Natural Language Processing – Projet

Analyse de sentiments

Afin de clôturer notre formation en Natural Language Processing, il nous a été demandé de faire un projet sur l’analyse de sentiments. L’analyse de sentiments peut s’avérer très utile, notamment pour une entreprise, dans le but de recueillir des informations sur ce que pensent leurs clients. L’objectif est, à partir d’un texte (dans notre cas des tweets), de déterminer s’il est positif ou négatif. Plutôt que d’analyser manuellement chaque tweet un par un, il peut alors être très pratique d’établir un modèle qui peut analyser automatiquement ces tweets et dire si ces derniers sont plutôt positifs ou plutôt négatifs, avec une certaine précision. Ces prédictions peuvent ensuite servir comme indicateurs pour de potentielles améliorations.

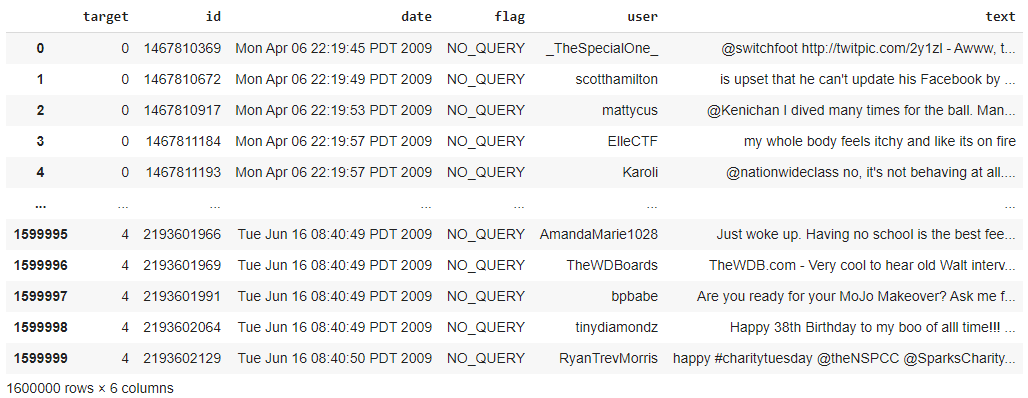
Notre objectif est donc de créer un modèle d’analyse de sentiments et de l’entraîner, le but étant, à l’issue de cet entraînement, de pouvoir prédire le sentiment d’un texte à partir de ce modèle.

Voici donc comment va se découper notre projet pour l’analyse de sentiments :

1. Importation des données d’entrée
2. Prétraitement des données
3. Préparation des données pour l’entraînement
4. Création du modèle, entraînement
5. Tests de prédiction
6. Importation des données d’entrée

La première étape est d’importer nos données. Les données que nous utilisons sont présentes dans un fichier .csv : *training.1600000.processed.noemoticon.csv*

Ce jeu de données comporte 1 600 000 tweets, tous répertoriés comme positifs ou négatifs.



Ci-dessus, un aperçu des données que nous avons à disposition : un tableau de 6 colonnes :

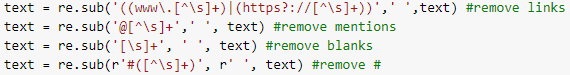
* target : 0 ou 4, valeurs définissant le sentiment du tweet (0 = tweet négatif, 4 = tweet positif)
* id : identifiant du tweet
* date : date du tweet
* flag : la validité du tweet (tous à NO\_QUERY : les tweets sont tous valides)
* user : l’utilisateur ayant posté le tweet
* text : le tweet en question

Pour notre application, les seules données qui nous intéressent sont les tweets et les sentiments. Nous ne gardons donc que ces deux colonnes pour la suite.

1. Prétraitement des données

Afin de faciliter le futur traitement des données, il faut traiter tous les tweets pour qu’ils soient plus facilement lisibles. Nous allons donc utiliser plusieurs traitements pour “nettoyer” les tweets :

* On supprime les liens (www., httpss://, etc)
* On supprime les mentions (@utilisateur)
* On supprime les espaces vides
* On supprime les hashtags
* On supprime les ponctuations (on ne garde que les lettres)
* On met toutes les lettres en minuscule
* Grâce à un dictionnaire de stopwords (qui sont les mots qui ne sont pas fondamentalement importants dans la phrase), on supprime les stopwords

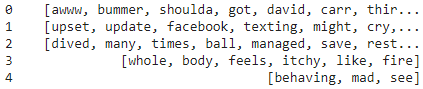




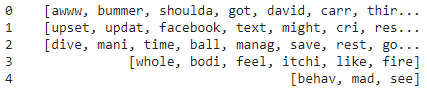
Ainsi nous obtenons des tweets propres, avec seulement les informations essentielles à l’intérieur

Ensuite, nous voulons appliquer le Stemming à nos tweets. Le but du Stemming (enraciment en français) est d’obtenir une forme tronquée d'un mot commune à toutes les variantes possibles de ce mot, c’est-à-dire d’obtenir le mot “racine”, le mot de base. Exemple : feels, felt, feeling => mot racine = feel

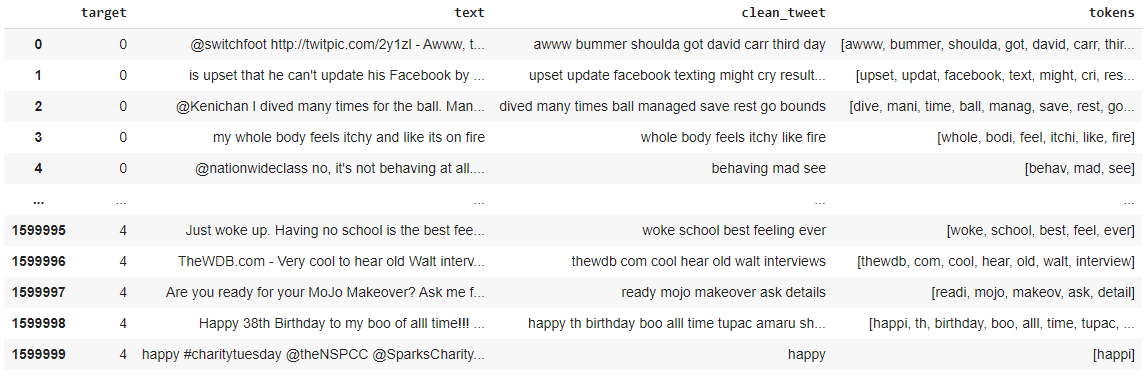
Pour procéder, nous devons tokenizer tous nos tweets, c’est à dire découper nos phrases en tableaux de mots :



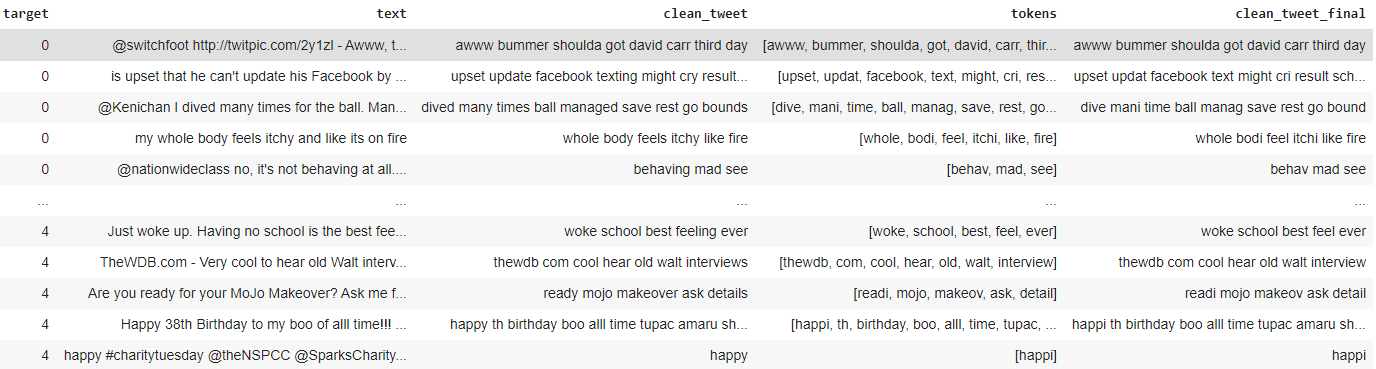
A présent on parcourt tous les mots et on applique le Stemming :



On crée une nouvelle colonne pour une visualisation



On rassemble les tokens pour créer les phrases finales traitées, et oon les mets dans une nouvelle colonne appelée *‘clean\_tweet\_final’*

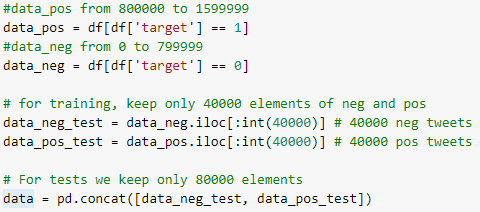


Nos données sont maintenant toutes traitées. La prochaine étape est de les préparer pour l’entraînement.

1. Préparation des données pour l’entraînement

Pour une question de lecture, nous choisissons de modifier toutes les valeurs de target valant 4 en 1. Ainsi, 0 = tweet négatif, 1 = tweet positif

Nous créons ensuite deux jeux de données, data\_pos et data\_neg, respectivement les données positives et les données négatives. Pour entraîner notre modèle, nous n’allons sélectionner que 40000 tweets positifs et 40000 tweets négatifs, donc un total de 80000 tweets. En effet, entraîner le modèle avec les 1600000 tweets prendrait beaucoup trop de temps, surtout si on veut faire un minimum d’époques pour l’entraînement du modèle.



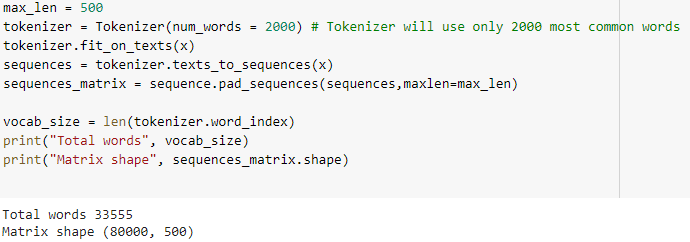
Puis nous définissons nos données d’entrée : x sera les textes et y les sentiments

Il faut maintenant transformer nos mots en vecteurs.

*max\_len* représente le nombre maximum de mots sélectionnés pour l'entrainement. Ces mots seront utilisés selon l'importance qui distingue les tweets positifs et négatifs. Nous en choisissons 500.

Nous utilisons ensuite un Tokenizer, dont le but est de créer des tokens pour chaque mot du jeu de données et de les mapper à un index à l'aide d'un dictionnaire. Nous lui choisissons un num\_words de 2000, c'est-à-dire qu'il n'utilisera que les 2000 mots les plus communs dans le jeu de données.

Etant donné que nous allons construire un modèle séquentiel, alimenté par des séquences de nombres, nous devons nous assurer que les séquences en entrée auront toutes la même forme, elles doivent toutes avoir la même longueur. Le problème est que les textes des tweets ont un nombre de mots différent. Pour pallier ce problème, nous utilisons la fonction *pad\_sequence* de keras. Cette fonction transforme toute la séquence dans une longueur constante *max\_len*.

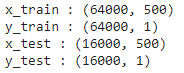


Puis nous définissons nos données de l'entrainement et nos données de test.

Pour cela on utilise la fonction train\_test\_split de sklearn, qui mélange le jeu de données et le divise. Cette fonction prend un paramètre appelé test\_size, qui correspond à la portion du jeu de données que nous souhaitons assigner aux données de test. Nous choisissons d'assigner 20% du jeu de données aux données de test

*x\_train, x\_test, y\_train, y\_test = train\_test\_split(sequences\_matrix, y, test\_size=0.2, random\_state=2)*

Avec un jeu de données de 80000 tweets, on a dons les portions suivantes :



1. Création du modèle, entraînement