FDS Université de Montpellier

Méthodes de la science de données

Projet

Classification de documents d'opinions

Groupe T :

MINKO AMOA Dareine
NGUYEN Thu Thao
RAIHAUTI Teiki
SCHMID Thomas
SKENDRAOUI Mira

Contents

1 Introduction

1.1 Présentation

Ce projet a pour finalité l'utilisation de procédés d'apprentissage automatique pour pouvoir classifier des données d'avis d'internautes sur des films. Nous avons à disposition un jeu de données de 10000 commentaires, chacun labellisé d'une valeur positive ou négative (1/-1).

Nous devons à terme établir la meilleure stratégie de polissage des avis et tester différents modèles de classification pour déterminer le meilleur d'entre eux. Ainsi nous pourrons tester notre modèle sur un nouveau jeu de données équivalent.

1.2 Méthode

Pour déterminer notre modèle de prédiction, nous avons procédé de cette façon :

- Déterminer et effectuer les prétraitements sur les données permettant leur exploitation optimale.
- Lancer plusieurs classifieurs avec les paramètres par défaut en utilisant la **k-fold Cross Validation**. Cette méthode consiste à diviser les données en k échantillons, dont un sert à la validation et les autres à l'apprentissage. Chaque échantillon est utilisé pour la validation tour à tour, permettant ainsi notamment d'éviter le surapprentissage.
- Prendre les meilleurs classifieurs et trouver les hyper paramètres optimisant la précision.
- Sauvegarder le modèle et le tester sur de nouvelles données.

1.3 Outils et langages

Nous avons utilisé **Scikit-learn**, une bibliothèque pour le machine learning en Python. C'est aussi celle que nous avons appris à utiliser dans les TP avec les notebook pour faciliter le partage du travail et l'ergonomie du code.

Pour les traitements sur les textes, nous avons utilisé **NLTK** et ses différents modules ainsi que la bibliothèque **Pandas** pour la visualisation de données et la lecture des fichiers *csv*.

2 Création du modèle

2.1 Prétraitements et choix

Nous avons mis en place plusieurs prétraitements permettant de polir les textes, à savoir la suppression des caractères non ASCII, la suppression des contractions, la mise en minuscule des mots, ainsi que d'autres expliqués en détails ci-après.

1. **Tokenisation**
   - Notre tokenisation se base principalement sur le modèle d'écriture des Tweets : nous gardons ainsi des éléments commes les émoticones ou les hashtags, porteurs de sens.
   - Les URLs, tags HTML et mentions sont également stockés pour être supprimés, ceux-ci étant inutiles à l'analyse textuelle.
   - Les signes de ponctuation répétés comme '!!!' nous semblaient importants à retenir, nous avons donc fait des tests pour évaluer leur pertinence. Ceux-ci se sont finalement révélés trompeurs, diminuant fortement le taux de réussite des prédictions, et nous les avons finalement supprimés.
2. **Stopwords**
   - Nous avons utilisé la liste de Stopwords correspondant à la langue de Shakespeare.
   - De plus, pour nous permettre d'être plus efficace dans notre domaine d'action (ici le cinéma), nous avons ajouté des mots qui nous semblaient inutiles à l'analyse d'opinions. En l'occurence des mots tels que : cinema, actor, plot.. ne servent pas dans l'analyse et sont ainsi ignorés.
   - Les signes de ponctuation simples font également partie des Stopwords. Bien qu'ils puissent changer le sens d'une phrase, ces cas ne nous semblent pas réalistiquement analysables.

3. **Lemmatisation**
    - Nous avons préféré la lemmatisation à la stematisation afin de ramener les mots vers une base commune. En effet, les mots comme 'love' ou 'lovely' sont similaires et il est inutile d'avoir les deux dans le dictionnaire.
    - Nous avons choisi d'appliquer la lemmatisation aux noms, aux adverbes ainsi qu'aux verbes.
    - Cette opération permet de traiter ces éléments de manière commune, leur attribuant une pondération qui porte plus de sens qu'en les gérant individuellement. En effet, les mots comme 'love' ou 'lovely' ont la même base et ont la même signification.
4. **Parts of Speech (POS) Tagging**
    - Le POS tagging, de la bibliothèque NLTK, est l'une des étapes les plus importantes de l'analyse de texte. Elle est utilisée pour générer des balises (tags) spécifiques à chaque mots afin de les classer selon leur rôle grammatical.
    - Nous l'avons utilisé pour supprimer des mots qui ne peuvent pas avoir d'importance.
    - Par exemple : until, before, .. (preposition/subordinating conjunction) ; I, he, she, .. (personal pronoun) ; for, yet .. (CC coordinating conjunction) ; ...
5. **N-gram**
    - Les n-grams permettent de rassembler des mots pour les traiter ensemble comme des séquences. Ils peuvent notamment permettre d'identifier les négations afin de les gérer correctement.
    - Nous avons testé plusieurs valeurs pour les n-grams mais seuls les bigrams ont pu améliorer les prédictions de notre classifieur.

2.2 Sarcasme

Le sarcasme n'est pris en compte que par les N-grams, en conservant le n-ordre de mots, TF-IDF ne le faisant pas. Ce raisonnement part du principe que dans une phrase sarcastique il peut y avoir une partie positive et une partie négative. Le problème est que ceci n'est pas la seule forme de sarcasme.

2.3 Choix du classifieur

Après avoir effectué les traitements sur notre jeu de données, nous avons lancé un 10-fold Cross Validation sur plusieurs classifieurs différents pour pouvoir décider lesquels seraient utiles à utiliser pour notre modèle. Nous aurions pu lancer un pipeline associé à plusieurs gridsearch pour avoir un choix plus précis, mais cela aurait demandé un temps extrêmement long et beaucoup trop de ressources. 2 classifieurs sont ressortis en particulier : Logistic Regression et LinearSVC Les deux présentaient la meilleure précision et un temps de calcul correct (de l'ordre de quelques secondes).

2.4 Sauvegarde du modèle

Nous avions des problèmes de compilation lorsque nous avons voulu associer un pipeline avec le gridsearch directement, nous avons donc séparé les deux dans deux cellules différentes du notebook.

Nous avons utilisé le classifieur *LogisticRegression* car après un *Grid Search* sur les 2 classifieurs trouvés précédemment, il s'est avéré être celui étant le plus précis et nous avons sauvegardé le modèle pour pouvoir faire des prédictions sur des données différentes.

3 Résultats du challenge

La meilleure précision aisément reproductible que nous avons obtenue avec les données de test est de 0.89. L'utilisation de ce classifieur sur les données du challenge a donné un résultat de 0.83 avant la mise en place de nos correctifs.

3.1 Interprétation

Il y a une différence significative (5%) entre la performance de notre modèle sur les données initiales et sur les données du challenge.

Pour expliquer cette différence, nous avons tout d'abord pensé au surapprentissage, une erreur souvent commise par les néophytes de la science des données. L'idéal pour un modèle étant qu'il soit performant avec les nouvelles données, or ce n'était pas

le cas du nôtre malgré le fait que les données du challenges aient le même contexte et la même nature que les données initiales.

Mais notre erreur était que, dans la pipeline, nous avions omis le prétraitement avant la vectorisation TF-IDF. La pondération se faisait donc sur les données de textes brutes alors que notre modèle a été entrainé sur des données prétraitées.

4 Améliorations

4.1 Prétraitements

Pour améliorer la précision du modèle, nous avons effectué quelques changements dans les prétraitements.
Tout d'abord, nous avons pris en compte :

- Les points de suspension qui peuvent avoir du sens dans un commentaire.
- Les mots contenant plusieurs lettres identiques à la suite. Par exemple *aaaaaaaah* n'était pas un mot retenu dans l'apprentissage de notre premier modèle, or il représente une émotion qui doit être traitée.
- Les smileys comme *:)* ou *:(* sont considérés comme des termes à part entière et font partie du dictionnaire. En effet, ils sont porteurs de sens.

4.2 Classification

En prenant en compte tous ces changements, nous avons relancé un pipeline avec différents classifieurs et plusieurs paramètres pour le gridsearch. Nous avons pu obtenir de meilleurs résultats, et surtout, nous nous sommes aperçus qu'il y avait un classifieur qui donnait une meilleure précision : **LinearSVC**. Nous avons donc modifié *LogisticRegression* pour ce dernier.

De plus, dans le pipeline, nous avions fait l'erreur de ne mettre aucun paramètre ce qui biaisait les résultats sur le challenge. En effet, il était primordial de rajouter nos prétraitements avec la ligne *preprocessor=clean_text* (clean_text étant la fonction rassemblant tous nos traitements) ainsi que les bigrammes.

Nous prenons aussi en compte plus de mots au moment de la vectorisation car avec le nouveau modèle, tous les termes apparaissant au moins 3 fois sont introduits dans le vocabulaire d'apprentissage.

4.3 Résultats finaux

Tous ces changements nous amènent à avoir une précision de 91.2% sur le jeu de données d'apprentissage et 90.8% sur le jeu de données de test.

Nous pouvons en conclure que nous avons réglé nos problèmes de surapprentissage car la différence de précision entre le jeu d'apprentissage et le jeu de test est minime. Enfin, notre nouveau modèle a obtenu des résultats plus précis donc nos changements se sont avérés justifiés.

Nous avons, par curiosité, étudié un échantillon des commentaires mal classés et essayé de comprendre où résidait le problème. Nous avons remarqué que notre modèle avait encore des difficultés à identifier le sarcasme.

5 Conclusion

Des tests plus complets sur les hyperparamètres de *LinearSVC* seraient intéressants, voire même de tous les autres classifieurs si nous avions des machines plus puissantes pour réduire les temps de calcul qui sont extrêmement longs.

Des méthodes plus avancées pourraient être mises en pratique pour identifier les commentaires sarcastiques. Les n-grams y participent mais ne sont pas suffisants car la présence seule du positif avec du négatif ne veut pas forcément dire qu'il s'agit forcément de sarcasme. Notamment, nous ne considérons pas les points d'exclamation et les "!?" (qui exprime une question rhétorique), les interjections par exemple, comme étant des stop-words car ils peuvent marquer le sarcasme.

En conclusion , ce projet nous a permis d'approfondir et acquérir des nouvelles compétences dans le domaine des méthodes de la science de données.

6 Remerciement

Nous souhaitons tout d'abord remercier nos enseignants qui nous ont encadrés et guidés tout au long de l'élaboration de ce projet. Tous les conseils qu'ils nous ont

donnés sont d'une grande importance et impactent positivement nos connaissances et compétences.