



MASTER 1 INFORMATIQUE - MACHINE LEARNING 1

Classification d'assertions selon leur valeur de véracité

Hugo Maitre

Awa Seck

Mitra Aelami

Amirhossein Nasri

Adrien Linares

Encadrants :

Pr. Pascal PONCELET

Pr. Konstantin TODOROV

21 mai 2022

Table des matières

1	Introduction	2
1.1	Organisation de notre travail	2
2	Prétraitements	3
2.1	Séparation des jeux	3
2.2	Equilibrage des classes - DownSampling	3
2.3	Ingénierie textuelle	3
2.4	Visualisation des données	4
3	Developpement des modèles et Evaluation	5
3.1	Validation Croisé	5
3.2	Grid Search	5
3.3	Evaluation sur le jeu Test	5
4	Approches Deep Learning	7
4.1	LSTM	7
4.2	BERT	7
5	Autres Approches	8
5.1	Logistic Regression	8
6	Conclusion	9
6.1	Perspectives d'amélioration	9

1 Introduction

L'objectif de ce projet est qu'en utilisant des méthodes d'apprentissage supervisé, plus particulièrement de classification, nous arrivions à prédire si des assertions (*proposition que l'on avance et que l'on soutient comme vraie*) politiques soient vraies, fausses ou encore mixte. Le jeu de données utilisé est ClaimsKG¹ <https://data.gesis.org/claimskg/>. Nous avons volontairement mis peu de graphiques pour détailler au maximum tout en restant synthétiques notre approche, mais nous vous invitons à observer nos résultats et graphiques sur le github du projet. <https://github.com/hugomtr/FakeNewsDetectionDifferentNLPApproach>

1.1 Organisation de notre travail

Une bonne partie du travail consistait premièrement à traiter les données en amont de la classification et deuxièmement à définir les features à choisir pour notre jeu de données. Nous avons commencé dans un premier temps par traiter de petit jeu de données environ 300 textes.

Environ 10-20% du temps a été consacré à l'ingénierie des données textuelles, étape fondamentale pour avoir un modèle qui fonctionne bien. Ceci comprend donc la constitution de nos jeux d'apprentissage, les nettoyages des données brutes, compré-

hension des features, vectorisation du texte en données numériques, suppression des "outliers", valeurs manquantes" etc. Puis le développement des modèles de ML qui comprend cross validation, grid search...ect, a pris environ 10-20% et environ 30-40% pour les modèle de deep learning qui ont demandé beaucoup de travail. Enfin le dernier tiers a été consacré à l'évaluation et l'interprétation des résultats, le test sur les différentes tâches de classifications à effectuer.

Nous avons pu mettre à profit durant ce projet les connaissances acquises au cours des différentes UE, celle de machine Learning bien sûr mais également d'autres telles que Probabilités et Statistiques pour la mesure de nos résultats ou encore en Langage Naturel et Web sémantique, notamment pour la partie pré-processing des données et sémantique.

Nous avons travaillé entièrement sous google colab, qui propose un environnement déjà installé et l'accès à du calcul sur GPU. Afin de mener à bien le projet, nous nous sommes inspirés en partie sur les notebooks des données textuelles (ingénierie des données textuelles et classification de celles-ci) ainsi que de nos cours et de ressources extérieurs, en particulier sur le NLP.

Pour la partie gestion de projet nous avons utilisé certains outils comme Trello, pour la gestion des tâches, ce qu'il reste à faire, ce qui a été

1. 3

fait, etc ainsi que Miro pour ses nombreux templates facilitant la conduite et l'organisation de ce projet.

2 Prétraitements

2.1 Séparation des jeux

Nous avons fais le choix de diviser en 3 notre jeu de données c'est à dire réserver 10% du jeu de donné pour la partie test, puis sur les 90% restant réserver entre 20% et 30% pour le jeu de validation. Ceci est en effet une très bonne pratique en machine learning puisque l'on a tendance à selectionner un modèle en fonction des resultat prédit par celui ci sur le jeu de validation. On choisit alors souvent le classifieur avec les paramètres qui donne le meilleur résultat, mais en faisant cela nous ne garantissons absolument pas d'avoir choisis le modèle qui généralisera le mieux et encore moins d'obtenir d'aussi bon résultat sur le jeu test qui lui est différent su jeu de validation. Le jeu de test a donc rôle bien différent des autres, puisqu'il ne sert pas à ajuster nos choix de classifieur et de paramètres mais bien d'évaluer les performances "normales" de notre classifieur.

Nous avons commencé dans un premier temps par traiter un petit jeu de données d'environ 100 à 150 individus par classe. Toutefois cette méthode cache un gros défaut. Nous avons pus observer des résultats très variables en terme de f1-score en fonction du jeu, ces différences tendent à diparaître au

fur à mesure que la taille du jeu augmente et fournisse des modèle de qualité plus solide. Nous reviendrons plus en détail sur ce point par la suite. Pour le reste du projet nous avons constitués un jeu plus conséquent environ 4500 individus par classes.

2.2 Equilibrage des classes - DownSampling

Nous avons pu observer des écarts de f1-score très notable sur les classes moins représentées d'où l'importance de constituer un jeu de données équilibrées. Nous avons opté pour une approche de down-sampling pour les 3 tâches de classification. La classe la moins représentée était celle des "True" environ 4500 individus. Par cette approche on a donc sélectionné aléatoirement 4500 individus dans la(ess) classe(s) majoritaire(s). Pour chaque tâche notre algorithme s'assure de constituer un jeu de données parfaitement équilibré au début du notebook.

2.3 Ingénierie textuelle

Les algorithmes de machine Learning se basent globalement sur des fonctions mathématiques qui prennent en entrée des données numériques. Il faut donc trouver une manière de "mathématiser" nos textes tout en conservant au maximum le sens du texte même si la sémantique est très difficile à conserver par des opérations de ce type. Une approche classique consiste

à calculer le tfidf qui transforme un texte d'un corpus en un vecteur où chaque entrée est un nombre entre 0 et 1 correspondant à un mot du corpus. Nous avons ensuite plusieurs approches possibles, l'approche N-Gram qui consiste à considérer non plus un seul mot dans la vectorisation mais des paquets de N mots. Cette méthode se voulait prometteuse mais a en fait elle s'est révélée plus médiocres encore. Ceci est d'ailleurs confirmé dans de nombreux papiers de recherche qui stipule que les meilleurs résultats sont observés pour $N=1$.²

Mais avant cela, il nous a fallu faire un bon nombre de pré-processing sur nos données brutes. Nous avons choisi des approches classiques en NLP, telles que la tokénisation des phrases, la conversion en lettres minuscules de notre corpus, la suppression des stop words (mots très communs et qui n'apportent rien lors de la phase d'apprentissage), nous avons supprimé les nombres et utiliser la méthode de stemming, donc de racinisation des mots. Concernant cette dernière, nous avons ensuite testé son analogue, la lemmatisation mais n'avons constaté aucun changement significatif sur la performance de nos modèles. Nous avons utilisé diverses combinaisons de prétraitements pour ensuite tester chacun d'eux sur un classifieur donnée.

Nous avons utiliser NLTK comme librairie pour principalement le pré-

traitement des données mais aussi pour de la visualisation avec word-cloud par exemple. Spacy quant à elle aura été utilisé uniquement pour les relations entre catégories grammaticales (POS) de différents mots du corpus.

2.4 Visualisation des données

A l'aide d'une analyse en composante principale on peut observer de manière visuelle l'espacement ou pas de nos classes. La magie de l'algèbre linéaire nous permet de projeter nos jeux de données dans un espace à 2 ou 3 dimensions sous sommes donc capable assez vite de voir si nos classes sont bien ou peu séparés. Comme on peut le voir ci dessous on observe un nuage de points où les classes sont très mélangés.

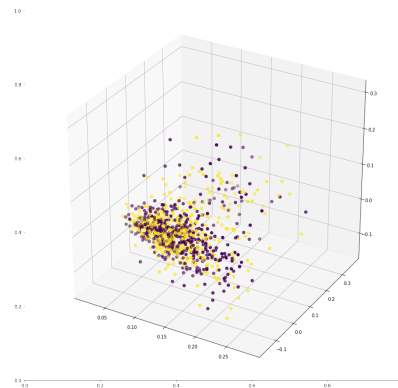


FIGURE 1 – Projection sur les 3 principales composantes du jeu True vs False

2. https://www.researchgate.net/publication/356618600_Performance_Study_of_N-grams_in_the_Analysis_of_Sentiments

3 Développement des modèles et Evaluation

3.1 Validation Croisé

Nous avons décidé de comparer 8 classifieurs assez classiques. Après la sélection de la meilleure combinaison pour le prétraitements nous avons effectuer une validation croisée K-fold de chaque modèle ce qui permet de constituer une évaluation moins "biaisée" de chacun des modèles. Nous avons utilisé la validation croisée K-fold pour la valeur de $K = 5$. Le but est de sélectionner 2-3 des meilleurs classifieurs pour effectuer ensuite une recherche des meilleures paramètres.

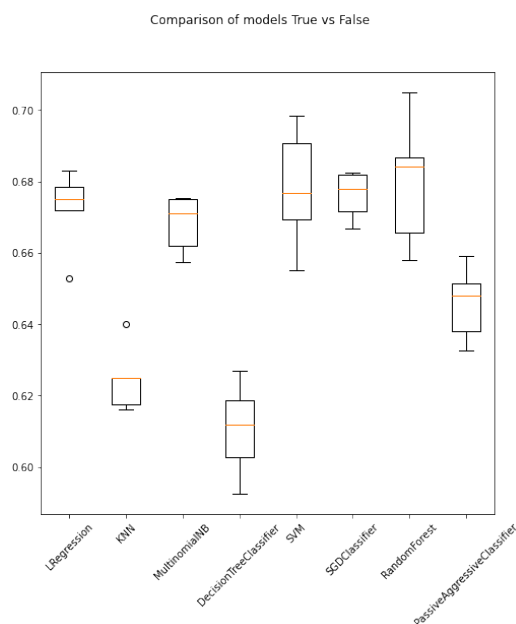


FIGURE 2 – Comparaison des modèles pour T vs F

3.2 Grid Search

Les classifieurs retenus comme plus performant sont le SVC, la LogisticRegression et le MultinomialNB qui pour ce dernier à l'avantage d'être rapide par rapport au SVC qui peut être atrocement lent. Nous effectuons comme d'habitude une recherche de la meilleure combinaison de prétraitements à appliquer sur le texte puis ensuite on définit un pipeline. Le pipeline applique les prétraitements sélectionnés, vectorise et normalise avec le TFIDF puis effectue le "grid search" qui permet de tester de nombreux paramètres d'un classifieur. Les meilleurs paramètres sont choisis par rapport à l'accuracy sur le set de validation. Un des défauts de cette méthode dans notre cas est que l'on avait très souvent tendance à obtenir un modèle qui avait bien appris et était sélectionné pour être bien précis sur le set de validation, mais du coup le set test étant a priori complètement différent de la validation celui ci performait beaucoup moins bien sur le set de test..

3.3 Evaluation sur le jeu Test

Pour évaluer le modèle on calcule ensuite une fonction de perte sur le jeu d'entraînement et de validation en augmentant petit à petit la taille du jeu. On peut ainsi voir comment se comporte le modèle en fonction de la taille du jeu. On observe en général que plus l'on augmente la taille plus

on augmente la stabilité des performances obtenues c'est ce qui nous à poussé comme expliqué au début à choisir tout de suite un jeu de données assez gros. L'erreur sur le set de validation décroît aussi légèrement.

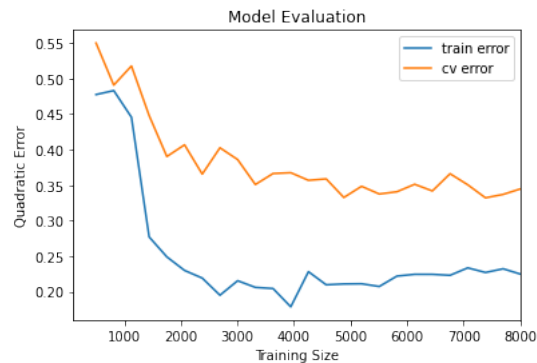


FIGURE 3 – Courbe d'apprentissage pour SVC

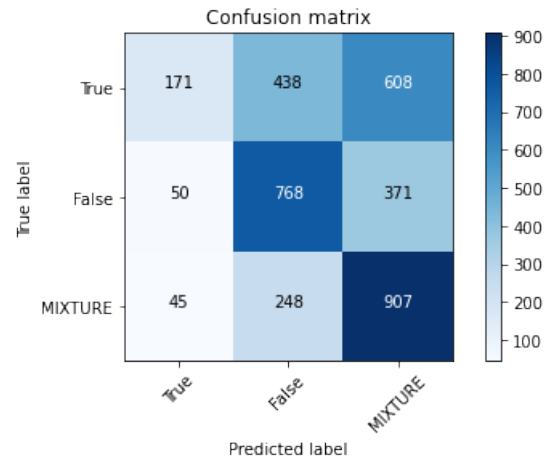


FIGURE 4 – Matrice de confusion

On effectue aussi une évaluation des performances sur le jeu de test et très souvent les résultats sont moins bons car comme dis plus haut le grid first search selectionne un modèle qui est souvent trop "biaisé" par rapport à la "réalité". Ci dessous on peut voir la matrice de confusion obtenu pour l'algorithme RandomForest sur le jeu de test sur la classif T vs F vs M.

4 Approches Deep Learning

Développer des modèles de deep learning nous a pris beaucoup plus de temps que l'implémentation des modèles de machine learning. Nous avons utilisé 2 bibliothèques différentes PyTorch et TensorFlow pour l'implémentation des modèles de deep learning. Toutefois les résultats ont été sensiblement meilleures de l'ordre de 10% supplémentaire avec BERT sur la classif T vs F par exemple.

4.1 LSTM

On peut voir ci-dessous la fonction de perte sur le jeu de validation et le jeu d'entraînement. On observe clairement ici du surapprentissage, puisque l'écart entre les 2 courbes augmente avec les epochs.

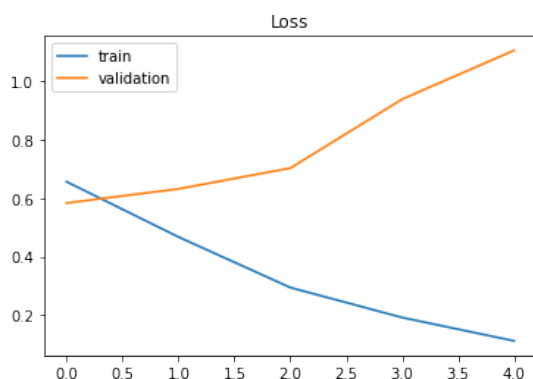


FIGURE 5 – Loss Plot for training and validation LSTM

Pour palier à ce problème nous avons essayé de "decomplexifier" notre modèle c'est à dire réduire le nombre et la complexité des couches intermédiaires et aussi d'augmenter le dropout³. Cela nous a permis d'améliorer un peu les résultats mais sans véritablement sensiblement meilleure. Les résultats obtenus ont tout de même supérieur à des classif classiques.

4.2 BERT

Le réseau BERT à lui offert des résultats bien meilleures toutes tâches confondus. Nous sommes partis donc du réseau préentraîné pour le "fine tuné" ensuite sur notre propre jeu de donnée. Nous avons ainsi réussi à obtenir 63% de f1-score pour la classif T vs F vs M ,80% pour T vs F et 75% pour la dernière.

```
Epoch 4
Training loss: 0.32715118148324707
Validation loss: 1.3467987546607407
F1 Score (Weighted): 0.7606349736855502
```

FIGURE 6 – Meilleurs résultats pour BERT

3. désactivation temporaire de certains neurones "intermediares" durant l'apprentissage

5 Autres Approches

partie eval des features :

Nous avons évalué nos résultats sous un autre angle : celui des features. En ajoutant certaines d'entre elles ou en retirant d'autres. On a effectué des dizaines de combinaison pour les catégories de classes différentes True,False, True,FalseMixture etc. On en a conclu qu'on ne peut pas faire d'observation notable sur l'accuracy de nos modèles qui sont de l'ordre de d'1 à 4% au moyenne pour chacune de ces catégories, ceci est donc dû au jeux de données. Nous avons trouvé un maximum de 6% pour l'algorithme Naive Bayes qui semble donc donner de meilleurs résultats avec des features plus nombreuses pour True,False. Nous nous sommes donc penchés sur d'autres métriques : le f1 score et le recall. On observe également peu de changement sur le rappel et le f-1 score en fonction des features choisies : de l'ordre de 4 à 5 % pour un même algorithme On constate aussi que sélectionner beaucoup de features n'est pas forcément corrélé à de meilleurs résultats, de même, l'ajout par exemple de Keyword dans ces dernières s'est révélée améliorative pour le rappel et le f1-score

lise des fréquences d'apparitions) au lieu d'avoir des vecteurs à n dimensions où n est le nombres total de mots différents dans le corpus nous avons des vecteurs à 3 dimensions pour un "claim". Avec une régression logistique comme classifieur nous avons pu obtenir des résultats très bons pour de petits jeux de données autour de 85-90 % mais se dégradant très vite à mesure que le jeu de données et donc le corpus augmentait.

Voici les scores obtenus par exemple pour un jeu de données de 300 claims.

5.1 Logistic Regression

Nous avons effectué un preprocessing bien différent du tfidf qui est bien détaillé dans le repository git uniquement pour la tâche T vs F (donc regarder le fichier fakeNewsTvsF.ipynb). Avec cette approche différente (qui uti-

6 Conclusion

Ce projet nous a posé les bases du Machine Learning et plus particulièrement sur des données textuelles. On a pu explorer les principaux algorithmes utilisés sur ce type de problème. La conduite d'un projet de Machine Learning nous été plutôt floue au départ mais nous avons réussi à définir les différentes étapes et l'organisation de celui-ci. On a pu de même poser les concepts théoriques appris en Probabilités et Statistiques sur des problèmes réels et pratiques. Enfin, il nous a ouvert à un champ prometteur qui est celui du traitement automatique du langage.

6.1 Perspectives d'amélioration

On pourrait pousser l'analyse à l'étude des faux positifs et faux négatifs pour essayer de dégager certaines propriétés qui leur sont propres. Par exemple nous avons remarqué que beaucoup des faux positifs dans la classif T vs F (c'est-à-dire pour nous les fake(false) classé non-fake(true)) comportaient beaucoup de mots entiers en majuscules. On pourrait par exemple rajouter une règle qui classe ce type de texte directement en fake. On pourrait ainsi définir une base de règles qui viendrait encadrer l'apprentissage et caractériser ses valeurs d'incertitudes pour le classifieur. Établir des règles prioritaire sur la la classif peut permettre de réduire drastiquement le nombre de faux positifs et faux négatifs.