KAFANDO Bila Ausni Martial
BERNARD Octave

Rapport de projet de Natural Language Processing

Dans le cadre du cours de NLP (Natural Language Processing), nous avons comme projet une classification de commentaires toxiques à partir d'un jeu de données.

I/ Etude du jeu de données

Nous avons donc d'abord analysé le jeu de données afin de trouver la méthode la plus efficace pour résoudre ce problème. Nous avons remarqué que les commentaires toxiques sont classés en 6 catégories (toxiques, très toxiques, obscènes, menace, insulte, haine identitaire). Cela nous indique donc qu'il s'agit d'un problème de classification à plusieurs labels. Afin de préparer les données, nous avons ensuite regardé s'il y avait des valeurs numériques manquantes. Comme les valeurs moyennes sont très faibles (de loin inférieures à 0,05), il y a beaucoup de commentaires qui ne sont pas labellisés comme positifs dans les six catégories. On vérifie donc le nombre de commentaires qui ne sont dans aucune catégorie. On remarque aussi que toutes les lignes de données dans les fichiers 'test' et 'training' contiennent des commentaires il n'y a donc pas besoin de nettoyer les champs 'null'. On teste ensuite la longueur des commentaires et on s'aperçoit que la majorité des commentaires sont compris entre 0 et 500 mots même si certains peuvent atteindre jusqu'à 5 000 mots. Enfin, on examine les corrélations entre les différentes catégories ciblées et on constate que certaines catégories sont très liées ('obscène' et 'insulte' par exemple).

II/ Préparation des données

Suite à l'étude des données, on modifie les données afin de les rendre facilement utilisables pour l'entraînement du modèle. On nettoie d'abord les commentaires en retirant les caractères inutiles à notre classification. On définit ensuite X dans les données de test et d'entraînement dans le but de 'tokenizer' les données et ensuite on les vectorise pour faciliter leurs utilisations.

III/ Entraînement du modèle

Comme dit précédemment, il s'agit d'un problème de classification à plusieurs labels. Les méthodes les plus communes pour ce genre de problème sont :

- La "Binary Relevance" : cette méthode est la plus simple et consiste à considérer chaque label comme un cas de classification seul. On présume donc ici qu'il n'y a pas de corrélation entre les différents labels.

- Les "Classifiers Chains" : Dans cette méthode, le premier classifier est entraîné sur l'entrée X. Puis les classifiers suivants sont entraînés sur l'entrée X et les prédictions de tous les classifiers précédents dans la chaîne. Cette méthode tente de tirer les signaux de la corrélation entre les variables cibles précédentes.

- Le "Label Powerset" : Cette méthode transforme le problème en un problème multi-classes où les labels multi-classes sont essentiellement toutes les combinaisons uniques de labels. Dans notre cas, où il y a six labels, le Label Powerset transformerait en fait le problème en 2^6 ou 64 classes.

IV/ Itération du modèle

Premier Model : Random Forest Classifier

Lorsque l'on utilise toutes les données, le GPU de l'ordinateur plante.
En raison de cela nous avons décidé d'entraîner le model sur 10000 donnée on a donc trouver un très bon f1_score.
Pour le la préparation des donnée on à opter pour un TF-IDF car étant donné que la majorité de tweet est positive si un mots apparaît trop de fois il y'a de force chance pour qu'il soit neutre ou positif et le TF IDF en lui donnant un faible poid permet de mettre cela en évidence.

Deuxième Model: RNN & LSTM

Le RNN est le meilleur modèle qu'on a pu obtenir pour parvenir à un tel modèle on a utilisé un LSTM de 128 unité le tout entrainer sur 10 epoch car lorsqu'on mets plus de couche de neurone le modèle fait un overfit moins il apprend pas bien.
En sortie on à 6 unités correspondant au 6 sortie et une fonction d'activation softmax.
Troisième Model:Embedding + glove Nous avons pas vraiment eu le temps d'approfondir ce modèle mais les résultats obtenus sont satisfaisants. Pour entraîner le modèle nous avons utilisé une couche de 64 neurones le tout entrainer sur 10 epoch.

Les modèles de deep learning ont tous été enregistrés afin que le travail puisse être utilisé dans une fonction pour tester les différents tweets mais aussi pour que d'autres personnes puissent utiliser notre travail comme base pour le leur.

Les graphes:
Nous avons fait deux graphes, un pour visualiser la perte et l'autre pour visualiser l'accuracy de chaque modèle.

Pour la fonction utiliser pour tester les tweets nous avons choisie le RNN parce que il est le meilleur parmi les modèles qu'on a pu tester.

V/ Résultats du modèle et amélioration possible

Bien que nous ayons obtenu un bon résultat au niveau de nos tests nous n'avons pas pu obtenir le f1 score pour le RNN et l'embedding car il y avait un problème entre les données de tester sur une dimension de 25 et la prédiction sur une dimension de 975.

Si nous avions eu plus de temps, on aurait amélioré notre modèle en utilisant le f1-score comme critère de validation afin de choisir un

meilleur modèle pour tester le projet. on aurait aussi testé le bag of words afin de voir s'il pouvait donner de meilleur résultat.
On aussi fait une fonction test tweet qui utilise tous les modèles afin d'avoir un meilleur résultat par exemple pour qu'un tweet soit prédit comme étant toxique il faudra qu'il le soit par plus de la moitié des modèles.