Reina Romain

# Compte rendu du Projet NLP Février 2021

## Introduction

Présentation du projet Pour une entreprise, il est important de bien comprendre ses clients. Il n'est pas toujours évident de savoir ce que vos clients pensent de votre entreprise. Aujourd'hui, grâce aux réseaux sociaux, les utilisateurs donnent leur ressenti de façon libre.

Une bonne application est d'utiliser ces données pour prendre la température des réseaux sociaux à propos de votre entreprise. Le but est de savoir si vos clients parlent en bien ou en mal de vos entreprises et de vos produits. Cette température peut ensuite être vue comme un indicateur à améliorer via des campagnes de communication ou une amélioration des offres ou des produits.

Dans le cadre de mon projet, j'ai eu la chance de réaliser l'implémentation de cette solution.

Lien Github : https://github.com/RomainReina/Projet_NLP

## Jeu de donnée utilisé

Les données sont issues d'un dataset de 1.6 Million de tweets (Sentiment140 dataset with 1.6 million tweets | Kaggle). Ce tableau Excel contient des phrases dites à sentiment positif, neutre et négatif.
Dans un premier temps, il est important de savoir comment traiter ces données. La colonne 1 contient le sens de la phrase, et la colonne 5 le texte en lui-même. La colonne 5 sera traité pour la rendre exploitable, comme la suppression des #, @, des usernames, des lien hypertextes ainsi que les majuscules et des chiffres.

## Création d'un dictionnaire

Dans l'implémentation du réseau de neurone, il est nécessaire de transformer les phrases en vecteurs de mots. Une fois tokenisé, les mots utilisés dans la construction du modèle sont placés dans un dictionnaire de vocabulaire. Ces mots sont associés à un chiffre qui sera ensuite utilisé pour afficher un vecteur. Dans sa construction, on l'initialise avec les sigles suivants :

```
Vocab = {'__PAD__': 0, '__</e>__': 1, '__UNK__': 2}
```

- __PAD__ est utilisé pour compléter une liste si cette derniere n'est pas la la bonne taille,
- /e designe la fin d'une ligne,
- __UNK__ désigne un mot inconnu du Vocabulaire,

Voici un exemple d'un vecteur de mot :

```
sentence: username awww that s a bummer you shoulda got david carr of third day to do it d
sentence into tensor: [3, 4, 2, 2, 2, 5, 2, 6, 7, 8, 9, 2, 10, 11, 2, 2, 2, 2]
```

## Création d'un data_generator

Toujours dans l'optique d'utiliser un réseau de neurone, la création d'un data_generator est une évidence. Cette fonction reprend les différents aspects vus jusqu'ici, et permet une itération dans le but d'un entrainement d'un modelé. Le data_generator utilisé ici est repris du site https://zhangruochi.com/Sentiment-with-Deep-Neural-Networks/2020/08/22/#2.

## Initialisation du modelé

```
] def SER(vocab_size=len(Vocab), embedding_dim=256, output_dim=2, mode='
    '''
      Input:
        vocab_size - integer containing the size of the vocabulary
        d_model - integer describing the embedding size
      Output:
        model - a trax serial model
    '''
     ### START CODE HERE (Replace instances of 'None' with your code)
    # create embedding layer
    embed_layer = tl.Embedding(
        vocab_size=vocab_size, # Size of the vocabulary
        d_feature=embedding_dim)  # Embedding dimension

    # Create a mean layer, to create an "average" word embedding
    mean_layer = tl.Mean(axis = 1)

    # Create a dense layer, one unit for each output
    dense_output_layer = tl.Dense(n_units = output_dim)


    # Create the log softmax layer (no parameters needed)
    log_softmax_layer = tl.LogSoftmax()

    # Use tl.Serial to combine all layers
    # and create the classifier
    # of type trax.layers.combinators.Serial
    model = tl.Serial(
      embed_layer, # embedding layer
      mean_layer, # mean layer
      dense_output_layer, # dense output layer
      log_softmax_layer # log softmax layer
    )
  ### END CODE HERE ###
    return model
```

Un modelé de réseau de neurone contient des couches, des nœuds, des poids ainsi de d'une fonction d'activation. Cette fonction a été repris et modifié d'un des précèdent exercice.

## Entrainement du modèle

L'entrainement est la où le modèle va se construire. Il va itérer des epoches. Suivant son nombre, le modèle sera plus ou moins précis.

Pour ma part, la compilation du modèle m'a fait défaut. Je n'arrivais pas à comprendre ce qu'est le jeu de donnée d'évaluation (eval_generator), si c'était soit des valeurs pour connaitre le sentiment du tweet, ou soit une autre série de tweets. Dans tout les cas les résultats de mon projet ne sont plus valable à partir de cette étape.

## Compilation de l'accurency

Maintenant que le modèle est monté, nous pouvons calculer l'accurency. Elle nous permet de visualiser si les prédictions sont proches ou non de la solution finale. Plus l'accurency est proche de 1, plus le modèle est bon.

## Prédiction

Si l'accurency est proche de 0.9, alors le modèle est bon pour prédire des phrases extérieures au jeu de donnée, ce qui nous est demandé pour ce projet.

(L'output est bien entendu faux)

```
sentence = "i'm feeling good today"
tmp_pred, tmp_sentiment = predict(sentence)
print(f"The sentiment of the sentence \n***\n\"{sentence}\"\n***\nis {tmp_sentiment}.")

print()
# try a negative sentence
sentence = "I hated my day"
tmp_pred, tmp_sentiment = predict(sentence)
print(f"The sentiment of the sentence \n***\n\"{sentence}\"\n***\nis {tmp_sentiment}.")
```

```
The sentiment of the sentence
***
"i'm feeling good today"
***
is positive.

The sentiment of the sentence
***
"I hated my day"
***
is positive.
```

## Conclusion et point d'amélioration

Contrairement aux exercices travaillés en cours, ce projet réellement appris à construire un RNN, les différentes étapes et méthodes, puisqu'en cours les notebooks étaient déjà prérempli. J'ai pu constater que le traitement des données en amont été tout aussi important que de modéliser un réseau de neurone.
Dans une optique d'amélioration, il y a bien entendu l'entrainement du modèle qui est a revoir, mais aussi une méthode plus simple pour traiter mes donnée en début de notebook.