



Parcours Ingénieur Machine Learning

Projet 8 : Participez à une compétition Kaggle !

Introduction



Objectif : comme son nom l'indique, le but de ce projet est de participer à une compétition Kaggle. Pour ma part, je suis parti sur une problématique de NLP consistant à classer des commentaires en fonction de leur toxicité. Il s'agit d'un sujet traitant des données multilingues et où l'utilisation d'un accélérateur (TPU) s'avère indispensable. Je présenterai tout d'abord ce que j'ai réalisé dans le pré traitement des données avant d'exposer la partie modélisation pour finir en présentant les résultats et le score obtenu.



Source de données : compétition Kaggle « **jigsaw-multilingual-toxic-comment-classification** ». Utilisation des fichiers *jigsaw-toxic-comment-train.csv* et *jigsaw-unintended-bias-train.csv* pour le jeu d'entraînement et *validation.csv* pour le jeu de validation.

A noter : Toutes les données et modèles remontés dans cette présentation font référence aux différents notebooks et documents fournis avec ce support.

Au programme

- 1. Présentation du jeu de données**
- 2. Preprocessing**
- 3. Data augmentation**
- 4. Modélisation**
- 5. Résultats**

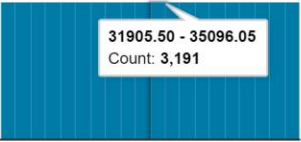
1- Présentation du jeu de données

Train dataset

- ✓ 2 fichiers de train disponibles :
 - « *jigsaw-toxic-comment-train.csv* » : data issue de commentaires provenant de Wikipedia. Le dataset contient 223 549 observations. On conserve 2 variables : la colonne « *comment_text* » qui sera mon input et qui est en anglais. La target est la colonne « *toxic* » qui est un booléen indiquant si le commentaire est toxique ou non.
 - « *jigsaw-unintended-bias-train.csv* » : data issue du « Civil Comments Dataset ». Fichier plus fourni que le premier avec 1 876 468 observations. On garde les mêmes variables que dans le fichier ci-dessus. La target « *toxic* » est transformée en booléen car elle contient à la base une probabilité de toxicité. Ce dataset ne sera pas utilisé pour les premiers tests mais une petite partie sera ensuite extraite pour challenger le score final.
- ✓ Répartition des commentaires toxiques : correspond à 10% environ des observations disponibles.

Validation & Test dataset

- ✓ Jeu de validation : « *validation.csv* ». Le jeu de données est cette fois-ci multilingue. Il sera utilisé lors de la phase d'entraînement pour tester la qualité du modèle. Le dataset contient 8 000 observations.
- ✓ Jeu de test : « *test.csv* ». Fichier comprenant les commentaires pour lequel la target sera à prédire. C'est ce résultat qui sera soumis pour déterminer le score obtenu pour la compétition. Il contient 63 812 commentaires de différentes langues.

id	content	lang
 0 63.8k	63812 unique values	tr 22% pt 17% Other (38800) 61%
0	Doctor Who adlı viki başlığına 12. doctor olarak bir viki yazarı kendi adını eklemiştir. Şahsen düze...	tr
1	Вполне возможно, но я пока не вижу необходимости выделять материал в отдельную статью. Если про пра...	ru
2	Quindi tu sei uno di quelli conservativi , che preferiscono non cancellare. Ok. Avresti lasciato ...	it

2- Preprocessing

Gestion des emojis

```
'♄': 'comet',  
'👁️': 'winking face with tongue',  
'✍️': 'writing hand',  
'♟️': 'chess pawn',  
'🎄': 'Christmas tree',  
'♟️': 'black nib',  
'😄': 'grinning face',
```

Tester la présence d'emojis dans les commentaires et les lister (utilisation de la librairie **demoji**).

```
In [10]: text = "game is on 🔥😄"
```

```
In [11]: convert_emoji(text)
```

```
Out[11]: 'game is on  fire  face with tears of joy '
```

Construction d'une fonction qui convertit l'emoji par sa « signification » en anglais. On applique cette fonction à la colonne « *comment_text* » du jeu d'entraînement.

Suppression des caractères indésirables

- ✓ Utilisation de la librairie **textthero**,
- ✓ Etapes de nettoyage :
 - Suppression des digits
 - Suppression des diacritiques (accents, retours charriot...)
 - Suppression des double espaces
 - Suppression des Urls
 - Suppression des tags HTML

	id	comment_text	toxic	lang	comment_text_clean
0	0000997932d777bf	Explanation\nWhy the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalism, just closure on some GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retired now.89.205.38.27	0	en	Explanation Why the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalism, just closure on some GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retired now. . . .
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC)	0	en	D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) : , January , (UTC)

Suppression des contractions

- ✓ Utilisation de la librairie **contractions**,
- ✓ Application de la modification aux commentaires anglais uniquement,
- ✓ Le principe : transformer par exemple “You’re” en “You are”,
- ✓ Construction d’une fonction pour effectuer cette opération.

```
In [22]:  
def remove_contractions(text):  
    text_clean = contractions.fix(str(text))  
    return text_clean
```

```
In [23]:  
remove_contractions("I'm ok")
```

```
Out[23]:  
'I am ok'
```

Gestion des fautes d'orthographe

- ✓ Utilisation de la librairie **autocorrect**,
- ✓ Correction des mots anglais uniquement,
- ✓ Paramètre « fast » positionné à True : traitement plus rapide mais correction moins efficace,
- ✓ Ecriture d'une fonction permettant de réaliser le travail.

```
In [27]:  
def make_corrections(text):  
    text_clean = check(text)  
    return text_clean
```

```
In [28]:  
make_corrections("It is veiry goood !")
```

```
Out[28]:  
'It is very good !'
```

Détection de langues

- ✓ Utilisation de la librairie **LanguageDetector** de Spacy et d'un dictionnaire multilingue,
- ✓ Calcul d'un score permettant de donner la probabilité que la langue détectée est bien la langue attendue,
- ✓ Méthode pas assez robuste à mes yeux : certains commentaires anglais se retrouvent avec un score à 0 à cause de certains mots mal identifiés ou inconnus du dictionnaire,
- ✓ Choix de garder l'ensemble des commentaires même si certains sont dans d'autres langues que celles attendues.



3- Data augmentation

Synonyme augmentation

- ✓ Utilisation de la librairie **nlpaug**
- ✓ Nombreuses possibilités de data augmentation comme :
 - Ajout ou remplacement de mots par plusieurs possibilités notamment via BERT pour rester dans le contexte,
 - Ajout de phrases aux phrases existantes,
 - Remplacement de mots par des synonymes pour enrichir le corpus (utilisation de wordnet),
- ✓ Choix de se baser la méthode des synonymes,
- ✓ Construction d'une fonction permettant de réaliser cette tâche et stockage dans une nouvelle colonne.

In [38]:

```
augmented_text = aug.augment(text)
print("Original:")
print(text)
print("Augmented Text:")
print(augmented_text)
```

```
Original:
The quick brown fox jumps over the lazy dog .
Augmented Text:
The immediate brown university charles james fox jumps over the
lazy dog .
```

4- Modélisation

Méthodologie et cadre général

- 1 Configuration du TPU
- 2 Tokenisation et encodage
- 3 Construction et compilation du modèle
- 4 Entraînement du modèle
- 5 Soumission du résultat sur Kaggle

Tokenisation et Encodage

- ✓ Permet d'avoir une représentation vectorielle du document,
- ✓ Utilisation de **Automodels** de HuggingFace : permet de récupérer le *tokenizer* correspondant au paramètre indiquant le modèle pré-entraîné sélectionné,

Tokenisation

- Attribution d'un index à chaque mot,
- L'index obtenu correspond à celui du modèle pré-entraîné sélectionné,
- Création d'une fonction *get_tokenizer* qui permet de faire ce travail,

Encodage

- Prend le token obtenu ci-dessus et retourne la représentation vectorielle du mot dans son contexte,
- Création de la fonction *encode* avec la méthode *batch_encode_plus* du *tokenizer*,
- Chaque séquence doit avoir la même taille (utilisation d'une fonction de padding),
- Résultat stocké dans un tenseur,
- Application de la fonction d'encodage aux inputs que l'on souhaite tester (donnée brute, nettoyée ou augmentée),
- Utilisation de l'API `tf.data.Dataset` pour créer un dataset optimisé (mémoire et temps de compute) à partir des inputs.

Construction du modèle

- ✓ Utilisation de modèles pré-entraînés (transfer learning),
- ✓ Création d'une fonction permettant de construire un modèle de type réseau de neurones,
- ✓ Prend en compte une couche de type *transformer* faisant référence au modèle pré-entraîné sélectionné.

Comme pour la partie « Tokenisation », on utilise TFAutoModel pour récupérer les poids du modèle choisi dans la méthode *from_pretrained*,

- ✓ Ici, je testerai 2 modèles pré-entraînés : BERT et XLM-RoBERTa,
- ✓ Exemple de modèle après compilation :

```
Model: "model"
-----
Layer (type)                Output Shape              Param #
-----
input_word_ids (InputLayer)  [(None, 192)]             0
-----
tf_roberta_model (TFRobertaM) ((None, 192, 1024), (None 559890432
-----
tf_op_layer_strided_slice (T [(None, 1024)]             0
-----
dense (Dense)                (None, 1)                 1025
-----
Total params: 559,891,457
Trainable params: 559,891,457
Non-trainable params: 0
-----
CPU times: user 2min, sys: 35.8 s, total: 2min 36s
Wall time: 2min 37s
```

Entrainement et soumission Kaggle

✓ 2 étapes pour entrainer le modèle :

- Entrainement du *train_dataset* avec validation sur le *valid_dataset* : on vient entrainer le jeu de données en anglais sur 3 époques,
- Entrainement du *valid_dataset* : on vient ajouter des étapes d'entrainement cette fois sur le jeu de validation composé de commentaires multilingues. Le jeu de données est plus petit et est entrainé sur le double d'époques, soit 6.

✓ Détermination du résultat et soumission :

- Prédiction de la toxicité ou non sur le *test_dataset*,
- Sauvegarde du résultat dans un fichier nommé « *soumission.csv* »,
- Obtention du score pour le concours

Liste des tests réalisés

Soumission	Public Score	Modèle	Data	Remarques
1	0.8674	distilbert-base-multilingual-cased	Originale	
2	0.8708	distilbert-base-multilingual-cased	Cleansée	
3	0.8730	distilbert-base-multilingual-cased	Augmentée	
4	0.8988	jplu/tf-xlm-roberta-large	Originale	
5	0.9308	jplu/tf-xlm-roberta-large	Cleansée	
6	0.8871	jplu/tf-xlm-roberta-large	Augmentée	
7	0.8384	jplu/tf-xlm-roberta-large	Cleansée	Ajout d'un EarlyStopping
8	0.8424	roberta-large	Cleansée	Couche de Dropout
9	0.9306	jplu/tf-xlm-roberta-large	Cleansée	Couche de Dropout
10	0.8892	bert-base-multilingual-cased	Cleansée	
11	0.9348	jplu/tf-xlm-roberta-large	Cleansée	Ajout de data (+250 000)

5- Résultats

Score final

- ✓ Meilleur score obtenu : **0.9346**
- ✓ Ce résultat me place autour de la 1000^{ème} place sur 1650 candidats (meilleur score obtenu : 0.9556)
- ✓ Critères retenus :
 - Data cleansée,
 - Data composée du fichier *jigsaw-toxic-comment-train.csv* et d'une partie de *jigsaw-unintended-bias-train.csv* (500 000 observations environ). A noter que cet ajout a engendré 1h de compute en plus,
 - Modèle pré-entraîné **jplu/tf-xlm-roberta-large**.
- ✓ Axes d'amélioration :
 - Améliorer encore le preprocessing, aller plus loin dans le nettoyage,
 - Ajouter l'intégralité du fichier *jigsaw-unintended-bias-train.csv*,
 - Tester une autre approche comme une traduction en anglais de l'ensemble des commentaires.

Conclusion et axes d'amélioration

Conclusion

✓ Participation à une compétition Kaggle :

- Projet enrichissant,
- Mise en place d'un projet de machine learning dans sa globalité et dans un contexte multilingue intéressant,
- Découverte de nouvelles librairies pour le preprocessing des datas,
- Découverte de nouveaux modèles de NLP pré-entraînés,
- Prise en main de Kaggle et des possibilités qu'offre cette plateforme collaborative,
- Découverte des TPU.

✓ Axes d'amélioration personnel :

- Aller plus loin dans la prise en main de modèles pré-entraînés (ajout de layers, multiplication des tests, ...),
- Continuer à s'autoformer et rester à l'écoute des innovations qui restent très fréquentes.

Merci !
Des questions ?