Rapport TD1 : Traitement Automatique de langue Naturelle

Parti 1 :

A-priori sur les caractéristiques et les modèles

Caractéristiques (features) :
Nous avons utilisé le nom des vidéos comme principale caractéristique pour prédire si une vidéo est comique ou non.
Notre hypothèse était que certains mots ou phrases dans le titre d'une vidéo pourraient indiquer sa nature comique.
Nous avons utilisé **TfidfVectorizer** pour convertir les noms des vidéos en vecteurs numériques. Ce vectoriseur prend en compte la fréquence des mots tout en réduisant l'importance des mots qui apparaissent fréquemment dans l'ensemble du corpus (comme les mots courants qui n'apportent pas beaucoup d'information).

Modèles :
Nous avons envisagé d'utiliser plusieurs modèles, notamment **RandomForestClassifier, LogisticRegression, et MultinomialNB.**
Notre a priori était que les forêts aléatoires pourraient bien fonctionner car elles peuvent capturer des relations non linéaires et des interactions entre les caractéristiques.
Apports individuels de chaque variation

Tokenization et Stemming :
La tokenisation a aidé à décomposer les titres des vidéos en mots individuels pour une meilleure analyse.
Le stemming, qui convertit les mots à leur racine, a permis de réduire la dimensionnalité et de regrouper les mots similaires, renforçant ainsi la robustesse du modèle.
Modèles :
RandomForestClassifier : Ce modèle a offert une capacité de généralisation élevée et a pu capturer des relations complexes entre les caractéristiques.
LogisticRegression : Utile pour comprendre l'importance de chaque mot ou groupe de mots. Cependant, il pourrait ne pas capturer des relations complexes.
MultinomialNB : Un modèle simple basé sur la probabilité, il est rapide à entraîner mais pourrait ne pas être le plus précis pour notre cas.
Conclusion

À l'heure actuelle, notre pipeline de traitement des données transforme efficacement les titres des vidéos en caractéristiques numériques grâce à la tokenisation, au stemming et au **TfidfVectorizer**.

En ce qui concerne le modèle, bien que nous ayons envisagé plusieurs modèles, des tests supplémentaires (cross vsalidation) sont nécessaires pour déterminer lequel offre les meilleures performances pour notre problème spécifique.

Nous recommandons également d'explorer d'autres caractéristiques, comme la description de la vidéo, les métadonnées, etc., pour améliorer davantage les performances du modèle. De plus, l'utilisation de techniques plus avancées telles que le word embedding ou les modèles basés sur des réseaux de neurones pourrait également être explorée pour améliorer les performances.

Parti 2 :

Méthode d'Extraction de Caractéristiques:

La méthode utilisée consiste en une itération sur chaque ligne du DataFrame. Pour chaque ligne, les tokens sont extraits et associés à des libellés correspondants. Ensuite, pour chaque token, un ensemble de caractéristiques est extrait en se basant sur des propriétés linguistiques clés.

Caractéristiques Extraites:

- **is_capitalized**: Indique si la première lettre du token est en majuscule, fournissant une information sur la capitalisation du token.
- **is_previous_capitalized**: Indique si la première lettre du token précédent est en majuscule, permettant de capturer des relations de casse entre tokens consécutifs.
- **is_next_capitalized**: Indique si la première lettre du token suivant est en majuscule, offrant une perspective sur la capitalisation du contexte immédiat.
- **length**: La longueur du token en termes de nombre de caractères, apportant une dimension sur la complexité morphologique des tokens.
- **position**: La position du token dans la séquence, fournissant des informations sur la structure séquentielle des tokens.

Stockage des Résultats:

Les résultats de l'extraction sont stockés dans trois listes distinctes :
features_list : Contient les caractéristiques extraites pour chaque token.
labels_list : Contient les étiquettes associées à chaque token (indiquant s'il s'agit d'un nom ou non).
tokens_list : Stocke les tokens eux-mêmes pour référence ultérieure.

Resultat obtenu

Le premier modèle utilisé est le GrandientBoostingClassifier. Ce modèle donne une accuracy de 95.8%. Le soucis c'est qu'il prédit quasiment que des 0 pour chaque mot ce qui explique le résultat.

Le second modèle est le randomForestClassifier et les criterions:

- 'gini' : il arrive à mieux prédire les noms mais il reste certain manque. L'accuracy est de 95.29
- 'entropy' : Même constat que pour le précédent. L'accuracy est de 95.34
- 'log_loss' : On obtient une accuracy de 95%

AdaBoostClassifier : donne des résultats similaires au randomForestClassifier, juste l'accuracy obtenu est plus élevé

Parti 3 :

Pour cette partie nous avons effectué la dernière tâche de « find_comic_name »,
Pour cela nous n'avons pas fait d'entrainement et mais assemblé les deux modèles.
On identifié le nom de la vidéo qui était prédit comic, dans la colonne
« is_comic_video_prediction » , alors on met les tokens qui ont un valeur égale à 1 venant de la prédiction « is_name_prediction »
A contrario, si la prediction dit que la vidéo n'est pas comique alors on met la valeur « None »

Enfin la prédiction obtenu à la fin est de 95%
Ce résultat parait très important si on considère que la tache is_name a du mal à prédire correctement.