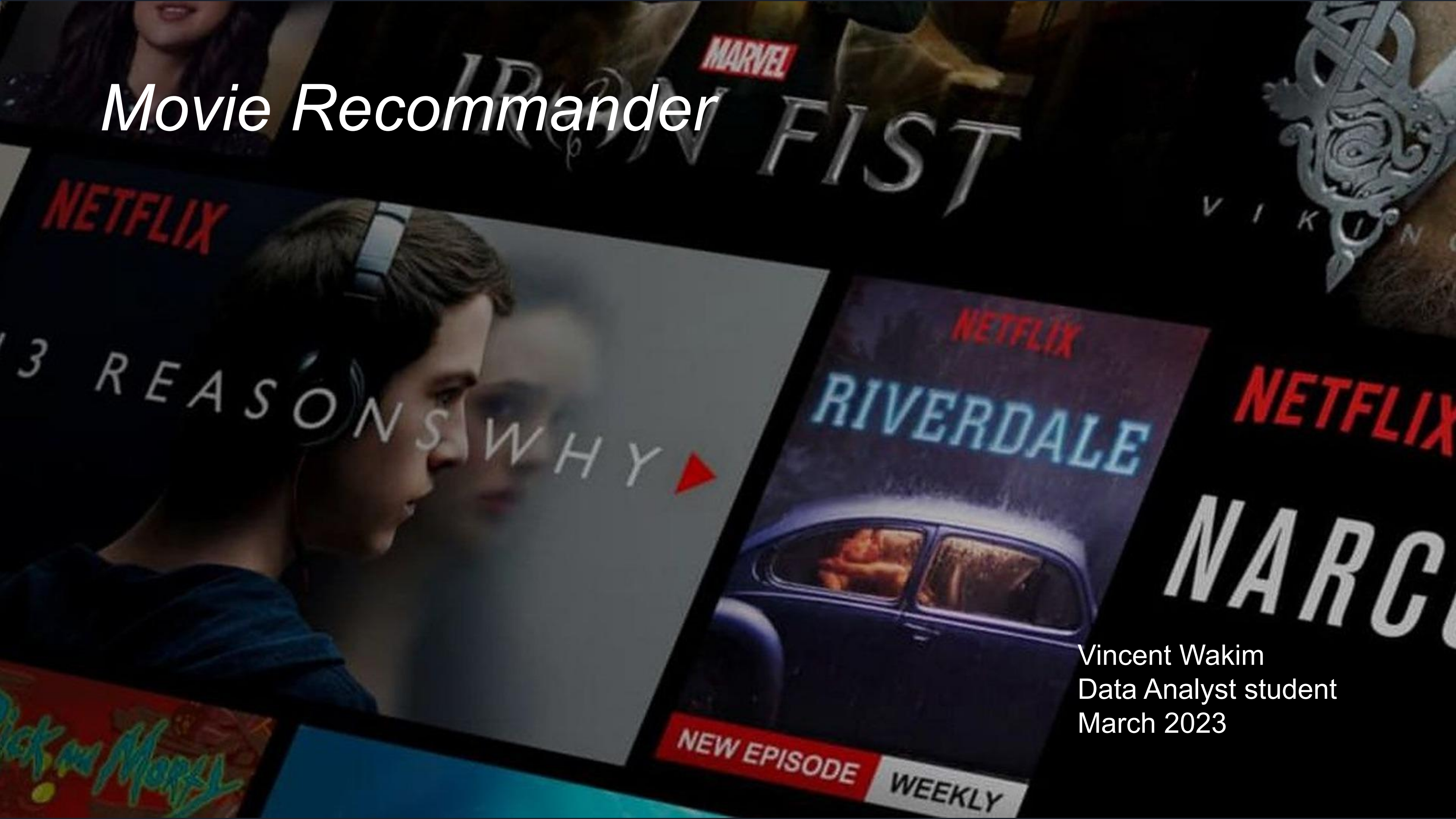
Movie Recommander



Vincent Wakim
Data Analyst student
March 2023

Table of contents

Data Collection

- Sourced from Kaggle

Files : - credits.csv

- movies_metadata.csv
- links.csv
- ratings.csv
- keywords.csv

```
movielens_credits: (45432, 3)
movielens_links: (45432, 2)
movielens_keywords: (45594, 3)
movielens_ratings: (26024289, 4)
movielens_metadata: (44985, 24)
```

Data Cleaning

- Some inconsistencies / missing values / duplicate ids

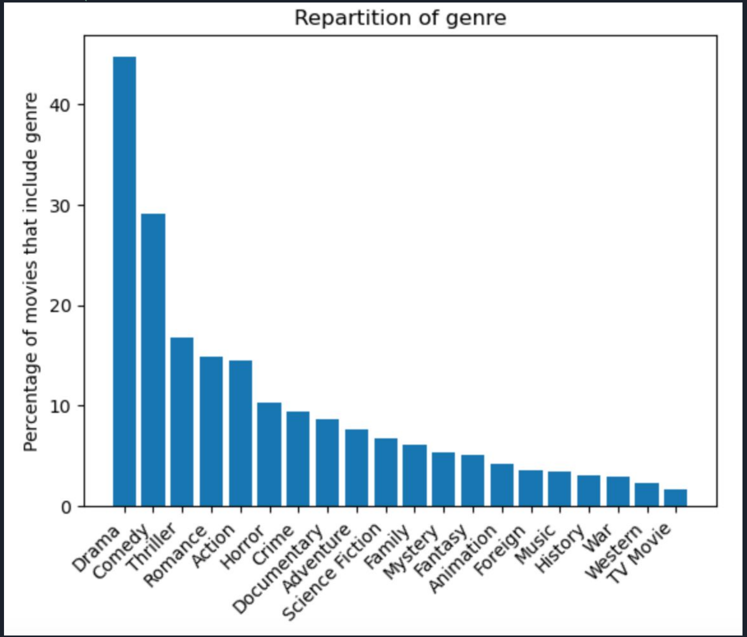
```
Inconsistency on id 99692 : names : Liao Fan /  廖凡
Inconsistency on id 111690 : names : Takako Matsu /  松隆子
Inconsistency on id 117642 : names : Jason Momoa /  杰森·莫玛
Inconsistency on id 9779 : names : Morris Chestnut /  Моррис Честнат
Inconsistency on id 23764 : names : Erika Eleniak /  Эрика Элениак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

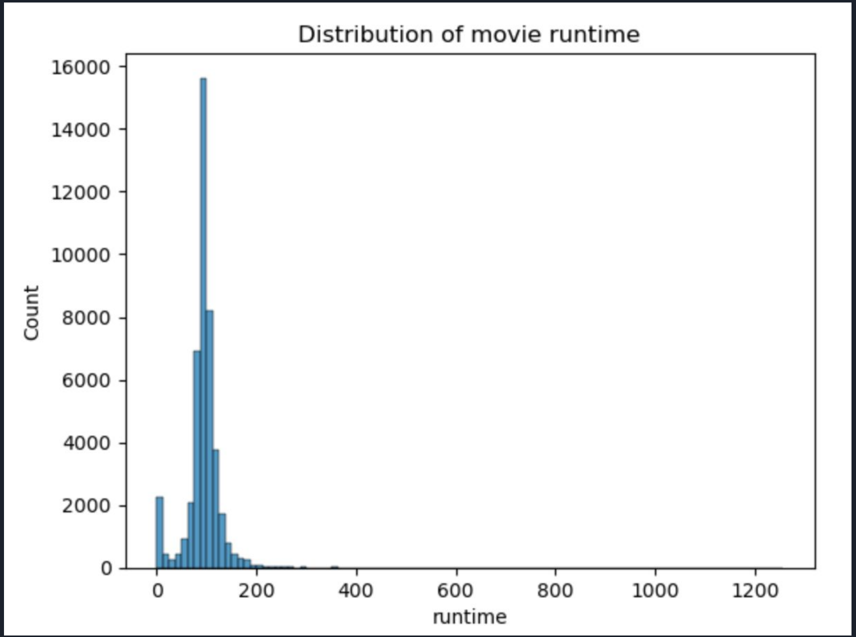
- Handling input format

```python
def extract_keyword_ids(l, id_key = 'id'):
    return {d[id_key] for d in l }
```

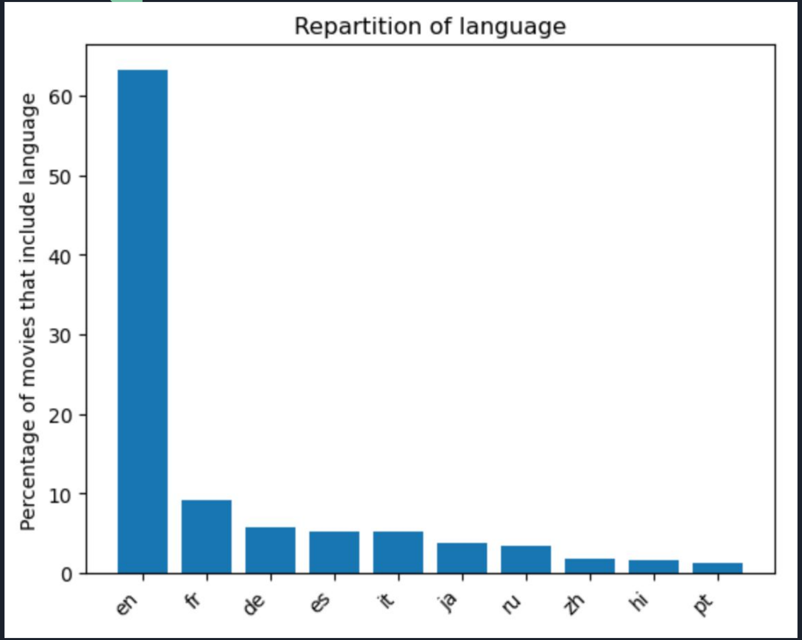
- Store the list as string

EDA

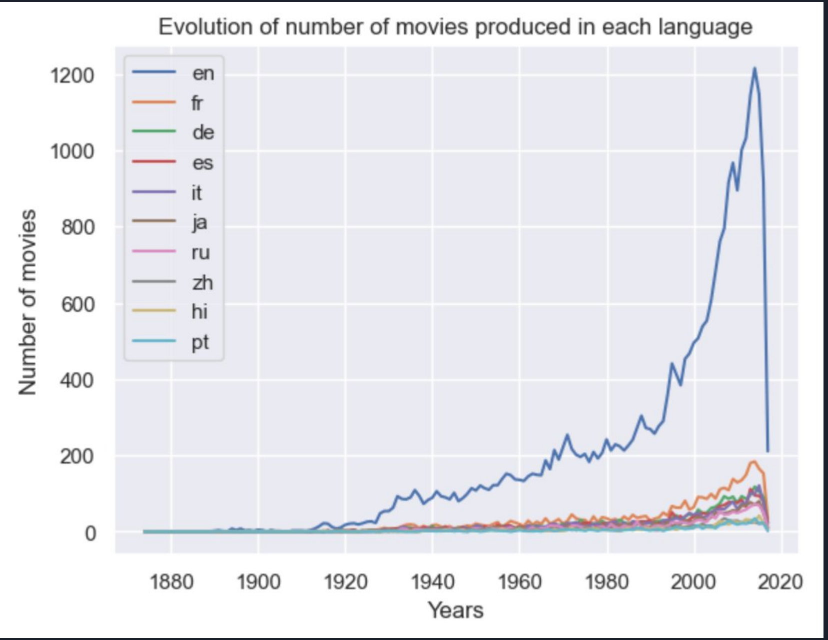

Repartition of genre
Percentage of movies that include genre
0
10
20
30
40
Drama
Comedy
Thriller
Romance
Action
Horror
Crime
Documentary
Adventure
Science Fiction
Family
Mystery
Fantasy
Animation
Foreign
Music
History
War
Western
TV Movie


Distribution of movie runtime
Count
0
2000
4000
6000
8000
10000
12000
14000
16000
0
200
400
600
800
1000
1200
runtime

EDA



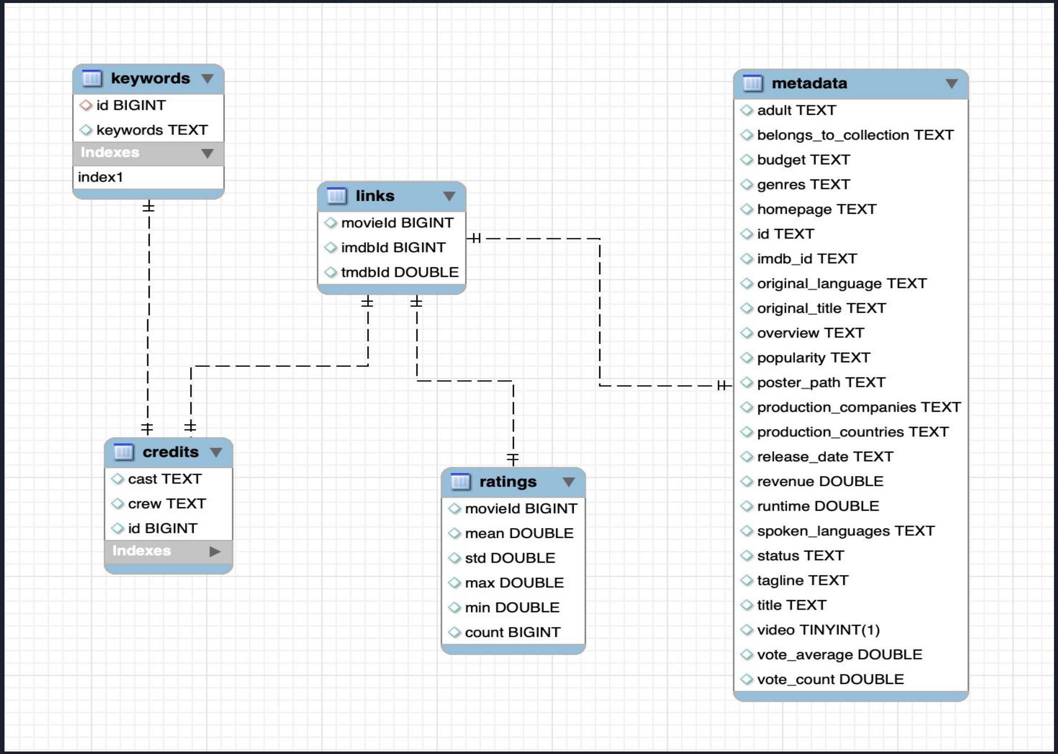
Repartition of language
Percentage of movies that include language
0
10
20
30
40
50
60
en
fr
de
es
it
ja
ru
zh
hi
pt



Evolution of number of movies produced in each language
Number of movies
0
200
400
600
800
1000
1200
en
fr
de
es
it
ja
ru
zh
hi
pt
1880
1900
1920
1940
1960
1980
2000
2020
Years

ERD

-


keywords
id BIGINT
keywords TEXT
Indexes
index1
links
movieId BIGINT
imdbId BIGINT
tmdbId DOUBLE
metadata
adult TEXT
belongs_to_collection TEXT
budget TEXT
genres TEXT
homepage TEXT
id TEXT
imdb_id TEXT
original_language TEXT
original_title TEXT
overview TEXT
popularity TEXT
poster_path TEXT
production_companies TEXT
production_countries TEXT
release_date TEXT
revenue DOUBLE
runtime DOUBLE
spoken_languages TEXT
status TEXT
tagline TEXT
title TEXT
video TINYINT(1)
vote_average DOUBLE
vote_count DOUBLE
credits
cast TEXT
crew TEXT
id BIGINT
Indexes
ratings
movieId BIGINT
mean DOUBLE
std DOUBLE
max DOUBLE
min DOUBLE
count BIGINT

CREATING DATABASE THRU SQL

```
SELECT links.movieId, imdbId, tmdbId, cast, crew, keywords, mean, count, adult, belongs_to_collection, budget, genres, homepage, original_l
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;
```

My SQL Queries

```sql
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

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

max_avg_runti...	avg_runtime
1256	94.26885590378683

My SQL Queries

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

title
Centennial

Data Processing

- Multi - Label encoding

- We kept only keywords / genre / Movie collection / Original language

Model

- K means with 8 clusters (low but still gives decent structure of data)

```
from sklearn.cluster import KMeans

kmean = KMeans()
clustered_movies = kmean.fit_predict(total_one_hot_encoded)
df_data['cluster_number'] =  clustered_movies
```

- Cosine Similarity Matrix in each cluster to rank

  We used this cosine similarity to have a more accurate result on the predictions

```
from sklearn.metrics.pairwise import cosine_similarity
```

LIVE DEMO

Conclusion

For conclusion,

I’m very happy of the recommander I provided but I also could tell that it really wasn’t an easy task.

During the programming, I noticed that recommend a movie was a little bit hard because I had a lot to do like data cleaning, data import to SQL and data processing to have the most accurate recommendation I could give.

Overall, it was a really good experience and I put in practice stuff I haven’t seen before like the cosine similarity matrix or the multilabelbinarizer.

Hope you enjoyed it !

Challenges/Next Steps

- Feature engeneering

- Evaluate the quality of the recommandation
- Implementing Streamlit front end

Highlights

- Learned multiple things on Python

- Very proud of the result of my movie recommendation