Movie Recommender

By

Salma Waqif, Hanaa Al Zahabi, Yasmine Souabi, Lapo Carrieri, Niko Picello

Departament de Ciències de la Computació
Universitat Politècnica de Catalunya

A thesis submitted for the degree of

FIB - MAI: IDSS

Barcelona, January 2023

Abstract

In a society where everyone have problems where they don't even exist, for many the most stressful part of the day is picking what movie to watch during a night. That's why we want to solve this issue as well, and to do that we created a movie recommender that suggests 5 movies that may correspond to the criteria one is giving as input to the program. To define the suggestions we used 2 models, one based on the overviews of the movies, and another based on their features (e.g., being a comedy, the year of release, the average vote, and so on). Also, the first model needs as input from the user a string that describe what he/she wants to watch (in this sense, the user is allowed to write literally whatever he/she wants, and the AI will try to understand what most suit words written). In order to obtain a result in a reasonable time, in particular for the first model, it is almost mandatory to filter the original dataset removing all those movies for which the overview is not really useful to extract information about the movie itself (indeed, there are samples for which that field is 'overview not found.' or it's an empty field)

Contents

List of Figures

Chapter 1

Introduction

The application we developed is made to suggest movies to the user as a list of 5 movies close to the criteria that one gives to the program. To do so, we are going to use 2 models: the first takes as input a string from the user in order to compute the similarity between the overviews of the movies in the dataset and what the user wrote, and then it picks the top 5 films for which the value of the distance is higher. The first has been presented in 2013 by a team at Google led by Thomas Mikolov, and there are many versions of it, different by complexity and vocabulary they learnt from during the training process; in particular, we decided to opt for one of the most complex one called "word2vec-google-news-300", which despite of the dimension, results also in the one that performs better (among the ones that we tried) in terms of time needed to compute the aforementioned similarities. To being able to get an answer in a reasonable amount of time, we opted to filter the dataset using certain constraints that will be explained later in the report, so to reduce the searching space of movies in the dataset. The second model instead...

Chapter 2

Everything specified in PRACTICALS:

Since we don't have the whole internet to look for movies that can be similar to the one given as input by the user, we limited our research to a "small" dataset containing 45572 movies. Movie metadata is used for tf-idf, in particular the Overview is used to get the summary and make comparison between movies plot. Then through the Netflix Prize Data dataset we obtain the reviews and ratings of all the users in order to filter the dataset and train a model that show the top 10 movie reccomanded for that specific user. All these datas are inside combination file.txt

2.1 a. Description of the domain of application

A movie recommendation system would target the entertainment industry, specifically the film industry.
Within this domain, it can be used by a variety of organizations and platforms such as streaming services like Netflix, Amazon Prime, Disney+ etc, movie rental companies like Redbox, Movie rental stores, cable/satellite providers, online movie ticket booking portals, and other platforms that offer a selection of movies for customers to choose from.

These organizations and platforms would use the movie recommender system to enhance the user experience by providing personalized recommendations to users. By using the system, the organizations and platforms can increase customer engagement, retention, and loyalty. Additionally, movie recommender systems can be used for the promotion of new movies, by identifying the users with similar taste, interests and demographics to the specific movie and target them with

personalized promotions and marketing campaigns.

Movie recommender systems can also be used by film production companies and studios, to identify the potential audience for their upcoming movies, and target them with promotions and marketing campaigns to increase the box office collection.

Furthermore, movie recommender systems can also be used in the field of academic research, for studying the patterns and trends in the movie industry, and understanding the audience preferences and behavior.

In summary, the domain of application for a movie recommender system is quite broad, and it can be used by a variety of organizations and platforms in the entertainment industry, to enhance the user experience, increase customer engagement, retention, and loyalty, and for the promotion and marketing of movies.

Hence, this intense demand and benefits of a movie recommender system encouraged us to choose this topic for the 3rd practical work.

2.2 b. Main identified decisions

Since we have done two different models we can obtain the top 10 movie recommendations. The tf-idf uses the user preferences to obtain the best movie suggestion, then it uses a Deep Learning model using the overview of the movies in order to select similar plots. So all the done preferences of the user present in the Netflix Dataset are useful to match the preference of the user. The doc2vec document instead is filtered in order to reduce the dataset and the amount of work necessary for the model. So the user preference is an uncertain event that influences the choices of filtering as the movie's length, genre, and date of release. From all these factors the top 10 movies are shown so, there is this decision of these movies. Then there is the top 10 movie satisfaction based on the result. Then the user must select one of these ten movies; this choice is personal. So the user humor, movie poster effectiveness, the description, and all the stuff shown on the top 10 display influence the choice of the user.

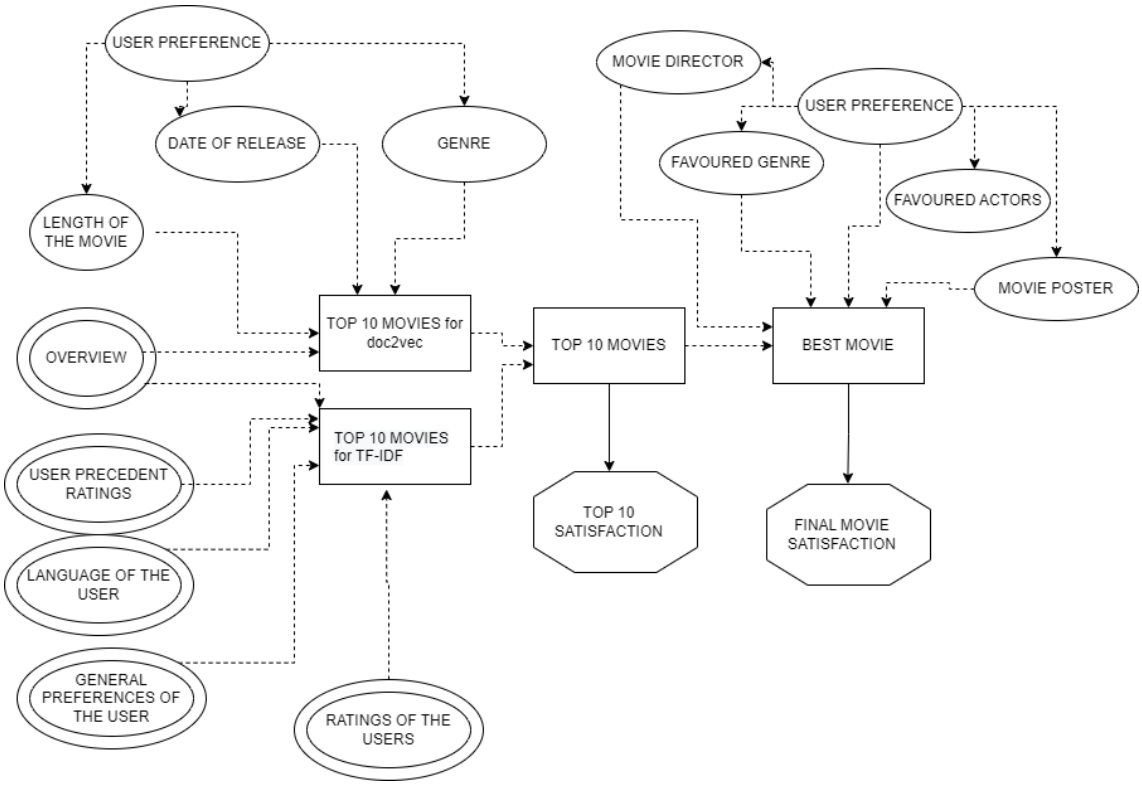


Figure 2.1: Influence Diagram

2.3 c. Functional architecture of the IDSS prototype

The system architecture follows a typical web development flask application with 3 levels : Data, Models and Frontend.

Hence, the system contains 3 separate folders :
- DATA : This folder contains the Data set used in our project.
- MODELS : In this folder, we have the two different models used for the recommendation system, word2vec and td-idf models.
- FRONTEND : It contains the interface and the app.py class.

The FRONTEND folder contains two different parts : templates and static, plus the app.py class that forms the bridge between the user interface and the models. The templates folder contains the html classes corresponding to the two pages that our front displays : The landing (a.k.a index) page and the result page.
The static folder contains the styling part (.css files) and the javascript files.
Finally, the class app.py which links the frontend to the backend. It basically gets the necessary information from the interface (Keywords, genre and year), then feeds it to the models and receive a response in exchange (list of recommended

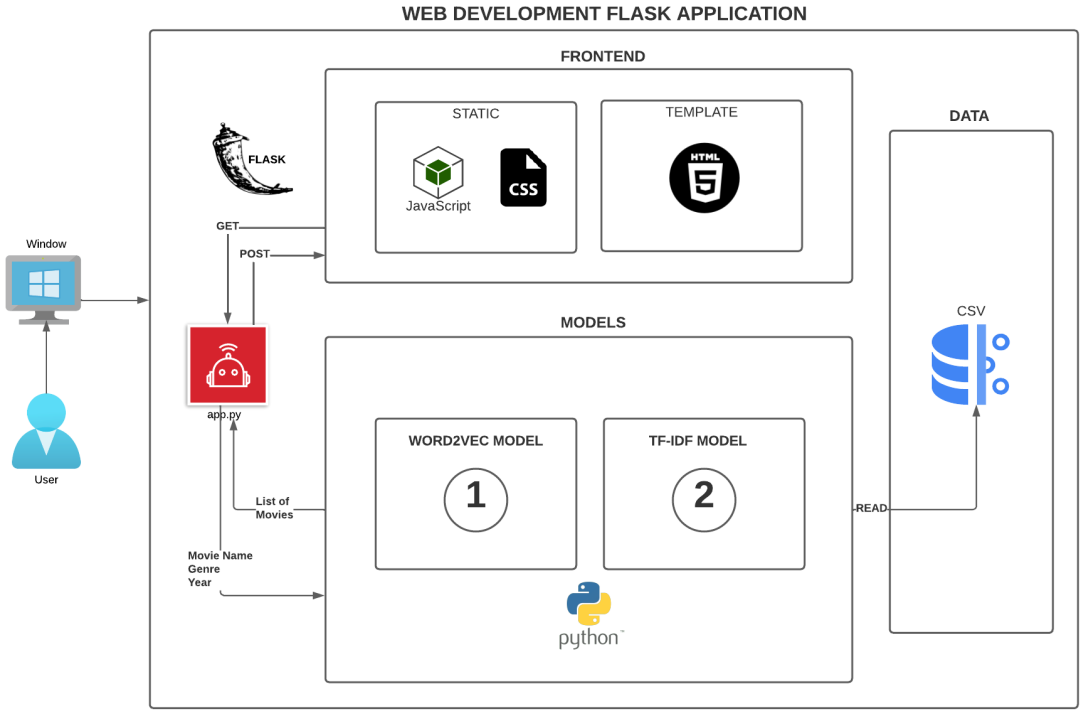


Figure 2.2: System Architecture

movies). Therefore, this list is sent back to the interface, always with app.py.
Not to forget to mention that the models generate a response, which is the list of recommendations, based on the informations received from the user and by using the DATA section, containing the dataset.

Concerning the languages and technlologies used in the system ;
We are using the micro web framework, FLASK for the application.
The dataset is stored in a csv file. Aditionnaly, the models are coded in python and the frontend is coded using javascript, html and css.

2.4 d. Data pre-processing summary

The Pre-Processing of the dataset for the tf-idf is very simple. We set some filters in order to take only the best movies in the dataset. For example, we filtered out all the movies with less than 10000 ratings and all the users in the Netflix dataset that did less than 200 user ratings. we had to clean and organize the data to remove irrelevant information. Sometimes, the data provided were in a JSON format like the genre of movie which has too many information, we had to make processing to extract the relevant information about the genre. We also had to remove movies that didn't have summaries because our model focus on summaries. We also applied a filter sparse movies, and a filter sparse user. For Data Analysis: we analyzed our data to identify patterns and trends, such as popular genres, and highly rated movies. Movies grouped by year of rehearsal, and distribution of Netflix ratings. As a data analysis, we find that the Shape User-Ratings unfiltered is (24053764, 4) and the shape User-Ratings filtered is (4178032, 4). When applying a model on user-ratings , we are able to select only few examples. we also Compute mean rating for all movies and we give the ranking of the top 10 mean-movie-ranking with an error of 0,99 RMSE.

```
# Filter sparse movies
min_movie_ratings = 10000
filter_movies = (df['Movie'].value_counts()>min_movie_ratings)
filter_movies = filter_movies[filter_movies].index.tolist()

# Filter sparse users
min_user_ratings = 200
filter_users = (df['User'].value_counts()>min_user_ratings)
filter_users = filter_users[filter_users].index.tolist()

# Actual filtering
df_filterd = df[(df['Movie'].isin(filter_movies)) & (df['User'].isin(filter_users))]
del filter_movies, filter_users, min_movie_ratings, min_user_ratings
print('Shape User-Ratings unfiltered:\t{}'.format(df.shape))
print('Shape User-Ratings filtered:\t{}'.format(df_filterd.shape))
```

Figure 2.3: Preprocessing tf-idf

2.5 e. Flowchart of the data-driven IDSS model/s gathering

Word-2-vec model is considered as a data-driven model because it makes decisions based on data and it uses data-mining techniques to detect trends and patterns. To explain better how our word2vec model works : Word2Vec is a method for learning a dense representation of words called "word embeddings" from text data. These embeddings capture the meaning of the words in a way that can be used for various natural language processing tasks, including movie recommendation. In our movie recommendation task, Word2Vec can be used to embed the movie titles and the plot summaries into vector space. These embeddings can then be used to calculate similarity between movies, which can be used to recommend similar movies to users. The basic idea behind Word2Vec is to use a neural network to predict the context words given a target word, or to predict the target word given a set of context words. The neural network is trained on a large corpus of text, and the weights of the network's hidden layer are used as the word embeddings. Once the word embeddings have been learned, they can be used to calculate similarity between words using a similarity metric such as cosine similarity, Euclidean distance, or Jaccard similarity. In our case , we chose the wmd distance. These similarities can be used to recommend movies based on the similarity of their embeddings. It's worth noting that word2vec is one of many methods to learn word embeddings, and it's not the only one, there are other methods such as Glove, BERT, ELMO.. For the word2vec model, we chose to integrate some filters for time computation improvment .The filters (like genre or year of rehearsal...) make the model suggest more relevant results.

2.6 models used

2.6.1 g. Description of model-driven IDSS techniques used

Our movie recommendation system includes two different models : tf-idf model and word2vec. the Tf-idf approach is model-driven because it is costumized to a predefined set of user requirements. It analyze different scenarios for user requirments and to improve user experience. To explain better the model, it is based on a numerical static model that defines how relevant a word in a series or corpus is to a text.Tf-idf is a famous algorithm in information retrieval and natural language processing fields. The word2vec model is a deep learning approach that

creates worb embeddings: it takes in words from a large corpus of texts as input and learns to give out their vector representation. Similar words will have close vector embeddings in the representation space. For this purpose, we have to choose a computational distance to measure the similarity between vectors and identify the most similar summaries in order to give a good movie recommendation. For our Tf-idf model, we choose the cosine-similarity distance. One of the main advantages of using cosine similarity is that it is relatively efficient to compute and does not require the vectors to have the same magnitude. Additionally, it is a commonly used and well-understood measure of similarity, so it is easy to interpret and explain the results. It is also easy to implement and can be used as a part of more complex algorithms.
To explain better how our tf-idf model works : The more frequent t is in d, the higher weight it should have. The more frequent t is in the whole collection, the less it discriminates among documents, so the lower its weight should be in all documents. A summary is a vector of weights : $s1 = [w1, w2, w3...]$ The weight of a term is calculated by multiplying its term frequency (TF) by its inverse document frequency (IDF).
The term frequency (TF) of a term in a document is the number of times it appears in that document. It is usually normalized by dividing the count by the total number of terms in the document. The inverse document frequency (IDF) of a term is a measure of how rare the term is across all documents in the corpus. It is calculated by taking the logarithm of the ratio of the total number of documents in the corpus to the number of documents that contain the term. The final weight of a term in a document is calculated as the product of its normalized TF and its IDF:

$$weight = TF \times IDF$$

This weighting scheme assigns a high weight to terms that are both frequent in the document and rare across the corpus, indicating that they are important in discriminating the document from others. Cosine similarity between two summaries is calculated by representing each text as a vector in a high-dimensional space, where each dimension corresponds to a unique word or token in the texts. The values of the vector in each dimension represent the importance or weight of the corresponding word in the text. To represent text as a vector we use the TF-IDF (term frequency-inverse document frequency) method, which assigns a weight to each word based on its frequency in the text and rarity across the corpus. Once the vectors for the two summaries are calculated, the cosine similarity is simply

the cosine of the angle between them. This is calculated as the dot product of the vectors divided by the product of the magnitudes of the vectors.

$$cosine\_similarity = \frac{A \times B}{||A|| \times ||B||}$$

Where A and B are the vectors representing the two texts and . represents the dot product, and $||A||$ and $||B||$ represents the magnitude of the vectors. This will give you a value between -1 and 1. A value of 1 means the texts are identical, a value of 0 means they are completely dissimilar, and a value of -1 means they are completely opposite.

for our data analysis , we give the top 10 weighted rating movies. It is a method used to compute the rating of a movie in a movie recommendation system. It is a way of combining the average rating of a movie with the number of ratings it has received, to give more weight to movies that have been rated by many users. The idea behind the weighted rating is that a movie with a high average rating but only a few ratings is less reliable than a movie with a lower average rating but many ratings. By taking into account both the average rating and the number of ratings, the weighted rating aims to provide a more accurate representation of a movie's popularity and quality.

2.6.2 Word2Vec and WMD distance

So, the other model used to suggest movies to the user is Word2Vec. Essentially, it embeds a document in lower-dimensional vector space using a shallow neural network; the result of the mapping is a set of doc-vectors where vectors that are close in the vector space have similar meaning, while doc-vectors that are distant have different meaning based on the context. Thus the user is giving the model a paragraph describing what he/she wants to watch, then a pre-processing phase splits the written text into tokens (that will be stored in a list) and removes the stopwords from it. The algorithm proceeds by computing the distances between the processed string given in input, and all the overviews in the dataset. In particular, the distance we used is the so called Word Mover's Distance (WMD) which enables to assess the "distance" between two documents even if they don't have any common words. Indeed, even if the two sentences do not have any words in common, by matching the relevant ones, WMD is able to measure the similarity between the strings we are giving it as input. Basically, the intuition behind the method is that we find the minimum "travelling distance" between documents.

Despite of that, the model is really heavy, and computing the distance for each movie in the dataset can take a long time. Based on the user's input, we need to decrease the size of the dataset in order to retrieve the results in a reasonable amount of time, and to do so we decided to filter out some samples of the dataset. The filtering is made in two phases: the first is guided by the user, while the second is done independently. So, when opening the application, the first thing the user will be asked to do is to set, if needed, any feature that must be present in the list of movies retrieved thanks to the model. In this sense, the program will reduce the size of the dataset by filtering out all those movies that do not contain the specification given by the user. Indeed, one will be able to:

- select the genre of the movie [e.g., comedy, adventure, thriller, and so on and so forth]
- the year from which to search for films
- the minimum vote given to the movie
- the duration of the movie [in particular, given a certain number as input, the program will take only those movies for which the duration is between the input +/- 10 minutes]

Also, since the dataset results in being very large but also very sparse (in the sense that there are many movies that are not completely described, and especially, they don't have a useful overview) it becomes mandatory, at least for the first model, to remove from the dataset all those movies for which the overview is too short, it's an empty value, or it's a value such as 'overview not found'.
This second approach to retrieve movies' suggestions is thus a mixture between a model and data driven approach: in particular, the optimization of the model and how it embeds each paragraph into a certain vector is crucial for its performance, at the same time though, it is also mandatory to clean the dataset as best we can since a very large and sparse one may be kind of difficult to analyze (sparse for the fields that we need, hence the title and the overview essentially). Also, if the user is able to select certain features that the suggested movies must have, so to reduce the size of the dataset, the algorithm can only benefits from this.

2.7 h. Validation and Results

2.7.1 tf-IDF

We asked our model to recommand movies same as the 'Assasins'. We got the following top 10 of recommanded movies :
['Der blaue Engel', 'Sharpshooter', 'Kinatay', 'Bad Teacher', 'Killers', 'MONDAY', 'King of the Ants', 'Colombiana', 'The Assassination of Jesse James by the Coward Robert Ford', 'Blast of Silence']
From analyzing the titles, we can imagine that those movies has the same topic about killing, shooting , the bad guy in movies, assasination.. In case our recommendation system didn't work well, we will have some recommendation about love stories or funny movies while our input is about assasins and violence. It would have been incoherent. Recommended movies same as 'the GodFather' :
['The Godfather: Part II', 'The Godfather Trilogy: 1972-1990', 'The Godfather: Part III', 'Blood Ties', '', 'Mobsters', 'Live by Night', 'Bad Turn Worse', 'Miss Bala', 'Family Business']
We see that the recommanded movies include other chapters of the godfather and other similar movies. So we succeed in this task with other types of movies too. Recommended movies for 'Batman Begins':
['Batman Unmasked: The Psychology of the Dark Knight', 'Batman: The Dark Knight Returns, Part 1', 'Batman: Bad Blood', 'Batman: Year One', 'Batman: Under the Red Hood', 'Batman Beyond: The Movie', 'Batman Forever', 'Batman: Mask of the Phantasm', 'Batman Bill', 'Batman']
In our case ,we are only referring to a human evaluation. In other cases, we can also evaluate our recommendation model using recall and precision.
Where,

$$Precision = \frac{TP}{TP + FP}$$

and

$$Recall = \frac{TP}{TP + FN}$$

To evaluate a movie recommendation system using precision and recall, we need to have a set of labeled data, where the relevance of the movies has been labeled by human experts or by users. It is not our case, so we cannot evaluate using those metrics , and labelling data takes too much efforts so we couldn't find a labelled dataset for our study.
We also plot the tf-idf scores for some words in the summaries :

astrology	10.287532867806089
barbarism	10.287532867806089
barca	10.287532867806089
astronomer	8.783455471029814
astronomers	9.882067759697923
bad	5.385968668764194
balance	6.761172343189927
bands	7.845185832436884
bar	6.307851213904128
barely	6.973346863133562

We see that for some words (astrology, barbarism, barca..), we have high scores because those words are significant and can inform us well about the topic of the movie, while other words like (bands, bar, barely..) don't provide too much information about the topic of the movie, that's why the scores are lower.

Concerning our data analysis , we have the following results for the top 10 weighted movie ratings :

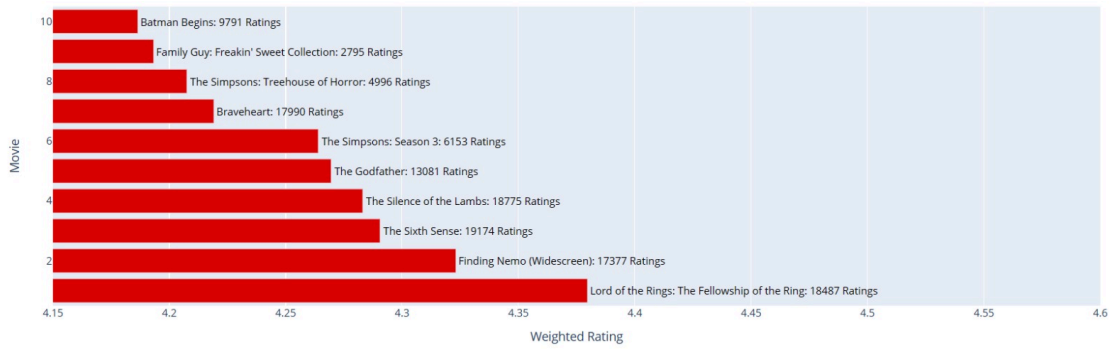
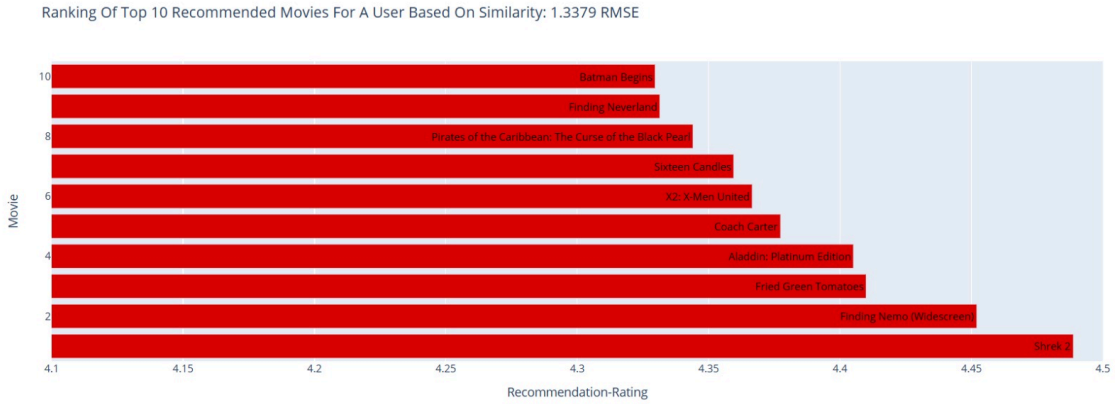


Figure 2.4: ratings of the top 10 movies

We had also the idea to plot the top 10 recommended movies for a user based on its similarity to other users.In this case, we choose the number of Number of similar users for recommendation, we compute similarity between all users, we sort similar users by index and score, we compute weight ratings of the top n most similar users with their rating and compute the mean for each movie. we select the ratings of the top n most similar users, and compute the mean rating for each movie by weighting the ratings with their similarity score.
For each user we compute the predicted rating of the movie by weighting the ratings of the top n most similar users with their similarity score . The purpose of this analysis is to implement a movie recommendation system that can recommend movies to users based on their preferences, interests, and viewing history. The sys-

tem uses cosine similarity to find similar users and recommends movies that those similar users have rated highly.



2.7.2 Word2Vec/WMD

For the Word2Vec model the results are a bit more generic: since it's not looking at other information but the overview of the movies, it's difficult that given a certain title it will also return that specific movie [this is because the title may not match the overview at all]. Indeed, the aim of this second model is not only to give a word as input, but an entire paragrah instead, so that the user can tell a "story" to the program, and the program in turn, gives a list of the movies which best suit the description depicted by the user (based, as I said, on the overviews).
Despite of that, since we want to compare the results of the 2 models too, we tried to give the same input used for tf-IDF. For example, by writing 'batman begins' the results are:

original_title	distance
LEGO DC Comics Super Heroes: Batman: Be-Leaguered	0.832
Bangkok Dangerous	0.892
The Dorm	0.925
Tokyo Fist	0.932
L'Enfer	0.9323

We can see how the most similar result, with the mimum distance among the all 5 movies, is indeed regarding batman, and its overview is 'Superman wants Batman to join his new superhero team, but Batman prides himself on being a self-sufficient loner'; since, we didn't give the program a lot of details regarding the description of the film we want to watch, it makes sense that the closest one is a batman-related film, with the word batman in the description.

Apart from that, the other movies are still somehow related to the batman atmosphere; take a look at Bancock Dangerous's overview for instance: 'The story is of a deaf-mute hitman and his partner who are based in Bangkok. He is friends with his partner's girlfriend who is a stripper at a local club. They go about their assassination business as usual as the boss climbs the underworld ladder and forms new alliances. Flashbacks explain how he got to this point in his life. He forms a relationship with a young woman from the pharmacy who is caring and innocent. The hunts continue as treachery begins. A big name hit makes him realize that good people are hurt by his actions. The first rule of assassination closes in on him as he strikes revenge'. We can readily understand how the film revolves around a dark and dangerous atmosphere, similar to the one of a batman movie. Also, the genres associated to the aforementioned film are [thriller, action, crime, drama], which are essentially the same as a typical batman film.
Now let's take a look at the results associated to the input 'assassins'.

original_title	distance
Mariposa Negra	0.831
Executive Action	0.862
Supersonic Man	0.872
Battaglie negli spazi stellari	0.880
Blablablà	0.889

In this case there is no film titled Assassins in the results, even though there surely is one in the dataset. It's obvious that such a generic word is not helpful to search for a specific movie named with that word, instead there are many overviews that contain the word assassin (or some derivatives). Mariposa Negra's summary, for instance, state 'A Peruvian schoolteacher conspires with a journalist to assassinate the official responsible for her fiancé's murder', and the words 'murder' and 'assassinate' are definitely really close to 'assassins'.
The last one in the list, Blablablà, has the following overview: 'The tensions experienced by three different people during the military dictatorship in Brazil: a politician, a revolutionary and a common citizen'. Eventually, there are no references of the word 'assassin', but many can be re-conduced to it, for example 'tensions', 'military', 'dictatorship'.
Finally, we want to show the results regarding the actual aim of this second model: the input to the program will be 'space battle and aliens fighting humans' and the list of results are:

original_title	distance
Survivor	0.774
The Ledge	0.789
Invaders from the Deep	0.816
Gojira: Fainaru uôzu	0.817
The Beast with a Million Eyes	0.831

just to report some of the actual overviews of the movies listed in the results, we have:

- Survivor: During their search for a habitable planet the last living humans crash-land on a barren world, inhabited by bloodthirsty aliens and mysterious post-apocalyptic warriors.
- Invaders from the Deep: When aquatic aliens plot to take over the planet, the world aquanaut security patrol are called in to battle the aliens.
- The ledge: A thriller in which a battle of philosophies between a fundamentalist Christian and an atheist escalates into a lethal battle of wills. Ultimately, as a test of faith, or lack of it, the believer forces the non-believer onto the ledge of a tall building. He then has one hour to make a choice between his own life and someone else's. Without faith in an afterlife, will he be capable of such a sacrifice?

'The ledge is definitely the less coherent with the requested description, it looks like that the model gave really importance to the word battle, that in the overview occurs twice, but also to the other part of the text, since we can state that it's a really intense movie revolving around a fight between two different groups (like humans and aliens in description we gave). Eventually, what we discover is that the biggest problem for this model is that it is very time-consuming: computing the distances for of the overviews of all the movies in the dataset might also take a couple of minutes to end, and thus a certain filtering from the user is almost mandatory in order to retrieve a list in less than 30 seconds.

2.8 i. Future work and improvements

2.8.1 Different measure of similarity

We can lead experiences on other type of distances (euclidian distance, jaccard distance..) to see what distance can perform better.

2.8.2 Different models

We can also test on other deep neural network methods to measure similarity between the summaries , or the provided keywords and summaries. If our user provide a query for searching a movie , we should be able to extract the key words from his query using deep learning and use it for selecting more relevant results.

2.8.3 Algorithm Connection

Another idea could be the connection of all the model in only one feature map in order to obtain only one result. So mixing up the two model it will be possible to improve the results without the necessity of train another model.

2.8.4 Performance improvements

Then another important improvement could be the Performance improvements of the already trained algorithm. In fact, it could be a great idea using Fine tuning or Domain Adaptation in order to have a more specific model for our purpose. A simple implementation could be retrain with the previous weights and train it with the Movie Dataset in order to specialize the model. Also it is possible to set all the hyperparameters and changing them to obtain better performance. Another performance improvement could be the increase of the server performances.

2.8.5 Functionality Extension

The front-end is very basic and a lot of additional features can be added, for example Save the user preferences that actually are simply taken from the netflix dataset, different movie inputs like

2.8.6 Filternig extension

Since the Movie dataset is full of metadata there is also the possibility to add filters of our choices during the user selection.

2.8.7 precomputation of the data

Obviously, computing the vector representation for all the overviews in the dataset become very expensive in terms of time when the size of the dataset becomes very large. A good way to reduce the needed time for the Word2Vec model would

therefore to pre-compute the matrices for all the reviews, and use them directly instead of converting them each times.

2.8.8 Dataset quality

We initially picked our actual dataset. However, we found other richer and bigger datasets that can be exploited better. In our actual dataset , we have many information provided and not only the movie summaries. Exploiting data related to users is also a good idea to take into account user preferences.

2.9 j. Gantt diagram with tasks planning

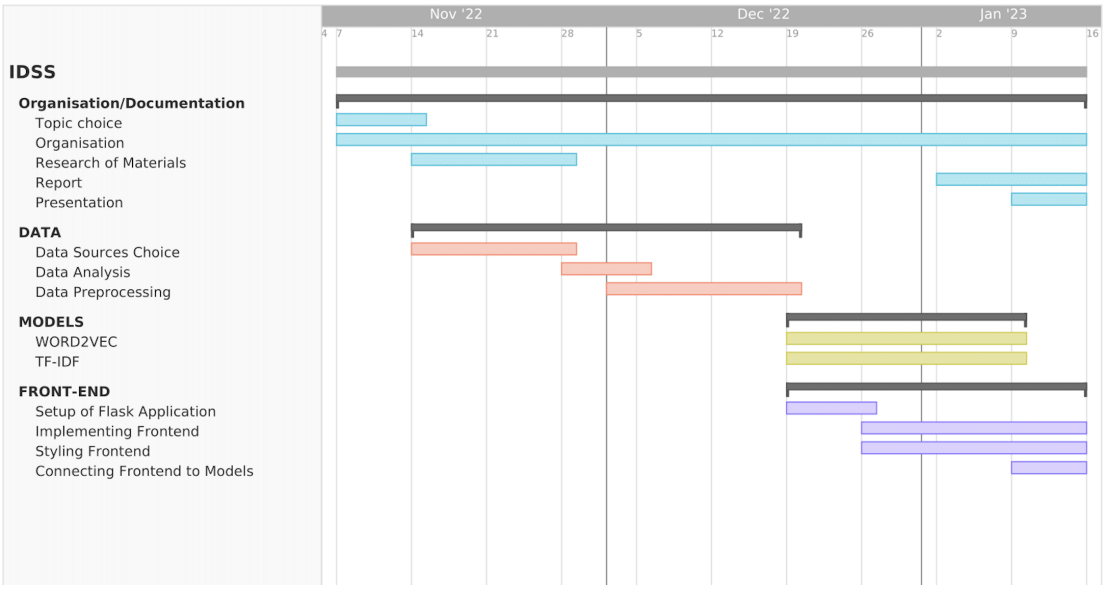


Figure 2.5: GANTT Diagram

2.10 k. Tasks assignment and responsibilities among teamwork members

TASK	MEMBERS				
	Niko Picello	Salma Waqif	Lapo Carrieri	Hanaa Al Zahabi	Yasmine Souabi
ORGANISATION/DOCUMENTATION					
Organisation	x	x	x	x	x
Topic Choice	x	x	x	x	x
Report	x	x	x	x	x
Research of material	x	x	x	x	x
Presentation	x	x	x	x	x
DATA					
Data Sources Choice	x	x	x	x	x
Data Analysis	x		x		x
Data Preprocessing	x		x		x
MODELS					
Implementing WORD2VEC Model	x				
Implementing TF-IDF Model			x		x
FRONT-END					
Setup of FLASK Application		x		x	
Implementing Frontend		x		x	
Styling Frontend		x		x	
Linking with the Models		x		x	

Figure 2.6: Tasks Assignment

2.11 l. Time sheet

TASK	MEMBERS					TOTAL
	Niko Picello	Salma Waqif	Lapo Carrieri	Hanaa Al Zahabi	Yasmine Souabi	
ORGANISATION/DOCUMENTATION						
Organisation	3	3	3	3	3	15
Topic Choice	3	3	3	3	3	15
Report	8	8	8	8	8	40
Research of material	8	8	8	8	8	40
Presentation	4	4	4	4	4	20
DATA						
Data Sources Choice	2	2	2	2	3	11
Data Analysis	4		2		5	11
Data Preprocessing	6		4		6	16
MODELS						
Implementing WORD2VEC Model	10		2		3	15
Implementing TF-IDF Model			12		5	17
FRONT-END						
Setup of FLASK Application		2		3		5
Implementing Frontend		6		8		14
Styling Frontend		4		7		11
Linking with the Models		8		2		10
TOTAL	48	48	48	48	48	240

Figure 2.7: Time Sheet