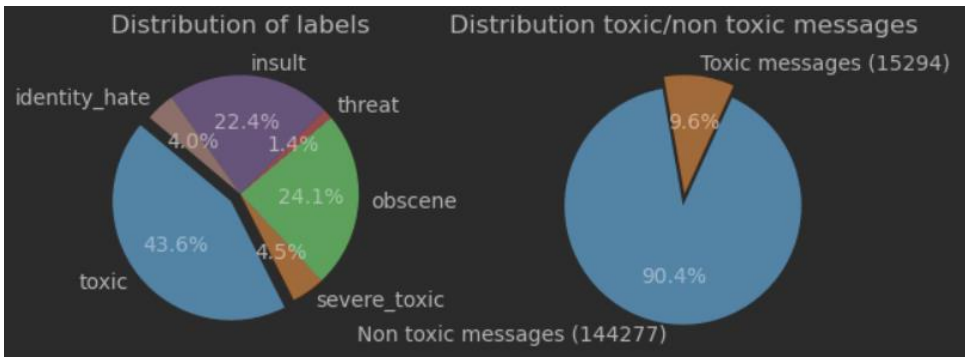
Approche globale du projet :

Le projet consiste donc en la création d'un Notebook dans lequel nous allons traiter les données d'entrée afin de les visualiser et de les préparer à pour un apprentissage de notre modèle, nous allons construire une architecture basée sur les RNN afin de traiter le texte de manière efficace puis construire des fonctions qui prennent un commentaire et nous ressort sa classification.

Nous utilisons Python 3.10 et les librairies suivantes : Pandas, numpy, matplotlib, tensorflow, nltk et Scikit-learn en plus de quelques librairies de traitement de chaine de caractères (voir requirements.txt sur github).

Data preprocessing :

Nous possédons donc un fichier csv d'environ 150 000 phrases catégorisées sous 6 labels, 0 si la phrase n'est pas catégorisée comme faisant partie du label, sinon 1 : toxic, severe_toxic, obscene, threat, insult, identity_hate.



Le but du data preprocessing est de "nettoyer" les données afin de rendre exploitable les données. C'est une étape nécessaire car les données peuvent contenir des caractères spéciaux et d'autres problèmes qui peuvent affecter la qualité de l'analyse et réduire les performances de notre modèle. Le but est de garder l'information importante dans une phrase qui permet de savoir si un message est problématique ou non.

Voici donc l'utilité de chaque fonction :

La fonction remove_special_chars supprime les caractères spéciaux tels que les espaces, les sauts à la ligne (entrée) etc...

La fonction to_lowercase convertit tous les caractères du texte en minuscules.

La fonction remove_punctuation supprime la ponctuation du texte.

La fonction replace_numbers supprime les nombres dans le texte.

La fonction remove_whitespaces supprime les espaces inutiles au début et à la fin du texte.

La fonction remove_stopwords utilise la liste de stopwords prédéfinie et enlève les mots inutiles tels que "the", "a", "an", "and" etc...

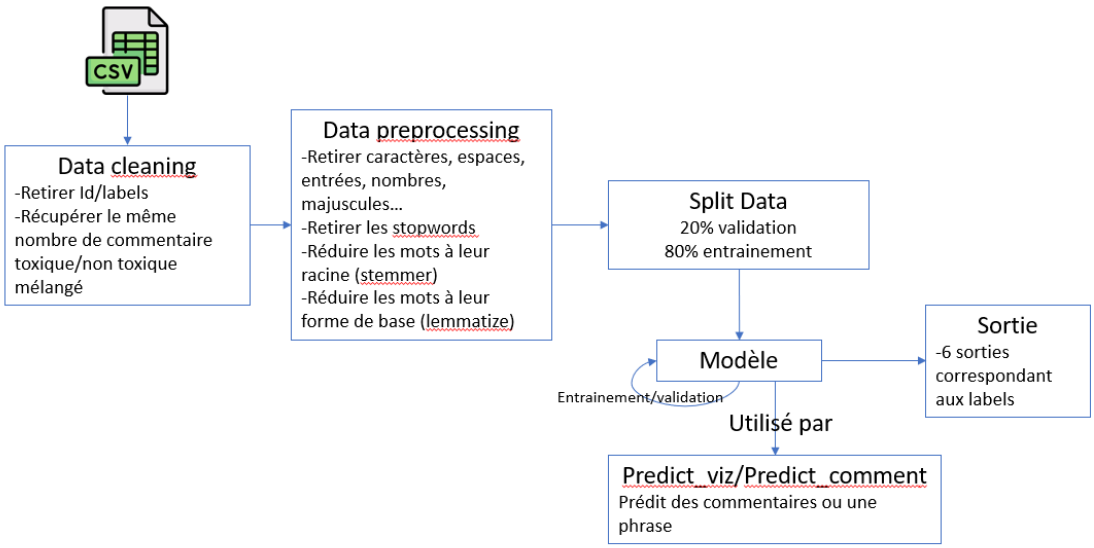
Les fonctions lemmatize_words et lemmatize_verbs réduisent les mots à leur forme de base.

Explication du modèle :

Le modèle commence par une couche d'embedding qui convertit les mots du texte en vecteurs de dimension 20. Ensuite, la couche Bidirectional GRU (Gated Recurrent Unit) permet d'analyser les séquences de mots dans les deux directions, en utilisant les RNN, nous avons choisi un GRU au lieu d'un LSTM afin de minimiser le nombre de paramètres et accélérer l'entrainement. Cette couche est suivie par une couche Flatten (réduire l'entrée à un vecteur d'une dimension). Enfin, une couche Dense d'output est utilisée pour produire les six scores de classification.

Notre modèle semble être performant est un bon compromis entre vitesse, taille et complexité. Nous avons exécuté plusieurs fois notre Notebook afin de sauvegarder un modèle qui nous parait suffisamment performant afin d'être certain de pouvoir le réutiliser.

Pipeline :



Conclusion :

Pour conclure, ce projet récupère des fonctionnalités vues dans nos Notebook en cours et trouvées sur d'autre Notebook participant au défi kaggle ce qui nous a permis de le construire à notre façon et en peu de temps. Néanmoins, il reste des axes d'amélioration notamment sur le traitement des commentaires en utilisant des fonctions afin de remplacer les abréviations. De plus, le faible nombre de cas pour certaines catégories nous empêche d'avoir des résultats optimaux. Avec plus de temps nous aurions aimé corriger certain biais au niveau des data et pourquoi pas faire un peu plus de visualisations sur notre modèle.