Data Analytics

Movie Recommander

Vincent Wakim

March, 2023

Table of content

Table of content 2
Introduction 2
Plan 3
Data and data sources 3
The Movie Dataset 5
Data collections 7
Data Cleaning 8
Export to MySQL 9
Data visualization 10
ERM 14
SQL 14
Conclusion 20

Introduction

Over the years, movie recommendation systems have become increasingly popular among movie enthusiasts. These systems analyze user preferences and recommend films that are likely to be enjoyed by the user. However, they often fail to provide a good way for the user to act on the recommendations given. Therefore, this project aims to address this issue by exploring the relationship between user preferences and various movie attributes, focusing on a selected film.

This project will examine data from a variety of sources, including movie attributes and user preferences. Specifically, it will use data from popular movie recommendation platforms like IMDb for example. The data includes various features, such as movie genre, director, actor, and user ratings.

The goal of the project is to investigate the relationships between the movie attributes and the user preferences of the selected film, as well as the recommendations given by the system over time. Additionally, the project aims to identify hidden patterns and trends in the data to create a more accurate movie recommendation model.

Overall, the objective of this project is to provide movie enthusiasts with a more effective movie recommendation system based on a selected film.

Data and data sources

The movie recommender project will use two data sources: a movie dataset and a IMDb dataset. The movie metadata dataset includes information on each movie, such as title, director, actors, and genre. The IMDb dataset includes titles and credits.

Since the data is personal, we will follow data protection regulations and only provide select portions of the datasets in our report and presentation.

The Movie Dataset

The data exported from The Movie Dataset consists of 5 data files:

- movie_metadata.csv
- credits.csv
- keywords.csv
- links.csv
- ratings.csv

Field Name	Description
adult	Boolean value to state if it is an adult sample or not.
belongs_to_collection	If it belongs to a connection of multiple movies
budget	This is the budget spent per movies
genres	The column describes the type of movie it is.
id	Integer value that define the movie id
imdb_id	Defines the movie id on imdb
original_language	String values that define the language of the movie

original_title	This describes the original title of the movie
overview	The ‘overview’ part sums up the movie with a little sentence
popularity	It defines the popularity of the movie on a specific scale.
poster_path	Gives us an image associated to the movie
production_companies	This is the company production of the movie
production_countries	The country where the movie takes place
release_date	Date that the movie came out to theaters
revenue	The revenue they made of this specific movie
run_time	The run time is pretty much the time of the movie
spoken_languages	The language speaking in the movie
status	Released movie or not
tagline	This contains the taglines of each movie
title	Title of the movie
vote_average	This is based on an average vote of the movie
vote_count	Number of votes on a movie
home_page	Gives us a link to a site

- credits.csv

cast	The cast column gives us the role the actor plays
crew	The crew of the production of the movie
id	Gives us the id of the cast and the crew

- keywords

id	This provides us the id for the keywords
keywords	Keywords themselves sum up the main movie

- links

movieId	Contains IMDB and TMDB IDs of all movies featured in the ratings.csv file
imdbId	References a movie id from the IMDb website
tmdbId	Foreign key that links to the metadata

- ratings

userId	ID of the user posting the rating of a movie
movieId	ID of a specific movie
rating	Rating is a 5 star system to rate
timestamp	Datetime object stored as an integer

Data collections

Since the data was taken from Kaggle it was already in a usable format and using it required only to read it with pandas. I only needed to download it and save it to my computer.

Data Cleaning

Since the data contained a lot of dictionary-like objects I had to deserialize them.

credit.csv contained a list of dictionaries that represented roles or staff position, keywords.csv contained à similar objects for keywords, and movies_metadata.csv also had some for language, genres, movie collections and production companies.

In order to handle them I decided to keep only à list of ids, and have a separate table to map them to their actual values. Those mapping tables would only be needed to present human-readable data to the user if needed. For the recommendation process, I will only use the ids.

While producing the mapping table I checked that an id was always mapped to the same value. There were some exception, of course, and they are as follows :

```
Inconsistency on id 99692 : names : Liao Fan /  廖凡
Inconsistency on id 111690 : names : Takako Matsu /  松隆子
Inconsistency on id 117642 : names : Jason Momoa /  杰森·莫玛
Inconsistency on id 9779 : names : Morris Chestnut /  Моррис Честнат
Inconsistency on id 23764 : names : Erika Eleniak /  Эрика Элениак
Inconsistency on id 9779 : names : Моррис Честнат /  Morris Chestnut
Inconsistency on id 117642 : names : 杰森·莫玛 /  Jason Momoa
```

After searching for those actor names we can see that they are only translations of the same name and that the id do represent unique persons. For the other objects (keywords, genre, language, companies, film collection) there were no mismatch.

There were 3 lines with odd data in movies_metada.csv. They, among other things, had ‘False’ as a value for production_companies. I discarded them as it was à tiny fraction of the total amount of movies.

movies_metada.csv had some inconsistent data type in the columns I needed to parse. Some fields with serialized data were sometimes empty. I assumed it was because there was no relevant data and replaced them with empty strings.

In order to use the result of the movie user rating I produced some aggregated values instead of keeping each individual notation. I also added the number of ratings given to have an estimation of the size of the audience.

```
1  aggregator_list = ['mean', 'std', 'max', 'min', 'count']
2  df_rating_stats = df_ratings.groupby('movieId').agg({'rating' : aggregator_list})
3  df_rating_stats.columns = aggregator_list
4  df_rating_stats['std'] = df_rating_stats['std'].fillna(0)
5  df_rating_stats = df_rating_stats.reset_index().sort_values('count', ascending=False)
```

Export to MySQL

Since the data contained a lot of serialized objects I had to make a choice. Normally I shouldn't store list-like data in à single row in an SQL table since that would defeat some of the purpose of a relational database. However the use I'll make of the data should not need such an extended table.

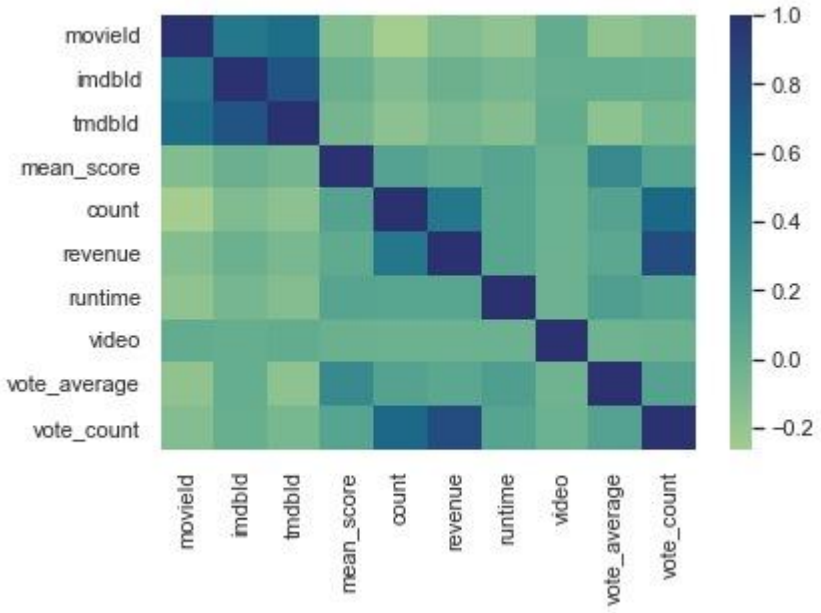
I decided to store those data as a comma separated list in à string field.

```
def is_na(x):
    return x != x or x is None or isinstance(x, pd._libs.missing.NAType)

def str_join(iterable, sep = ','):
    if is_na(iterable):
        return ''
    if isinstance(iterable, str) or isinstance(iterable, int):
        return str(iterable)
    return sep.join(str(i) for i in iterable)

df_keywords['keywords'] = df_keywords['keywords'].apply(str_join)
for col in METADATA_COLUMNS_TO_PARSE:
    df_metadata[col] = df_metadata[col].apply(str_join)
```

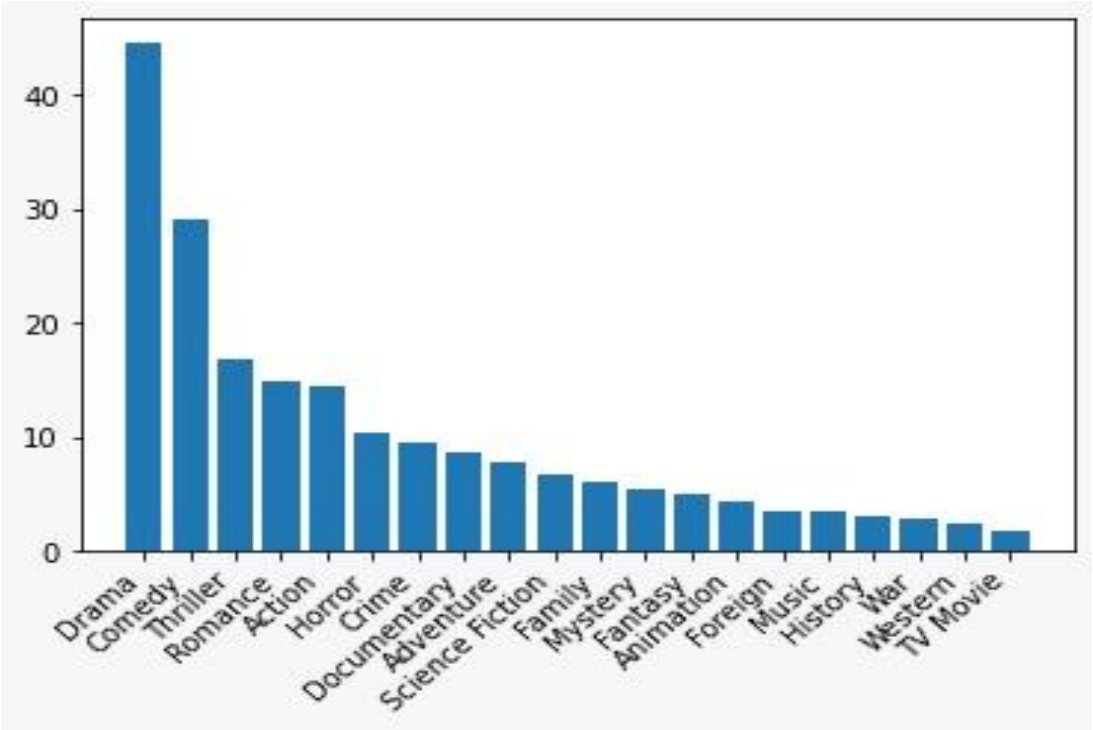
This should be sufficient unless I need to answer questions like "What are all the films using the keyword 'boy' ?" using only SQL. Fortunately, this data will be used to construct a list of clusters with python and will be loaded in its entirety so it shouldn't matter.

Data Visualization

I looked at the correlation of the numeric fields :

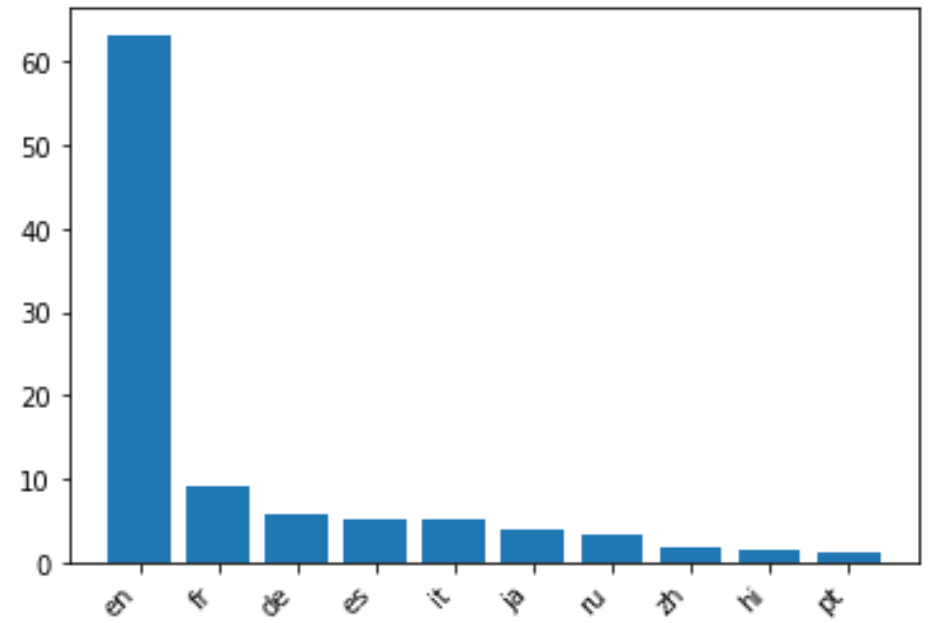


Surprisingly the score of the movie was not very correlated between the two platforms : (columns ‘vote_average’ and ‘mean_score’). I didn’t spot any meaningful correlation in the numeric fields. So I looked more closely into the genres of the movies :

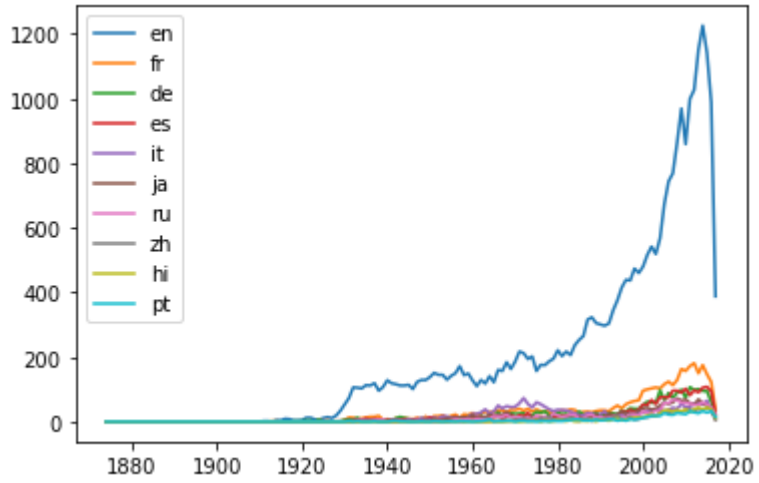


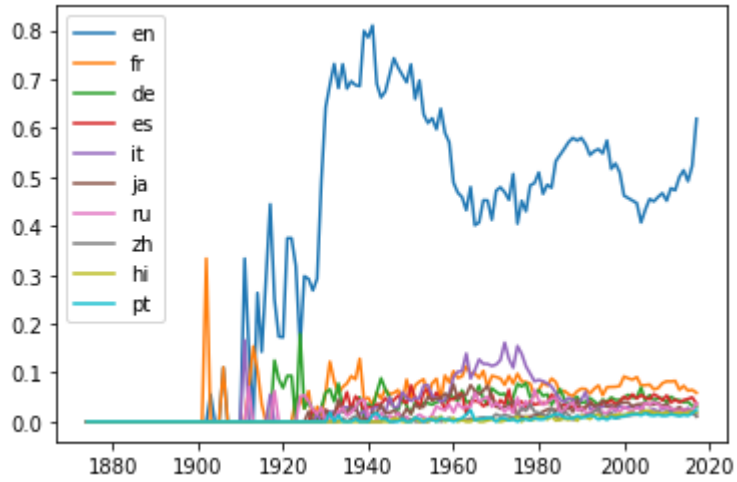
Above is displayed the popularity per genre (in percentage of presence in the totality of the movies). I noticed that some genres are quite rare and can then be a very good indicator of similarity between two movies !

I made a similar analysis on the spoken languages and found an overwhelming proportion of English speaking movies, leading me to believe this field will not be very helpful for the recommendation . Bellow the top 10 languages and in which proportion of the films they appear :

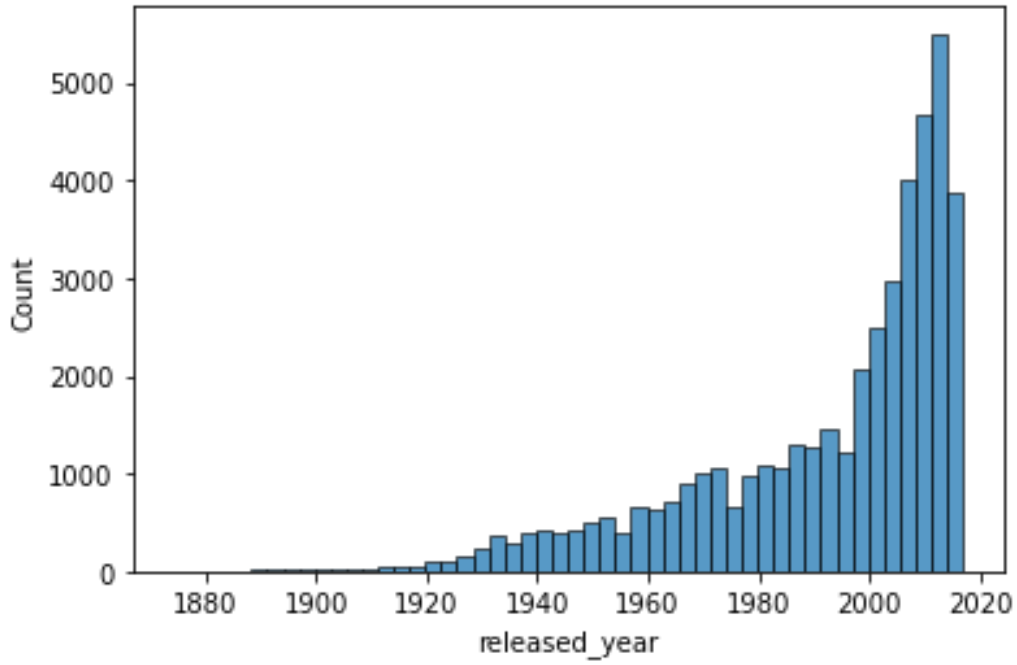


To see if this distribution evolved over time I plotted the number of film per spoken language and the proportion of film that included each of the top 10 languages for each year :

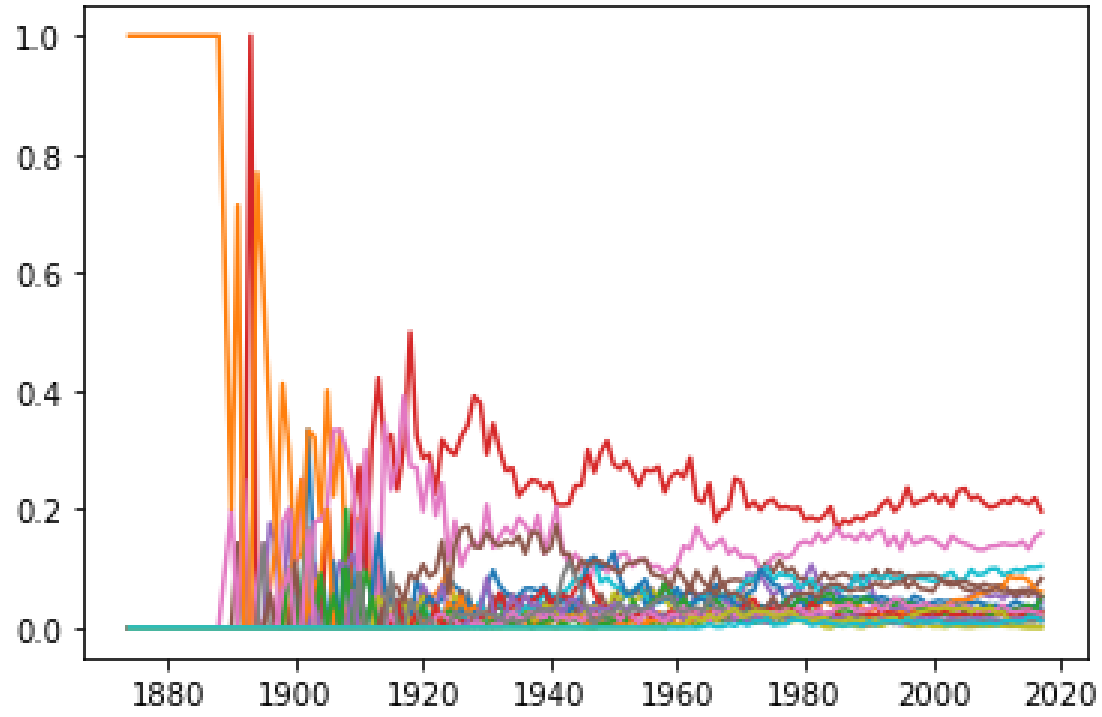




We can note a decrease in the number of movies in 2017, probably because of the data collection period. Also, as we could expect, English stayed predominant over time with some variability. I also displayed the number of films per year to confirm my observation on the data collection period and it confirms the data stops in 2017:



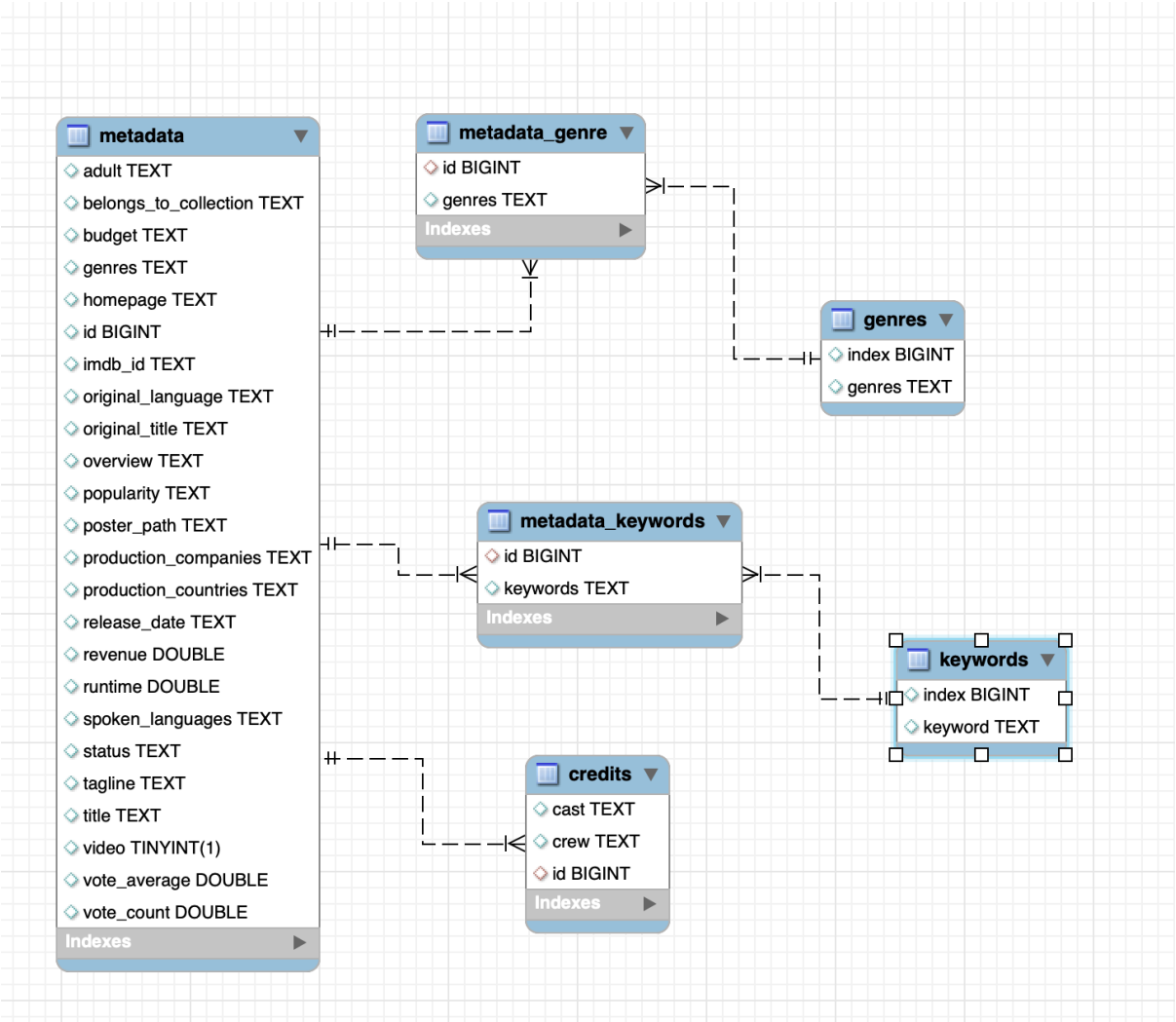
With a similar analysis on proportion of movies per genre (excluding the legend for clarity's sake):



We can see some changes over time but no major shift of the cinematographic scene.

ERM

This is an example of the diagram for our entity relation model that can be seen as follows :



SQL

This was the first query I ran to be able to link the tables :

```sql
SELECT links.movieId, imdbId, tmdbId, cast, crew, keywords, mean, count, adult, belongs_to_collection, budget, genres, homepage, original_
FROM credits
    INNER JOIN
    keywords
    ON credits.id = keywords.id
    INNER JOIN links
    ON credits.id = links.tmdbId
    INNER JOIN ratings
    ON links.movieId = ratings.movieId
    INNER JOIN metadata
    ON credits.id  = metadata.id;
```

Here are some queries to be able to dig a little bit more inside the data and have some nice information.

```
SELECT title, release_date
FROM metadata WHERE release_date is NOT NULL
ORDER BY release_date
LIMIT 5;
```

title	release_date
Passage of Venus	1874-12-09
Sallie Gardner at a Gallop	1878-06-14
Buffalo Running	1883-11-19
Man Walking Around a Cor…	1887-08-18
Accordion Player	1888-01-01

On this query, we have the minimum and the maximum runtime of a movie :

```
SELECT MIN(runtime) AS min_avg_runtime, MAX(runtime) AS max_avg_runtime, AVG(runtime) AS avg_runtime
FROM metadata;
```

min_avg_runti…	max_avg_runti…	avg_runtime
0	1256	94.12819945578833

We can see that there are movies of a minimum average run time at 0 minutes so I asked a query that could give them to me. There were so many of them that I had to put a limit to the query :

```
SELECT title FROM metadata WHERE runtime = 0
LIMIT 5;
```

title
Dream Man
Destiny Turns on the Radio
Dos Crímenes
The Beans of Egypt, Maine
The Run of the Country

Same thing for the maximum average run time that was 1256 minutes, I asked the same query but of course with the maximum average and it gave me a movie :

```
SELECT title FROM metadata WHERE runtime = 1256;
```

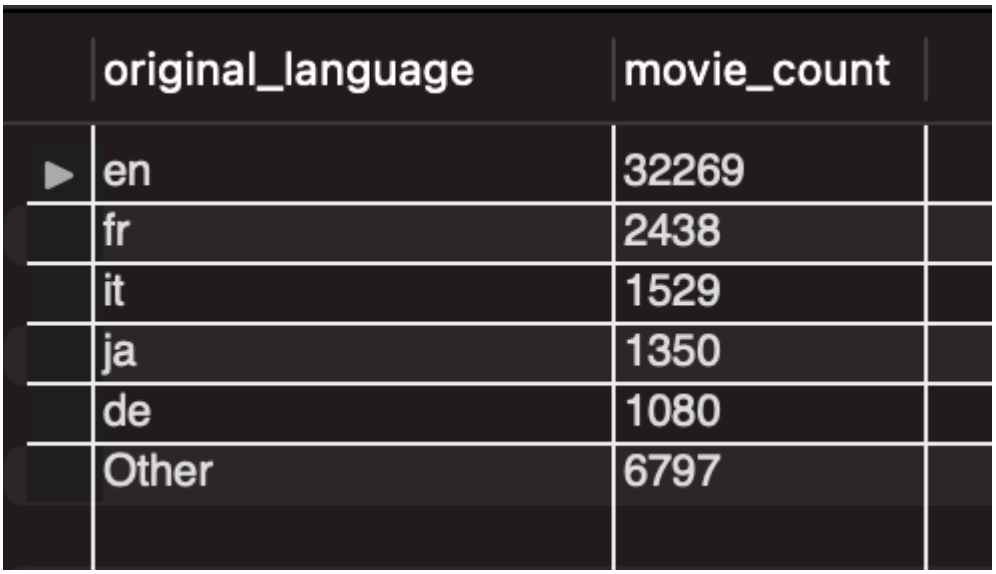
title
Centennial

I went to check for it on the internet and it turned out that it’s real.

For the last query, I searched for the most numbers of movie in a specific language :

```
SELECT case WHEN top_five.original_language IS NULL then "Other"
            ELSE top_five.original_language
            END as original_language
, SUM(all_films.num_films) AS movie_count
FROM (SELECT original_language, COUNT(*) AS num_films
    FROM metadata
    GROUP BY original_language
    ORDER BY COUNT(*) DESC) all_films
LEFT JOIN (SELECT original_language, COUNT(*) AS num_films
    FROM metadata
    GROUP BY original_language
    ORDER BY COUNT(*) DESC LIMIT 5) top_five
ON all_films.original_language = top_five.original_language
GROUP BY 1;
```

Here is the output :



original_language	movie_count
en	32269
fr	2438
it	1529
ja	1350
de	1080
Other	6797

You can tell that most of the movies from this dataset are in english, followed by France, Italy, Japan, Deutschland and all other languages that are in a category called “Other”

Conclusion

After this movie analysis which took into account the films published until 2017 Ihave a pretty good idea of what to use for our recommendation model. First of all, the genre is a very good indicator of the similarity of the two movies. Indeed there are multiple genres that can be combined for each movie and they are pretty well distributed across the dataset.
Even though there is à significant increase in the number of films, there is no major change of style, quality or type of film over time.
That way we can safely proceed without making any major adaptations around the release date of the film and still have a reliable way to compare movies even across a large time period.

The keywords will also be a relevant addition since they are very specific and numerous.

In order to move forward with the classifier I will focus on the categorical data such as genre and keyword, and will use a one-hot-like encoding. I will also most likely discard a lot of numerical feature such as the rating or revenue of the movies which seem irrelevant to recommendation.