FACULTÉ DE SCIENCES ET INGÉNIERIE

PROJET

MU4IN813 - RITAL

Traitement Automatique de la Langue

Modèles sac de mots pour la classification de documents

Auteurs:
Mathis Koroglu 3711799
Karima Sadykova 28610100

Mars 2023

Sommaire

Introduction

Ce projet a pour but de nous amener à être capable de concevoir, entraîner, évaluer et utiliser des modèles de traitement du langage sur deux jeux de données:

- des extraits de dialogues menés entre les ex-présidents Chirac et Mitterrand. La tâche correspond à prédire, d'après une phrase, si celle-ci a été prononcée par l'un ou l'autre de ces hommes politiques. Les données sont déséquilibrées, 86.9% des exemples sont labellisés Chirac et seulement 13.10% le sont pour Mitterand.

- des extraits de commentaires en anglais de films issus de la base de données IMDB classés selon la polarité du sentiment exprimé en deux classes négative et positive. La tâche est d'analyser et prédire la polarité de nouveaux commentaires. Les données sont équilibrées.

Pour ce travail, nous utiliserons la représentation sac de mots pour extraire des vecteurs des chaînes de charactères liées aux données textuelles.

Le code employé sur ce projet est très similaire sur ces deux tâches. Nous évaluerons différentes stratégies de pré-traitement, de normalisation, de réduction de dimension ainsi que des modèles de classification.

La particularité de notre implémentaton est de ne pas fixer manuellement ces stratégies dans la chaîne de traitement, mais plutôt de considérer cela comme un tout dans un modèle dont ces stratégies sont des hyper-paramètres et où les données d'entrées sont laissées brutes.

Les avantages de cette aproche sont nombreuses. Ainsi, il devient beaucoup plus simple, avec des techniques d'optimisation d'hyper-paramètres non étudiées ici, de trouver un modèle maximisant une métrique adaptée à chacune des tâches ci-dessus. Par ailleurs, cela simplifie grandement notre étude.

1 Pré-traitements du texte

Dans notre implémentation, parmi les étapes de notre chaîne de traitements, on peut noter les étapes suivantes :

- Vectorizer
- Transformer en pondération TF-IDF (facultatif)
- Réduction de dimension (facultatif)
- Scaler
- Classifier

La première partie 'Vectorizer' consite à extraire des sac de mots depuis le texte brut. Elle est elle même constituée des étapes suivantes où les pré-traitements étudiés sont également indiqués :

- pre-process
  - la mise en minuscule
  - enlever les accents
  - enlever la ponctuation
  - enlever les nombres
  - ajouter des marqueurs pour les mots en majuscule
- tokenizer (extractions des mots du texte)
  - avec et sans stemming
  - avec et sans lemmatization
- enlever les stopwords
  - d'après une liste
  - d'après des seuils min et max
- fixer ou non une taille maximum de vocabulaire : (2000, 5000, 10000)
- considérer ou non des n-grammes de mots

  On étudie un modèle où l'on ne considère plus uniquement des mots comme unité du vocabulaire, mais aussi des tuples de n mots.

2 Transformation suivantes des sac de mots

2.1 Variantes

Les approches suivantes et les traitements listés précedemment peuvent être combinées entre-eux.

Exemples: ponderation TF-IDF de N-grammes.. ou BoW binaire de N-grammes, etc..

2.1.1 Pondération TF-IDF

Afin de pondérer les mots plus rares d'un corpus, apparaissant davantage dans un document, on transforme le vecteur sac de mots comportant le nombre d'occurences en un vecteur comportant le score tf-idf de ce terme.

2.1.2 BoW binaire

On étudie un modèle sans présence du nombre d'occurences, i.e. le vecteur sac de mots contient uniquement un booléen indiquant ou la présence ou non d'un terme du vocabulaire.

2.2 Réduction de dimension

Le nombre de dimension est jusqu'ici égal à la taille du dictionnaire et les vecteurs sont très peu denses. On va donc employer une technique de réduction de dimension basée sur LSA (utile sur des données sparses) avec des valeurs comprises entre 10 et 300 composantes de sorties

Note: la pondération TF-IDF est systématiquement appliquée si une technique de réduction de dimension est appliquée.

2.3 Mise à l'échelle (scaler)

Il s'agit d'une étape habituellement de centrage des données à la moyenne avec un écart-type unité. On n'effectuera ici pas de centrage pour éviter de transformer les matrices sparse en matrices denses. Notons que cette étape n'est pas utile avec un classifieur de Bayes.

3 Modèles d'apprentissage

3.1 Métriques d'évaluation

3.1.1 Données locuteurs

Étant donné le déséquilibre des données, il va falloir ici choisir ici un compromis entre précision et rappel. Un classifieur classe majoritaire aurait une accuracy de près de 90%, l'accuracy n'est donc pas une métrique adaptée ici.

Dans la suite du projet, on considèrera la classe Mitterant comme la classe positive, qu'on cherche à détecter. Ainsi, nous justifions le choix du score F-β avec $\beta = 1.5$ permettant de préférer légèrement le rappel à la précision.

3.1.2 Données films

Les données étant équilibrées, et comme nous souhaitons autant détécter les commentaires positifs que négatifs, nous utiliserons le score F-1. Notons que nous aurions aussi pu utiliser l'accuracy.

3.2 Stratégies d'entraînement

On étudiera les trois algorithmes suivants :

- SVM
- Classifieur à regression logistique
- Classifieur de Bayes naïf

Dans le cas des classifieurs SVM et regression logistique, on testera l'ajout de regularisation l1 et l2. Par ailleurs, on ponderera les classes par leur distribution dans les données d'entrainement pour la tâche de reconnaisance de locuteur.

Pour le classifieur de Bayes naïf, comme on conserve la repartition des classes dans la création des folds de validation-croisée, on évaluera aussi la probabilité à priori.

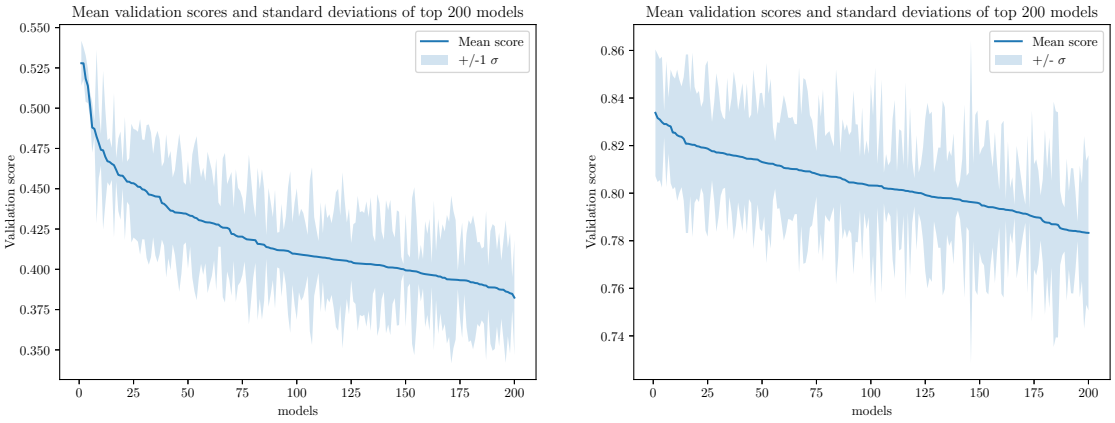
Nous utiliserons pour chaque modèle (incluant les paramètres de pré-traitements) les performances par validation croisée sur k=4 folds de données.

L'espace de recherche étant très vaste, nous utilisons les fonctionnalités de réduction de grille de recherche (`HalvingGridSearchCV`) sur l'ensemble de notre `pipeline` dans `scikit-learn`. Nous obtenons ainsi quelques centaines de modèles à comparer.

4 Résultats et évaluation de la méthode

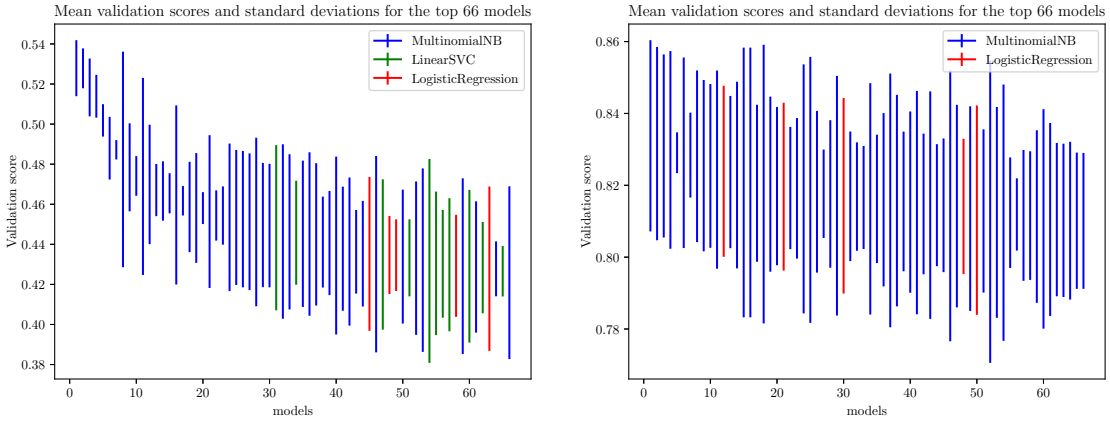
4.1 Détermination du meilleur modèle

Après quelques heures de calcul sur un ordinateur portable, on peut obtenir les résultats suivants. Ceux-ci montre le score de validation et l'écart-type des modèles les plus prometeurs.



(a) Tâche locuteurs (b) Tâche films

Figure 1: score de validation des 200 premiers modèles



(a) Tâche locuteurs (b) Tâche films

Figure 2: score de validation des 65 premiers modèles

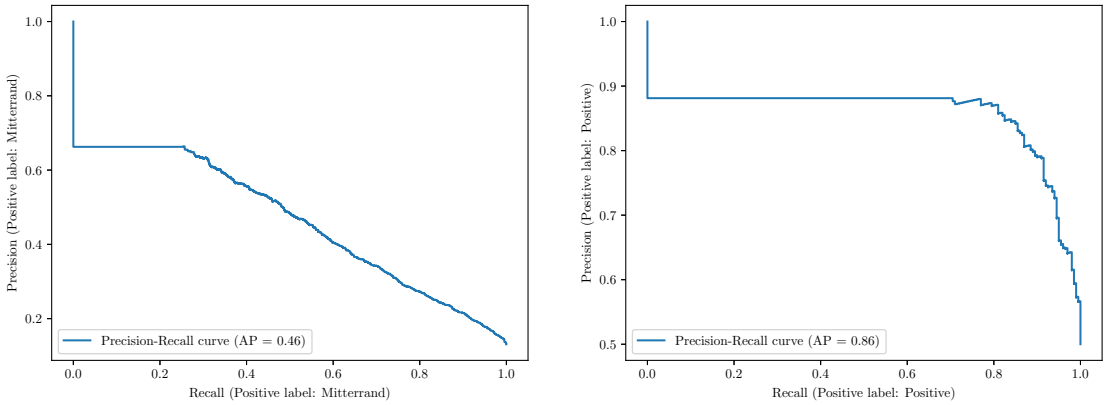
Pour les deux tâches, on constate la supériorité du classifieur de Bayes naïf. Cependant, l'espace de recherche étant très important, on ne peut affirmer qu'on obtiendrait des résultats similaires avec une recherche d'hyper-paramètres exhaustive.

Par ailleurs, l'écart-type reste important, notamment dans le cas de la tâche de détection de sentiments où certains classifieurs à regression logistique obtiennent aussi de bons résultats.

4.2 Évaluation finale du meilleur modèle

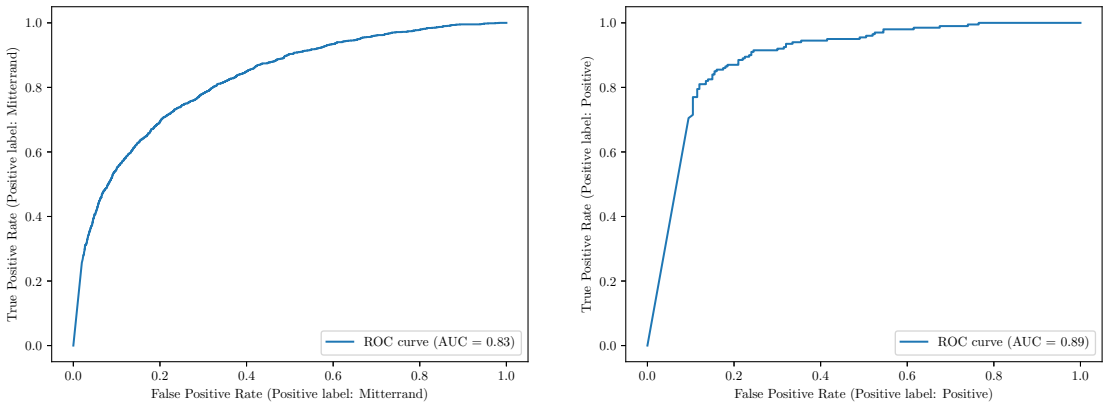
Le meilleur modèle trouvé correspond dans les deux cas à un classifieur naïf de Bayes où l'on considère aussi les n-grammes de taille 2. Les pré-traitements diffèrent.

Après ré-entrainement du meillleur modèle sur l'ensemble des données d'entrainement et de validation, on estime sa performance sur le jeu de données de test. Les résultats obtenus sont les suivants.



(a) Tâche locuteurs (b) Tâche films

Figure 3: Courbes précision-rappel



(a) Tâche locuteurs (b) Tâche films

Figure 4: Courbes ROC

Ces courbes permettent d'avoir un aperçu des performances de notre modèle final pour differentes valeurs de coupures (ou categorisation). Dans la tâche de détection de sentiment par exemple, la courbe ROC nous donne l'idée d'un seuil procurant autour de 20% de faux positifs pour 80% de vrais positifs.

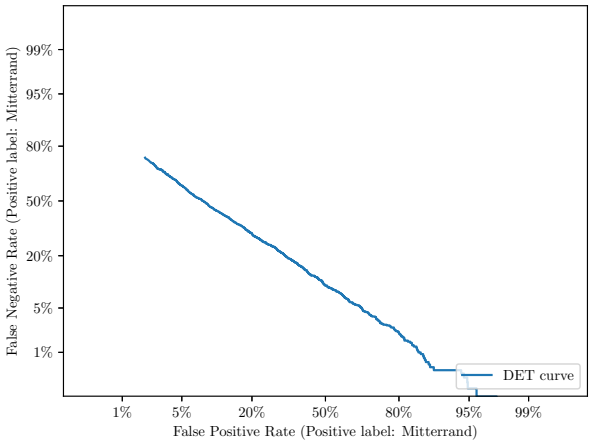


Figure 5: Courbe DET (Detection error tradeoff) tâche locuteurs

Dans la tâche de détection de locuteur, cette détermination est bien moins évidente et dépend grandement de l'objectif à la base de la conception d'un tel modèle, ex: annotation de sous-titres ou autres. Ce compromis est également visible sur la courbe de détection d'erreur indiquant le taux de faux négatifs en fonction de faux positifs.

Conclusion

Notre stratégie de recherche de pré-traitement et de type de classifieur combiné poour former un seul modèle fonctionne. Nous pouvons raisonnablement supposer qu'en augmentant le nombre de modèles à comparer en explorant davantage la grille de recherche et / ou utilisant des techniques d'optimisations d'hyper-paramètres nous obtiendrions de meilleur résultats.

Toutefois, cette approche, bien que puissante et utile dans un cadre appliqué à la production s'effectue au détriment de l'interprétabilité : pourquoi l'association de ces traitements avec ce type de classifieur fonctionne ?

Par ailleurs, il n'est pas évident d'optimiser des modèles où le problème est "mal (ou incomplètement) posé" comme dans la tâche de détection de locuteur.

Enfin, à travers ce projet, on peut noter la relative efficacité des modèles basés sur les sacs de mots dans des tâches de traitement automatique du langage naturel.