Doron Israel Elliot muller Nicolas Wagner
doron.israel@dauphine.eu elliot.muller@dauphine.eu nicolas.wagner@dauphine.eu

Rapport projet final : Toxic Comment Classification

Intro :

Pour ce projet, on dispose d'un grand nombre de commentaires Wikipédia labellisés selon 6 types de toxicité par des évaluateurs humains.
Le but de ce challenge est de créer un modèle qui prédit une probabilité de chaque type de toxicité pour chaque commentaire.
La particularité est qu'il s'agit d'une tâche de classification multi-label : en effet, un même commentaire peut être classé dans plusieurs catégories. Nous avons envisagé 2 méthodes pour réaliser cette tâche.

Preprocessing :

Nous nous sommes inspirés d'un TP fait en classe qui utilise la librairie re permettant de travailler avec expressions rationnelles. Nous avons créé une fonction appelée "clean_text" qui enchaîne les opérations usuelles de pré-processing de texte. La fonction convertit ainsi les commentaires en minuscules (lowercase), enlève les éventuels liens qui sont laissés par les commentateurs, enlève la ponctuation, les nombres etc...
Pour la tokenisation nous avons ensuite utilisé le Tokenizer de Pytorch (qui utilise spacy) pour construire notre vocabulaire. Nous avons volontairement décidé de ne pas procéder aux tâches de lemmatisation et au retrait des "stops words" qui dans certains cas peuvent se révéler utiles mais dans le cas d'une tâche de classification avec beaucoup de données présentent peu d'intérêt et peuvent même affecter la performance du modèle. (Beaucoup d'articles sur le machine Learning sur internet faisait mention de cela même s'il aurait été mieux de le tester mais nous ne l'avons pas fait par souci de temps).

Modèle :

Le classifieur a la structure suivante :

- Une couche de Word Embedding, suivi d'un BI-LSTM,concaténation des deux états cachés finaux,suivi d'un Dropout, et d'une couche linéaire.
-

La couche de Dropout est là pour prévenir de l'overfitting en abandonnant certaines couches du réseau lors de l'entraînement du modèle.

Nous avions pensé au début puisque nous avons un problème de classification multi label qu'après avoir concaténer les deux états finaux du BI-LSTM de les envoyer chacun dans 6 couches linéaires différentes afin de prédire chacun des labels. Nous avons vite abandonné cette idée face au risque d'overfitting qu'elle présentait de plus les labels n'étant pas indépendantes il n'est pas absurde de considérer un seul réseau linéaire pour traiter

l'ensemble des labels (mais là encore idée intéressante à tester si nous disposions de plus de temps).
L'embedding dimension est de 100 tout comme la dimension des états cachés du BI-LSTM qui est aussi de 100. Le dropout rate est à 0.5

Entraînements du modèle et résultats:

La loss utilisée pour entraîner notre modèle est la "binary cross entropy loss" avec l'algorithme d'optimisation ADAM avec un pas de 1e-3.
Le score afin de sélectionner les paramètres du modèle est la moyenne du roc auc obtenu pour chaque label.

Dans un premier temps, le classifieur a été entraîné et évalué sur 5 epochs avec une taille de batch de 50. Les paramètres sélectionnées furent ceux qui donnèrent le plus grand score pour ce modèle.

Remarquant qu'après les 5 epochs le score du modèle était toujours en augmentation, nous avons conservé le premier modèle et créé un second modèle sur les bases du premier mais nous l'avons entraîné pour 2 épochs supplémentaires.

Le score n'a pas plus évolué pendant ces deux épochs, nous avons donc décidé d'arrêter l'entraînement. Nous nous retrouvons donc avec deux modèles qui ont eu respectivement un score de 0.981 et 0.982 sur le notre ensemble de validation.

Au final sur le test set nous obtenons les scores suivants :

	Public Leaderboard	Private Leaderboard
BI-LSTM 1 (paramètres =meilleur des 5 epochs)	0.96726	0.96723
BI-LSTM 2 (paramètres =meilleur des 7 epochs)	0.97108	0.96831

Conclusion:

Suite à l'avancée du rendu du projet d'une semaine, nous nous sommes concentrés à construire un modèle simple et fonctionnel. Au cours de la réalisation de celui-ci nous avons pensé à plusieurs idées permettant de l'améliorer comme ajouter des couches convolutionnelles entre le couche d'embedding et la couche du BI-LSTM, créer une couche d'attention, ou bien encore jouer avec le preprocessing, … En espérant malgré tout vous avoir montrer à l'aide de ce projet de notre bonne compréhension des enjeux du traitement naturel du langage.

Doron Israel, Eliot muller, Nicolas Wagner Master Mash