


FACULTÉ DES SCIENCES

RAPPORT PROJET MÉTHODES DE LA SCIENCES DES DONNÉES

Membres :

- BELLEL Oussama **21817080**
- BOURGIN Jérémy **20140477**
- DIABATE Amara **21509428**
- HERRI Abdallah **21712664**

Encadrants :

- FATI Chen
- CHRISTOPHE Menichetti
- KONSTANTIN Todorov
- PASCAL Poncelet

Table des matières

I. Introduction

Ce projet consiste à mettre en oeuvre la détection des avis positifs et négatifs. Le but consiste à trouver des caractéristiques communes entre divers commentaires positifs et négatifs. Pour cela, il faut utiliser un jeu de données assez important sur lequel on connaît la polarité de chaque commentaire. Ainsi on va pouvoir classifier ce qui caractérise les avis positifs des avis négatifs, et essayer de prédire la polarité d'un commentaire. On appel cela le machine learning. Dès lors, plusieurs problèmes se posent :

- comment extraire les caractéristiques communes ?
- comment classifier les données ?
- comment détecter la négation ?
- comment détecter le sarcasme ?

Pour aborder ces différentes problématiques, on va commencer par analyser quelques commentaires, ensuite on va essayer d'extraire les caractéristiques de chaque commentaire, puis on va implémenter un classifieur naïf pour avoir un algorithme de détection des avis sur lequel on va pouvoir se baser, par la suite on va utiliser différentes méthodes de classification, et sélectionner la meilleur. Enfin, on va évaluer le pourcentage de réussite des différentes méthodes de classification, sur des jeux de données destinés aux tests, et voir comment on peut améliorer ces résultats.

II. Développement

1. Approche

Pour aborder ce projet, nous avons commencé par prendre quelques commentaires au hasard. Cela nous a permit de constater l'état de notre jeu de données (la propreté du texte). De plus, cela nous a ouvert le voit sur les caractéristiques communes entre les différents avis. Dès lors, on a pu appliquer certains prétraitements (à l'aide de librairies comme NLTK, Stanford, SickitLearn...) qui permettront de rendre la classification efficace.

2. Prétraitements

Le prétraitement consiste à enchaîner une séquence de méthodes qui transforme les avis d'un texte brute lisible par l'humain, à un ensemble de mots facile à classifier.

2.1. Nettoyage du texte

Dans un premier temps, on a commencé par enlever les textes inutiles, et faire quelques ajustements sur le texte. Voici la liste des tâches :

- Suppression des URLs.
- Suppression des adresses emails.
- Suppression du code HTML.
- Suppression des doubles quotes (NLTK va considérer "word" et word comme 2 mots distinct).
- Ajout d'un espace avant et après chaque ponctuation (si quelqu'un fait une faute de frappe et écrit (word1.word2 NLTK va considérer cela comme un seul mot).
- Suppression des simples quotes inutiles.

2.2. Extraction des caractéristiques

Maintenant qu'on a un texte propre, on passe à la tokenization. Cela consiste à associer à chaque mot son un type correspondant (nom commun, adjectif, verbe, adverbe...). Pour cela, on a utilisé le NLTK.

Une fois les mots tokenisé, on récupère uniquement les mots avec des types qui seront potentiellement utiles pour la classification (on a choisit de garder uniquement les mots du type "adjectif", "comparatif", "superlatif").

2.3. Détection de la négation

Pour la détection de la négation, on commence par enlever tous les stopwords. Ensuite, on va chercher à concaténer le mot “not” avec le mot qui le succède. Par exemple : “this is not so good” devient “this is notgood”. Ici initialement le mot “good” qui est censé avoir une connotation positive se trouve dans un contexte positif. Après la transformation, on a le mot “notgood” qui a une connotation négative, ce qui est plus approprié au contexte courant (ici la suppression des stopword a permis d’éliminer le mot “so”).

2.4. TfidfVectorizer

Le TfidfVectorizer est une autre méthode pour extraire les caractéristiques à partir du texte nettoyé. Il fait une matrice des n-grams qui apparaissent dans les avis et associe un “poids” qui correspond au nombre d'occurrences de ce n-gram.

Le TfidfVectorizer nous permet d’affiner le résultat en manipulant quelques paramètres. Parmi ces paramètres il y a “min_df”, “max_df” et “ngram_range”.

- min_df : utilisé pour supprimer les termes qui apparaissent rarement, Par exemple (min_df = 0.01 signifie "ignorer les termes qui apparaissent dans moins de 1% des documents").
- max_df : utilisé pour supprimer les termes qui apparaissent trop fréquemment, par exemple (max_df = 0.50 signifie "ignorer les termes qui apparaissent dans plus de 50% des documents").
- ngram_range(min, max) : Les bornes inférieure et supérieure de la plage de valeurs n pour différents n-grams à extraire. Toutes les valeurs de n telles que min <= n <= max seront utilisées.

3. Algorithme naïf

Avant de commencer à utiliser les classifieurs fournis dans la bibliothèque “scikit-learn”, on a commencé par implémenter notre propre classifieur. L’idée est de créer un tableau {“mot”, “score_positif”, “score_négatif”} pour les mots qui étaient sélectionnés dans le prétraitement. Pour cela, le score positif associé à un mot est le nombre d'occurrence pour lequel ce mot apparaît dans les avis positifs, et donc le score négatif est le nombre d'occurrence pour lequel un mot apparaît dans les avis négatifs. Ainsi, pour calculer le score d’un mot, il suffit de faire simplement score_positif - score_négatif.

Une fois la classification terminé, on peut prédire un résultat sur un commentaire. Pour cela, on commence par appliquer les mêmes prétraitements. Ensuite, pour chaque mot du commentaire on va récupérer le score associé et faire la somme. Si le résultat est < 0 alors le commentaire est négatif sinon il est positif.

4. Classification

Après avoir fait différents prétraitements sur notre jeu de données, on a testé différents classifieurs comme le MultinomialNB, GaussianNB, DecisionTreeClassifier, RandomForestClassifier, LinearSVC, KNeighbors et AdaBoostClassifier. Ensuite, il faut faire la prédiction sur ce même jeu de données (et si possible sur un jeu de données tests) afin de connaître le taux de réussite de chaque méthode de classification. Pour cela, on a utilisé KFold.

KFold (validation croisée), est une méthode d'estimation de fiabilité d'un modèle fondé sur une technique d'échantillonnage. Son principe est de diviser nos données en k échantillons, et faire l'entraînement sur les k-1 echantillons et le reste des échantillons seront utilisés pour la prédiction (validation).

5. MultinomialNB

Après avoir testé différents classifieurs avec différents prétraitements, nous avons porté notre choix vers MultinomialNB. En effet, l'utilisation de ce classifieur a donné de meilleurs résultats que les autres. Pour cela, l'utilisation des prétraitements que nous apporte le TfidfVectorizer étaient suffisant pour obtenir le meilleur résultat possible. Nous nous sommes donc intéressé à son fonctionnement pour comprendre pourquoi il est efficace.

MultinomialNB est un classifieur qui se base sur les modèles bayésiens naïfs. Il est souvent plus efficace lorsque qu'il est entraîné dans un contexte d'apprentissage supervisé. Ce classifieur repose sur "le maximum de vraisemblance" pour l'estimation des différentes caractéristiques. Pour cela, il fait une approximation par rapport aux fréquences relatives des mots dans l'ensemble des données d'entraînement afin d'estimer les différentes caractéristiques.

Dans le cadre de notre projet, notre jeu de données contient des avis positifs et négatifs sur des films. On constate qu'il y a beaucoup de ressemblances entre les commentaires. Par exemple : l'utilisation des mots "good, nice, like" pour les commentaires positifs, et des mots : "hate, bad, worst" pour les commentaires négatifs, sont souvent employé. Sachant que le classifieur MultinomialNB se base sur les ressemblances et les dépendances entre les les caractéristiques, alors cela justifie son efficacité par rapport à d'autres méthodes de classification.

III. Analyse

1. Résultats

1.1. Algorithme naïf

```
accuracy du jeu de données initial:

0.7638

 matrice de confusion
 [[3168 1832]
 [ 530 4470]]
              precision    recall  f1-score   support

          -1       0.86      0.63      0.73      5000
           1       0.71      0.89      0.79      5000

   micro avg       0.76      0.76      0.76     10000
   macro avg       0.78      0.76      0.76     10000
weighted avg       0.78      0.76      0.76     10000


accuracy du jeu de données du challenge:

0.7765

 matrice de confusion
 [[1308  692]
 [ 202 1798]]
              precision    recall  f1-score   support

          -1       0.87      0.65      0.75      2000
           1       0.72      0.90      0.80      2000

   micro avg       0.78      0.78      0.78      4000
   macro avg       0.79      0.78      0.77      4000
weighted avg       0.79      0.78      0.77      4000


accuracy du jeu de données de IMDB:

0.71842

 matrice de confusion
 [[13733 11267]
 [ 2812 22188]]
              precision    recall  f1-score   support

          -1       0.83      0.55      0.66     25000
           1       0.66      0.89      0.76     25000

   micro avg       0.72      0.72      0.72     50000
   macro avg       0.75      0.72      0.71     50000
weighted avg       0.75      0.72      0.71     50000
```

1.2. Classifieur MultinomialNB

```
accuracy du jeu de données initial:

0.9514


matrice de confusion
[[4729  215]
[ 271 4785]]

              precision    recall  f1-score   support

          -1       0.96      0.95      0.95      5000
           1       0.95      0.96      0.95      5000

   micro avg       0.95      0.95      0.95     10000
   macro avg       0.95      0.95      0.95     10000
weighted avg       0.95      0.95      0.95     10000


accuracy du jeu de données du challenge:

0.91675


matrice de confusion
[[1808  141]
[ 192 1859]]

              precision    recall  f1-score   support

          -1       0.93      0.90      0.92      2000
           1       0.91      0.93      0.92      2000

   micro avg       0.92      0.92      0.92      4000
   macro avg       0.92      0.92      0.92      4000
weighted avg       0.92      0.92      0.92      4000


accuracy du jeu de données de IMDB:

0.86506


matrice de confusion
[[20475  2222]
[ 4525 22778]]

              precision    recall  f1-score   support

          -1       0.90      0.82      0.86     25000
           1       0.83      0.91      0.87     25000

   micro avg       0.87      0.87      0.87     50000
   macro avg       0.87      0.87      0.86     50000
weighted avg       0.87      0.87      0.86     50000
```

2. Analyse des résultats

On constate qu'avec un algorithme assez simple on obtient de bons résultats. Par conséquent, il n'est pas étonnant que l'utilisation d'un classifieur avec des mécanismes plus complexe donne des résultats excellent. Il était aussi important que l'écart entre le résultat de la prédiction sur le jeu de données d'entraînement et le jeu de données de test soit le plus faible que possible. Dans notre cas, l'algorithme naïf à un écart assez faible (5%), et MultinomialNB a un écart de 5 à 10% ce qui est plutôt bon puisqu'il a un score très élevé. Un autre point intéressant dans ces résultats est la différence entre le score des prédictions positifs et négatifs. En effet, cela nous permet de pouvoir voir les faiblesses d'une classification.

Dans l'algorithme naïf, on remarque qu'il est moins bon pour détecter les avis positifs des avis négatifs. Il y a 2 raisons qui peuvent expliquer cela : la détection de la négation n'est pas assez poussé donc cela ne fonctionne pas toujours, et on n'a pas mis en place la détection du sarcasme. Par conséquent, dans des commentaires sarcastiques, on enregistre des mots comme étant négatifs alors qu'ils devraient être enregistré comme étant positifs.

On remarque qu'avec la classification MultinomialNB il y n'a quasiment aucun écart entre le score de la prédiction des avis positifs et négatifs. Premièrement, on remarque (après avoir fait quelque tests) que la détection de la négation fonctionne bien en faisant simplement des N-Grams (et donc sans avoir appliqué notre algorithme de détection de négation). De plus, on suppose que lors de l'entraînement, le classifieur a pu caractériser certains aspect du sarcasme, et donc prédire qu'un avis est négatif lorsque celui-ci est négatif.

IV. Conclusion

Pour conclure, ce projet nous a permis de détecter des avis positifs ou négatifs sur des données textuelles à l'aide de classifieurs. Nous sommes plutôt satisfait du résultat obtenu après avoir effectué plusieurs séries de tests. En effet, Parmis les classifieurs que nous avons utilisé, certains étaient plus ou moins efficace en fonction des prétraitements que l'on appliquait sur notre jeu de données. Cela nous nous a permis de comprendre le concept du machine learning, et aussi de voir un autre aspect de la programmation avec python avec Jupyter NoteBook. Pour finir, ce projet a été très enrichissant pour nos connaissances personnelles.