Discours des Présidents

Rapport

Valentin Fontanger

&

Julie Lascare

Master 1 DAC, Traitement automatique de la langue

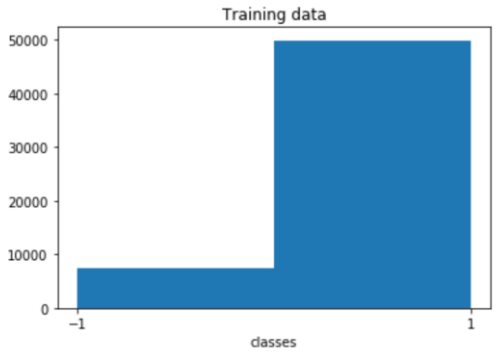
02/03/2022

1. Introduction

L'objet de ce projet est d'utiliser le Machine Learning afin de détecter quel candidat, Mitterrand ou Chirac, est l'auteur d'un passage de discours. Nous avons utilisé le jeu de données ***présidents,*** comportant 57413 passages de discours des deux présidents. L'objectif principal est d'évaluer plusieurs modèles de classification, reposant sur la méthodologie ***BagOfWord (BOW).*** Les BagOfWords (sac de mots en français), permettent de constituer une matrice dont les lignes sont les documents et les colonnes les différents mots du vocabulaire. Pour chaque vecteur ligne, les coefficients traduise une information sur la présence d'un mot dans le document. Nous avons utilisé les occurrences des mots dans chaque document.

2. Problématique et Algorithmes

L'un des principaux enjeux de ce dataset est son ***important déséquilibre***. La classe majoritaire est Chirac. On observe ici ce déséquilibre :



Distribution des classes (-1 : Mitterrand, 1 : Chirac)

49890 Chirac et 7523 Chirac

Il est donc facile pour nos classifieurs de prédire aveuglément tous les échantillons comme étant de la classe majoritaire, et d'obtenir une ***accuracy en validation croisée*** supérieure à 84 %. Nous nous sommes donc intéressés aux mesures se focalisant sur la classe minoritaire. Ces mesures sont le ***rappel***, la ***capacité*** du classifieur à ***identifier correctement les Mitterrands***, et la ***précision***, sa capacité à ***ne pas prédire des échantillons Chiracs en tant que Mitterrands***. Voici les formules de ces deux mesures :

Précision : tp / (tp+fp) (où tp désigne true positive)

Rappel : tp/ (tp + fn)

Le ***f1 score est une moyenne harmonique entre la précision et le rappel*** et a servi de mesure principale dans notre étude.

Par conséquent, l'objectif est de trouver un classifieur et de l'optimiser, dans le but de maximiser le f1 score Mitterrand, sans altérer les performances sur les échantillons Chirac.

Nous avons utilisé les algorithmes suivants : Regression Logistique, Support Vector Machine linéaire, naive Bayes, perceptron. Ces algorithmes ont fait leurs preuves pour de nombreux problèmes de classifications

3. Expériences et évaluations

3.1. Méthodologie

Nous avons souhaité nous concentrer sur les algorithmes maximisant le f1 score Mitterrand pour une configuration naïve d'un BOW (feature = 50000, min_df = 0 , max_df = 1.0). L'idée était de déterminer les ***hyperparamètres*** du BOW permettant d'obtenir les meilleures performances, puis de déterminer les ***hyperparamètres du modèle***.

En plus d'être déséquilibré, le dataset fourni des bloques de discours pour chaque candidat. Nous avons décidé de ne pas mélanger les données, afin de profiter de cet aspect.

Dès lors, nous avons évalué les modèles optimisés sur un échantillon de test, pour finalement lisser les prédictions afin d'obtenir des blocs. Ce lissage implique de nouveaux hyperparamètres.

3.2 Résultats

Parmi les algorithmes entraînés naïvement, le Naive Bayes a été plus performant concernant le f1 score Mitterrand en validation croisée (0.51). En optimisant le CountVectorizer, nous avons obtenu un f1 score Mitterrand de 0.48 sur les données de tests pour les paramètres max_features=50000, min_df=1, max_df=0.65, ngram=(1,2) :

F1 Mittérand TEST : 0.4815039417828987

	precision	recall	f1-score	support
-1	0.68	0.37	0.48	2135
1	0.87	0.96	0.91	9348
accuracy			0.85	11483
macro avg	0.78	0.67	0.70	11483
weighted avg	0.84	0.85	0.83	11483

Performances sur le jeu de test avant gridsearch

Après optimisation des hyperparamètres du modèle, nous obtenons un f1 score de 0.51 :

F1 Mittérand TEST : 0.5105124835742444

	precision	recall	f1-score	support
-1	0.67	0.41	0.51	1881
1	0.89	0.96	0.93	9602
accuracy			0.87	11483
macro avg	0.78	0.69	0.72	11483
weighted avg	0.86	0.87	0.86	11483

Performances sur le jeu de test après gridsearch

Nous nous sommes intéressés à différentes méthodes de lissages des prédictions. Nous avons établi deux méthodes de lissages, que nous avons combiné afin d'obtenir les meilleures performances. La première a consisté à remplacer toutes les valeurs ayant pour voisin de gauche et de droite une prédiction de classe opposée. La seconde méthode a permis de lisser des groupes de prédictions par fenêtre de tailles 9, en se basant sur la fréquence de prédictions Mitterrand et Chirac dans cette fenêtre. Les hyperparamètres

impliqué dans ce processus de lissage sont la taille de la fenêtre ainsi que le threshold de conversion des prédictions.

Après avoir optimisé ces deux méthodes de lissages, nous obtenons les résultats suivants sur l'ensemble de test :

Prédictions KERNEL_SIZE=9 | TH=0.55 | MIT_WEIGHT=22.854638641349908 | STRATEGY=mean

	precision	recall	f1-score	support
-1	0.65	0.75	0.70	1002
1	0.98	0.96	0.97	10481
accuracy			0.94	11483
macro avg	0.81	0.86	0.83	11483
weighted avg	0.95	0.94	0.95	11483

Performances après optimisation du vectorizer, du modèle, et du lissage, sur l'ensemble de test

Nous obtenons un f1 score final de 0.70.

Après soumission des prédictions sur ***l'ensemble officiel de test***, nous obtenons un ***f1 score Mitterrand supérieur à 0.74***.

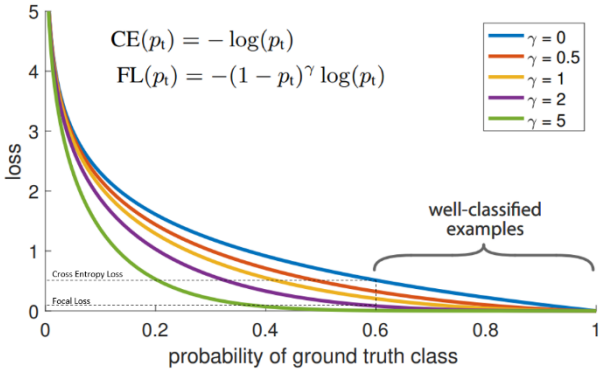
3.2 Discussion

Nos expériences ont été coûteuses en temps, dû à l'optimisation intensive des hyperparamètres. Après de nombreux faux départs (nombre d'hyperparamètres trop élevé, une seule fonction pour optimiser tous les modèles et leurs hyperparamètres d'un coup), nous avons réussi à établir une méthodologie basée sur les performances des modèles sans optimisation. Bien que ceux-ci ne figurent pas dans le rapport, nous avons entraîné des modèles de régression logistique et SVM, qui proposaient des résultats intéressants. Nous avons souhaité exposer un travail plus structuré et ordonné, celui portant sur le Naïve Bayes.

Nous retenons les conclusions suivantes sur la démarche à suivre : Procéder par campagnes divisées en plusieurs étapes d'optimisation, sauvegarder les résultats afin de les analyser, réduire le nombre d'hyperparamètres dans le but de converger petit à petit vers les valeurs optimales. Nous avons appliqué ces principes dans le projet ***movie review***.

4. Travaux futurs

Le déséquilibre des datasets est un problème récurrent en Machine Learning. Nous souhaitons explorer de nouvelles pistes afin de réduire son impact sur les performances. La ***focal loss*** (***Facebook, 2018, Retinanet : focal loss for dense object detection***) est une fonction de coût permettant de se focaliser sur les échantillons compliqués à classifier, en poussant la perte vers le bas lorsque la prédiction est bonne et sûre (0.90 ou 0.80) :



Focal loss (Focal Loss for Dense Object Detection)

https://arxiv.org/abs/1708.02002

La librairie ***imblearn***, propose de nombreuses méthodes de sampling (over/under) afin de palier au problème du déséquilibre. Nous souhaitons étudier l'impact de ces méthodes sur les performances de notre modèle.