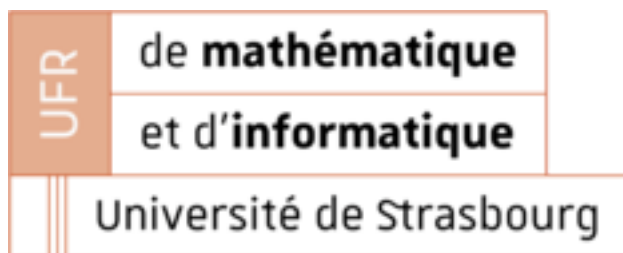


# Traitement Automatique de Langage

## Rapport de Projet

**Equipe:**  
ALLEMAND Fabien  
LEBOT Samuel



## Table des matières

Liste des figures	iii
<b>1 Analyse et Pré-traitement des Données</b>	<b>1</b>
<b>2 Méthodes Basiques</b>	<b>1</b>
2.1 Pré-traitement des Données . . . . .	1
2.2 Apprentissage Simple . . . . .	2
2.3 Apprentissage par Plis et Comparaison des Algorithmes . . . . .	3
<b>3 Réseau Neuronal Convolutif</b>	<b>3</b>
3.1 Pré-traitement des Données . . . . .	3
3.2 Indexation du Vocabulaire . . . . .	4
3.3 Chargement de Plongements de Mots Pré-entraînés . . . . .	4
3.4 Construction et Entraînement du Réseau Neuronal Convolutif . . . . .	4
<b>4 BILSTM</b>	<b>5</b>
<b>5 Transformer</b>	<b>5</b>
<b>Bibliographie</b>	<b>7</b>

## Liste des figures

1	Effectifs des classes dans les données d'entraînement . . . . .	1
2	Comparaison des matrices de confusion pour des données déséquilibrées et équilibrées: les classes les moins représentées dans le jeu de données d'entraînement sont moins bien apprises lorsque les classes sont déséquilibrées. . . . .	2
3	Pipeline de pré-traitement des données . . . . .	2
4	Pipeline d'entraînement d'un modèle de régression logistique . . . . .	3
5	Précision de différents algorithmes lors d'un apprentissage par validation croisée avec 5 plis . . . . .	3
6	Architecture du réseau neuronal convolutif . . . . .	4
7	Précision du réseau neuronal convolutif au cours de l'entraînement . . . . .	5
8	Architecture du réseau BILSTM . . . . .	5
9	Interface web Solr pour une requête sur le mot <i>star</i> . . . . .	6

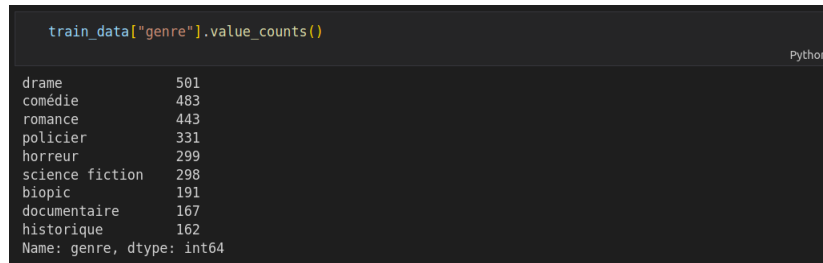


Figure 1: Effectifs des classes dans les données d'entraînement

## Introduction

L'objectif du projet est de réaliser un système de recherche d'information dans une collection de descriptions de films publiées sur Allociné.

Le projet se décompose en deux parties:

- Prédiction du genre des films par TAL
- Visualisation des résultats

## 1 Analyse et Pré-traitement des Données

Dans un premier temps, les données d'entraînement (*allocine\_genres\_train.csv*) peuvent être chargées grâce à la fonction `read_csv` de la bibliothèque Pandas [7] en précisant le séparateur (`sep=","`). L'utilisation des méthodes `head`, `tail`, `describe`, `info` et `hist` permettent de visualiser et comprendre les données contenues dans le jeu de données complet.

Dans le cadre de ce projet seules les données contenues dans les colonnes *titre* et *synopsis* seront utilisées pour déduire la valeur contenue dans *genre*. Le jeu de données peut être réduit à ces trois features.

Etant donné que les données vont être utilisées pour de l'apprentissage automatique, il faut vérifier s'il y a des valeurs manquantes. Les méthodes `isna` et `sum` ne révèlent aucune valeur manquante dans les données d'entraînement.

La proportion des classes dans le jeu de données peut avoir un impact sur l'apprentissage. Les classes ayant un effectif plus faible seront généralement moins bien "appries". La figure 1 montre que les classes n'ont pas toutes la même proportion dans le jeu de données: il y a beaucoup d'individus de la catégorie *drame* alors que les classes *biopic*, *documentaire* et *historique* sont très peu représentées.

Après de nombreuses expériences, il s'avère que rééquilibrer les classes par *oversampling*, c'est à dire: dupliquer des individus des classes les moins représentées, donne de meilleurs résultats quelque soit la méthode utilisée. (Figure 2)

**Dans tout la suite**, les résultats présentés correspondront aux résultats obtenus avec le jeu de données d'entraînement rééquilibré par *oversampling* à l'aide de l'objet `RandomOverSampler` de la bibliothèque Imbalanced-learn [5].

**Remarque** Le jeu de données d'entraînement contient trop peu de données pour effectuer un équilibrage des classes par *undersampling*, c'est à dire: supprimer des individus des classes les plus représentées. Les résultats obtenus avec cette méthode étaient généralement moins bons que sans équilibrage.

## 2 Méthodes Basiques

### 2.1 Pré-traitement des Données

Il est possible de mettre en place une pipeline (Figure 3) afin de vectoriser les données d'apprentissage et d'en extraire des informations statistiques.

Dans cette pipeline, les données déjà tokenisées vont être vectorisées selon la méthode TF-IDF en supprimant

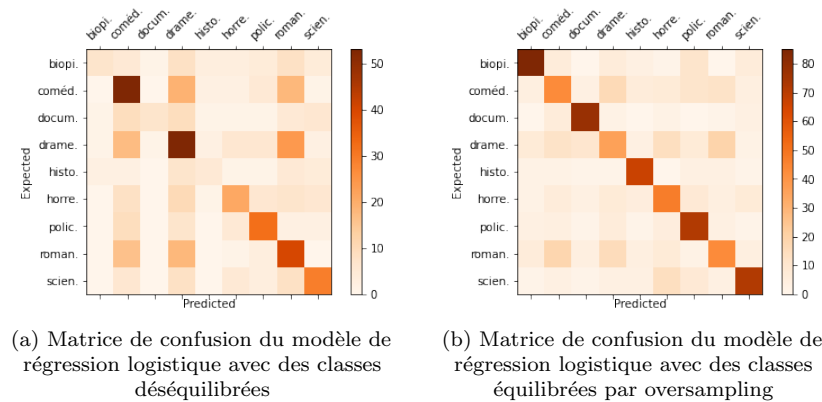


Figure 2: Comparaison des matrices de confusion pour des données déséquilibrées et équilibrées: les classes les moins représentées dans le jeu de données d'entraînement sont moins bien apprises lorsque les classes sont déséquilibrées.

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

column_trans = ColumnTransformer(
    [
        # Titre: tf-idf
        ("titre_tfidf", titre_vectorizer, "titre"),
        # Synopsis: tf-idf
        ("synopsis_tfidf", synopsis_vectorizer, "synopsis"),
        # Synopsis: statistiques
        (
            "synopsis_stats",
            Pipeline(
                [
                    ("text_stats", text_stats_transformer),
                    ("vect", text_stats_vectorizer),
                    ("scaling", min_max_scaler)
                ]
            ),
            "synopsis"
        )
    ],
    # Others
    remainder="passthrough"
)
```

✓ 0.1s Python

Figure 3: Pipeline de pré-traitement des données

les stop-words. Puis des objets `FunctionTransformer` et `DictVectorizer` de la bibliothèque Scikit-learn [8] vont être utilisés pour obtenir les valeurs statistiques telles que la longueur du synopsis en nombre de mot et en nombre de phrases. Un `MinMaxScaler` est utilisé pour normaliser ces données afin qu'elles aient le même poids lors de l'apprentissage.

## 2.2 Apprentissage Simple

Avant tout apprentissage supervisé, il faut définir un jeu d'apprentissage et un jeu de test avec par exemple la méthode `train_test_split` de Scikit-Learn en spécifiant la proportion du jeu de données sélectionnée pour les données de test (typiquement: `test_size=0.2`) et précisant `shuffle=True` pour que les données ne soient pas sélectionnées séquentiellement (ce qui pourrait impacter le résultat si le jeu de données est trié par classes).

**Remarque** || Le jeu de données *allocine\_genres\_test.csv* correspond au jeu de validation qui ne sera utilisé qu'après avoir définitivement choisi la méthode de prédiction.

Il est alors possible de créer une nouvelle pipeline pour automatiser le pré-traitement et l'apprentissage (Figure 4).

On peut ensuite faire des prédictions sur le jeu de test avec la méthode `predict` pour évaluer le modèle. Ce modèle de régression logistique donne une précision de 0.6 et un rappel de 0.61 (donc un score f1 de 0.6).

```
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression

# Pipeline pré-traitement et apprentissage
classifier_pipeline = make_pipeline(
    # Préparation des données pour l'apprentissage
    column_trans,
    # Algorithme d'apprentissage
    LogisticRegression()
)

# Apprentissage avec les données d'entraînement
classifier_pipeline.fit(X_train, y_train.to_numpy())
```

Figure 4: Pipeline d'entraînement d'un modèle de régression logistique

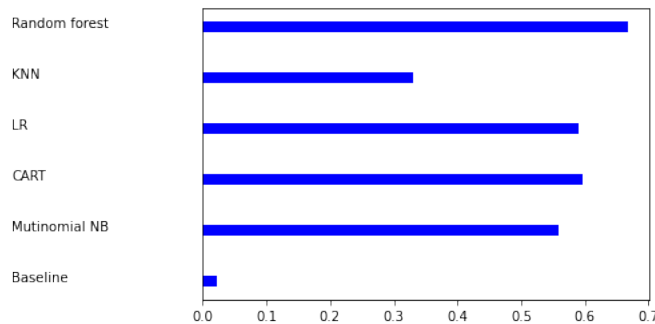


Figure 5: Précision de différents algorithmes lors d'un apprentissage par validation croisée avec 5 plis

## 2.3 Apprentissage par Plis et Comparaison des Algorithmes

Comme vu précédemment, la taille et les individus sélectionnés pour entraîner le modèle peuvent avoir une influence sur la qualité de l'apprentissage. Pour contrer ce biais on peut utiliser des plis pour l'apprentissage: au lieu de diviser les données d'entraînement en un jeu d'entraînement et un jeu de test, on peut diviser les données d'entraînements en plusieurs plis qui seront tour à tour des données d'entraînement ou de test. Cela permet d'obtenir une idée de la performance moyenne du classifieur. On utilise la méthode `StratifiedKFold` de Scikit-Learn qui a la particularité de conserver la proportion d'individus de chaque classe dans les plis.

En utilisant cette méthode, on peut comparer le modèle de régression logistique avec d'autres algorithmes. Les meilleurs résultats proviennent d'un modèle de forêt aléatoire (précision proche de 0.7). (Figure 5)

## 3 Réseau Neuronale Convolutif

Les réseaux neuronaux convolutifs (ou Convolutional Neural Network)[3] sont des réseaux de neurones qui utilisent des opérations de réduction de dimension et de convolution afin d'extraire des caractéristiques sur les données.

### 3.1 Pré-traitement des Données

Ces réseaux de neurones sont généralement associés à un réseau dense pour effectuer la prédiction après avoir extrait les informations des données. Dans le cadre du projet, la prédiction correspond à une classe (genre du film) parmi neuf à l'aide du titre et du synopsis du film. La dernière couche du réseau comporte donc neuf cellules qui donneront chacune la probabilité que le film appartienne à chaque classe. Pour l'apprentissage et la prédiction, il faut lister tous les genres, leur associer un indice et remplacer les valeurs dans les données (voir dictionnaires `genre2id` et `id2genre`).

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, None)]	0
embedding (Embedding)	(None, None, 700)	5600000
conv1d (Conv1D)	(None, None, 64)	224064
max_pooling1d (MaxPooling1D)	(None, None, 64)	0
conv1d_1 (Conv1D)	(None, None, 64)	20544
global_max_pooling1d (GlobalMaxPooling1D)	(None, 64)	0
dense (Dense)	(None, 64)	4160
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 9)	585
Total params: 5,849,353		
Trainable params: 5,849,353		
Non-trainable params: 0		

Figure 6: Architecture du réseau neuronal convolutif

Dans le dataframe, les colonnes *titre* et *synopsis* doivent être fusionnées afin de pouvoir les vectoriser dans la suite.

### 3.2 Indexation du Vocabulaire

Les réseaux neuronaux convolutifs effectuent des opérations sur des données numériques. Pour une application en TAL, il faut transformer le contenu sous forme de texte en données numériques, par exemple représenter chaque mot comme un vecteur de nombres réels dans un espace de grande dimension.

Dans le programme, l'objet de type `TextVectorization` de la bibliothèque Tensorflow [9] permet d'effectuer la vectorisation, chaque mot correspond alors à son indice dans le vecteur. Par exemple: si `TextVectorization` place le mot *bonjour* à la troisième position du vecteur alors le mot *bonjour* sera traité par le CNN comme la donnée 2.

### 3.3 Chargement de Plongements de Mots Pré-entraînés

La vectorisation des mots du vocabulaire est une opération complexe mais il est possible d'utiliser des plongements pré-entraînés.

Dans ce projet, le plongement *frWiki-no\_phrase-no\_postag\_700\_cbow\_cut100.bin* [6] de dimension 700 a été utilisé pour vectoriser les données.

### 3.4 Construction et Entraînement du Réseau Neuronal Convolutif

Un CNN est composé de plusieurs couches effectuant différentes opérations (réduction de dimension, convolution, couches denses). Le réseau utilisé dans ce projet contient seulement deux couches de réduction de dimension et deux couches de convolution (Figure 6).

**Remarque** || Après plusieurs expériences avec différentes architectures, il semblerait qu'ajouter des couches de réduction de dimension et de convolution n'améliorent pas la qualité des résultats dans le cadre de cette étude.

Les opérations de convolution du CNN sont des opérations qui doivent être apprises au cours de l'entraînement. [2] La taille des batchs est de 64 et le nombre d'époques choisi est 25, permettant un apprentissage plus précis mais aussi plus long. Pour contrer cela, un callback de type `EarlyStopping` a été ajouté (arrêt de l'apprentissage lorsque la précision du modèle diminue).

Le réseau neuronal convolutif obtenu a une précision de 72.91%.

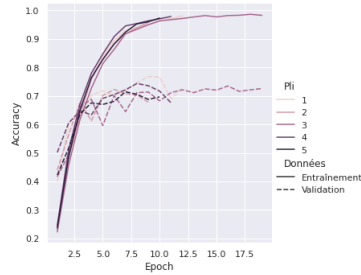


Figure 7: Précision du réseau neuronal convolutif au cours de l'entraînement

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, None)]	0
embedding (Embedding)	(None, None, 700)	5600000
bidirectional (Bidirectional)	(None, 128)	391680
dense (Dense)	(None, 9)	1161
Total params: 5,992,841		
Trainable params: 5,992,841		
Non-trainable params: 0		

Figure 8: Architecture du réseau BILSTM

## 4 BILSTM

Les réseaux (bidirectionnels) à mémoire longue à court termes permettent généralement d'obtenir de meilleurs résultats en TAL car ils ont la capacité de retenir le contexte dans lequel ils travaillent.

Le pré-traitement des données (jusqu'au chargement du plongement pré-entraîné) est similaire au CNN. Cependant, l'architecture du modèle est très différente (Figure 8). L'entraînement est le même que pour le CNN et on obtient une précision de 74.51%.

## 5 Transformer

Les transformers sont des modèles de réseaux de neurones permettant la prise en compte de dépendance à très grande distance grâce à un mécanisme d'auto-attention.

Pour des tâches complexes cet entraînement requiert une grande quantité de données ainsi qu'une grande puissance de calcul afin d'obtenir des résultats corrects. Pour éviter cette phase d'entraînement, on peut utiliser un modèle déjà entraîné et l'adapter aux données et à l'objectif souhaité.

Le site Hugging Face [4] recense de nombreux modèles pré-entraînés en fonction de la langue et de la tâche. Malheureusement, que ce soit en local ou en utilisant Google Colab, nous n'avons pas trouvé de modèle français pouvant être entraîné...

## Conclusion

Avec le temps et la puissance de calcul à disposition, le meilleur modèle pour la classification des films semble être le réseau neuronal convolutif. Il offre des résultats proches du BILSTM pour un temps d'entraînement significativement plus faible.

En entraînant le même modèle avec l'ensemble des données du jeu d'entraînement et en testant sur l'ensemble des données du jeu de test, on observe une précision de 27%.

Ce niveau de précision semble faible, cependant si on observe le vecteur de probabilités à la sortie des couches denses, dans 60% des cas le CNN place le genre attendu dans les trois premiers.



Modèle	Radom Forest	CNN	BILSTM	Transformer
Précision	66%	72.91%	74.51%	???

Table 1: Résultats des modèles testés

[Solr Admin](#)

**Filtrer par...**

**Nationalité**

[Américain](#) (6)

**Langue**

[Anglais](#) (6)

**Réalisateur**

[George Lucas](#) (2)

[Derek Fillion](#) (1)

[Ivan Kerschner](#) (1)

[J.J. Abrams](#) (1)

[Richard Marquand](#) (1)

6 results found in 166ms Page 1 of 1

**Année de Production:** 2019

**Synopsis:** La conclusion de la saga Skywalker . De nouvelles légendes vont naître dans cette bataille épique pour la liberté .

**Titre:** Star Wars : L' Ascension de Skywalker

**Genre:** science fiction

**Genre (prédiction):** documentaire

**uri:** [567](#)

**Année de 2008**

**Production:**

**Synopsis:** La galaxie est en proie à la Guerre des Clones , un conflit à grande échelle qui oppose les maléfiques Séparatistes et leurs immenses armées d' androïdes à la République . Les Chevaliers Jedi , protecteurs de la République , luttent pour maintenir l' ordre et restaurer la paix tandis que de nouvelles planètes succombent chaque jour aux puissances du mal . Pour prendre l' avantage , le Chevalier Jedi Anakin Skywalker et sa jeune Padawan , Ahsoka Tano , sont chargés d' une mission capitale qui va les confronter au redoutable " parrain " de Tatooine , Jabba le Hutt . D' autres épreuves et de nouveaux dangers attendent nos héros sur Tatooine , car le Comte Dooku ne tarde pas à se lancer à leur poursuite avec ses sinistres agents - dont la cruelle Asajj Ventress , prête à tout pour faire échouer Anakin et Ahsoka . Pendant ce temps , sur le front , Obi - Wan Kenobi et Maître Yoda s' efforcent vaillamment de préparer l' armée des clones à résister aux forces des Ténébres ...

**Titre:** Star Wars : The Clone Wars

**Genre:** science fiction

**Genre science fiction**

**(prédiction):**

**uri:** [60](#)

**Année de 1999**

**Production:**

**Synopsis:** Avant de devenir un célèbre chevalier Jedi , et bien avant de se révéler l' âme la plus noire de la galaxie , Anakin Skywalker est un jeune esclave sur la planète Tatooine . La Force est déjà puissante en lui et il est un remarquable pilote de Podracer . Le maître Jedi Qui - Gon Jinn le découvre et entrevoyt alors son immense potentiel . Pendant ce temps , l' armée de droïdes de l' insatiable Fédération du Commerce a envahi Naboo , une planète pacifique , dans le cadre d' un plan secret des Sith visant à accroître leur pouvoir . Pour défendre la reine de Naboo , Amidala , les chevaliers Jedi vont devoir affronter le redoutable Seigneur Sith , Dark Maul .

**Titre:** Star Wars : Episode I - La Menace fantôme

**Genre:** science fiction

**Genre science fiction**

**(prédiction):**

**uri:** [20](#)

**Genre**

**Genre (prédiction)**

[science fiction](#) (5)

[documentaire](#) (1)

**Genre**

[science fiction](#) (6)

Figure 9: Interface web Solr pour une requête sur le mot *star*

Après avoir enregistré les prédiction dans un fichier au format `csv`, les résultats obtenus peuvent être parcourus et analysés au moyen d'une interface web basée sur Solr [1]. L'utilisation des panneaux latéraux permet de filtrer les résultats des requêtes. (Figure 9)

## Bibliographie

- [1] apache solr. <https://solr.apache.org/>. Accessed: 2023-04-30.
- [2] Cnn et couche de convolution. <https://inside-machinelearning.com/cnn-couche-de-convolution/>. Accessed: 2023-04-30.
- [3] Convolutional neural networks. <https://www.ibm.com/topics/convolutional-neural-networks>. Accessed: 2023-04-30.
- [4] Hugging face. <https://huggingface.co/models>. Accessed: 2023-04-30.
- [5] Imbalanced-learn. <https://imbalanced-learn.org/stable/>. Accessed: 2023-04-30.
- [6] jean-philippe fauconnier. <http://fauconnier.github.io/>. Accessed: 2023-04-30.
- [7] Pandas. <https://pandas.pydata.org/>. Accessed: 2023-04-30.
- [8] Scikit-learn. <https://scikit-learn.org/stable/>. Accessed: 2023-04-30.
- [9] Tensorflow. <https://www.tensorflow.org/?hl=fr>. Accessed: 2023-04-30.