UNIVERSITÉ DE MONTRÉAL

Combinaison de deux classifieurs appliquée à une tache de classification de commentaires Reddit

Dos Santos Luis, Kletz David

October 26, 2020

IFT 6390 - Fondements de l'apprentissage machine

Nom de l'équipe : Top 1 nous voilà
Membres de l'équipe :

- Luis Dos Santos - 20164783
- David Kletz - 20152270

Contents

1 INTRODUCTION

Ce projet d'apprentissage automatique porte sur la classification de commentaires Reddit. Nous avions 70.000 commentaires de test et 30.000 d'entrainement, avec des caractéritiques différentes (taille des commentaires, fautes d'orthographe, abréviations...), et les classes indiquant le sujet dans lequel le commentaire est classé. Ce projet se décline en deux phases:

1) Beat the baselines : Nous avons codé un Naïve Bayes en s'inspirant du lab n°3. La distribution utilisée pour calculer les probabilités qu'un commentaire appartienne à chacune des 20 classes s'appuie sur celle qu'un mot du commentaire fasse partie de la classe (d'après la fréquence observée de ce mot à l'entrainement). Le meilleur score atteint par notre Naïve Bayes est **55,5%**.

2) Best accuracy : Le but ici était d'avoir la meilleure accuracy. Nous avons essayé plusieurs classifieurs en cherchant à optimiser leurs hyperparametres. Puis nous nous sommes interessés à la combinaison des meilleurs d'entre eux. Enfin nous avons investigué des techniques de pré-traitement (notamment la normalisation des mots). Le meilleur score fut par combinaison de deux classifieurs : notre Naïve Bayes de la phase 1 et SVM (sans pré-traitement) pour atteindre une accuracy de **59,8%**.

2 FEATURE DESIGN

Les inputs, présentés sous forme de blogs ont été traités afin d'être exploitables par nos algorithmes. Pour ce faire, nous les avons mis sous forme de bag of words.

2.1 PRE-PROCESSING METHODS

Nos méthodes de pré-processing ont été séparées en deux étapes : d'abord, nous nous sommes attachés à transformer les commentaires redit en bag of words, puis nous avons essayé d'appliquer des methodes de traitement du contenu aux commentaires eux-mêmes.
- Création des bag of words : nous avons utilisé la classe sklearn.feature_extraction.text.TfidfVectorizer, afin de vectoriser les commentaires reddit. Puis la méthode ".toarray()" nous a permis d'obtenir des bag_of_words.
- Méthodes de pré-processing : Deux ajouts ont été tentés : dans un premier temps, nous avons essayé de remplacer toutes les majuscules des commentaires par des minuscules, afin de normaliser leur vectorisation (i.e. les mots "Savoir" et "savoir" sont vectorisés comme un seul mot).Puis nous avons essayé de normaliser les mots. Dans un autre cours, nous avions eu à créer un algorithme remplaçant les mots écrits de façon particulière par leur écriture normale : i.e remplacer "HAHAHAHAHAH" par "haha", ou l'expression "I haaaaaaaaate it" par "I hate it", nous avons donc décidé de le tester ici.

2.2 Design et sélection des features

Afin d'aider à la sélection des features, nous avons essayé d'utiliser les modules de selection de SkLearn : sklearn.feature_selection.
De même, nous avons essayé de supprimer les mots outils (ou stop words), mots très présents, mais ne permettant pas d'aider à la classification. Pour ce faire, nous nous sommes appuyés sur la liste de stop-words anglais distribuée par NLTK : nltk.corpus.stopwords

3 Algorithms and methodology

Afin d'augmenter la précision de nos classifieurs, nous avons décidé de nous appuyer sur trois classifieurs. Pour chacun d'entre eux, nous avons cherché à optimiser ses paramètres, puis nous nous sommes intéressés à la possibilité de combiner ces classifieurs afin de maximiser les capacités de chacun d'entre eux, pour améliorer nos résultats.

3.1 Choix des classifieurs

Nous avons choisi trois classifieurs : Naives Bayes codé pour la première partie de la compétition, et deux classifieurs de la librairie SkLearn : Logistic Regression (sklearn.linear_model.LogisticRegression) et SVM (sklearn.linear_model.SGDClassifier).
Ces trois algorithmes nous ont parus complémentaires : Naive Bayes n'est pas un classifieur linéaire (alors que le SVM sélectionné correspond une forme linéaire). Et la regression logistique et SVM ont des frontieres de decision diférentes..
De plus, dans une logique de classification non-binaire, nous avons ajouté une méthode (aux classifieurs SVM et regression logistique) de comparaison "un contre tous", grâce à la méthode Sklearn : sklearn.multiclass.OneVsRestClassifier.

3.2 Methodologie

- Création des sets d'entrainement et validation : Pour tester nos méthodes, nous avons séparé nos "bag-of-words" entre deux sets respectivement d'entraînement et de validation. Nous avons fixé à 10000 commentaires la taille du set de validation, à chaque fois sélectionnés de manière aléatoire.
- Naive Bayes : Nos classifieurs Naives Bayes calculaient leur répartition d'après la fréquence observée ; c'est à dire que $P(c|m) = n_m c$ avec c l'évenement : le commentaire appartient à la classe c, m le mot donné, et $n_m c$ le nombre d'occurences du mot dans l'ensembles des commentaires du set d'entrainement.Afin d'optimiser les performances de comparaison des Naives Bayes entre les différentes classes, nous avons décidé de ramener les probabilités à une fréquence $f \in [0,1]$. Pour ce faire, nous avons décidé de choisir : $\hat{P}(c|m) = n_{mc}/n_m$, avec $n_m c$ la fréquence du mot m dans la classe c et n_m la fréquence du mot dans l'ensemble du set d'entrainement.

- Autres classifieurs : Pour les autres classifieurs, à savoir SVM, Random Forest, Logistic

Régression et MultiLayer Perceptron, nous avons cherché à optimiser leurs hyperparamètres, en comparant leurs performances en les entrainant et testant respectivement sur les mêmes sets d'entraînement et de validation, et en faisant varier la valeur des hyperparamètres.

-Combinaison d'algorithmes : Afin de valoriser les particularités de nos classifieurs, nous avons également essayé de normaliser les probabilités d'appartenance à une classe proposées par chaque classifieur : ce vecteur de taille 20 contient des probabilités dont le total n'est pas forcément comparable d'un classifieur à un autre. Ainsi, nous divisons chaque probabilité par la somme de probabilité, afin que le total soit bien de 1. Puis, nous additionnons les probabilités des trois classifieurs pour chaque classe, par exemple:

$$[p_{SVM,0}, p_{SVM,1}, ...p_{SVM,19}] + [p_{NB,0}, p_{NB,1}, ...p_{NB,19}] = [p_0, p_1, ...p_{19}]$$

avec $p_{a,b}$: la probabilité normalisé selon le classifieur a, que le commentaire soit de la classe b (classifieurs : NB = Naive Bayes, lr = Logistic Regression)

4 Résultats

4.1 Pré-traitements

L'utilisation des pré-traitements s'est avérée très décevante: l'utilisation des modules de sélection de features de Kaggle a fait chuter nos résultats : en moyenne, et sur notre meilleur classifieur , nous sommes passés de 59.54% à 11.3%. Et le remplacement des majuscules par des minuscules a eu un effet neutre : conservation de la même moyenne de précision.
La suppression des mots outils a entrainé une perte de 0.75% de précision.
De même, l'utilisation de la normalisation du texte des commentaires, détaillée dans la section 2.1, a légèrement diminué la précision de notre algorithme, de 59.54% à 58.86%. Tous les caractères des smileys ou des sites web ont été remplacés par SMILEY et WEBSITE. Nous avons récupéré des statistiques concernant le nombre de modifications effectuées pour chaque catégorie (cf. Tableau 4.1). Ce tableau démontre que de nombreuses modifications ont été apportées.

Table 4.1: Nombre de modifications faite pour chaque catégorie de problèmes traités

Modification traitée	Nombre de cas rencontrés
Lettres répétitives	120.042
Ponctuation répétitive	23.299
Sites web	4.113
Smileys	1.492

La baisse d'accuracy suite à ce pré-traitement peut peut-être s'expliquer par le fait que le texte a été trop uniformisé, et donc que de l'information a été perdue.

4.2 Choix des hyperparamètres

Dans une perspective de trouver les meilleurs classifieurs afin d'avoir une idée précise de quelles combinaisons pourraient résulter en une meilleure accuracy, il nous a fallu optimiser les hyperparamètres de chacun d'eux.
Naïve Bayes n'a pas d'hyperparamètre. Nous prenons dans cette partie l'exemple de SVM car il fait partie de la meilleure combinaison que nous ayons obtenue. La Figure 4.1 illustre l'accuracy obtenue en fonction de la loss function utilisée pour SVM, sur le même set d'entraînement et de validation.
On voit que le choix de celle-ci a une très grosse influence, avec quasiment 10% d'écart entre le meilleur et le moins bon. Notre choix s'est ainsi porté sur *modified_huber*.

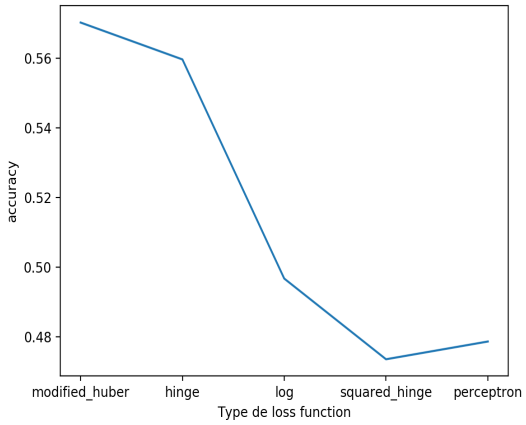

Figure 4.1: Accuracy de SVM en fonction de différentes loss functions

4.3 Comparaison des classifieurs

De nombreux classifieurs ont été testés. La Tableau 4.2 récapitule les meilleures accuracy que nous ayons réussi à avoir avec chacun d'entre eux, en optimisant les hyperparamètres.

On voit que SVM et Naïve Bayes sont ceux qui ont la meilleure accuracy.
MLP était optimisé pour 15 couches de 25 perceptrons. Au delà, le classifieur était surentraîné et l'accuracy chutait.

4.4 Combinaison des classifieurs

En analysant les prédictions des classifieurs, nous nous sommes aperçus que leurs erreurs n'étaient pas toujours réalisées sur les mêmes commentaires. Ainsi, nous avons pensé qu'il serait intéressant de combiner les classifieurs afin d'optimiser la précision des prédictions. Pour un commentaire donné, nous obtenons un vecteur indiquant la probabilité pour chaque classe que le commentaire fasse partie de celle-ci; deux axes ont été explorés :

Table 4.2: Accuracy obtenue avec les différents classifieurs

Classifieur	Meilleure accuracy obtenue
Naïve Bayes	57,4%
SVM	56,6%
Logistic Régression one vs all	54,2%
Logistic Régression one vs one	51,4%
MultiLayer Perceptron	50,8%
Random Forest	42,8%
Logistic Régression boosté (avec AdaBoost)	40,8%

1) en classant ces classes par ordre de probabilité, quelles sont les classes les plus probables partagées par les classifieurs?
2) l'écart de probabilités entre la première et la deuxièmes classe la plus probables est-il conséquent?

Nous avons ainsi determiné que les classifieurs Naive Bayes et SVM (en conditions One Vs All) étaitent assez complémentaires : lorsque les deux ont la même prédiction , dans 74.9% des cas, ils prédisent la même catégorie. De même, lorsque la différence de probabilité entre la première et deuxième classe est supérieure à un seuil, ils obtiennent un fort taux de réussite : 96% pour SVM , avec un seuil à 0.5, et 98.6% pour Naive Bayes avec un seuil à 0.05 (les deux seuils n'étant pas sur une même échelle). Ceci nous a poussés à essayer de les combiner afin que les erreurs de l'un soit compensées par l'autre : en ramenant leur probabilité à une échelle de 1, nous pouvons les additionner. En expérimentant cette méthode, nous nous sommes ensuite aperçus qu'il était possible d'optimiser ces résultats; en effet, en les sommant nous donnions autant de poids à l'un qu'à l'autre, or Naive Bayes obtenant des résultats sensiblement meilleurs, il serait judicieux de lui donner davantage de poids. En faisant varier ce nouvel hyperparamètre (coefficient de poids accordé à Naive Bayes) nous avons pu trouver une valeur optimale.

La figure 4.2 met en évidence la présence d'une valeur optimale de poids accordé à Naive Bayes par rapport à SVM. En réitérant plusieurs fois l'expérience, nous avons trouvé des valeurs optimales comprises entre 3 et 5, et avons choisi un coefficient de 4 : avec celui-ci nous observons une augmentation de 2.3%.

5 Discussion

Un des avantages de notre approche est que nous avons testé de nombreuses méthodes pour améliorer notre accuracy (boosting, pré-traitement, combinaisons de classifieurs, optimisation des classifieurs). La combinaison de classifieur a notamment été pensé afin de combiner un classifieur non-linéaire (notre Naïve Bayes), avec un classifieur linéaire, (ici, SVM), ce qui est d'ailleurs la combinaison qui nous a permit d'avoir le meilleur résultat.

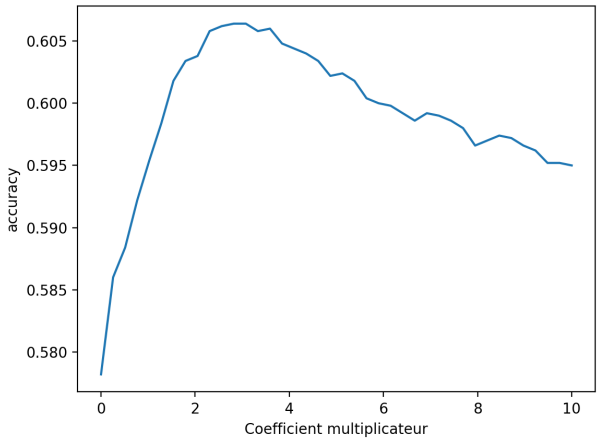

Figure 4.2: Accuracy de notre systeme de classifieurs en fonction du coefficient de poids donné à Naive Bayes

Nous avons également étudié les cas où les deux classifieurs avaient des prédictions différentes et lequel des deux était le plus apte à avoir raison.
Il aurait cependant été plus judicieux de commencer directement par tester différentes combinaisons puisque comme l'a mentionné un TA durant un lab, c'est souvent ce qui permet d'améliorer l'accuracy. De plus, certaines combinaisons de classifieurs n'ont pas été testées, comme celles impliquant Random Forest par exemple. Aussi, nous ré-utilisons le pré-processing que nous avions codé pour un autre cours, sans adapter ce traitement spécifiquement aux commentaires du train set. Les 'fautes' présentes dans les commentaires sont cependant très semblables.
Nous n'avons pas fait de statistiques sur les commentaires du test set, ainsi il existe peut être de petites disparités entre le set d'entraînement et de test.

6 Contribution des membres de l'équipe

Le travail a constamment été réalisé en collaboration entre les deux membres de l'équipe. Pour les Naive Bayes, nous avons essayé d'obtenir les résultats attendus. Dès lors que l'un d'entre nous avait un résultat, il partageait son code avec l'autre.
Pour la deuxième partie, nous avons partagé les taches: Luis Dos Santos s'est occupé du pré-processing (normalisation des mots), des tests (gestion des hyperparamètres) de la régression logistique et de MLP, et a optimisé les performances du Naïve Bayes. David Kletz s'est interessé aux tests de nouveaux classifieurs, de leur optimisation (hyperparamètres), et enfin a créé et optimisé la combinaison de Naïve Bayes et SVM.
Le rapport a été écrit par les deux membres. Nous nous engageons sur le fait que le travail qui a été présenté ici, est bien celui des auteurs. We hereby state that all the work presented in this report is that of the authors.