Recommendation Engine on IPTV

Vedad Njuhović*, Samed Jukić*
*International Burch University,
vedad.njuhovic@stu.ibu.edu.ba
samed.jukic@ibu.edu.ba

Original research

Abstract: *In recent years IPTV (Internet Protocol Television) platforms are becoming one of the most popular entertainment multimedia services which are used to serve movies, tv-series and other video and audio attractive content using the Internet Protocol. VoD (Video on Demand) is the most popular multimedia IPTV service, which provides content without the need for the old traditional way of using video playback devices. Except that it is necessary to have high-quality VoD content, IPTV platforms must provide the best end-user experience. Moreover, it is imperative to provide new features to attract new customers and keep the existing ones. We confirmed the efficacy of this classifier thru simple trial and error. When we searched for movies that have sequels, our engine recommended those sequels. Since Cosine Similarity Classifier considers multiple factors, such as actor, genre, year, etc. Even if the movie does not have prequels or sequels this algorithm was able to provide us with movies that share other common characteristics.*

Keywords: IPTV, engine, unicast, broadcast, multicast, python, CSS, HTML, JS.

1. Introduction

In the next couple of chapters, this paper will explain what IPTV is, what is a Recommendation engine, and how a recommendation engine can be used to improve the end-user experience. In the first part, we will define general terms which need to be understood before proceeding. The second part will cover current solutions which are implemented on market for this purpose. The methodology will be covered in the third part, which includes technologies used for the implementation of this project. The last part will summarize the results obtained after the implementation of this system.
IPTV stands for "Internet Protocol Television". The "IP" in IPTV is the same as your IP or VoIP (Voice over IP) address. All this means that television programs are distributed using Internet protocols [1].
A recommendation engine is a data filtering tool that uses machine learning algorithms to recommend the most relevant items to specific users or customers. It works by looking for patterns in consumer behavior data, which can be collected implicitly or explicitly [2-5].

2. Literature Review

It is really difficult to enumerate all the different fields in which recommender systems are applied. For example, they are used in e-commerce [6], [7], for recommending music, movies, scientific papers, etc. GroupLens [8] was one of the first implementations of recommender systems. Unlike our hybrid technique, it uses the traditional collaborative filtering approach to calculate new recommendations. Later, the same research group developed MovieLens, a movie recommender system that is also based on traditional collaborative filtering. We proposed an approach based on Cosine Similarity Classifier to increase recommendation accuracy. Based on this approach, we developed a movie recommendation system that combines both content-based and collaborative information about movies. This system takes into account all separate information about a movie such as actors, year, genre, etc, and by searching most similar items based on multiple factors it delivers the most similar movies.

A more interactive movie recommender system, named MovieGEN is presented in [20]. This is a hybrid recommendation system, which uses machine learning and cluster analysis to calculate recommendations. However, this system uses the personal information of users to predict their movie preferences using well-trained support vector machine (SVM) models. Then, based on SVM predictions, it selects movies from the dataset, clusters them, and generates questions for users. Finally, it uses the information collected through the answers to refine user recommendation lists. The system recommended by Conti's IPTV [5] is a recommended social system based on the profiles of the social network and the analysis of the activity of personal users according to similar activities. Social recommendations. In addition, we analyzed the information of the live program of Electronic Program Guide (EPG) and we made the time division for recommendations to distinguish users. Jinni [8] is a semantic search engine for movies and television content and a recommendation of them. Another engine analyses only the information of the standard film (as titles, actors, lists, director, scenarios, etc.) approach such as this one, enables significant content searches and allows quick adaptations to the user's benefit. The authors are based on the implicit feedback of the user who divides the user profile into multiple subfiles called Microfofiles based on the user, each based on multiple subfiles representing the user's technology introduced. Two different context Recognition technologies for film recommendations are shown in [7]:

- Greater performance of conventional C.
- Approaches that use electrical contextual time factors.
- Implementation of machine learning.

The authors propose the use of temporary information to improve the recommended quality. For example, [13] proposes a technique to model the temporary dynamics of client preferences by separating the transient factors from the last. The recommendation service of Recommendation, based on the interesting location, which takes a Temporary context, is [15], but [16] is feedback from implicit users and information on the user's purchase time and the start-up of the article to achieve the accuracy of the recommendation.

3. Methodology

A. Acquisition

To prepare the recommendation engine, data should be collected, cleaned, and prepared for the engine needs. In this project, we have collected data from 6 different sources, cleaning of data was performed in 5 pre-processing steps which are going to be explained in detail below.
Pre-processing 1:
The first dataset was retrieved from Kaggle and is called "IMDB 5000 Movie Dataset" with a CSV file called "movie_metadata.csv".
Movie_metadata.csv consists of movie information with columns such as director name, main actors, movie name, genre, number of reviews, number of likes, duration, etc. From all the columns we will keep only the most important columns such as 'director_name',' movie_title', 'genres' etc.

Since in each column there are a lot of NaN values which are replaced with 'unknown'.
In this dataset genres are written with pipe '|', so it should be replaced with just blank/space instead, for the better processing of the data. Once all NaN values are replaced it is time to transform all movie titles to lower case. After completing this step, we realized that each movie name consists of \xa0 (single character) which should be removed to have correct names. Using the lambda function (iterating each movie title) and using the syntax [:-1] we are excluding the last part of the movie titles. Finally, the last step is saving cleaned and prepared data to the common CSV file.

B. Pre-processing 2

Since the first movie dataset included data till 2016, this dataset includes data from 2017. So using this dataset we will take data of the 2017 year and merge it with the first dataset. This dataset has separated actors and directors in another dataset called 'credits.csv', this data should be merged with the main dataset by id. Genres in this dataset are like an array of objects but in form of a string, they should be converted like in the first dataset, like genres with space-separated. For converting this, we have used AST (abstract syntax trees) and 'literal_eval' which constructs an object from string. Using a custom function for iterating through the objects in the array we achieved the same result for genres like for the first dataset. The same process is repeated for extracting actor names and also for the directors.
After this, we need to take only the most important columns, like for the first dataset with dropping NaN values and renaming all columns to match the first dataset.
The most important step is the creation of a new column called 'comb' which consists of director names, actor names, and movie genres. The last but not least step is combining the first and second movie datasets into the same file.

C. Pre-processing 3

The first and second data set was taken from Kaggle and includes movies till 2018, so we decided to take movies from Wikipedia by extracting a list of films in 2018. After taking all the

movies from Wikipedia, I realized that there is no genres column. Using the tmdb3api for each movie by its name retrieve its id which is used for the new request for getting genres.
Since the cast and crew column included all information including directors and actors we needed some custom functions with basic splitting for retrieving its names. Last but not least step is renaming all columns to match the previous dataset. The last step is the creation of a comb column with merging all previous data with 2018 data.

D. Pre-processing 4

In pre-processing 4, we have performed the same steps as in pre-processing 3 for retrieving movies of 2019 and 2020.

E. Pre-processing 5

In pre-processing 5, we have performed the same steps as in pre-processing 3 for retrieving movies of 2021.

F. Similarity matrix

After the prepared data, we are ready to move forward with the creation of a recommendation engine.
Our dataset contains as we already explained info such as movie name, genre, casts, director, etc. The last field in the dataset contains a field that has all this data combined, as plain text.
Let us consider that our dataset has only 10 movies so we simplify the explanation. Since we have 10 movies, we will generate a matrix of size 10x10, where for example element [3], [5] quantifies how much movie 3 is similar to movie 5. These quantities are calculated by cosine similarity, which is explained earlier in this chapter two plain texts are compared, each movie is compared with each one, and then the algorithm calculates similarity as a number between 0 and 1. These calculations are only performed once, and this matrix is saved.

Now consider that the user is searching for movie number 7 in our list (that is movie from our 10-movie dataset which we mentioned above). Our program will take all movies from row number 7 because that row contains similarity of movie 7 with all other movies in our database. If the numeric value in the field is higher that means the movie is more similar to the one user searched originally. If we want to get the top 5 most similar movies, we will request the top 6 movies, and ignore or remove the one with similarity one, or the most similar movie, because this is the value where movie number 7 was compared to itself, or in our case that would be [7].

4. Results

In the figure below, the homepage of our application is presented.

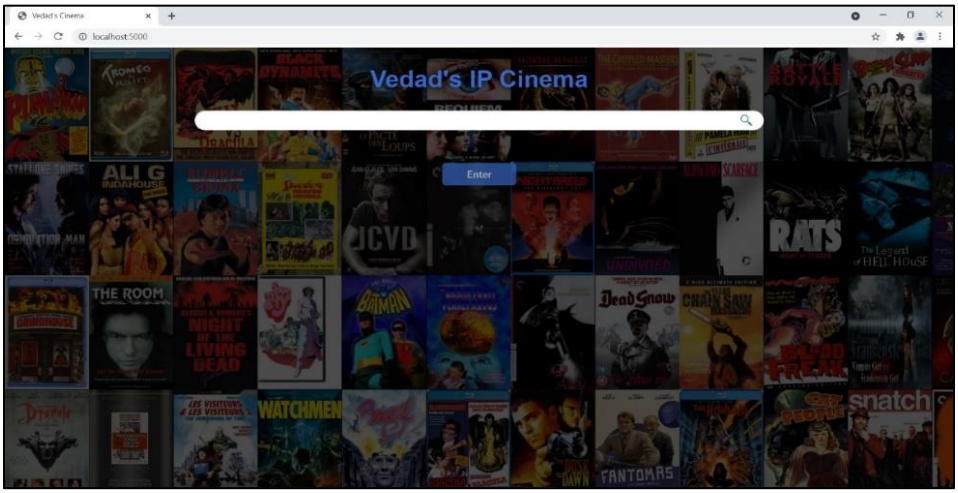

FIGURE 1. Homepage.

The user has can browse thru movies. After entering three letters recommended movies start to pop up. Recommendations or autocomplete are obtained by calling the function "get_suggestions()", which gets data from AJAX requests. After the user selects a movie, the screen from figure 2 below appears.

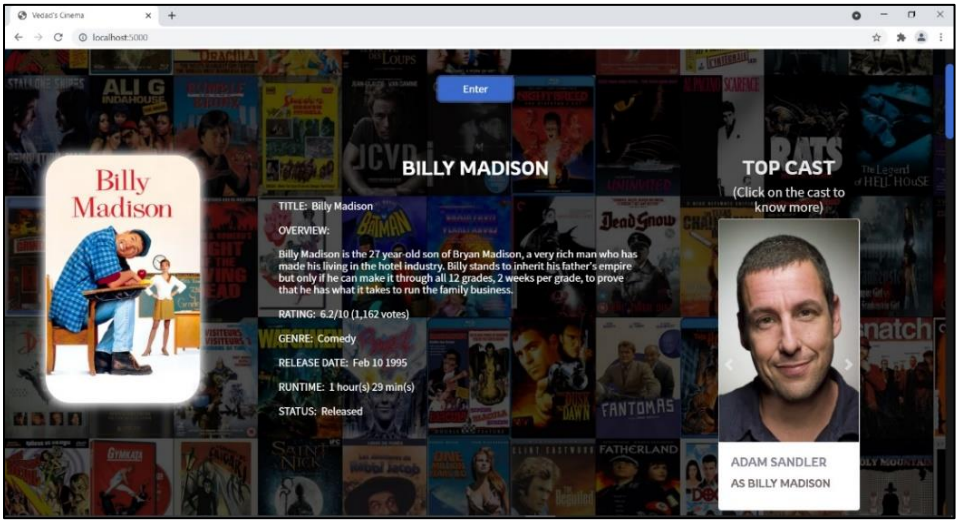

FIGURE 2. Movie page.

If we scroll down, we may see user reviews, which are obtained from IMDB. Figure 3 contains reviews and is presented below.

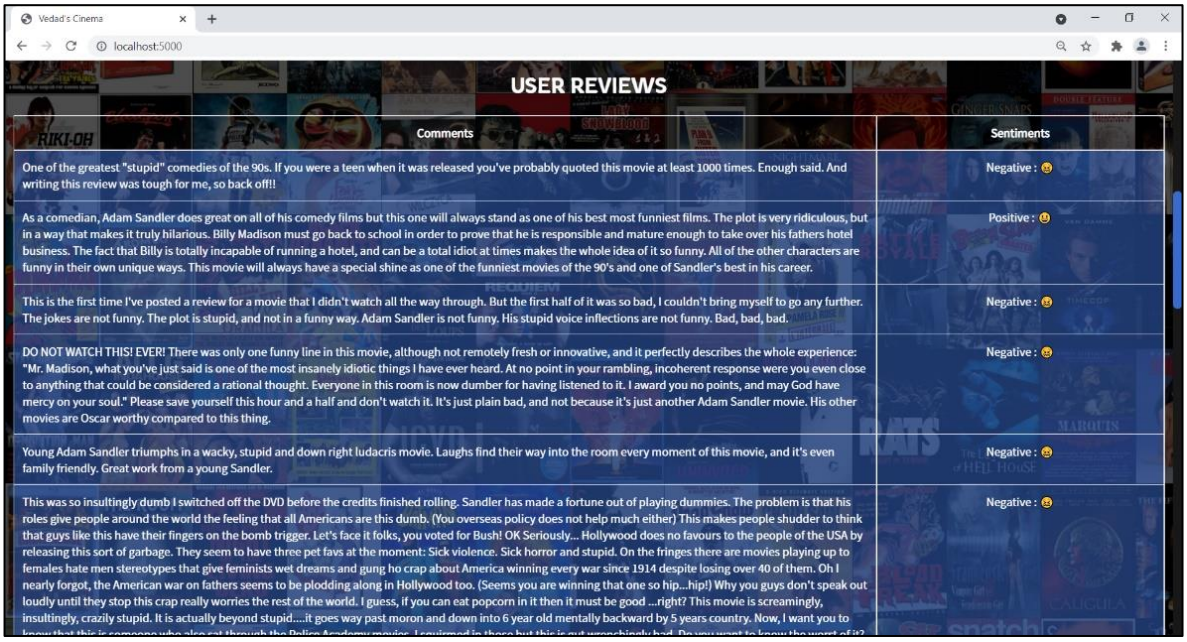

FIGURE 3. Representation of user reviews.

At the bottom of the page, we may find suggested movies, which are delivered by API based on genre, actors, and similar factors.

A. Accuracy

Since the recommendation engine needs to be personalized toward users' requirements, we surveyed a small group of people and asked them which movies they expect to be recommended if they searched for certain movies.
Survey returned the following results:

- System recommended 1 out of 4 expected movies 2 times
- System recommended 2 out of 4 expected movies 5 times
- System recommended 3 out of 4 expected movies 38 times
- System recommended 4 out of 4 expected movies 5 times

Graphical representation of results is presented in Figure 4

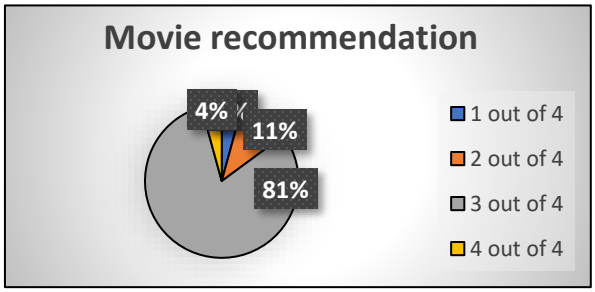

FIGURE 4. Pie chart representing survey results

5. Conclusion

IPTV as a concept first time appeared in the 1990s, and it presented a way of delivering video content over networks [1]. The factors such as reliable package delivery protocol and video compression algorithms allowed this technology to be developed in commercial installations. Primary underlaying protocols which are used in stand-based IPTV are [3]: Unicast web-based live and VoD streaming and Web-based multicast. A set of hardware equipment and software which are interconnected via a network is called an IPTV system [5]. Video on demand servers pr storage for media such as tv shows, video clips, movies, and similar. Their job is to provide secure and perfect access to content that is stored [8]. Session level protocol which is defined by the IETF to provide transportation functions over the network that facilities the delivery of data such as video or audio in real-time using either unicast or multicast technology is called Real-Time Protocol. Technologies that we used for the development of this project are HTML, CSS, JS, AJAX, and Python. Cosine Similarity presents a measurement that quantifies the similarity between at least two vectors**.**

Considering that we spent most of our time preparing the data for processing, it is clear how important this is for the modeling itself. In comparison with the already made recommendation engines, we have shown that in a much simpler way it is possible to make a good enough recommendation engine, which proved to be of good quality taking into account the statistics of the respondents.

The main reason why I chose this topic and embarked on the adventure of making a recommendation engine is the current trend of watching Netflix and the marathon search for a movie we would like and would love to watch. Taking into account the result of the project, I managed to reach the goal I had set.

6. References

[1] International Telecommunication Union ITU, «Agentia National Pentru Reglementare», January 1998. [En línea]. Available: http://www.anrceti.md/files/filefield/Recomandarea%20ITU%20H.323_0.pdf.

[2] Committed to connecting the world, «Telecommunication Standardization Sector», ITU, [En línea]. Available: http://www.itu.int/en/ITU-T/Pages/default.aspx.

[3] I. K. Park, O. Seung Hun, Y. S. Kwon and H. Young Song, «An implementation of user-participated interactive IPTV service system», IEEE International Symposium on consumer electronics, Perth, 2008.

[4] K. Kerpez, D. Waring, G. Lapiotis, J. Lyles and R. Vaidyanathan, «IPTV service assurance», IEEE Communications Magazine, vol. 44, n°9, pp. 166-172, 2006.

[5] G. Myoung, L. Chae, S. Lee, W. Seop Rhee and J. Kyun Choi, «Functional architecture for NGN-based personalized IPTV services», IEEE Transactions on Broadcasting, vol. 55, n°2, pp. 329-342, 2009.

[6] Arnold, D., Bond, A., Chilvers, M., & Taylor, R. (1996, June). Hector: Distributed objects in python. In *Proceedings of the Fourth International Python Conference, Livermore CA*.

[7] P.S. Pacheco, Parallel Programming, Morgan Kaufmann Publishers, 1997.

[8] López Sarmiento, D. A., Villanueva Ocampo, B. F., & Rivas Trujillo, E. (2013). Iptv: Next-generation network technologies and protocols. *TECCIENCIA*, *7*(14), 39–48. https://doi.org/10.18180/tecciencia.2013.14.5

[9] Mirri, S., Peroni, S., Salomoni, P., Vitali, F., & Rubano, V. (2017). Towards accessible graphs in HTML-based scientific articles. *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 1067–1072. https://doi.org/10.1109/CCNC.2017.7983287

[10] Zeadally, S., Moustafa, H., & Siddiqui, F. (2011). Internet Protocol Television (IPTV): Architecture, Trends, and Challenges. *IEEE Systems Journal*, *5*(4), 518–527. https://doi.org/10.1109/JSYST.2011.2165601