Projet Semestriel - Text Mining

Antoine Adam, Dylan Monfret, Loïc Rakotoson

22 Janvier 2021

Table des Matières

1 Introduction

Dans le cadre de ce projet, nous nous sommes focalisés sur la tâche 2.1 du défi DEFT 2015, défi de fouille de texte autour de tweets en français. La tâche 2.1 consiste en *"l'identification de la classe générique de l'information exprimée dans le tweet"*. Les tweets en question sont donc labelisés en **4 catégories** de la manière suivante :

- `INFORMATION` : le tweet contient une information factuelle. (Exemple: Il a neigé cette nuit.)
- `OPINION` : le tweet exprime une opinion (une expression intellective) (Exemple: Je suis d'accord, il faut plus de solutions alternatives!)
- `SENTIMENT`: le tweet exprime un sentiment (une expression affective-intellective) (Exemple: Satisfait! de ce qu'ils ont fait!
- `EMOTION` : le tweet exprime une émotion (une expression affective) (Exemple: Le bruit de ces éoliennes, me met en colère.)

Le problème donné rentre donc dans les deux catégories du Traitement Naturel du Langage et de la Classification Supervisée. L'exécution de nos méthodes seront disponibles dans le répertoire GitHub suivant : Tweets Generic. Dans ce rapport nous développerons les méthodes de Machine Learning et de Deep Learning utilisés pour répondre à cette problématique, ainsi que les résultats obtenus.

"Les corpus DEFT 2015 ont été réalisés dans le cadre du projet uComp financé par l'ERA Net CHIST-ERA (ANR-12-CHRI-0003)."

2 Pré-traitements

Dans des contextes de traitement naturel du langage, il est important de ce demander comment retranscrire, comment extraire les informations pertinentes depuis un nombre important de texte. Dans le cadre de ce projet, nous avons donc aborder plusieurs pré-traitements différents à partir du script `base` avec la fonction `clean_tweet`. Les données traités sont par la suite stockées dans le sous-répertoire `data/datasets` suffixées de la sorte:

- `basic` : suppression des URLs uniquement (puisqu'ils n'importent pas d'information particulière, surtout les URL raccourcis de Twitter).
- `classic` : passage des caractères en minuscule, suppression des mots vides et de la ponctuation (idéal pour une vectorisation TF-IDF).
- `classic_nos_nol` : `classic`, mais sans stemming et lematisation (idéal pour des Embeddings)
- Sans suffixe : aucun pré-traitement

Et nous prenons garde à ne supprimer ni Hashtag, ni @ (arobase) de nom d'utilisateur car, dans le contexte de Twitter, ce sont des caractères qui ont de l'importance. Les données sont aussi séparées en 3 sets : `train`, `val` et `test`.

3 Méthodologie

3.1 Machine Learning classique

Avec un premier notebook `benchmark`, nous avons mis à l'épreuve les méthodes d'apprentissage machine classique pour savoir quelle méthode, sans optimisation d'hyper-paramètre, fonctionne le mieux selon les scores macro-F1, AUC et Accuracy, et par rapport à chaque pré-processing. Il sera ensuite intéressant d'optimiser la méthode qui marche le mieux.

3.2 Perceptron multicouche

Nous avons essayé plusieurs architectures et c'est celle qui a le mieux gérer le score F1. Une architecture trop complexe rend le label `SENTIMENT` invisible car vraiment trop peu de données pour beaucoup de paramètres, même avec une régularisation. Cette méthode a été appliqué sur les pré-processing `classic`.

3.3 Transformer : camemBERT

Utilisation du Transfert Learning sur les paramètres de CamemBERT : RoBERTa entraîné sur le corpus français OSCAR. Utilisation des données basic car les tokens qui viennent du corpus OSCAR comprennent quand un mot est un Hashtag et pareil quand c'est un username qui commence par @. Le processus est répertorié dans le notebook `camemBERT` et utilise les données `basic` et `nos_nol`.

Pour l'entraînement :

- 1ère entrainement en figeant les poids de CamemBERT et en mettant à jour uniquement les poids de la tête du réseau avec un learning rate de 1e-5 et un early stopping en surveillant la perte sur le set de validation et avec une patience de 2.
- 2ème entrainement de tout le réseau avec un learning-rate plus bas (1e-7) + early stopping

4 Résultats

4.1 Benchmark

Premier constat au niveau du Benchmark, `SGDClassifier` bat tous les autres classifiers, tout pré-processing confondu et sur toutes les métriques. C'est donc la méthode qui sera optimisé par grille `nestedCV` avec les données `classic`.

4.2 SGD Classifier

SGD requiert une standardisation des données, mais TF-IDF donnant déjà d'une certaine manière des données "standardisées", une sur-couche en devient inutile (Model 1). Pour prendre en compte le déséquilibre, on effectue :

- du sur-échantillonnage (ROS : Random Over Sampling)
- du SMOTE : sur échantillonnage en créant des individus en fonction de k-voisins.
- du SMOTE Tomek : sur-échantillonnage de la classe sous-représentée suivi d'un sous-échantillonnage des individus la classe dominante qui se chevauchent avec les classes sous-représentées (même si c'est déconseillé pour TFIDF).

Ce qui nous donne donc 19 modèles. En général ces 5 models mais avec plusieurs versions optimales selon les `innerCV`. On y évalue les scores sur les `outerCV` pour les 19 modèles, puis on regarde le top 5 sur le score F1 et l'AUC.

Les meilleurs modèles sont SGD sans standardisation et SGD avec SMOTE (k=15) qu'on évalue sur les données de test.

Et *in fine* SGD + SMOTE est le meilleur en AUC et F1. SMOTE a réussi à corriger le handicap du déséquilibre, le TP de SENTIMENT sur la matrice de confusion a doublé par rapport au SGD simple. Même si c'est pas ouf niveau équilibre.

4.3 MLP

Les réseaux récurrents n'ont pas données de très bons scores (Bi-LSTM et RNN).

Du coup, deux modèles nous nous rabattons sur deux modèles MLP à 2 couches : avec et sans ROS. ROS est appliqué sur les batchs lors de l'entraînement, sans resampling de type SMOTE sur le tokenizer (TF-IDF) à cause de `keras` et de la limitation de mémoire.

Le modèle n'est cela dit pas si bon que ça par rapport au SGD. L'application de ROS améliore le F1 d'un point sur les données de test.

4.4 camemBERT

La faible taille de nos données ne permet pas au modèle Transformer CamemBERT d'acquérir une bonne capacité de généralisation.

Le handicap de la sous-représentation du label `SENTIMENT` dans le set d'entraînement est plus important avec l'utilisation des batch. Si les labels `EMOTION` et `OPTINION` sont peu représentés dans les batch, `SENTIMENT` est complètement écrasé par le label `INFORMATION`. Ce fort déséquilibre pousse le modèle à sur-entraîné en prédisant toujours `INFORMATION`.
Sur les données `basic`, l'application de ROS améliore les 3 métriques, cependant le sur-échantillonnage

de `SENTIMENT` en passant de 73 à 1749 engendre des batch d'un seul individu répété plusieurs fois, ce qui rajoute un autre problème de sur-apprentissage.

En passant aux données `nos_nol`, le modèle baisse en performance puisqu'on perd les informations données par les HashTags et les usernames.

5 Conclusion

5.1 Meilleur classifier

Finalement, et avec nos essais, on remarque que la combinaison SGD avec SMOTE est la meilleure. On peut aussi se poser quelques questions de comparaison avec le "défi IA 2021", en effet les tweets sont des textes beaucoup moins propres que les descriptions de métiers que nous avions, rendant le preprocessing plus important. De plus, la faible quantité de données rend le déséquilibre entre les classes plus gênant (surtout pour les transformers, qui n'ont pas assez d'occurrences pour bien apprendre lors du fine-tuning). Enfin l'utilisation de SMOTE améliore le score, et montre qu'une méthode de rééquilibrage bien choisie est très utile !

Model	Preproc	F1	AUC	Accuracy
SGD	classic	43.92	64.18	58.93
SGD + SMOTE	classic	45.16	65.74	58.51
MLP	classic	41.23	70.03	57.26
MPL + ROS	classic	41.85	70.12	57.08
Camembert	basic	16.70	55.66	44.68
Camembert + ROS	basic	16.96	57.61	44.32
Camembert	nos_nol	15.55	53.27	44.90
Camembert + ROS	nos_nol	15.47	53.76	44.68

Table 1: Résultats des modèles retenus

5.2 Piste d'amélioration

Les données sont très déséquilibrées (deux ordres de grandeur de différence entre label `INFO` et `SENTIMENT`). En plus du resampling, de la Data Augmentation aurait été envisageable. C'est par manque de temps que nous n'avons pas opté pour cette option, mais c'est une possibilité.

References

- Thierry Hamon, Amel Fraisse, Patrick Paroubek, Pierre Zweigenbaum, and Cyril Grouin (2015). Analyse des émotions, sentiments et opinions exprimées dans les tweets : présentation et résultats de l'édition 2015 du défi fouille de texte (DEFT). In: Actes de DEFT. Caen, France.

- Amel Fraisse and Patrick Paroubek (2014). Twitter as a Comparable Corpus to build Multilingual Affective Lexicons. In: Proceedings of the 7th International Workshop on Building and Using Comparable Corpora at LREC 2014 (BUCC 2014), pages 17-21. Reykjavik, Iceland.

- Chawla, N. V. et al. "SMOTE: Synthetic Minority Over-Sampling Technique." Journal of Artificial Intelligence Research 16 (2002): 321–357. Crossref. Web.

- Martin, Louis, B. Muller, Pedro Javier Ortiz Suárez, Y. Dupont, L. Romary, 'Eric de la Clergerie, Djamé Seddah and Benoît Sagot. "CamemBERT: a Tasty French Language Model." ArXiv abs/1911.03894 (2020): n. pag.