Web scraping

sur IMDB et Rotten Tomatoes

SOMMAIRE

1/ SCRAPING

2/ CLEANING

3/ VISUALISATION

4/ SCRIPT

1/ SCRAPING

Répartition du travail en 2 étapes (IMDB)

Partie 1 : Scraping du site IMDB

>> Analyse de la structure HTML du site.

>> Récupération des balises contenant les informations nécessaires.

>> Scraping des données sur chaque film (titre/année/durée/genre)

Répartition du travail en 2 étapes (ROTTEN)

Partie 2 : Scraping du site ROTTENTOMATOES

>> Analyse de la structure HTML du site.

>> Blocage sur un élément du dom qui n'était pas accessible via les balises html (#shadowroot).

>> Récupération des données non complètes (problème de titre du film différent de notre recherche)

Fonction pour récupérer les données sur IMDB

```
def imdb_fetch():

    # Create a list for later use (adding each movie data into this list)
    movies_data = []

    # Starting page
    page = 1

    # Looping through the five page of top 250 movies
    while page <= 250:
        url = f'https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&start={page}

        # Getting data from imdb url
        data = requests.get(url, headers=headers)

        # Add html data to BeautifulSoup
        soup =  BeautifulSoup(data.text, 'html.parser')

        # Loop through each div with the content that we need (class = 'lister-item-content')
        for div in soup.find_all('div', { 'class' : 'lister-item-content' }):
            # Get movie's title and keep only the text
            title = div.find('a')
            title_text = title.text
```

Fonction pour créer un fichier csv de nos données IMDB

```
def create_imdb_dataframe(data):
    imdb_data = data
    # Create dataframe from movies_data list
    movies_df = pd.DataFrame (imdb_data, columns = ['title', 'year_of_release', 'duration_in_minutes', 'genre'

    # Keep only useful characters and change type of duration_in_minutes
    movies_df["duration_in_minutes"] = movies_df["duration_in_minutes"].str[:3].astype(int)
    movies_df["year_of_release"] = movies_df["year_of_release"].str[1:5]

    # Check if dir_name already exists before creating csv file
    if not os.path.exists(dir_name):
        os.mkdir(dir_name)
        print("Directory " , dir_name ,  " Created ")
    else:
        print("Directory " , dir_name ,  " already exists")

    # Export the dataframe with to_csv()
    movies_df.to_csv(f'{dir_name}/top_250_imdb_eng_scraper_module.csv', encoding='utf-8', index=False)
```

Fonction pour récupérer les scores sur rottentomatoes

```
def rotten_scores_fetch(imdb_dataframe):
    # Define empty list
    score_data = []

    # Define header language for getting data in english
    headers = {"Accept-Language": "en-US,en;q=0.5"}

    # Define static part of the url
    uri = 'https://www.rottentomatoes.com/m/'

    # Loop through titles in the dataframe
    for title in imdb_dataframe['title']:
        changed_title = title.replace(" ", "_").replace("The_", "").replace(":", "").replace(".", "").replace("'", "").replace
"").replace(",", "").replace("ä", "").replace("__","_")
        url = f'{uri}{changed_title}'
        data = requests.get(url, headers=headers)

        if data.status_code == 404:
            score_list = ['not found', 'not_found']
            score_data.append(score_list)

        else:
            soup = BeautifulSoup(data.text, 'html.parser')
```

Fonction pour créer un fichier csv des scores rottentomatoes

```
def create_rotten_df(score_data, imdb_dataframe):
    # Store imdb_dataframe in final_df
    final_df = imdb_dataframe

    # Create score_df with score_data from previous function
    score_df = pd.DataFrame(score_data, columns = ['tomato_meter','audience_score'])

    # Create new columns in our final dataframe
    final_df['tomato_meter'] = score_df['tomato_meter']
    final_df['audience_score'] = score_df['audience_score']

    # Check if dir_name already exists before creating csv file
    if not os.path.exists(dir_name):
        os.mkdir(dir_name)
        print("Directory " , dir_name ,  " Created ")
    else:
        print("Directory " , dir_name ,  " already exists")

    # Export the dataframe with to_csv()
    final_df.to_csv(f'{dir_name}/top_250_imdb_with_rottenscore.csv', encoding='utf-8', index=False

    return final_df
```

2/ CLEANING

Technique de traitement et cleaning

- Export, en python, dans une base de données pour traiter les données en SQL

🎬 MOVIES IN SQL

Entrée [58]:
```
import pandas as pd
import numpy as np
from sqlalchemy import create_engine
```

Entrée [59]:
```
df = pd.read_csv("data/data.csv")
```

Entrée [63]:
```
# database connection
hostname="127.0.0.1"
dbname="hello_movies"
uname="root"
pwd="pixel"

# create SQLAlchemy engine to connect to MySQL Dat
engine = create_engine("mysql+pymysql://{user}:{pw

# connect to the database
engine.connect()
```

- Cleaning via SQL : remplacer les valeurs incorrectes
- Transformer via Power Query pour la visualisation : modification des types de données selon les colonnes et pivot

3/ VISUALISATION
SUR POWER BI

US BOX OFFICE

1921 > 2021

Premier film

1921

Film le plus récent

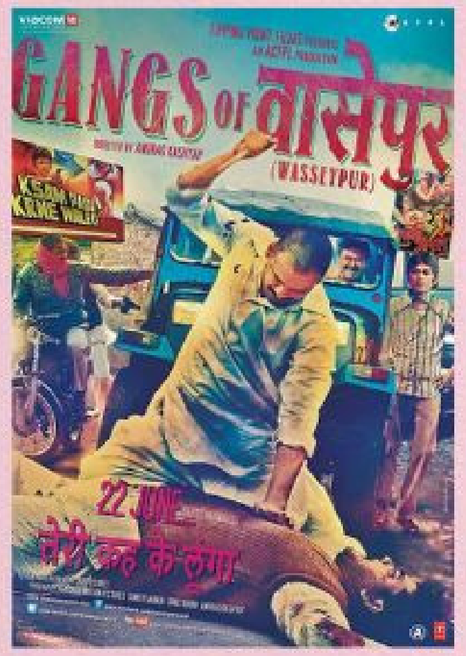
2021

Genre de films les plus populaires

Film le plus long

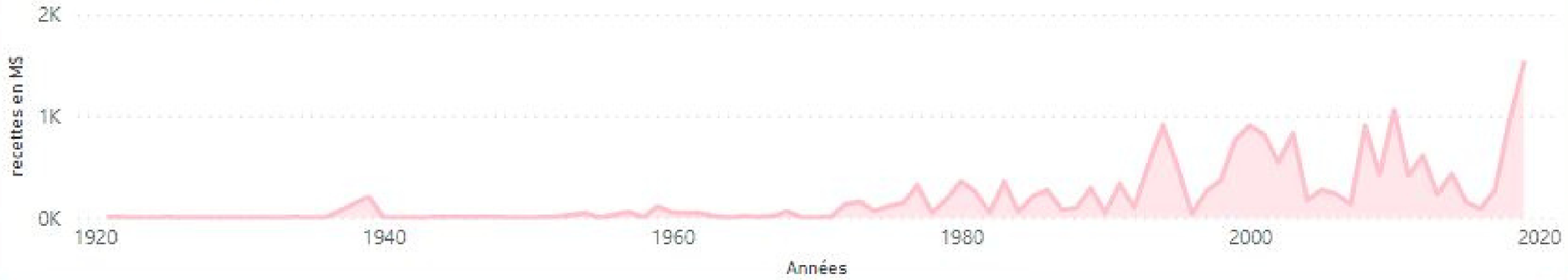
321

minutes, soit 5h21

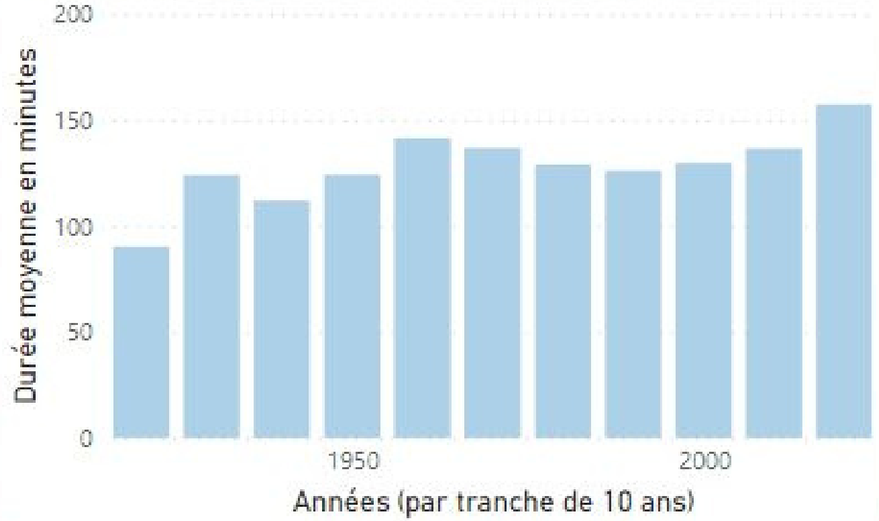


Gangs of Wasseypur
Film "Bollywood" - 2012

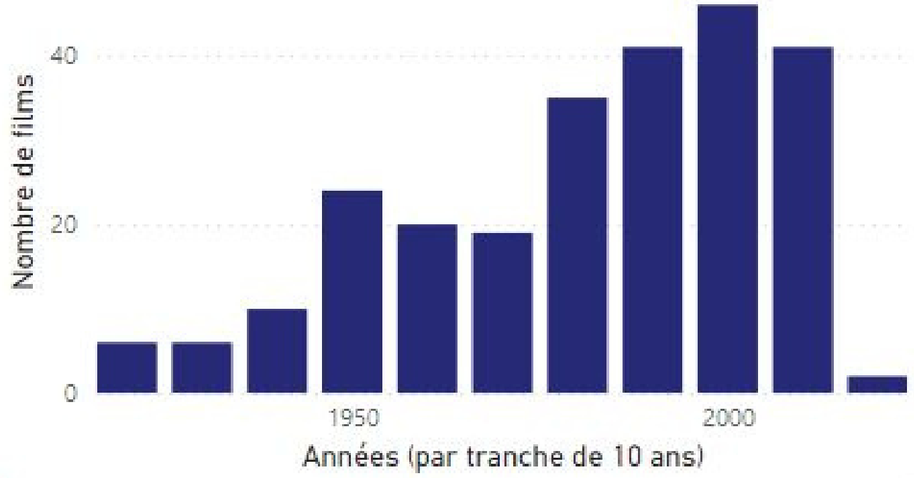
Recette totale par année



Durée moyenne des films de 1921 à nos jours



Nombre de films au US box office par années



4/ AUTOMATISATION
DU PROCÉDÉ

Technique de traitement et cleaning

Utilisation de notre script final pour créer un fichier csv avec les données IMDB et un autre avec les mêmes données mais aussi les scores de Rottentomatoes

1 Scraper.py Instructions

This notebook will guide you if you want to have a csv file with data about the top 250 movies. You can go to 4 and get only a dataframe with imdb data, or execute every steps in order to get 2 csv (one with only imdb data, and one with imdb data with rottentomatoes scores added).

- 1.1 STEP 1 : import scraper.py
- 1.2 STEP 2 : use imdb_fetch() function in order to fetch data about top 250 movies on IMDB and store them in imdb_data variable
- 1.3 STEP 3 : use create_imdb_dataframe() function in order to create a new csv file with data about top 250 movies (will also create a folder /data in your current directory if it doesn't exist yet)
- 1.4 STEP 4 (optional) : check the dataframe created
- 1.5 STEP 5 : use rotten_scores_fetch() function in order to fetch tomatoscore and audience score of the movies, from rottentomatoes (Caution : this step will take between 5 and 10 min, so you need to wait before executing cells below this one)
- 1.6 STEP 6 (optional) : check the list created
- 1.7 STEP 7 : use create_rotten_df() function in order to create a new csv file with rottentomatoes scores added to the initial dataframe (will also create a folder /data in your current directory if it doesn't exist yet)
- 1.8 STEP 7 (optional) : check the final dataframe created