Machine Learning Applications

Final project

Patrycja Wysocka: 100502920
Nicolas Buhringer: 100503977
Åshild Tenold Fridtun: 100503474

Table of contents

Acknowledgement of authorship - Dataset

We are using a dataset from Kaggle called “The Movies Dataset” found in the following link: https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=movies_metadata.csv. The dataset has metadata on over 45 000 movies, with 26 million ratings from over 270 000 users. In addition, the dataset has 7 csv files in which we use keywords.csv and movies_metadata.csv. The name of the author of the dataset is Rounak Bandik and was updated over 5 years ago.

Project contents

The project is divided into following directories:

- `/preprocess` - contains all the files performing data preprocessing:
  - `rating_preprocess.ipynb` - preprocess of the ratings
  - `nlp_vectorization.ipynb` - all the contents essential to the Task 1 - data cleaning and preparation, SpaCy text preprocessing, BoW, TF-IDF, Word2Vec, Doc2Vec Vectorization, LDA Topic Modelling
- `/recommender_system` - directory with recommender system models
  - `content_based_tfidf.ipynb` - performing content based recommendation model using TF-IDF Vectorizer
  - `content_based_doc2Vec.ipynb` - performing content based recommendation model using Doc2Vec representation
  - `collaborartive_filtering_params.ipynb` - notebook which allows to find optimal hyperparameters of a model and also helps finding the most suitable model using GridSearch and Surprise library to get the best evaluation metrics
  - `collaborative_filtering.ipynb` - performs collaborative filtering using model and parameters which had the best performance
- `/assets` - contains pictures necessary for dashboard
- `/png` - helpful plots in .png format generated during data exploration
- `helper_functions.py` - additional file with functions used by dashboard
- `dash_app.py` - dashboard application implementation

NOTE: In the content based approach we were using `ratings.csv` dataframe, while in collaborative filtering we were forced to work on `rating_small.csv` dataframe (which is reduced `ratings.csv`) due to lack of RAM memory and kernel crashing while computing rating matrix.

Libraries

To implement this project we used following technologies:

- Python3
- Pandas
- Numpy
- SpaCy
- Gensim
- Sklearn
- Seaborn
- Regex
- Surprise
- Sklearn
- Dash
- Fuzzywuzzy
- Pickle

Task 1: Text Preprocessing and vectorization

In this first part of the assignment, the group has first implemented a pipeline for the preprocessing of the text and then done several vectorization schemes. This will be discussed in detail in the following section.

Preprocessing

In this section we have done data cleaning and prepared the dataset to be ready for the vectorization. We have loaded and prepared the keywords.csv file as well as the movies_metadata.csv file. Due to keywords being saved inside of a raw string, we have to parse them out with Regex:

```
# keywords need to be parsed
data_keywords.iloc[[0]].keywords[0]
```

```
"[{'id': 931, 'name': 'jealousy'}, {'id': 4290, 'name': 'toy'}, {'id': 5202, 'name': 'boy'}, {'id': 6054, 'name': 'f
riendship'}, {'id': 9713, 'name': 'friends'}, {'id': 9823, 'name': 'rivalry'}, {'id': 165503, 'name': 'boy next door
'}, {'id': 170722, 'name': 'new toy'}, {'id': 187065, 'name': 'toy comes to life'}]"
```

```
def key_word_grabber(text):
    """
    Function to retreive the keywords from the above string
    """
    if text is not np.nan:

        return re.findall(r"'name':\s*'([^']*)'", text)
    else:
        text
```

Then a basic NLP preprocessing was performed using spaCy. Due to one third of the movies in the dataset not having keywords we joined the overview description and keywords together for the recommendation system.

Vectorization

Bow & TF-IDF

Both Bow and TF-IDF representations were created using spaCy.

BoW

```
reviews_bow = [D.doc2bow(doc) for doc in corpus]

n_review = 1000
print('============= Review (lemmas) =============')
print(' '.join(corpus[n_review]))

print('\n============= Sparse vector representation =============')
print(reviews_bow[n_review])

print('\n============= Word counts for the review =============')
print(list(map(lambda x: (D[x[0]], x[1]), reviews_bow[n_review])))
```

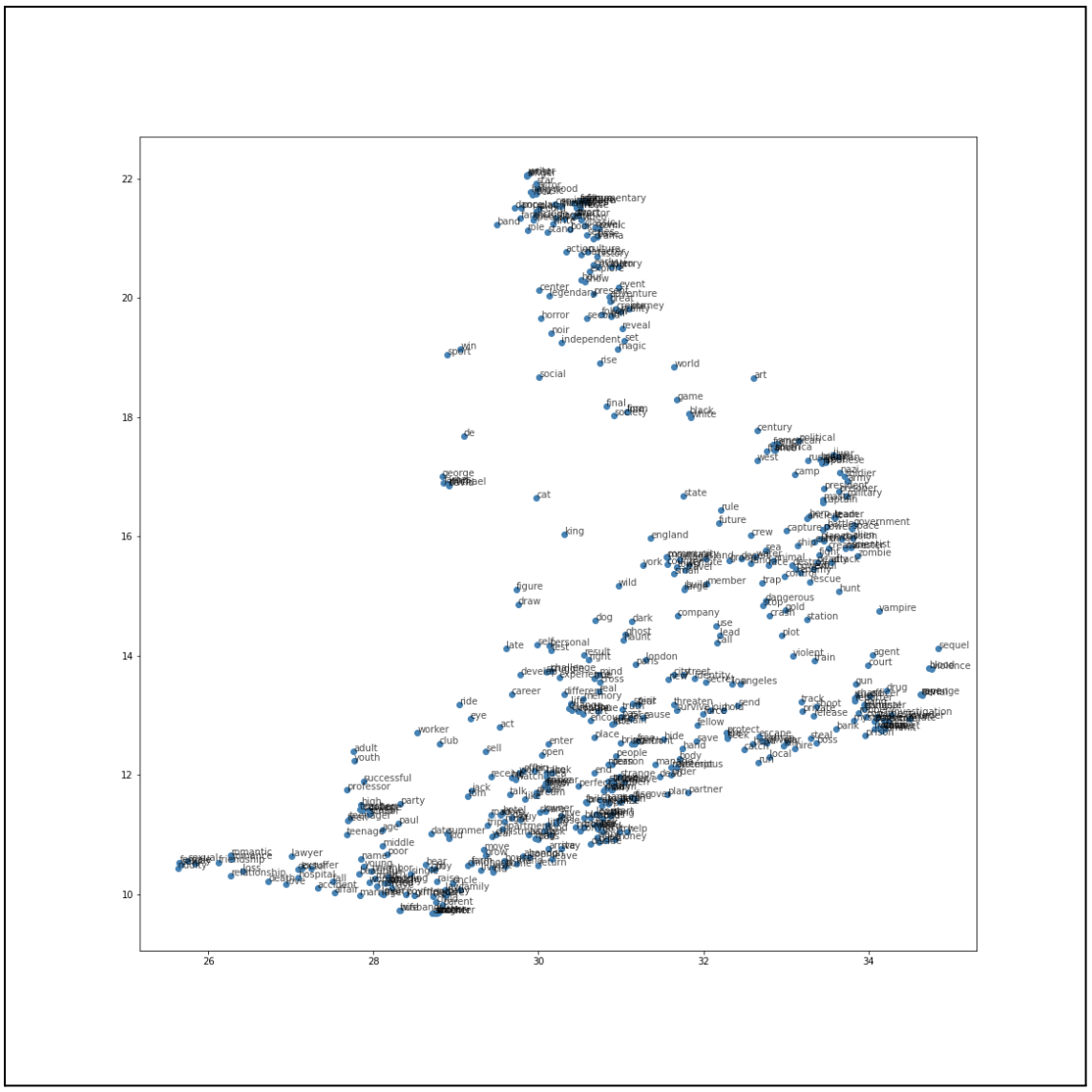
```
============= Review (lemmas) =============
Pennsylvania band score hit ride star make machinery lot help star musical record label music music band

============= Sparse vector representation =============
[(389, 2), (753, 2), (817, 1), (1143, 1), (1294, 2), (1302, 1), (1323, 1), (1796, 1), (2383, 1), (2769, 1), (3709, 1), (7661, 1), (9073, 1)
, (9074, 1)]

============= Word counts for the review =============
[('star', 2), ('music', 2), ('help', 1), ('musical', 1), ('band', 2), ('hit', 1), ('make', 1), ('lot', 1), ('record', 1), ('ride', 1), ('Pe
nnsylvania', 1), ('score', 1), ('label', 1), ('machinery', 1)]
```

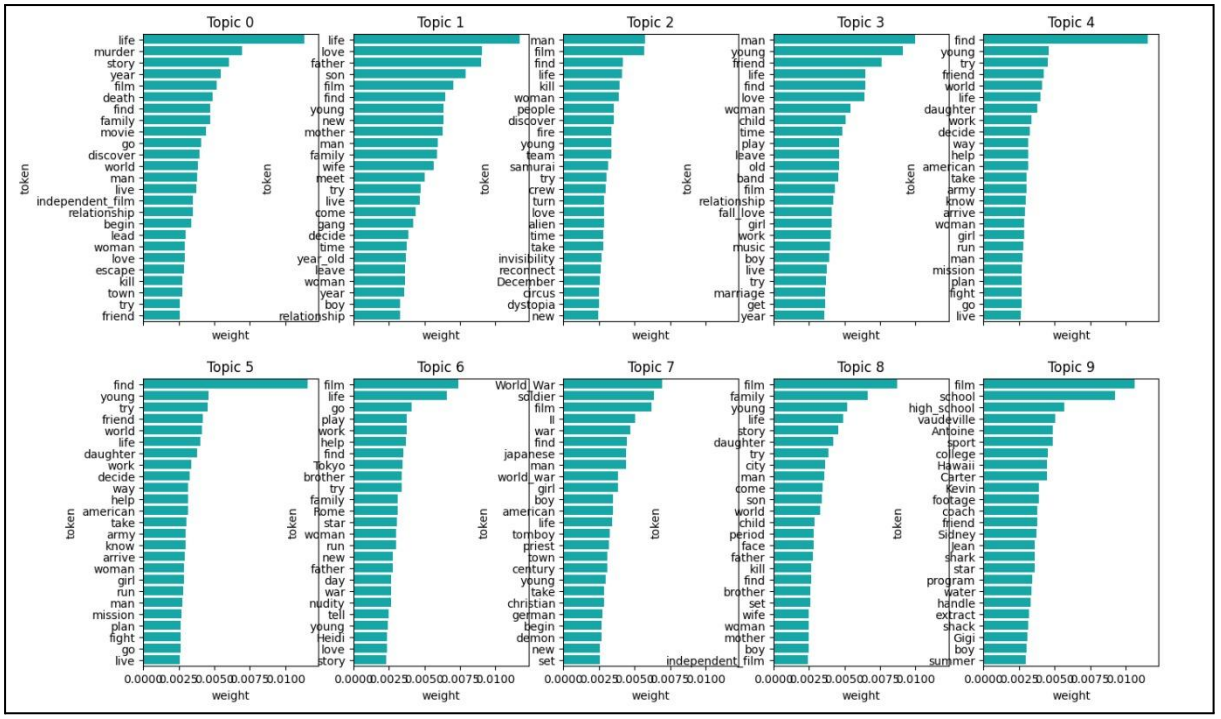
Word2Vec & Doc2Vec

Furthermore, we used Gensim and the text_to_word_sequencer from Tensorflow to create Word2Vec and Doc2Vec Models. For Doc2Vec we created two models, one model on the combined overview description and keywords and the other one just on the keywords. Surprisingly, the keywords showed a better performance when testing it for selected movies. Maybe the overview had too many non-describing words which lowered performance. Here is a plot showing 500 Word2Vec Embeddings:



LDA

We performed a Latent Dirichlet Allocation Analysis on the combined overview description and keywords leading to the following discovered topics. These align with commonly known genres in the movie industry:



Task 2.3: Machine Learning model - Recommender systems

For this part of the assignment, the group has implemented a content based filtering based on the document vectorization using both TF-IDF vectorizer and doc2Vec representation, and this is compared to the performance of a collaborative based filtering system. This will be further discussed in the next section.

Content based recommendation system

The group has implemented a content based recommendation system based on the document vectorization from task 1. From this the system will:

- Calculate how similar the movies will be to each other, and one input movie will give many output movies that are the similar ones.
- Predict rating that one particular user will give to the chosen movie based on users' previous ratings.

We designed a neighbourhood based system.

Content based recommendation system - TF-IDF

To implement a content based recommendation system the group decided to use TfidfVectorizer from sklearn library. To find similar movies we are using cosine similarity and calculating pairwise similarity scores of all movies with one particular movie, to finally sort all of them and obtain the most related.

To predict rating that one particular user will give to the chosen movie we were using the following formula:

$$\hat{r}_{u,i} = \frac{\sum_{j \in N_i^K} \text{sim}(i,j) r_{u,j}}{\sum_{j \in N_i^K} \text{sim}(i,j)}$$

Where N is the number of K similar neighbours to the item i.

Below are examples of inputs and outputs from the content based recommendation system using TF-IDF vectorizer:

Movie	Top 5 recommended movies (in general - similarities)
Batman Forever	Batman Returns: 0.328 Batman: 0.294 Batman: Mask of the Phantasm: 0.262 Batman & Robin: 0.259 Batman Beyond: Return of the Joker: 0.245
Star Wars	The Empire Strikes Back: 0.431 Star Wars: Episode III - Revenge of the Sith: 0.302 Return of the Jedi: 0.288 Star Wars: Episode II - Attack of the Clones: 0.193 Hot Shots! Part Deux: 0.192
Spider-Man	Arachnophobia: 0.342 Spider-Man 2: 0.302 X-Men: 0.183 The Giant Spider Invasion: 0.164 Hangman's Curse: 0.15

Recommending using TF-IDF Vectorization

The following are different tables of unrated movies by a given user. The user has already rated similar movies to the given movie based on cosine similarity. The predicted rating is calculated using the tables below.

User Id	Movie title	Predicted rating
107720	Batman Forever	4.849

	userId	movieId	rating	sim	title
230	107720	364	6.0	0.327897	Batman Returns
172	107720	268	4.0	0.293520	Batman
248	107720	415	4.0	0.259142	Batman & Robin
174	107720	272	8.0	0.223875	Batman Begins
1015	107720	2661	4.0	0.106883	Batman
470	107720	820	4.0	0.089430	JFK
807	107720	1924	2.0	0.088710	Superman
797	107720	1902	4.0	0.081993	Open Your Eyes
898	107720	2118	6.0	0.073846	L.A. Confidential
798	107720	1903	4.0	0.071586	Vanilla Sky

For Batman Forever the predicted rating is 4.849 based on the output from the screenshot. This rating is based on the similar movies, and we see that the most similar movies are all the Batman movies and they have in general ratings by the user close to 4.0, 6.0 and 8,0. It is quite reasonable that the predicted rating then is 4.849.

User Id	Movie title	Predicted rating
179792	Star Wars	6.517

	userId	movieId	rating	sim	title
592	179792	1895	6.0	0.302491	Star Wars: Episode III - Revenge of the Sith
589	179792	1892	4.0	0.287634	Return of the Jedi
591	179792	1894	5.0	0.192938	Star Wars: Episode II - Attack of the Clones
590	179792	1893	4.0	0.156073	Star Wars: Episode I - The Phantom Menace
424	179792	957	8.0	0.131279	Spaceballs
1243	179792	8584	8.0	0.125353	Shanghai Noon
1313	179792	25952	10.0	0.124495	Conquest of Space
417	179792	947	10.0	0.113156	Lawrence of Arabia
1408	179792	44864	7.0	0.104112	The Magic Flute
661	179792	2067	10.0	0.098405	Mission to Mars

For Star Wars the predicted rating is 6.517 based on the output from the above screenshot. We see that the most similar movies are the other Star Wars movies, but then other movies also score quite well. The predicted rating for Star Wars, based on the most clearly similar movies, is in this case in the rather higher scope.

User Id	Movie title	Predicted rating
8659	Spider-Man	6.708

	userId	movieId	rating	sim	title
1283	8659	6488	6.0	0.342202	Arachnophobia
300	8659	558	6.0	0.301980	Spider-Man 2
722	8659	1924	6.0	0.131815	Superman
130	8659	235	8.0	0.102616	Stand by Me
683	8659	1807	8.0	0.092426	Elephant
1377	8659	8491	8.0	0.091782	Weekend at Bernie's
847	8659	2294	6.0	0.091779	Jay and Silent Bob Strike Back
181	8659	314	8.0	0.084330	Catwoman
1383	8659	8592	6.0	0.083017	Dick Tracy
833	8659	2255	6.0	0.083006	Chasing Amy

For Spider-Man the predicted rating is 6.708. Most clearly similar movies here are Arachnophobia and Spider-Man 2, and then in the lower scope of similarities are Superman, Stand by Me and Elephant to mention some. The predicted rating for the movie looks pretty decent for these examples.

Content based recommendation system - Doc2Vec

Implementation of that model is quite similar to the previous one - the difference is that here we are using Doc2Vec representations from the gensim library, which easily allows us to find the most similar movies to the given one. To predict ratings we are following the same steps as in the previous implementation.

Movie	Top 5 recommended movies (in general - similarities)
Batman Forever	Panic in the Streets: 0.804 The Devil's Own: 0.798 Sky Captain and the World of Tomorrow: 0.767 Mystery of the Wax Museum: 0.764 Panic Room: 0.758
Star Wars	The Empire Strikes Back: 0.932 Missing in Action: 0.931 Monty Python Live at the Hollywood Bowl: 0.913 The Pawnbroker: 0.913 The Sound of Music: 0.907
Spider-Man	Spider-Man 2: 0.827 House of the Dead: 0.816 If Looks Could Kill: 0.815 Young Sherlock Holmes: 0.796 Agent Cody Banks: 0.79

The table above shows the top five recommended movies given by similarities implemented by Doc2Vec. The output is for the same movies as evaluated before: Batman Forever, Star Wars and Spider-Man. Note how doc2vec has quite different similarity movies from the TF-IDF implementations as of which has the highest importances.

Recommending to user using Doc2Vec

User Id	Movie title	Predicted rating
107720	Batman Forever	4.377

	userId	movieId	rating	sim	title
659	107720	1493	4.0	0.507480	Miss Congeniality
681	107720	1572	4.0	0.510050	Die Hard: With a Vengeance
713	107720	1642	6.0	0.517861	The Net
1237	107720	4477	2.0	0.552443	The Devil's Own
1330	107720	5137	6.0	0.528078	Sky Captain and the World of Tomorrow

The predicted rating of 4.377 of Batman Forever is based on all pretty equivalent similarities of the five given similar movies.

User Id	Movie title	Predicted rating
179792	Star Wars	Nan

From 10000 similar movies to this one it didn't find any that this user rated before.

User Id	Movie title	Predicted rating
8659	Spider-Man	6.0

	userId	movieId	rating	sim	title	id
300	8659	558	6.0	0.568164	Spider-Man 2	558

The predicted rating of 6 is based on Spider-Man 2 which was the only rated similar movie of the user.

Collaborative based recommendation system

For the collaborative based system we are predicting what a given user will rate a movie. We give a user and a movie as an input, and as an output comes a predicted scoring of how that user will rate the movie. The output score will be a number between 1 and 10.

Below are examples of inputs and outputs from the collaborative based recommendation system.

Movie	User	Estimation
Batman Forever	564	5.42
Star Wars	547	6.79
Terminator 3: Rise of the Machines	15	7.04

Comparison of the systems

User	Movie	Content based - TFIDF	Content based - Doc2Vec	Collaborative based - user based
480	Batman Forever	6.3143013785 3958	nan	6.47
260	Star Wars	9.4365870720 76145	nan	7.64
15	Terminator 3: Rise of the Machines	7.3042365823 77969	nan	7.04

In the table above we have collected three different users and their predicted rating on Batman Forever, Star Wars and Terminator 3: Rise of the Machines implemented by Content based TFIDF, and Collaborative based - user based. The Content based implementation by Doc2Vec did not give us any valuable insights here as it didn't find any similar movies that the given user had rated before.

From the table above we see that the Content based system implemented by TF-IDF have generally a higher recommendation score than the Collaborative based implementation,

apart from the movie Batman Forever where the scores are fairly similar but 0.16 higher with the Collaborative based system. These differences can be caused by the use of different datasets as stated in the introduction, where for the content based approach we use the full dataframe whilst in the collaborative part we only work on a reduced part of the dataset.

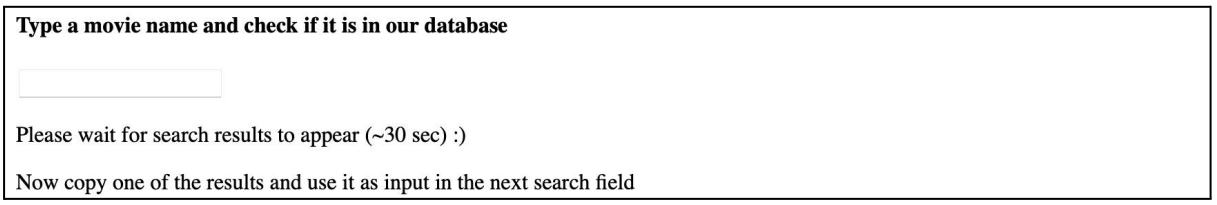
Task 3: Implementation of a dashboard

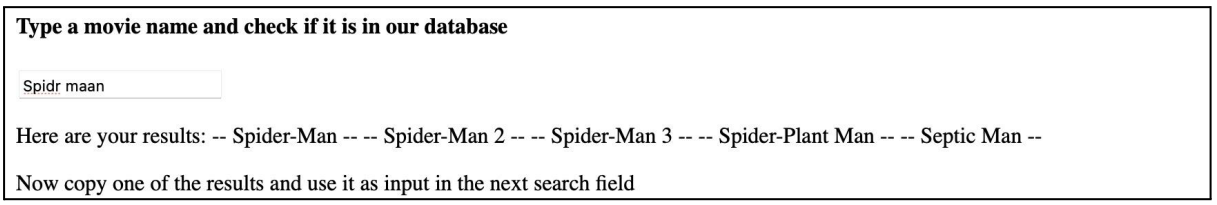
For the visualisation of the data we have used Dash. We offer 5 functionalities:

1. Performing a smart lookup of movies in our database
2. Recommending five similar movies based on a provided movie
3. Providing insights for a selected movie in our database
4. Showcasing an interactive popularity graph based on genre
5. Showcasing the results of a LDA Analysis

Movie lookup

Here we allow a user to check if a movie is contained in our database. Since we wont know how the title is written, we use the library fuzzywuzzy to perform a similarity analysis. The function returns a maximum number of five movie titles which could be the users intended input. Since some search inputs could return in less results, we check if that is the case and return the reduced amount of movies.





Movie recommendation

Based on the title of a movie, five similar movies are recommended using a Content Based Approach. A check is performed if the movie is contained in the database. If not, the user can use the search function from 3.1 to find the correct way of writing the movie title.

From here on please only verified inputs from the search bar above!

Your favorite movie you have already watched a houndred times:

Spider Man

Your recommendation based on that movie:

Recommendation 1: Please try another movie
Recommendation 2: Please try another movie
Recommendation 3: Please try another movie
Recommendation 4: Please try another movie
Recommendation 5: Please try another movie

From here on please only verified inputs from the search bar above!

Your favorite movie you have already watched a houndred times:

Spider-Man

Your recommendation based on that movie:

Recommendation 1: Spider-Man 3 ---- with a certainty of: 0.38%
Recommendation 2: Arachnophobia ---- with a certainty of: 0.35%
Recommendation 3: Spider-Man 2 ---- with a certainty of: 0.3%
Recommendation 4: Spiderman: The Ultimate Villain Showdown ---- with a certainty of: 0.21%
Recommendation 5: X-Men ---- with a certainty of: 0.18%

Movie insights

The user can use a movie title to find out more information about the movie:

a) Title
b) Genre
c) Budget
d) Runtime
e) Popularity

A check is performed whether the movie is in our database. Otherwise the user can use the search function from 3.1 to find the correct way of writing the movie title.

Get information on a movie in our database

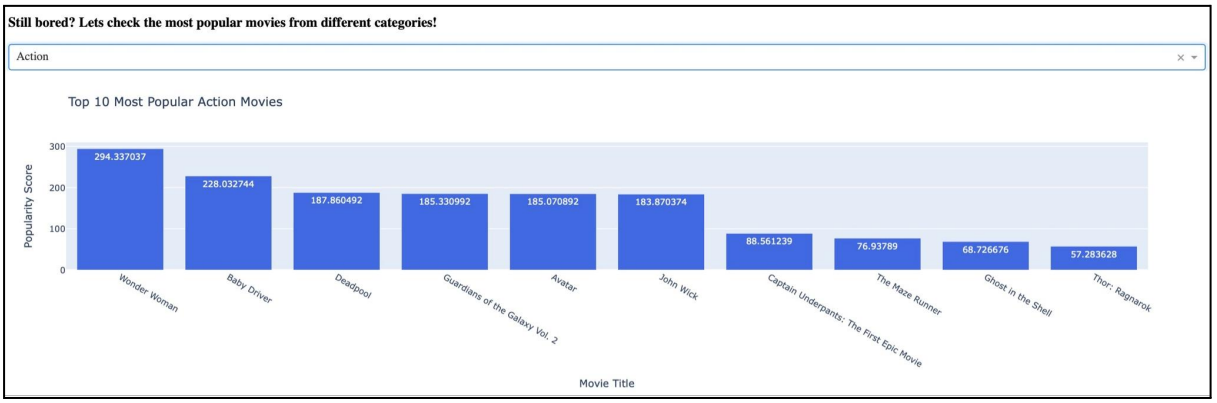
Spider-Man

Title: Spider-Man
Genre: Fantasy
Budget: 139000000
Runtime: 121
Popularity: 29.413341

Interactive popularity graph

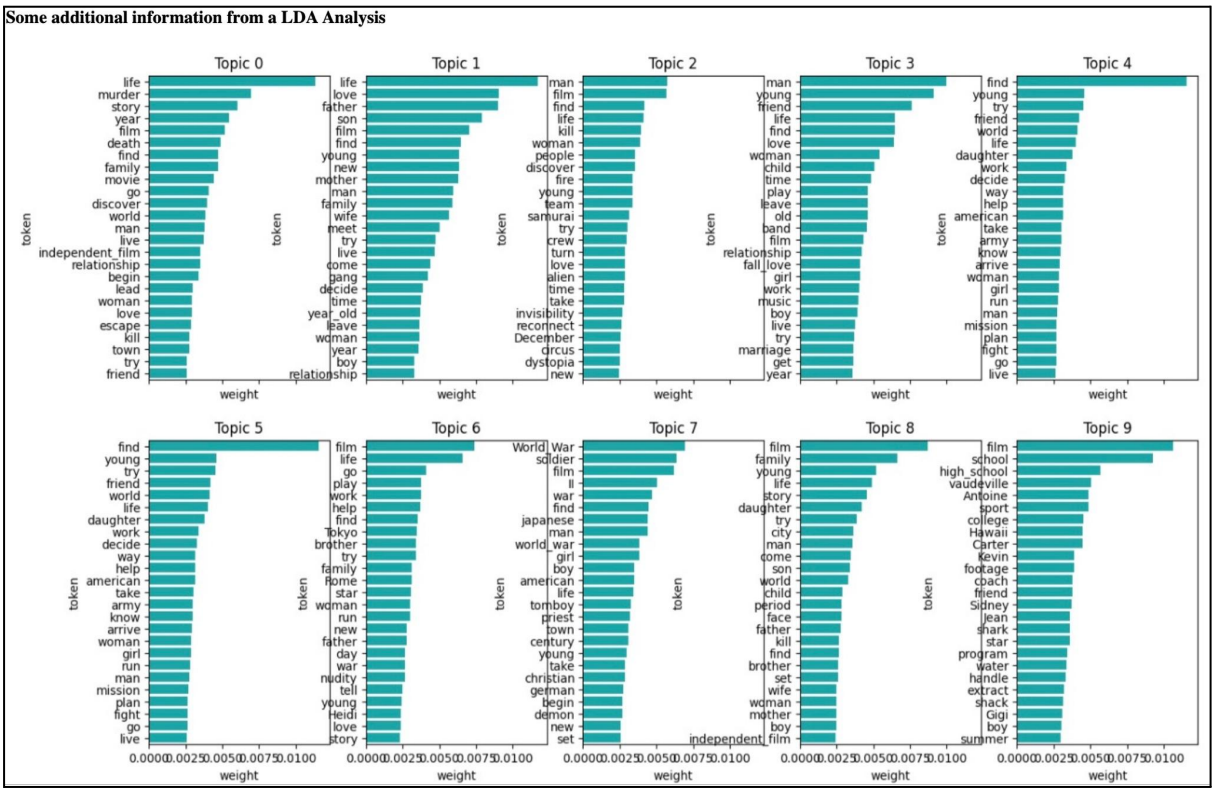
We also included an interactive plotly graph that showcases the top ten movies per genre based on the popularity score. The genre is being selected via a dropdown menu by the

user.



LDA Analysis

Here we are showing the results of performing a LDA Analysis on a preprocessed version of all of the movie descriptions and keywords. Due to the long computational time we provide an exported png version of the graph from our notebook. It is very interesting to see varying genres emerge from the different topics.



Annex: Dashboard backend