Mohand Ameziane ZAIDI et Kamel MESSAOUDENE

Université de paris 7 |

Classification des documents

Analyse sentimentale des gens vis-à-vis de films

**Sommaire**

[1 Dictionnaire de données 3](#_Toc58955312)

[2 Importation de données 3](#_Toc58955313)

[3 Nettoyage de données 3](#_Toc58955314)

[4 Conversion de texte en nombre 4](#_Toc58955315)

[5 Encoding TF-IDF 4](#_Toc58955316)

[6 Ensemble de formation et de test 4](#_Toc58955317)

[7 Algorithme et évaluation de modèle 5](#_Toc58955318)

[7.1 Algorithme de forêt aléatoire 5](#_Toc58955319)

[7.2 Evaluation du modèle 5](#_Toc58955320)

[8 Conclusion 6](#_Toc58955321)

[8.1 Résultats 6](#_Toc58955322)

[8.2 Difficultés rencontrées 6](#_Toc58955323)

[8.3 Code 6](#_Toc58955324)

[8.4 Contribution de chaque membre 6](#_Toc58955325)

***Table de figures :***

[Figure 1 - Importation de données 3](#_Toc58955871)

[Figure 2 - Nettoyage de données 4](#_Toc58955872)

[Figure 3 - Conversion de texte en nombre 4](#_Toc58955873)

[Figure 4 - Train et Test 5](#_Toc58955874)

[Figure 5 - Random Forest Classifier 5](#_Toc58955875)

[Figure 6 - Visualisation et évaluation du modèle sélectionné 5](#_Toc58955876)

[Figure 7 – Résultats 6](#_Toc58955877)

# Dictionnaire de données

Afin de mettre en place notre analyse et application on a exploité les données dont la source est ci-dessous :

Source : <http://www.cs.cornell.edu/people/pabo/movie-review-data>

L’ensemble des données comprend un total de 2000 documents dont la moitié contient des critiques positives concernant un film tandis que la moitié restante contient des critiques négatives.

# Importation de données

Ensuite afin de les utiliser on doit les importer depuis les différents fichiers pour cela nous avons utilisé la fonction LOAD\_FILES de la Bibliothèque SKLEARN\_DATASETS.



Figure - Importation de données

Depuis la capture ci-dessous la variable X contient l’ensemble des données (fichiers positifs et négatifs) tandis que les catégories cibles (0 : négatif et 1 : positif) sont stocké dans Y.

# Nettoyage de données

L’objectif de la présente opération est de rendre nos données la plus clean possible grâce à la succession des opération suivantes :

* Suppression des caractères uniques (début, milieu et fin)
* Suppression des caractères spéciaux…
* Ecrire en minuscule.
* Lemmatisation grâce à la fonction split.

Une image contenant texte

Description générée automatiquement

Figure - Nettoyage de données

Nous avons utilisé pour cela la bibliothèque python RE et les expressions régulières pour effectuer les diverses tâches de prétraitement de texte.

# Conversion de texte en nombre

Dans cette étape nous avons convertie le texte brut des fichiers en nombres en appliquant le modèle du **sac de mots**.



Figure - Conversion de texte en nombre

Dans le bout de code ci-dessus nous avons utilisé la fonction **CountVectorizer** qui prend les paramètres suivants :

* **max\_features**: il représente le nombre souhaité des mots fréquents.
* **min\_df**: le nombre de document dans lesquels les mots doivent être présents impérativement afin de les sélectionner.
* **max\_df**: le pourcentage maximum de présence de mots afin de les sélectionner.
* **stop\_words**: supprime les mots vides.

# Encoding TF-IDF

Tel que vu en cours le TF représente le « Terme Frequency » donc on va associer à chaque terme une fréquence, et l’IDF c’est la fréquence inverse d’un document « Inverse Document Frequency »

# Ensemble de formation et de test

Afin de deviser nos données en ensemble de formation et de test nous avons utilisé la fonction **train\_test\_split** qui prend en argument :

Les données, les catégories, pourcentage de l’ensemble de test et le reste pour l’ensemble de formation.



Figure - Train et Test

# Algorithme et évaluation de modèle

## Algorithme de forêt aléatoire

A présent nous avons divisé nos données en ensemble de formation et de test. Pour entrainer notre modèle nous avons implémenté l’algorithme de forêt aléatoire.

Une image contenant texte

Description générée automatiquement

Figure - Random Forest Classifier

Nous avons utilisé la fonction **RandomForestClassifier** de la bibliothèque **SKLEARN**. De plus on a utilisé la méthode **fit** afin d’entrainer l’algorithme.

Ensuite, on a prédit grâce à la fonction **predict** méthode de **RandomForestClassifier** tel que présenté ci-dessus.

## Evaluation du modèle

Dans le but d’évaluer la performance de notre modèle de classification on a utilisé la matrice de confusion, la mesure F1 et la précision.

Afin de trouver ces valeurs sous python on a utilisé « **classification\_report** », « **confusion\_matrix** » et les « **accuracy\_score** » services publics de la « **sklearn.metrics** ».

Une image contenant texte

Description générée automatiquement

Figure - Visualisation et évaluation du modèle sélectionné

# Conclusion

## Résultats

Une image contenant table

Description générée automatiquement

Figure – Résultats

## Difficultés rencontrées

Parmi les difficultés rencontrées au cours du projet on a les suivantes :

* Travail en distanciel à cause de la crise sanitaire.
* La charge de travail (plusieurs projets en parallèle et les examens)
* Problèmes liés au système d’exploitation différents.

## Code

Le lien GitHub suivant vous permet de visualiser notre code :

<https://github.com/ZAIDIMDAMZ/documentsClassificator>

## Contribution de chaque membre

* **Idée du projet** : Kamel MESSAOUDEN et Mohand Ameziane ZAIDI
* **Code** :
  + Partie “Choix de sujet” : Les deux en collaboration avec le prof.
  + Partie “Nettoyage et importation de données” : Kamel MESSAOUDENE
  + Partie “Algorithme de classification” : Mohand Ameziane ZAIDI.
  + Partie “Evaluation et visualisation” : Kamel MESSAOUDENE
  + Partie “Slides et rédaction” : Mohand Ameziane ZAIDI.
* **Nettoyage des données** : Kamel MESSAOUDENE
* **Algo des forêts aléatoire** : Mohand Ameziane ZAIDI
* **Evaluation et vérification de travail** : Kamel MESSAOUDENE
* **Slides** : Mohand Ameziane ZAIDI