المدرسة العليا للتكنولوجيا - الداخلة
ⵜⵉⵎⵙⵙⵓⵔⵉⵜ ⵜⴰⵙⵙⴰⵡⵉⵜ ⵏ ⵜⵙⵏⵙⵉⵜ - ⴰⴷⴰⵅⵍⴰ
ÉCOLE SUPÉRIEURE DE TECHNOLOGIE - DAKHLA

École Supérieure de Technologie - DAKHLA

Module : **Machine Learning**

Professeur : **Pr. DIB**

PROJET Système de Recommandation

Machine Learning

Rapport théorique et pratique complet

Sur Jupyter NoteBook

Réalisé par :

Lucien YAOGO

Filière : Cybersécurité & IA

Année universitaire : 2025-2026

Table des matières

Table des figures

Introduction

Avec l'essor des plateformes numériques et la croissance exponentielle des données générées par les utilisateurs, les systèmes de recommandation sont devenus des outils essentiels pour personnaliser l'accès à l'information. Dans le domaine du divertissement, notamment sur les plateformes de streaming comme Netflix ou Amazon Prime Vidéo, ces systèmes permettent de proposer des films adaptés aux préférences des utilisateurs, améliorant ainsi leur expérience et leur engagement.

Ce projet, réalisé dans le cadre du module *Machine Learning*, porte sur la conception d'un système de recommandation de films à partir du jeu de données **MovieLens**. L'objectif est d'exploiter des données réelles comprenant les films, leurs genres et les évaluations attribuées par les utilisateurs afin de développer un moteur de recommandation capable de suggérer des films pertinents.

Objectifs du Projet

Ce travail pratique a pour objectif principal de concevoir et d'implémenter un système de recommandation de films (MovieLens) basé sur le contenu. Plus spécifiquement, nous visons à :

- Explorer et prétraiter un dataset de films pour extraire des caractéristiques pertinentes
- Vectoriser le contenu textuel (overviews et genres) pour le rendre exploitable par des algorithmes de machine Learning
- Calculer des similarités entre films à l'aide de la similarité cosinus
- Implémenter un moteur de recommandation capable de suggérer des films similaires à un film donné.

Méthodologie Adoptée

Notre approche repose sur le filtrage basé sur le contenu, une technique qui recommande des éléments similaires à ceux qu'un utilisateur a appréciés dans le passé, en se basant sur les caractéristiques intrinsèques des items. Pour les films, ces caractéristiques incluent :

- Les overviews
- Les genres
- Les mots-clés (extraits du contenu textuel)

Le processus comprend plusieurs étapes :

- Prétraitement des données : Nettoyage et fusion des colonnes pertinentes
- Vectorisation : Transformation du texte en représentations numériques
- Calcul de similarité : Utilisation de la similarité cosinus pour mesurer la proximité entre films
- Recommandation : Suggestion des N films les plus similaires

Structure du Rapport

Ce rapport est organisé comme suit :

- **Partie 1 :** Téléchargement, Chargement, Exploration et prétraitement des données
- **Partie 2 :** Extraction de caractéristiques et vectorisation
- **Partie 3 :** Calcul de similarités entre les films et Implémentation du moteur de recommandation
- **Partie 4 :** Système de recommandation personnalisé
- **Partie 5 :** Conclusion et perspectives d'amélioration

Outils et Technologies

Le projet a été réalisé avec les technologies suivantes :

Langage : Python 3.11

Bibliothèques principales :

- Pandas pour la manipulation des données
- Scikit-learn pour la machine Learning et la vectorisation
- NumPy pour les calculs numériques
- Matplotlib pour la visualisation
- Données : Dataset de films contenant 10,000 entrées avec informations sur les titres, overviews et genres etc.

Partie 1 : Chargement, Exploration et Prétraitement des données

1. Téléchargement et Chargement du Dataset

a) Téléchargement

La première étape du projet consiste à préparer les données nécessaires à la mise en place du système de recommandation. Le fichier utilisé provient du dataset MovieLens et contient principalement des informations sur les films ainsi que les évaluations attribuées par les utilisateurs.

Lien de téléchargement :

https://www.kaggle.com/datasets/ahsanaseer/top-rated-tmdb-movies-10k?fbclid=IwAR2MpWrWpcw2QNCv_FZg2losjBh9xAvhrqtnZBO9K-QS6PHI1aHkdB6qLao

Nous allons télécharger le fichier nommé : '**top10K-TMDB-movies.csv**'

Ce fichier est sous format compresser (**.zip**), donc il est nécessaire de le compresser avant de le charger pour le début de notre du projet. Dans ce cas, nous avons jugé nécessaire de renommer le fichier comme « **movies.csv** ».

b) Chargement du dataset

Avant tout, il est crucial d'importer tous les bibliothèques python nécessaire pour le TP :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Figure 1 : Importation des bibliothèques

Cette phase consiste à charger notre donnée dans l'environnement Python avec le Notebook de jupyter à l'aide de bibliothèques adaptées à la manipulation de données comme ***pandas***.

- Chargement des données

```
import pandas as pd
```

```
data = pd.read_csv('movies.csv')
data.head()
```

	id	title	genre	original_language	overview	popularity	release_date	vote_average	vote_count
0	278	The Shawshank Redemption	Drama,Crime	en	Framed in the 1940s for the double murder of h...	94.075	1994-09-23	8.7	21862
1	19404	Dilwale Dulhania Le Jayenge	Comedy,Drama,Romance	hi	Raj is a rich, carefree, happy-go-lucky second...	25.408	1995-10-19	8.7	3731
2	238	The Godfather	Drama,Crime	en	Spanning the years 1945 to 1955, a chronicle o...	90.585	1972-03-14	8.7	16280
3	424	Schindler's List	Drama,History,War	en	The true story of how businessman Oskar Schind...	44.761	1993-12-15	8.6	12959

Figure 2: Chargement de données

Comme on peut constater dans la figure ci-dessus, notre fichier a été bien chargé et cela nous permet d'afficher les premières lignes et les colonnes constituant notre fichier « **movies.csv** » avec la fonction ***head()***.

Il est possible d'afficher les dernière lignes de notre dataset avec la fonction ***tail()***.

```
data.tail()
```

	id	title	genre	original_language	overview	popularity	release_date	vote_average	vote_count
9995	10196	The Last Airbender	Action,Adventure,Fantasy	en	The story follows the adventures of Aang, a yo...	98.322	2010-06-30	4.7	3347
9996	331446	Sharknado 3: Oh Hell No!	Action,TV Movie,Science Fiction,Comedy,Adventure	en	The sharks take bite out of the East Coast whe...	12.490	2015-07-22	4.7	417
9997	13995	Captain America	Action,Science Fiction,War	en	During World War II, a brave, patriotic Americ...	18.333	1990-12-14	4.6	332
9998	2313	In the Name of the King: A	Adventure,Fantasy,Action,Drama	en	A man named Farmer sets	15.159	2007-11-30	4.7	668

Figure 3 : Afficher les dernières lignes

2. Exploration et prétraitement des données

a) Exploration des données avec 'pandas'

Une exploration initiale va permettre d'analyser la structure des fichiers, d'identifier les différentes variables disponibles :

- L'identifiants des films (Id),
- Le titre des films (titre)
- Le genre des films (genre),
- La langue originale des films (original_language),
- L'aperçu (overview),
- La popularité du film (popularity),
- La date de publication (release_date),
- La moyenne de note de chaque film (vote_average),
- Le nombre de note par film (vote_count).

Cela permet de comprendre la distribution globale des données.

Le bibliothèque ***pandas*** permet également de faire d'autre analyse profonde pour avoir des informations importantes de notre dataset « **movies.csv** » comme suit :

✓ Pour connaitre le nombre de film et le nombre de variables (colonnes) :

```
data.shape
```

```
(10000, 9)
```

Figure 4 : Afficher les nombres de films & variables du dataset

Avec figure, on peut clairement dire qu'il y a 10000 films, et 9 colonnes.

- ✓ Pour afficher les colonnes de notre dataset

```
data.columns
```

```
Index(['id', 'title', 'genre', 'original_language', 'overview', 'popularity',
       'release_date', 'vote_average', 'vote_count'],
      dtype='object')
```

Figure 5 : Afficher les colonnes du dataset

- ✓ Pour afficher le contenu d'une colonne. Exemple la colonne 'genre' :

```
data['genre']
```

```
0                                       Drama,Crime
1                                 Comedy,Drama,Romance
2                                       Drama,Crime
3                                  Drama,History,War
4                                       Drama,Crime
                              ...
9995                          Action,Adventure,Fantasy
9996    Action,TV Movie,Science Fiction,Comedy,Adventure
9997                        Action,Science Fiction,War
9998                    Adventure,Fantasy,Action,Drama
9999                           Thriller,Action,Crime
Name: genre, Length: 10000, dtype: object
```

Figure 6 : Afficher le contenu d'une colonne

b) Prétraitement des données

Le prétraitement des données consiste à gérer :

- ✓ Les valeurs manquantes

Dans cette section, la fonction qui permet de rechercher les valeurs manquantes est la fonction ***isnull ().*** Cette fonction est ***booléenne***, c'est-à-dire que son utilisation unique retourne des résultats ***True*** (si la valeur est null) ou ***false*** (si la valeur est non-null).

```
data.isnull().head()
```

	id	title	genre	original_language	overview	popularity	release_date	vote_average	vote_count
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False

Figure 7 : Utilisation unique de la fonction booléenne .isnull()

Cette figure montre qu'il est difficile de trouver les valeurs manquantes de notre dataset. On ajoute la fonction ***sum()*** pour mieux voire les valeurs manquantes.

```
data.isnull().sum()
```

```
id                   0
title                0
genre                3
original_language    0
overview            13
popularity           0
release_date         0
vote_average         0
vote_count           0
dtype: int64
```

Figure 8 : Association de isnull().sum() pour mieux afficher les valeurs manquantes du dataset

Cette application de la fonction ***sum()***(compte le nombre de valeur manquante du dataset) en accompagnement de la fonction ***isnull()*** montre dans la figure ci-dessus qu'il y a trois (3) lignes nulls (ou vide) dans la colonne *'genre'* et treize (13) lignes nulls dans le colonne *'overview'*.

- ✓ Supprimer les les valeurs manquantes de dataset

```
: data = data.dropna(axis=0)
  data.shape

: (9985, 9)
```

Figure 9 : Supprimer les valeurs manquantes

Nous pouvons voire qu'après la suppression des valeurs manquantes suivant l'axe horizontale (***axis=0***) ou les lignes, le dataset comprend :

- **9985 lignes** (Comparé à **10000 lignes** avant suppression)
- **9 colonnes**

✓ Pour supprimer les éventuels doublons

```
data.drop_duplicates()
```

Figure 10 : Supprimer les éventuels doublons

✓ Pour supprimer une colonne de notre dataset

```
data = data.drop(['original_language', 'release_date'], axis=1)
data.head()
```

	id	title	genre	overview	popularity	vote_average	vote_count
0	278	The Shawshank Redemption	Drama,Crime	Framed in the 1940s for the double murder of h...	94.075	8.7	21862
1	19404	Dilwale Dulhania Le Jayenge	Comedy,Drama,Romance	Raj is a rich, carefree, happy-go-lucky second...	25.408	8.7	3731
2	238	The Godfather	Drama,Crime	Spanning the years 1945 to 1955, a chronicle o...	90.585	8.7	16280
3	424	Schindler's List	Drama,History,War	The true story of how businessman Oskar Schind...	44.761	8.6	12959
4	240	The Godfather: Part II	Drama,Crime	In the continuing saga of the Corleone crime f...	57.749	8.6	9811

Figure 11 : Suppression des colonnes

Ici, nous voyons que les colonnes *'original_language'* et *'release_date'* ne figurent plus dans le dataset suivant les colonnes (***axis=1***).

✓ Statistiques descriptives

```
data.describe()
```

	id	popularity	vote_average	vote_count
count	10000.000000	10000.000000	10000.000000	10000.000000
mean	161243.505000	34.697267	6.621150	1547.309400
std	211422.046043	211.684175	0.766231	2648.295789
min	5.000000	0.600000	4.600000	200.000000
25%	10127.750000	9.154750	6.100000	315.000000
50%	30002.500000	13.637500	6.600000	583.500000
75%	310133.500000	25.651250	7.200000	1460.000000
max	934761.000000	10436.917000	8.700000	31917.000000

Figure 12 : Statistiques descriptives

Cette étape montre de informations nécessaires sur notre dataset, comme :

- **Count :** compte les nombres de lignes du dataset
- **Mean :** calcule la moyenne sur les valeurs numeriques du dataset
- **Std :** calcule l'écartype des données numeriques
- **Min :** calcule le minimum
- **Max :** calcule le maximum

✓ L'utilisation de la fonction ***info ()***

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            10000 non-null  int64
 1   title         10000 non-null  object
 2   genre         9997 non-null   object
 3   overview      9987 non-null   object
 4   popularity    10000 non-null  float64
 5   vote_average  10000 non-null  float64
 6   vote_count    10000 non-null  int64
dtypes: float64(2), int64(2), object(3)
memory usage: 547.0+ KB
```

Figure 13 : La fonction info ()

Cette fonction permet aussi d'afficher les informations importantes sur les *'types'* nommé ***'Dtype'*** des valeurs contenant dans notre dataset.

c) *Exploration des données en utilisant des Graphes avec Matplotlib*

La visualisation graphique joue un rôle fondamental dans l'analyse des données du projet. Elle permet de mieux comprendre la structure du dataset MovieLens, d'identifier rapidement les tendances générales, telles que :

✓ La distribution des notes moyennes (*'vote_average'*) par film

```
plt.hist(data['vote_average'], edgecolor='white', bins=20)
plt.xlabel('Note moyenne')
plt.ylabel('Nombre de films')
plt.title('Distribution des notes moyennes')
plt.grid(True, alpha=0.3)
plt.show()
```

Nous avons utilisé ici l'histogramme pour représenter les graphes.

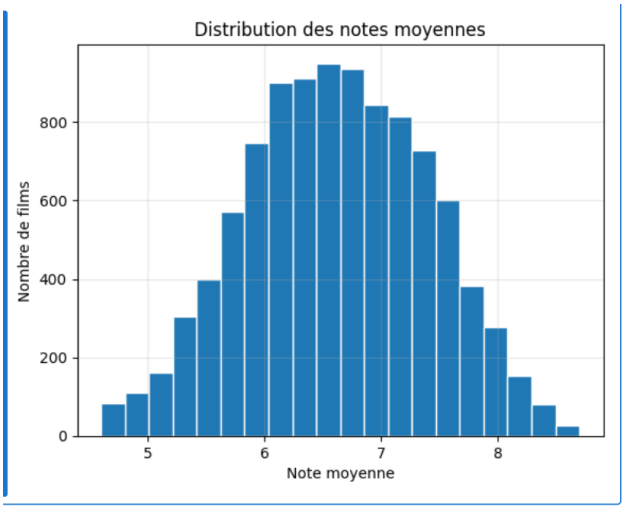

Figure 14 : La distribution des notes moyenne par films

✓ La distribution de nombre de votes par film(*'vote_count'*)

```
plt.figure(figsize=(10, 6))
plt.hist(data['vote_count'], edgecolor='white', bins=30)
plt.xlabel('Nombre de votes')
plt.ylabel('Nombre de films')
plt.title('Distribution du nombre de votes par film')
plt.grid(True, alpha=0.3)
plt.show()
```

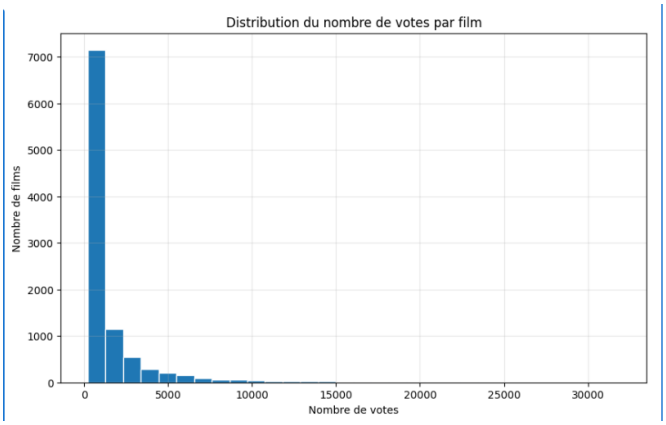

Figure 15 : Distribution du nombre de votes(vote_count) par film

✓ Relation entre les notes moyennes (*'vote_ count'*) & les nombre de notes (*'vote_average'*) par film.

```
plt.scatter(data['vote_count'], data['vote_average'], alpha=0.5)
plt.xlabel('Nombre de votes')
plt.ylabel('Note moyenne')
plt.title('Relation votes vs notes')
plt.show()
```

Nous avons utilisé la fonction ***scatter()*** pour ce graphe. Cela à pour but de montrer qu'il existe plusieurs types de graphe dans la bibliothèque ***matplotlib***.

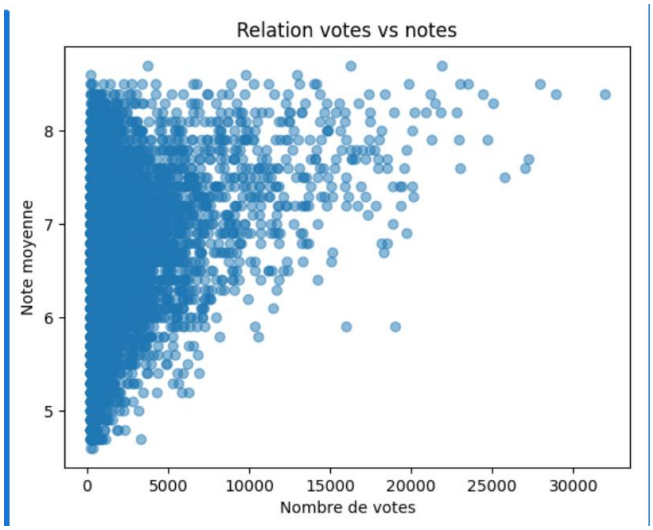

Figure 16 : Relation entre vote_count & vote_average

✓ Top 10 des films les plus notés

```
:
top_voted = data.nlargest(10, 'vote_count')
plt.barh(top_voted['title'], top_voted['vote_count'])
plt.xlabel('Nombre de votes')
plt.title('Top 10 films les plus votés')
plt.tight_layout()
plt.show()
```

Ici la fonction ***nlargest()*** est utilisée pour la représentation horizontale des graphes.

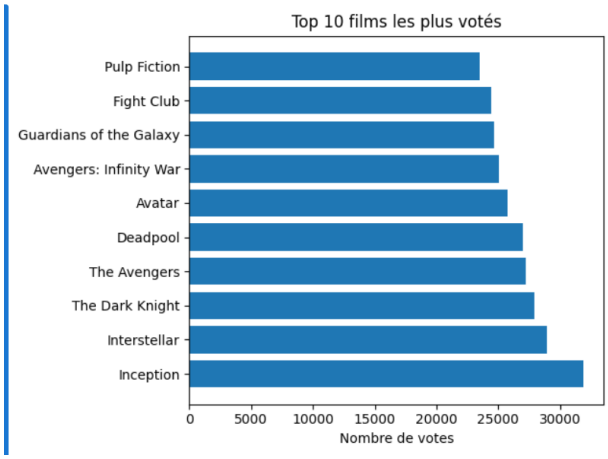

Figure 17 : Top 10 des films les plus notés

✓ Top 10 films les plus populaires

```
top_popular = data.nlargest(10, 'popularity')
plt.barh(top_popular['title'], top_popular['popularity'])
plt.xlabel('Popularité')
plt.title('Top 10 films les plus populaires')
plt.tight_layout()
plt.show()
```

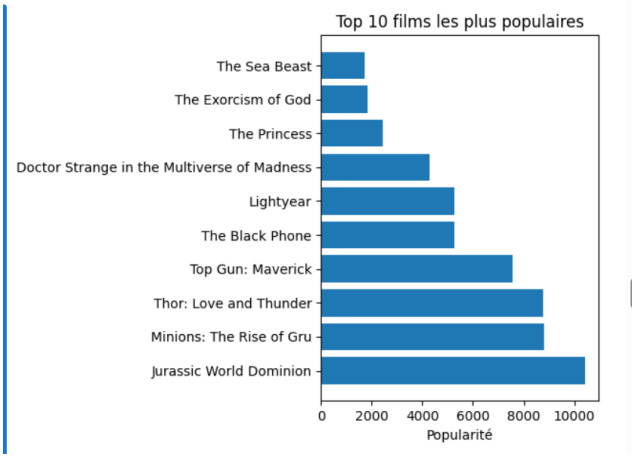

Figure 18 : Top 10 films les plus populaires

L'utilisation du module de visualisation est essentielle, car il transforme des données complexes en représentations clairs et interprétables. Grâce aux graphiques générés, il devient plus facile de justifier les choix de prétraitement, d'orienter la modélisation du système de recommandation. Ainsi la visualisation constitue un outil indispensable pour renforcer la compréhension, la fiabilité et la qualité globale du projet.

Partie 2 : Extraction de caractéristiques et vectorisation

1. Extraction des caractéristiques

Après le prétraitement, les données doivent être transformées en représentations numériques afin d'être utilisées par les algorithmes de Machine Learning. Cette étape repose sur l'extraction de caractéristiques pertinentes à partir des données disponibles. Dans notre dataset, nous avons pris en compte les valeurs de quatre (04) colonnes à savoir : *'id', 'title', 'overview', 'genre'*.

```
data = data[['id', 'title','overview', 'genre']]
data.head()
```

	id	title	overview	genre
0	278	The Shawshank Redemption	Framed in the 1940s for the double murder of h...	Drama,Crime
1	19404	Dilwale Dulhania Le Jayenge	Raj is a rich, carefree, happy-go-lucky second...	Comedy,Drama,Romance
2	238	The Godfather	Spanning the years 1945 to 1955, a chronicle o...	Drama,Crime
3	424	Schindler's List	The true story of how businessman Oskar Schind...	Drama,History,War
4	240	The Godfather: Part II	In the continuing saga of the Corleone crime f...	Drama,Crime

Figure 19 : L'extraction des caractéristiques

Les colonnes extraites constituent une information essentielle pour la recommandation.

Pour ce TP, nous allons créer une nouvelle colonne ('***tags***') dont la valeur sera les valeurs concaténées de la colonne '***overview***' et '***genre***'. Cela a pour but de réduire la la largeur de notre nouvelle data.

```
data['tags'] = data['overview'] +data['genre']
data.head()
```

	id	title	overview	genre	tags
0	278	The Shawshank Redemption	Framed in the 1940s for the double murder of h...	Drama,Crime	Framed in the 1940s for the double murder of h...Drama,Crime
1	19404	Dilwale Dulhania Le Jayenge	Raj is a rich, carefree, happy-go-lucky second...	Comedy,Drama,Romance	Raj is a rich, carefree, happy-go-lucky second...Comedy,Drama,Romance
2	238	The Godfather	Spanning the years 1945 to 1955, a chronicle o...	Drama,Crime	Spanning the years 1945 to 1955, a chronicle o...Drama,Crime
3	424	Schindler's List	The true story of how businessman Oskar Schind...	Drama,History,War	The true story of how businessman Oskar Schind...Drama,History,War
4	240	The Godfather: Part II	In the continuing saga of the Corleone crime f...	Drama,Crime	In the continuing saga of the Corleone crime f...Drama,Crime

Figure 20 : Création de la colonne tags

Après la création de cette nouvelle colonne '***tags***', il n'est plus nécessaire de garder les colonnes '***overview***' et '***genre***'. D'où leur suppression de notre nouvelle dataset.

```
new_data = data.drop(columns=['overview', 'genre'])
new_data.head()
```

	id	title	tags
0	278	The Shawshank Redemption	Framed in the 1940s for the double murder of h...
1	19404	Dilwale Dulhania Le Jayenge	Raj is a rich, carefree, happy-go-lucky second...
2	238	The Godfather	Spanning the years 1945 to 1955, a chronicle o...
3	424	Schindler's List	The true story of how businessman Oskar Schind...
4	240	The Godfather: Part II	In the continuing saga of the Corleone crime f...

Figure 21 : Novelle data après suppression des colonnes 'overview' et 'genre'

2. Vectorisation du dataset

Cette phase de vectorisation permet de traduire notre nouvelle dataset nommée ***'new_data'*** en une forme de mathématique adaptée au calcul de similarités et à la modélisation du système de recommandation.

Dans le cadre de ce TP, le module utilisé est : '***CountVectorizer***' du bibliothèque '***scikit-learn***' de Python.

```
cv = CountVectorizer(max_features=10000, stop_words='english')
```

```
cv
```

CountVectorizer

▸ Parameters

Figure 22 : CountVectorizer

Cette section vise à la création d'un objet de vectorisation de texte avec scikit-learn :

- **CountVectorizer** transforme du texte en nombres pour que les algorithmes de Machine Learning puissent l'exploiter.
- **max_features=10000** : taille des lignes.
- **stop_words='english'**: Supprime les mots inutiles en Anglais comme is, to, as, at ...

L'étape suivante est de transformer toutes les données (textes) de la colonne '*tags*' en chiffre :

```
vector = cv.fit_transform(
    new_data['tags'].values.astype('U')).toarray()
```

```
vector.shape
```

```
(9985, 10000)
```

Figure 23 : Convertion en chiffre

- Elle lit tous les textes de tags.

- Elle apprend les mots importants.
- Elle convertit chaque film en un tableau de chiffres c.à.d. transformer les mots en un tableau de chiffre.
- Le résultat (vector) est une matrice de nombres utilisable par la machine Learning.

Partie 3 : Calcul de similarités et implémentation du moteur de recommandation

1. Calcul de similarités

Le cœur du projet repose sur la mise en place du moteur de recommandation. Une approche de filtrage collaboratif a été adoptée afin d'exploiter les similarités entre les films, en se basant sur leurs évaluations respectives.

```
similarity = cosine_similarity(vector)
similarity
```

```
array([[1.        , 0.05634362, 0.13041013, ..., 0.07559289, 0.11065667,
        0.06900656],
       [0.05634362, 1.        , 0.07715167, ..., 0.        , 0.03636965,
        0.        ],
       [0.13041013, 0.07715167, 1.        , ..., 0.02300219, 0.0673435 ,
        0.09449112],
       ...,
       [0.07559289, 0.        , 0.02300219, ..., 1.        , 0.03253   ,
        0.03042903],
       [0.11065667, 0.03636965, 0.0673435 , ..., 0.03253   , 1.        ,
        0.04454354],
       [0.06900656, 0.        , 0.09449112, ..., 0.03042903, 0.04454354,
        1.        ]])
```

Figure 24 : Calcule de Similarité

Des mesures de similarité ont été calculées afin d'évaluer la proximité entre les vecteurs représentant les films. Ces similarités permettent d'identifier les éléments les plus proches et de prédire les films similaires à un utilisateur qui fait une recherche d'un titre de film recherché.

2. Implémentation du moteur de recommandation

À partir de ces calculs, un moteur de recommandation a été implémenté afin de générer des suggestions personnalisées. Pour chaque utilisateur, le système est capable de proposer une liste de films recommandés de type *Top-N*, basée sur les similarités calculées et les évaluations existantes.

```
new_data[new_data['title']=="Blood Simple"].index[0]
```

```
1995
```

```
distance = sorted(list(enumerate(similarity[278])),
                  reverse = True, key = lambda vector:vector[1])
for i in distance[0:5]:
    print(new_data.iloc[i[0]].title)
```

```
Dial M for Murder
Blood Simple
The Getaway
Mr. Brooks
Presumed Innocent
```

Figure 25 : Trie la similarité entre les films

Cette partie permet de trier les films similaires en tapant uniquement un ***'id'*** de film. Cela va automatiquement afficher les 5 premières films qui ont une similarité commune.

Partie 4 : Système de recommandation personnalisé

Le but de la création de cette fonction est de rendre le système plus interactif avec les utilisateurs.

Cela permettra à un utilisateur de taper le titre d'un film et le système va lui recommander les films similaires de ce dernier.

```
: def recommand(data):
    index = new_data[new_data['title']==data].index[0]

    distance = sorted(list(enumerate(similarity[index])),
                  reverse = True, key = lambda vector:vector[1])
    for i in distance[0:5]:
        print(new_data.iloc[i[0]].title)
```

```
n = input("Tapez le titre du film que vous recherchez : ")
print(f"Les films que nous vous recommandons similaire à ({n}) sont : \n")

recommand(n)
```

```
Tapez le titre du film que vous recherchez :  Avengers: Age of Ultron
Les films que nous vous recommandons similaire à (Avengers: Age of Ultron) sont :

Avengers: Age of Ultron
Iron Man
Iron Man 3
Deathstroke: Knights & Dragons - The Movie
Robot Overlords
```

Figure 26 : Système de recommandation personnalisé

Partie 5 : Conclusion et perspectives d'amélioration

1. Conclusion

Ce projet a permis de mettre en œuvre un système de recommandation de films en s'appuyant sur des techniques de Machine Learning et des données réelles issues du dataset MovieLens. Les différentes étapes, allant du prétraitement des données à la mise en place du système de recommandation, ont permis de comprendre les mécanismes fondamentaux des systèmes de recommandation.

Les résultats obtenus montrent que le système est capable de fournir des recommandations personnalisées pertinentes, bien que certaines limites subsistent.

2. Perspectives à améliorer

Parmi les perspectives d'amélioration possibles, on peut citer l'intégration de méthodes plus avancées telles que la décomposition matricielle (SVD), les mesures de performance métriques (RMSE, MAE). La mise en place d'un système hybride combinant filtrage collaboratif et recommandation basée sur le contenu, ou encore l'ajout de nouveau données de films.

Ce travail constitue ainsi une base solide pour le développement de systèmes de recommandation plus performants et plus adaptés aux besoins réels des utilisateurs.