Détecter la polarité d'un tweet

Comment parvenir à identifier le sentiment d'un tweet ? Inutile de dire qu'il s'agit d'un enjeu majeur pour les entreprises. Identifier un tweet négatif signifie anticiper un buzz potentiellement nuisible à la réputation.

L'objectif de cet article est de parvenir à classifier la polarité des tweets (selon qu'ils soient plutôt positifs ou négatifs) à l'aide de différents modèles disponibles en Python et surtout de comparer leurs performances.

1. Tweets et NLP

Comme les machines ne savent pas ce qu'est un mot ou une phrase, il est nécessaire d'encoder les données de sorte à obtenir des vecteurs et des données numériques.

Prétraitement

L'encodage des tweets est assez délicat, car leur structure est quelque peu chaotique : les gens s'expriment avec des emojis, des ponctuations et… des fautes d'orthographe. Tout cela demande un sérieux travail de nettoyage préalable. Une fois nettoyés, les tweets ont été transformés grâce aux techniques de prétraitement de texte : le stemming et la lemmatisation.

Vectorisation & sequencing

Suite au prétraitement, les tweets doivent être transformés en vecteurs. Grâce à Keras et son tokenizer, il est possible de classer chaque mot en fonction de sa fréquence, pour ensuite lui attribuer un index. Nous obtenons ainsi un dictionnaire contenant les entiers correspondant à l'index d'un mot. Le dictionnaire, quant à lui, permet de transformer l'ensemble des tweets en arrays remplis d'indexes, en d'autres termes transformer le texte en chiffres.

2. Word Embeddings

Il s'agit d'approches qui permettent **d'identifier les mots qui partagent des contextes similaires**, et par conséquent partagent également des significations similaires. Nous avons utilisé deux techniques de Word Embeddings : Word2Vec et GloVE.

Word2Vec entraîne les données textuelles avec un réseau de neurones. L'embedding qui en résulte capture les mots qui apparaissent dans des contextes similaires. **GloVe,** à son tour, se concentre sur les cooccurrences de mots dans l'ensemble du corpus. Ses embeddings sont le résultat d'une probabilité que deux mots apparaissent ensemble.

Les deux approches donnent la possibilité de mesurer la relation entre les mots grâce à la similarité cosine (dans l'espace vectoriel, les mots sémantiquement similaires sont proches les uns des autres). Ce qui est absolument fascinant, c'est que nous pouvons également utiliser l'arithmétique sur ces plongements pour en déduire le sens.

3. Approches de modélisation

Une fois les tweets plongés dans un espace vectoriel, nous n'avons plus qu'à faire tourner les modèles afin de détecter les sentiments positifs ou négatifs.

Modèle sur mesure simple

Le modèle pour lequel nous avons opté est la **Régression Logistique.** La régression logistique utilise la fonction sigmoïde qui permet de mesurer si une entrée a dépassé le seuil de classification.

- ➢ Les scores obtenus par le modèle : ACC 0.72, F1 0.72, AUROC 0.72.

Modèles sur mesure avancés

Dans un premier temps nous avons utilisé un réseau de neurones simple avec une couche Embedding de Keras, puis deux types de réseaux de neurones profonds : CNN et RNN.

- ❖ **Keras et sa couche Embedding**

Il s'agit d'un modèle de réseau de neurones simple avec une couche embedding basique. Pour améliorer les résultats, nous avons optimisé les hyperparamètres suivants : le nombre de neurones dans la première couche dense, le learning rate et le nombre d'epochs.

Voici les résultats obtenus suite à l'entraînement du modèle Keras :

- ➢ Sur un corpus lemmatisé : ACC 0.69, F1 0.53, AUROC 0.76.
- ➢ Sur un corpus stemmé : ACC 0.69, F1 0.47, AUROC 0.76.

- ❖ **RNN**

Les Réseaux de Neurones Récurrents interviennent lorsque l'on a affaire à des séquences temporelles. Cette structure introduit un mécanisme de mémoire des entrées précédentes qui persiste dans les états internes du réseau. Le problème c'est qu'un réseau RNN traditionnel peut facilement oublier des données anciennes (ou des mots assez éloignés du mot courant dans un texte) lors de la phase d'apprentissage, car sa mémoire est "courte". C'est grâce au **réseau de neurones `LSTM`** que l'on peut remédier à cela (savoir quelles décisions ont été prises dans le passé afin de prendre une décision optimale à l'instant t).

Voici les résultats obtenus suite à l'entraînement du réseau profond RNN :

- ➢ Sur un corpus lemmatisé :
  - avec Word2Vec : ACC 0.51, F1 0.51, AUROC 0.51
  - avec GloVe : ACC 0.50, F1 0.48, AUROC 0.50
- ➢ Sur un corpus stemmé :
  - avec Word2Vec : ACC 0.5, F1 0.54, AUROC 0.5
  - avec GloVe : ACC 0.5, F1 0.51, AUROC 0.5

- ❖ **CNN**

Les réseaux de neurones convolutifs (CNN ou ConvNets) appliquent une opération de convolution sur les données d'entrée pour détecter des patterns. CNN se composent d'une ou plusieurs couches de convolution, chacune contenant des noyaux internes. Les CNN avec **convolution 1d** peuvent être utilisées pour des tâches NLP telles que la classification de texte, la génération de texte, etc., car les données textuelles sont considérées comme des données séquentielles (un signal).

Voici les résultats obtenus suite à l'entraînement du réseau profond CNN :

- ➢ Sur un corpus lemmatisé :
  - avec Word2Vec : ACC 0.5, F1 0.54, AUROC 0.5
  - avec GloVe : ACC 0.51, F1 0.54, AUROC 0.51
- ➢ Sur un corpus stemmé :
  - avec Word2Vec : ACC 0.51, F1 0.52, AUROC 0.51
  - avec GloVe : ACC 0.51, F1 0.53, AUROC 0.52

Modèle avancé BERT

BERT (Bidirectional Encoder Representations from Transformers), contrairement aux modèles précédents qui lisent le texte entré de manière séquentielle (de gauche à droite ou de droite à gauche), permet l'entraînement bidirectionnel.

BERT fait appel à **Transformer**, un mécanisme d'attention qui apprend les relations contextuelles entre les mots. De plus, BERT utilise une nouvelle technique appelée **Masked LM**. L'idée est de lire la séquence entière de mots en une seule fois grâce à l'encodeur du Transformer. Mais attention, avant d'introduire des séquences de mots dans BERT, 15 % des mots de chaque séquence sont remplacés par un token [MASK]. Le modèle tente alors de prédire la valeur originale des mots masqués, sur la base du contexte fourni par les autres mots non masqués de la séquence.

Voici les résultats obtenus suite à l'entraînement du modèle BERT :

- ➢ Sur un corpus lemmatisé : ACC 0.6, F1 0.70, AUROC 0.68.
- ➢ Sur un corpus stemmé : ACC 0.49, F1 0.62, AUROC 0.62.

4. Interprétation des résultats

La métrique d'évaluation que nous avons choisi est le **score F1**. C'est la moyenne pondérée de la précision et du rappel.

Les résultats ne sont pas spectaculaires. L'aire sous **la courbe ROC** peut être interprétée comme la probabilité que, parmi deux tweets choisis au hasard, la valeur du marqueur soit plus élevée pour le tweet négatif que pour le positif. Par conséquent, une AUC de 0,5 (50%) indique que le marqueur n'est pas vraiment informatif. C'est le cas de la plupart des performances de nos modèles. Pour remédier à cela, il faudrait avoir un échantillon plus important, mieux prétraiter les données textuelles et dans l'idéal optimiser les hyperparamètres de façon plus poussée.

Le modèle BERT semble avoir les meilleurs résultats parmi les modèles à base de réseaux de neurones - le score F1 monte à 0.70 sur un corpus lemmatisé. Néanmoins, la classification la plus réussie est celle obtenue avec la Régression Logistique : le score F1 atteint 0.72 points et AUROC s'élève à 0.72.