MovieSearch

Davide Liu
Tsinghua University
liud20@mails.tsinghua.edu.cn

May 6, 2020

Abstract

MovieSearch is a content specific search engine with the aim to retrieve movies information given the contents of a user's query. The search engine relies on the OkapiBM25 algorithm and takes into consideration the text present in the overview, the title, the names of the cast and the production companies of each movie. The back-end has been developed with the framework Django while the front-end extensively relies on Bootstrap 4 and Html5. Movies data, reviews and a reverse index used to speed up the research are stored in a local MongoDB database. It has also been implemented a movies recommendation system to recommend similar movies exploiting a nearest neighbors machine learning algorithm. Moreover, before uploading, movies reviews have been classified into two categories, positive and negatives, exploiting a pretrained Bert model fine tuned on the Imdb reviews dataset.

1 Introduction

In our reality already exist many context specific search engines focused on movies retrieval. Among them, one of the most popular is without doubts the website IMDb (1). It contains information about every possible movie since the early beginning of the history of the cinema to not yet released movies such as Avatar 5 which is set to be released in 2027 (2). Despite our project being less complete and complex than IMDb, however MovieSearch relies on a more powerful, fast and easy to use retrieval system based on the OkapiBM25 algorithm. It also benefits of a simple, friendly and mobile responsive user interface, an automatic way to classify movies reviews without relying on humans judgments and a reliable recommendation system.
overall, the website is composed of two main pages: the first page is the *"search results"* (fig 2) page which shows the movies retrieved given an input query. When clicking on the *"More info"* button of a specific retrieved movie, the system will dynamically generate its relative *"movie details"* (fig 1) page where it is possible to visualize more complete information about a movie, the results of the recommendation system and, whenever present, the reviews given by other users. The textbox to where input a new query is always present in any page and is located on top of it so to allow the user to perform a new search without requiring to be in a specific page.
The next paragraphs will explain each of the above mentioned features in more details.

2 Dataset

Overall, three different datasets have been used to make the contents of the search engine. The first dataset has been crawled from IMDb and is publicly available on Kaggle (3). It

contains information of about 5000 movies including: title, overview, release date, main actors, director, production companies and production countries, original languages, genres, plot keywords, popularity, runtime, IMDb average score, votes and id as well budget and revenue in dollars. Given the different types present in the data (there are integers, floats, strings and arrays) SQL like relational database were discarded preferring a document-based database like MongoDB. As common in many data science projects, most of the work has been consumed for cleaning the dataset. Many empty cells have been filled with N.A (not available) or 0, depending on the type of the data. Movies whose 'overview' field was absent have been directly discarded since it is a fundamental field to make the search engine algorithm work.
The second dataset has also been crawled from IMDb and available on Kaggle (4) and contains the poster of many movies. The title and the poster of a movie are the first two things users look at in order to judge whether they may like it or not. Thus it has been retained important to show the poster of a movie along with its title in both the search results and even more in the details page.
The third dataset is a typical text classification dataset containing over 50000 movies reviews (5), always taken from IMDb, to be displayed in the details page of a specific movie. Reviews are very important to let a user further understand whether a movie is suitable for him and whether it is worth to be watched based on the experiences of other users.

3 Movies retrieval

Before diving into the details of this part, it's more convenient do define a function f that takes as input a text x and processes it. This function f splits the text into single words, converts them to lowercase, removes stopwords, removes puntuactions and applies stemming. Its return is thus a list of words w. Moreover, let's also define the *text* of a movie as the concatenation of the words obtained by applying f on its title, overview, actors, director and production companies fields. These operations are very common in the text preprocessing pipeline of many NLP applications and their goal is to make a text more understandable for machines. As already introduced, Okapi25 is the algorithm chosen for retrieving relevant movies given an input user query. A query is first processed with $f(\text{query})$ and subsequently are retrieved from the database all movies containing at least one of the words w_q in their text. This allows a user to enhance its search by taking into consideration also cast and production companies names as keywords rather than just relying on the titles and overviews. Moreover, words in the title have a weight double respect to the other words in the text since usually users tend to retrieve a movie directly by writing it's title or some keywords contained in it.
Given the whole database containing over 5000 movies, searching for query keywords in all of the movies one by one would result in a very slow operation. Has then been implemented a reverse index whose keys are the union of the words present in the text of each movie and the items are arrays of indices of those movies whose text includes

the key. In this way the number of movies retrieved depends on the frequencies of the words in the processed query as well as its length and it's usually not required to scan the whole database. To further speed up the retrieval phase, the system temporarily caches movies retrieved from the database for the first time so to avoid retrieving them in the next searches. Table 1 shows the first 5 search results for the queries "Avengers heroes" and "Kung Fu".

Query	Avengers heroes	Kung Fu
1	The Toxic Avenger	Legend of a Rabbit
2	Avengers: Age of Ultron	Kung Fu Panda 2
3	The Toxic Avenger Part II	Kung Fu Panda
4	The Avengers	Kung Fu Panda 3
5	Last Action Hero	Bulletproof Monk

Table 1: Top 5 search results for the queries "Avengers heroes" and "Kung Fu".

4 Recommendation system

The goal of the recommendation engine is to recommend *n* movies whose content is similar to a target movie. In our implementation *n*=5. Hence, movie plot keywords play an important role in the functioning of the system. Luckily, the movie dataset already includes a set of keywords for each entry. Anyway, before using them, keywords have first been cleaned suppressing rare terms that appeared less than 5 times and replaced by a synonymous of higher frequency. Secondly, have been suppressed all the keywords appearing in less than 3 films in total. To compute the similarity between a target movie *t* and the rest of the movies, has first been built a similarity matrix (see table 2) with as many rows as the number of movies and a number of columns depending on *t*. In particular, the columns correspond to the fields title, director, actors, genres and keywords of the movie *t*. Beside the title which is a string, all the other fields a(i,2<=j<=N+K+H+2) have a boolean value that is true only if the there is a correspondence between the significance of column *j* and the content of film *i* respect to the movie *t*. For example, if *keyword1* of movie *t* is in movie *i*, with *t*!=*i*, we will have *a(i,N+K+3)*=1 and 0 otherwise. Once this matrix has been defined, we determine the distance using K-neighbours between two films according to the

title	director	actor1...N	genre1...K	keyword1...H
Movie1	a(1,2)	a(1,3..N+2)	a(1,N+3...N+K+2)	a(1,N+K+3...N+K+H+2)
Movie2	a(2,2)	a(2,3..N+2)	a(2,N+3...N+K+2)	a(2,N+K+3...N+K+H+2)
MovieI	a(i,2)	a(i,3..N+2)	a(i,N+3...N+K+2)	a(i,N+K+3...N+K+H+2)

Table 2: Movies similarity matrix.

Target	Avatar	I Robot
1	Guardians of the Galaxy	The Matrix
2	X-Men: Days of Future Past	The Hunger Games: Catching Fire
3	Avengers: Age of Ultron	Divergent
4	Captain America: The Winter Soldier	Minority Report
5	Star Trek	Lucy

Table 3: Top 5 recommendation results for the movies "Avatar" and "I Robot".

formula:

$$d_{t,i} = \sqrt{\sum_{j=2}^{J}(a_{t,j} - a_{i,j})^2}$$

Where J=N+K+H+2 is the number of features for target movie *t*. Next, are simply selected the m, (m>=n), movies with the lowest *d*. Now given these selected m movies, their similarity score is refined taking into consideration their IMDb score, the number of votes they received, the popularity and the year of release. The reason is that the less is the difference between these last 3 parameters, the more related are two movies. For example, it's very likely that people's favorite movies will be most of the time from the same epoch. Then the new score for each movie is thus calculated according to the formula:

$$score = IMDb_{score}^2 \times N_{votes}(votes_i) \times N_{popularity}(popularity_i) \times N_{year}(year_i)$$

Where N is a gaussian with mean and standard deviation equal to the max of its relative parameter among the subset of selected movies. With the gaussians, we put more weight to the entries with a large number of votes and popularity and to the movies whose release year is close to the title selected by the user. In addition, movies with higher IMDb score would tend to appear first since the goal of the recommendation system is to suggest similar but also of good quality movies. One last attention has to paid to the sequel and prequels of the target movie. Many blockbusters have sequels that share the same director, actors and keywords and so on. However, most of the times, the fact that sequels exist means that it was a "fair" box-office success, which is a synonym of a good IMDb score and similar popularity. Usually, there's an inheritance of success among sequels and according to how the current recommendation system is built, it is quite probable that if the engine matches one movie of a serie, it will end recommending several of them. To overcome this drawback, movie whose title is too similar to the title of the movie *t* are discarded. For example, "Pirates of the Caribbean: Dead Man's Chest" and "Pirates of the Caribbean: The Curse of the Black Pearl" have a very similar title, thus the system would recognize one of them to be the sequel of the other. Finally, table 3 shows some recommendation results for the movies "Avatar" and "I Robot".

5 Reviews classification

To make the website more interesting, movie reviews have been labeled with a pretrained model Bert (6) into 2 categories: positive and negative. The model has been fine-tuned on the IMDb movie review dataset [5] using the 25000 sentences contained in the training set. Next, have been predicted the other 25000 reviews contained in the test set before uploading all of them in the database. However, given the discrepancy between the reviews dataset and the movies dataset, not all movies have some associated reviews. Hence, future work may consist in crawling more reviews such that most movies can have at least one associated review. We believe that the labeled review may reward the user with an immediate feedback of how much a specific movie has been appreciated by the public without needing to read all of them.

6 Conclusions

The aim of this project was to design and implement a simple but effective content specific search engine. Moreover, we enhanced it with a recommending system and movie reviews classificator in order to provide a better user experience. However, this work could still be extended by crawling more data. In particular, should be crawled that posters and review that are still missing in the present work. Other enhancements could be adding movie trailers, build a login system to allow the user to create its own account and writing more reviews and modifying the recommendation system such that recommendation would not be based only on the content of the movies but also on user interactions.

References

IMDb website
`https://www.imdb.com/`

Avatar 5
`https://www.imdb.com/title/tt5637536`

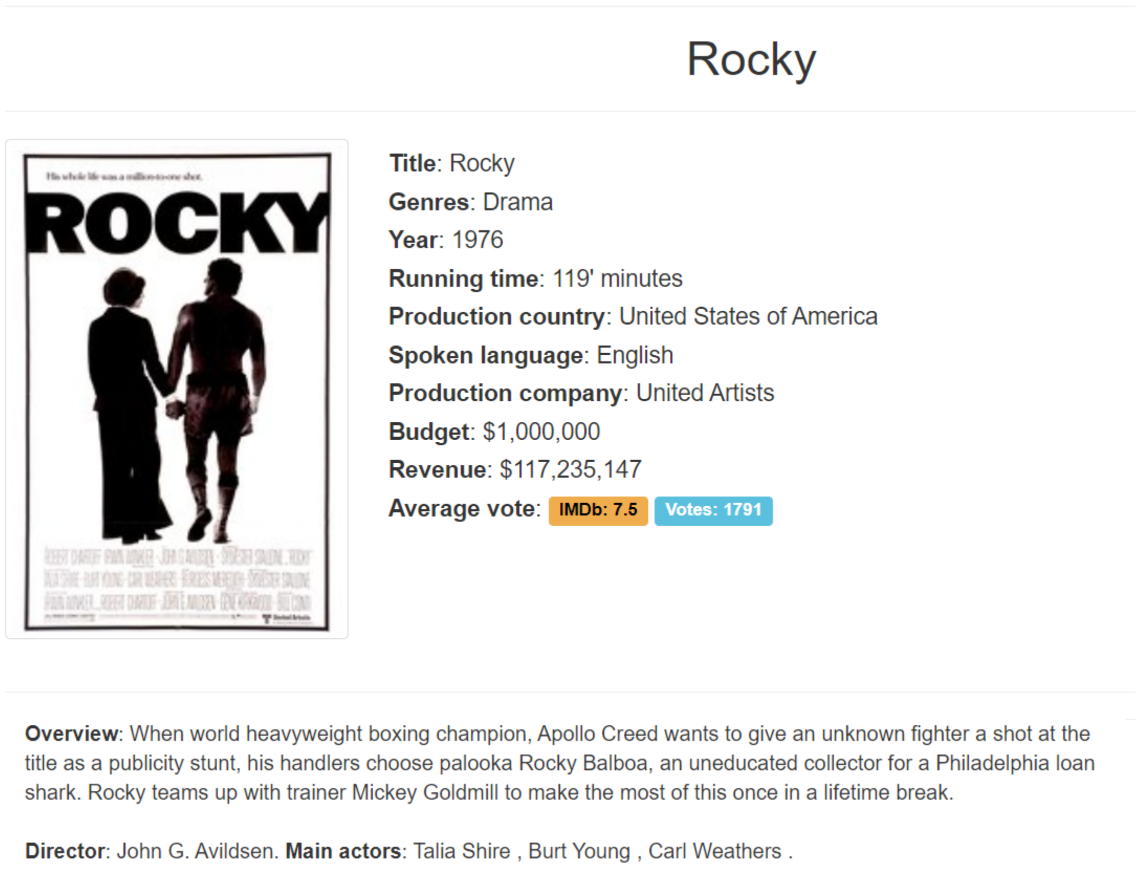
Movies dataset
`https://www.kaggle.com/tmdb/tmdb-movie-metadata`

Posters dataset
`https://www.kaggle.com/neha1703/movie-genre-from-its-poster`

Reviews dataset
`http://ai.stanford.edu/ amaas/data/sentiment`

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. [*On the electrodynamics of moving bodies*]. Annalen der Physik, 322(10):891–921, 1905.

7 Extra



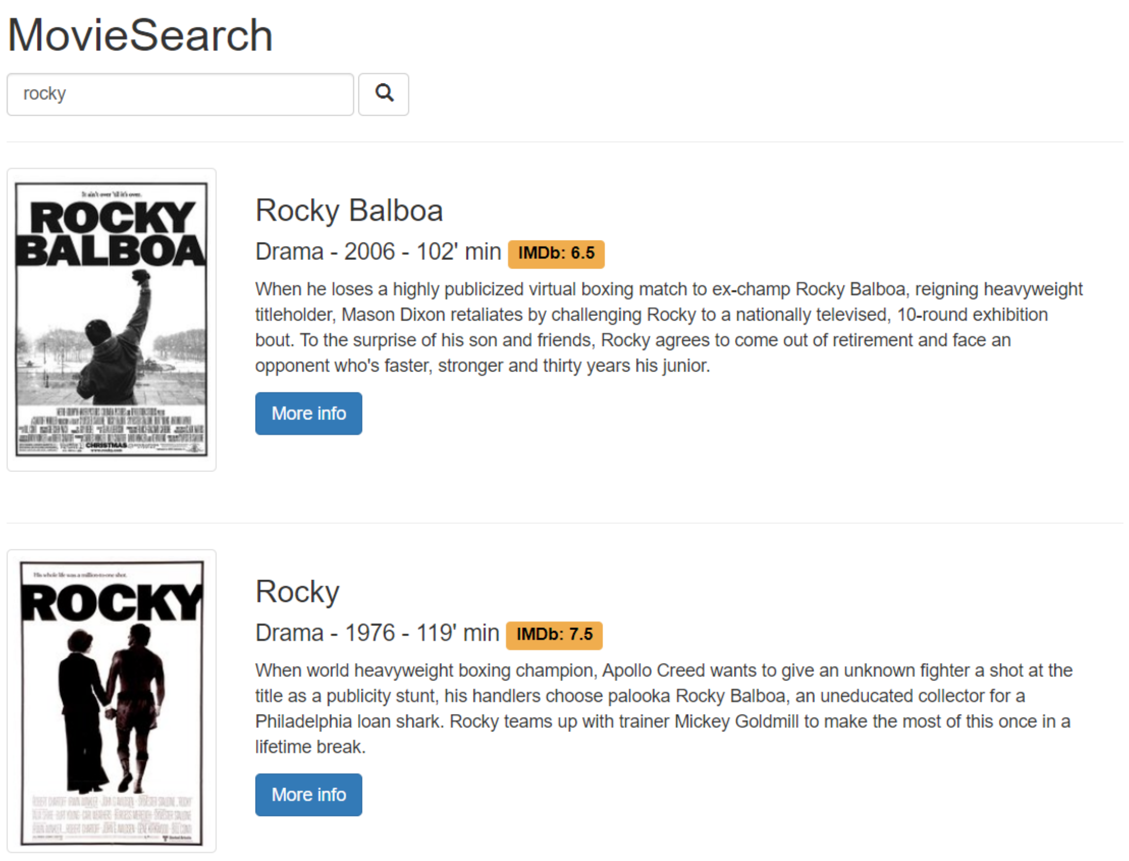
Rocky

Title: Rocky
Genres: Drama
Year: 1976
Running time: 119' minutes
Production country: United States of America
Spoken language: English
Production company: United Artists
Budget: $1,000,000
Revenue: $117,235,147
Average vote: IMDb: 7.5 Votes: 1791

Overview: When world heavyweight boxing champion, Apollo Creed wants to give an unknown fighter a shot at the title as a publicity stunt, his handlers choose palooka Rocky Balboa, an uneducated collector for a Philadelphia loan shark. Rocky teams up with trainer Mickey Goldmill to make the most of this once in a lifetime break.

Director: John G. Avildsen. **Main actors**: Talia Shire , Burt Young , Carl Weathers .

Figure 1: Details page of the movie "Rocky".


MovieSearch

rocky

Rocky Balboa

Drama - 2006 - 102' min IMDb: 6.5

When he loses a highly publicized virtual boxing match to ex-champ Rocky Balboa, reigning heavyweight titleholder, Mason Dixon retaliates by challenging Rocky to a nationally televised, 10-round exhibition bout. To the surprise of his son and friends, Rocky agrees to come out of retirement and face an opponent who's faster, stronger and thirty years his junior.

More info

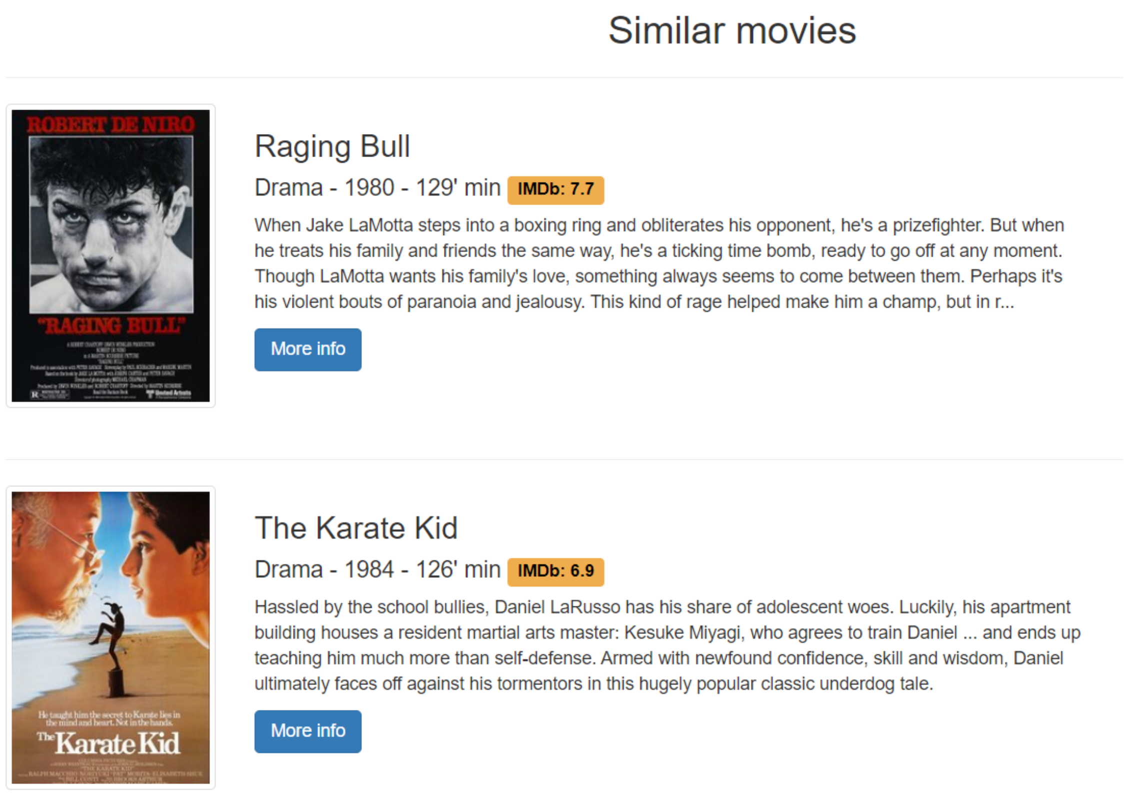
Rocky

Drama - 1976 - 119' min IMDb: 7.5

When world heavyweight boxing champion, Apollo Creed wants to give an unknown fighter a shot at the title as a publicity stunt, his handlers choose palooka Rocky Balboa, an uneducated collector for a Philadelphia loan shark. Rocky teams up with trainer Mickey Goldmill to make the most of this once in a lifetime break.

More info

Figure 2: First 2 search results for the query ”rocky”.


ROBERT DE NIRO
"RAGING BULL"
He taught him the secret to Karate lies in the mind and heart. Not in the hands.
The Karate Kid

Similar movies

Raging Bull

Drama - 1980 - 129' min IMDb: 7.7

When Jake LaMotta steps into a boxing ring and obliterates his opponent, he's a prizefighter. But when he treats his family and friends the same way, he's a ticking time bomb, ready to go off at any moment. Though LaMotta wants his family's love, something always seems to come between them. Perhaps it's his violent bouts of paranoia and jealousy. This kind of rage helped make him a champ, but in r...

More info

The Karate Kid

Drama - 1984 - 126' min IMDb: 6.9

Hassled by the school bullies, Daniel LaRusso has his share of adolescent woes. Luckily, his apartment building houses a resident martial arts master: Kesuke Miyagi, who agrees to train Daniel ... and ends up teaching him much more than self-defense. Armed with newfound confidence, skill and wisdom, Daniel ultimately faces off against his tormentors in this hugely popular classic underdog tale.

More info

Figure 3: First 2 recommendation results for the movie "Rocky".