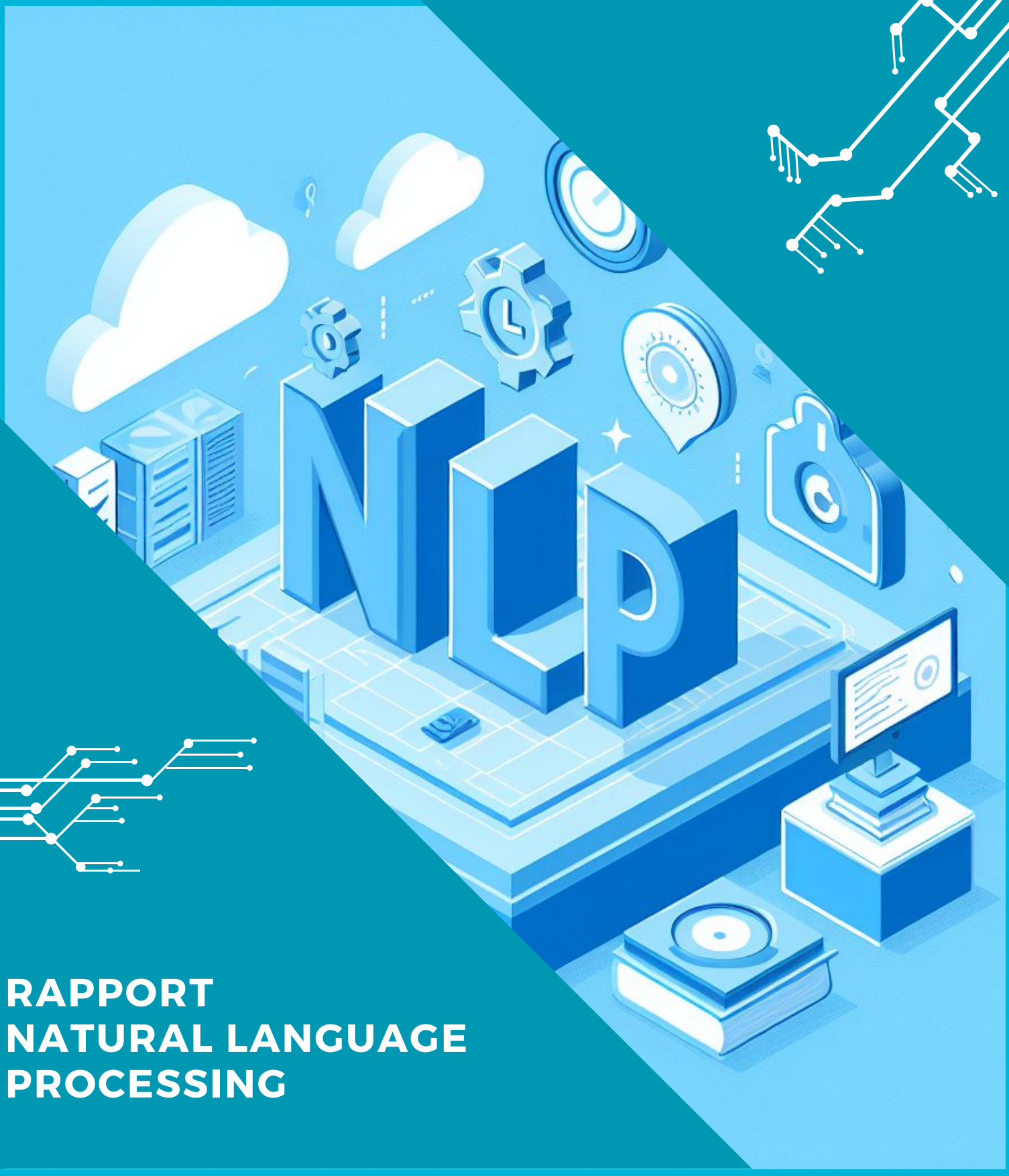

RAPPORT
NATURAL LANGUAGE PROCESSING

PRÉSENTÉ PAR:

Adam Mohemmane et Aymane Lamqari

SOMMAIRE

INTRODUCTION & CONTEXTE

Le traitement du langage naturel, est un domaine qui se concentre sur l'interaction entre les ordinateurs et le langage humain. Il s'agit de donner aux machines la capacité de lire, comprendre et interpréter le langage humain pour effectuer des tâches.

Ce projet, en rapport avec le traitement du langage naturel, a pour objectif d'explorer le monde des commentaires sur internet.

Notre rôle est mettre au point un système qui sait lire et organiser les commentaires tout seul.

On débute par des choses simples pour saisir comment ça fonctionne. Plus tard, on utilisera des techniques plus poussées pour améliorer notre système grâce au Deep Learning.

Dans les pages qui suivent, on vous expliquera comment on a construit notre outil, ce qu'on a découvert et les idées qu'on a pour le rendre encore plus performant.

MÉTHODOLOGIE & RÉSULTAT

• *PARTIE 1 : PRÉPARATION DES DONNÉES*

Pour traiter et analyser des données textuelles, comme par exemple le cas des tweets, on utilisé une combinaison de bibliothèques Python qui sont efficaces. L'utilisation de ces outils nous a permis de nettoyer, d'analyser et de modéliser efficacement les données textuelles à notre disposition.

Ces bibliothèques incluent **NumPy** et **Pandas** pour la manipulation des données, **Matplotlib** et **TensorFlow** pour la visualisation et le modelage des données, ainsi que **nltk** (Natural Language Toolkit) pour le traitement du langage naturel.

Ce dernier a été employé pour télécharger et utiliser des ressources linguistiques comme **punkt** pour la tokenisation des phrases et **stopwords** pour filtrer les mots peu informatifs dans l'analyse des tweets. Tout cela rentre dans la section importation des données.

Pour charger des données, on a mis en place deux méthode : soit en utilisant Google Colab après avoir ajouté le dossier partagé à votre Google Drive via le lien fourni. Soit en important le CSV avec **pandas,** en remplaçant **chemin/du/fichier.csv** par le chemin réel du fichier.

On a comme résultat un tableau de commentaires classés selon leur nature, avec des identifiants et des marques qui montrent qu'ils ne contiennent pas de langage toxique. Chaque catégorie, comme la toxicité générale, sévère, l'obscénité, les menaces, les insultes et la discrimination, est cochée avec des zéros, indiquant que ces commentaires sont considérés comme non offensants. Les codes qui suivent sont mis en place pour uniquement différents affichages de la data.

En utilisant Seaborn pour visualiser les données, on a fait un diagramme en barres. On voit que la catégorie "**toxique**" a le plus de commentaires et "**haine identitaire**" le moins.

Pour la distribution de la longueur des commentaires, un histogramme a été tracé, révélant que la majorité des commentaires sont courts, avec une fréquence qui diminue rapidement au fur et à mesure que la longueur augmente.

On a utilisé une carte de chaleur qui montre quelles catégories vont souvent ensemble. Par exemple, "**toxique**" et "**insulte**" apparaissent souvent en même temps, tandis que d'autres ne sont pas autant liées, ce qui montre que les commentaires méchants ne se ressemblent pas tous.

MÉTHODOLOGIE & RÉSULTAT

Pour explorer les données, on a d'abord affiché un échantillon de commentaires. Ensuite, on a cherché des motifs spécifiques, comme les URL, et on a trouvé qu'il y en a 4778 dans le jeu de données.

Par la suite, on a créé un histogramme qui montre la répartition de leur longueur mais cette fois pas de fréquence par rapport la longueur mais cette fois ci le compteur par rapport à la longueur.

Et enfin, nous avons répertorié les mots les plus courants pour découvrir ceux qui reviennent le plus souvent dans les commentaires.

On est passé à la préparation des données après avoir fait le côté visualisation pour analyser les résultats obtenus.

Dans la préparation des données, le processus a commencé par nettoyer les commentaires pour enlever les URLs, les caractères spéciaux et les chiffres, en plus de convertir les abréviations.

Après cela, on a appliqué une tokenisation et éliminé les stopwords pour ne garder que le contenu pertinent. Une nouvelle colonne a été ajoutée pour marquer si un commentaire est toxique selon l'un des critères.

Pour finir, on a divisé les données en ensembles de formation et de test avec la méthode de stratification pour s'assurer que chaque ensemble avait une répartition équilibrée des catégories de commentaires.

Cela a permis de préparer un modèle pour l'entraînement qui pourrait comprendre et classifier de manière équilibrée les différents types de toxicité des commentaires.

MÉTHODOLOGIE

• *PARTIE 2 : MODÉLISATION*

La Régression Logistique est souvent utilisée comme point de départ dans les problèmes de classification, notamment parce qu'elle est simple et rapide. Elle fonctionne bien quand il existe une séparation linéaire entre les classes.

La Forêt Aléatoire (Random Forest) est une méthode d'apprentissage en ensemble qui utilise la combinaison de plusieurs arbres de décision pour améliorer la robustesse et la précision du modèle. Elle est efficace pour capter les non-linéarités dans les données.

XGBoost est un algorithme d'optimisation basé sur des arbres de décision qui utilise le gradient boosting pour améliorer la performance et la vitesse. Il est reconnu pour sa performance sur un large éventail de problèmes de classification.

CatBoost est une autre forme de boosting d'arbres qui gère automatiquement les variables catégorielles et réduit le surajustement, ce qui est utile pour les données avec beaucoup de features catégorielles.

Pour les réseaux de neurones utilisant **TensorFlow/Keras**, ils permettent de capturer des relations complexes et des hiérarchies de features, avec des architectures pouvant inclure des couches d'embedding, des couches convolutionnelles pour les séquences (CNN pour texte), ou des architectures récurrentes (comme LSTM ou GRU) pour gérer les dépendances temporelles.

L'analyse des résultats doit considérer à la fois la précision, le rappel et le score F1 pour chaque catégorie de toxicité. Un modèle peut avoir une haute précision mais un faible rappel, indiquant qu'il est prudent dans ses prédictions mais en manque beaucoup, ou inversement. Le score F1 équilibre ces deux mesures.

En conclusion, le choix du modèle dépend de la spécificité du problème et de l'équilibre souhaité entre performance, complexité, et puissance de calcul. Les modèles plus simples comme la régression logistique et RandomForestClassifier peuvent servir de bonnes baselines, tandis que XGBoost et CatBoost peuvent apporter une performance accrue, souvent au prix d'un temps de calcul plus long.

RÉSULTAT & CONCLUSION

Analysons et commentons les résultats qu'on a obtenus avec différents modèles, :

XGBoost :
- Pour la catégorie 'toxic', nous avons une précision élevée et un score F1 raisonnable, ce qui indique une bonne performance globale.
- Pour 'severe_toxic', la précision chute et le rappel est très faible, signifiant que le modèle ne capture pas bien cette catégorie, probablement en raison de moins d'exemples dans les données d'entraînement.

CatBoost :
- Les résultats sont similaires à ceux de XGBoost, avec de bonnes performances sur 'toxic' mais pas aussi bien sur 'severe_toxic'.
- La performance globale est légèrement meilleure que celle de XGBoost, ce qui peut indiquer une meilleure gestion des caractéristiques catégorielles et des features complexes.

Dans l'ensemble, nos modèles semblent mieux performer pour la classification de commentaires 'toxic' que 'severe_toxic'. Cela peut être dû à un déséquilibre des classes, où il y a probablement beaucoup plus d'exemples de commentaires 'toxic' que 'severe_toxic'. Pour améliorer cela, on doit envisager des techniques de rééquilibrage des classes, comme le suréchantillonnage de la classe minoritaire ou l'utilisation de poids de classe lors de la formation.

Aussi, il est important de regarder au-delà des métriques globales. Un modèle peut avoir une excellente précision globale, mais notre objectif est de minimiser les faux négatifs pour certaines catégories, on doit privilégier le rappel ou le score F1 pour ces catégories.

En vue de renforcer l'analyse des commentaires, il est essentiel d'affiner les architectures des CNN pour une extraction plus précise des caractéristiques textuelles, d'améliorer la qualité et la pertinence des embeddings afin de mieux saisir les subtilités linguistiques et contextuelles, et de perfectionner les modèles LSTM et GRU au sein des RNN pour améliorer la gestion des séquences de texte longues et complexes, en augmentant leur capacité à comprendre les nuances et les intentions derrière les mots, tout en minimisant les problèmes de disparition ou d'explosion des gradients.