Documentation

Projet : « English proficiency prediction »

Réalisé par : Asma NAIFAR et Marouane El ADBI

1. Pré-traitement des transcripts des participant

Avec la fonction **“generateCleanFile(filename)”**, on commence par récupérer toutes les lignes qui se trouvent entre les balises "B" et de supprimer le retour à la ligne pour chaque fichier et génère à la fin un seul fichier avec une seule ligne.

La fonction “**generateCleanline(filename)”** permet à chaque ligne de chaque fichier de supprimer toutes les balises et leurs données qui ne nous intéressent pas grâce à la fonction **“removeAllInternalTags”** afin de générer un fichier avec une ligne épurée.

Problème rencontré:

Le problème qu’on a rencontré dans la partie de pré-traitement c’était le fait qu’il existait des balises qui s'ouvrent sur une ligne et se ferment dans la ligne qui suit. Cela résultait en suppression incorrecte des balises et leurs contenus. Donc, pour rectifier ce mal-fonctionnement du système, on a décidé de supprimer tous les retours à ligne et traiter chaque fichier textes en tant qu’une seule ligne.

2. Traitement du jeu de données Sac de mots (Bag of Words)

2.1 Correspondance transcript-score

La fonction “**extractScore(filename)”** permet de récupérer le score qui se trouve dans la balise "SST_level" pour chaque transcript et l’associer à ce fichier..

2.2 Création du vocabulaires

Dans un premier temps, nous avons créé la fonction **“word_extraction(line)”** qui permet de supprimer les mots et tout ce qui ponctuation qu'on voit c’est pas utiles dans sur la phrase.
Ensuite nous avons créé notre propore function **“createVectorizer(filename)”** qui permet de générer un sac de vocabulaire.

Problème rencontré:

Nous avons trouvé un problème de temps de calcul en générant les vecteurs de Bag of Words à la main. Alors la solution c’était de basculer en utilisant la bibliothèque **sklearn** et la fonction **CountVectorizer** qui fait automatiquement la génération du vecteur Bag of words.

2.3 Diviser notre base de données en ensembles d'entraînement et de test de manière aléatoire

2.4 Créer sac de vocabulaire à partir de l'ensemble de données d'entraînement

Dans cette partie nous avons appliqué la méthode word_extraction pour chacun des fichiers d'entraînement et des tests afin de générer des fichiers propres (entraînement et test). Ensuite nous avons tokenizer et construit la liste de vocabulaires à partir des données d'entraînement et enfin encoder les fichiers et les convertir en un tableau

2.5 faire un dataframe qui est une structure de données permettant de stocker des données en deux dimensions : lignes et colonnes.

Nous avons créé Dataframe pour les données d'entraînement (input) **"x_train"** qui ont comme lignes **"nom de fichier"** et les colonnes **"vocabulaires"**, ensuite nous avons fait la même choses pour (output) qui ont comme lignes **"nom de fichier"** et un seul colonne **"Score"**

- Nous avons fait de même pour les données de test. on récupère même les données de test (output) afin de faire les comparer avec les (output) qu'on va prédire

3. Entraîner le classificateur pour prédire le score SST

Tout d'abord, nous avons utilisé différents classificateurs classiques pour prédire notre sortie.

- **La classification naïve bayésienne**
- **La régression logistique**
- **SVC**
- **eXtreme Gradient Boosting**

D'après le résultat, nous avons remarqué qu' à chaque fois ça donne une prédiction différente par exemple la régression et svc on peut dire que c'est très proche avec une légère différence. Avec le XGBoost et SVC nous obtenons souvent une prédiction globale légère plus forte que celles des autres classifieurs.

4. la précision et Matrice de confusion

Nous avons fait un résumé des résultats de prédictions sur un problème de classification sous la forme matrice de confusion. Les prédictions correctes et incorrectes sont mises en lumière et réparties par classe.

5. Construction du réseau de neurones

Nous avons utilisé un modèle à 3 couches :

- première couche 32 neurone en utilisant la fonction "relu" et comme taille des données d'entrée --> la taille du vocabulaire
- Deuxième couche 16 neurones en utilisant la fonction "relu"
- La dernière couche 9 neurone est de la même taille que nos données de sortie

6. Utilisation de Word2vec

word2vec permettra de représenter les mots en tenant compte de leur similarité et du contexte actuel.

On a utilisé la fonction Word2vec de la bibliothèque `gensim` puis on a suit les étapes suivantes:

- Faire entraîner le modèle word2vec sur la totalité des données text
- Générer les vecteurs correspondants pour les données d'entraînement et de test.
- Faire entraîner les mêmes modèles de prédiction sur les nouveaux vecteurs générés et déduire leurs précisions..