**Movie Recommender Implementations with Sentiment Analysis on a**

**Chatterbot System: Foxie**

**Can Mergen, 0078490**

Graduate School of Sciences and Engineering, Koç University

***Abstract:***Recommender systems encompass the future for both service providers and users. The sector has great market potential attracting organizations eager to invest in the new frontier. This project aims to create a movie recommendation system based on this approach while including the movie reviews to predict their sentiments; all the visualization, information, sentiment analysis, and recommenders were also included inside a chatterbot to create a more user-friendly environment for non-developer users to investigate it. The datasets are taken from the 45,000 movies listed in the Full MovieLens and Netflix-Hulu-Disney-Amazon Prime title datasets on Kaggle, IMDB movies dataset on their website, and lastly, asked for permission and granted from IMDB to use web-scraping for gathering movie reviews for the selected film. The access is only granted to not use it for commercial purposes and not scrape repeatedly to keep the system busy. Furthermore, all the movie recommender systems are used, from the simple recommendation systems for top-scored movies to more complex approaches such as using only poster images for movie score prediction and only genre selection for movie score prediction. Since the poster data is used for image classification and genre data for regression, a convolutional neural network joined with a fully connected multi-layer perceptron is trained. The results show that the proposed MLP+CNN model beats both the MLP model and the CNN model baselines with a mean absolute error of 0.1262 (0.02 loss) compared to 0.9284 (loss: 1.2753) and 0.9258 (loss:1.419). This shows that our models are viable for accurately predicting movie scores. Further analysis of other recommender parts shows that building a more complex model creates a better prediction environment with the highest positive impact; however, if it is too complex, it has the highest negative impact on users by giving inappropriate recommendations. In addition, the results suggest increasing the users' data mainly for collaborative filtering and combining it with other methods such as content recommender to create a better service solution. This will increase user attraction and satisfaction while increasing the overall market potential, to prove this, the project aimed to build every recommender system and display their results.

***Keywords:*** Simple Recommender, Content-based Recommender, Collaborative Filtering Recommender, Hybrid Recommender, Machine Learning, Deep Learning, EDA, ANN, MLP, CNN, Sentiment Analysis, K-Nearest Neighbor, Chatterbot

1. **Introduction**

Recommender systems are one of the most widely used data science applications today. The possibility of a future where people do not have to waste their time and gather their needs immediately is an appealing idea for individuals and organizations. The eagerness to have more personalized recommender systems in daily life presents an opportunity for service providers and organizations interested in diversifying their businesses. Recommendation-affiliated companies and other businesses that successfully tap into this sector will benefit from the huge market potential. The global recommendation engine market size was valued at USD 1.77 billion in 2020 and is expected to expand at a compound annual growth rate (CAGR) of 33.0% from 2021 to reach USD 54 billion by 2030 (Straits Research, 2022). Recommender systems are used to anticipate a user's "rating" or "preference" for a certain item. Almost every major IT firm has used them in some capacity. It's used by Amazon to recommend products to customers, YouTube to determine which video to play next on autoplay, and Facebook to suggest pages to like and people to follow. Recommendation systems, particularly in streaming services, are a key element of proposing goods to its users. Recommendation algorithms are critical for streaming movie services like Netflix, Disney, Amazon, etc., to help its customers to discover new movies to watch and have better user-friendly websites to stay in the competition, among others. Additionally, because of how congested the world is now, recommendation engines are necessary when endorsing goods or services; they reduce transaction costs and enhance the quality of users' decision-making (Nagamanjula, 2019). To do that, the collection of a ton of data on customer preferences for a variety of goods is needed to perform. It keeps records of data in a wide range of beneficial and undesirable ways and records user feedback for movies seen, locations visited, and goods bought. Compared to other recommendation systems, which depend on quick computation and processing from service providers and product distributors, the demand from consumers for shopping products, service providers (such as travel and restaurants), etc., movie recommendation systems possess a significant design challenge. To propose movies, one must first compile user ratings before presenting the target user with the top recommendations. In addition, users may look at testimonials from other users. Collaborative filtering, content-based recommender systems, and hybrid recommender systems are only a few of the numerous recommendation algorithms that have been described. Machine learning (ML) plays a key role in the adoption of recommender systems, so developers use it to input instructions that facilitate velocity, mobility, and safety. This project includes a historical scheme from the very beginning to more futuristic recommenders and creates a user-friendly chatbot to display the recommendations.

1. **Related Work**

Over the years, several different recommendation systems have been created. These systems employ a variety of methodologies, including collaborative, content-based, utility-based, hybrid, etc. A collaborative filtering approach system MOVREC presented by D.K. Yadav et al.; content and collaborative filtering approach by Herlocker et al.; hybrid approach by H. Kaur et al.; sentiment analysis by P. Baid et al.; and chatbot design by Lin L. et al. The majority of today's recommendation systems rely on user evaluations to locate new clients. These evaluations are also used to forecast and suggest the item of a person's preference. Simple, content-based, collaborative filtering and even hybrid recommendation systems are used in most recommendation projects. This project also contains a historical part of recommender systems; so, to preserve this idea, these recommendation systems are also used in this project. However, they are upgraded from their basic models to create a unique perspective on these models while evaluating new recommendations such as generating movie score prediction from only looking at posters and from only looking at their genres. Users can also input their own posters and genres to predict how would be the outcome, using a web-scraping tool to perform sentiment analysis on selected films and all the methodologies in a unique chatbot design used in this project. This project may also lead to a new era of recommendation systems based on the sentiment analysis combined with the recommender engines to perform a better user approach, and producers may evaluate their genres and posters to get more people and eventually gain more than before.

1. **Approach**

This section will contain the necessary information about the datasets used in this project, their EDA reports, preprocessing steps, structure with implementations, and design process.

**A) Dataset Description**

The datasets are gathered from many sources. The main dataset comes from the movies dataset and **appended Netflix, Disney, Prime, and Hulu platform datasets on Kaggle, whereas used IMDB’s main website to gather information to create a comparison of recommendations. Furthermore, used web-scraping to gather IMDB reviews of the selected film.**

***A.1) The Main Dataset with Platforms Added:***

Table

Description automatically generated with medium confidenceIn total, there are 44970 rows and 29 columns. The columns include numerical, categorical, and others that don’t fit with both categorical and numerical. Numerical columns include budget, popularity, revenue, runtime, vote\_average, and vote\_count. Categorical columns include adult, genres, id, imdb\_id, original\_language, production\_companies, production\_countries, spoken\_languages, status, and video. Other columns include belongs\_to\_collection, homepage, original\_title, overview, poster\_path, release\_date, tagline, title, and platform. The dataset doesn’t include any duplicate rows. However, it contains null values.

**Table

Description automatically generated with low confidence*A.2) The IMDB Dataset:***

In total, there are 40108 rows and 5 columns. The only numerical column is IMDB score, and the only categorical column is movie\_genre; others are imdb\_id, imdb\_link, and poster.

***A.3) The Web-scraping Dataset:***

Text

Description automatically generatedIn total, there are 251 rows and 5 columns. However, the row size can change from one selected movie to another. It only gathers the ratings lower or equal to four and higher or equal to seven. There are no numerical columns in the dataset. Categorical columns include rating and sentiment. Others are title, text, and id.

**B) Data Preprocessing and Feature Extraction**

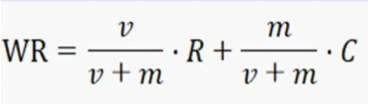
Only the main dataset gathered from the sources had ambiguous and unnecessary columns for my future predictions and the EDA visualizations. To create a more appropriate dataset, some columns are removed, replaced, changed, and others are added. Removed columns include adult, status, original title, release\_date after extracting only year, poster\_path, tagline, release\_year, imdb\_id duplicated rows. Adult and status columns were useless, so after gathering status to only released ones, dropped both. Some original title column values were the same as the title column, whereas others were different and created a biased dataset for future work. The poster path was no longer available and wasn’t gathering any poster images for CNN or Multi Input processes, so it was dropped and replaced with the available poster path from the appropriate IMDB dataset. The others were unnecessary for the evaluation. Apart from that, turned belongs\_to\_collection into gathering only collection name, genres column to have only genres separated by a comma, replaced missing revenues and budgets with zero, profit column is created with the combination of budget and revenue columns, year column created from release\_date before removal, tagline combined with an overview before tagline removal, production\_countries, spoken\_languages, keywords, cast, crew dictionaries’ values combined with commas, and if any imdb\_id value is NaN, it dropped from the dataset.

**C) Learning Approach and the System Structure**

This project contains many different approaches and inputs combined to give one selected output to the user. Some approaches are coding and mathematical related, some are machine learning related, and others are for user-based approximations of the problem.

***C.1) Simple Recommender (Popularity Based Recommender):***

It offers a generalized recommendation to every user in the system based on movie popularity and/or genre. The basic idea behind this system is movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience. For this section, I used the IMDB's rating formula and sorted them in descending order. The simple rating formula is:

“WR” is the weighted rate output, “v” is the number of votes for the movie, “m” is the minimum votes required to be listed in the chart, “R” is the average rating of the movie, “C” is the mean vote across the whole report.

That part also contained three different simple recommender ideas. One of them was the same as IMDB as described before, it was giving the most watched/scored titles in descending order, the other was selecting a specific genre as Romance, Action, etc., and getting the most scored films for that genre, the last one is a couple based simple recommender which gathers two genres and outputs the most scored movies based on that.

***C.1.1) The Top 250 Films Based On IMDB's Weighted Rating Formula:***

Table

Description automatically generated

This is the IMDB’s similar top 250 movies based on popularity order. The order is different from IMDB’s because all the dataset information to form this recommender is contributed by the main metadata movies dataset, not from IMDB. However, the results are quite similar, with few differences.

### ***C.1.2) The Top Genre Films Based On IMDB's Weighted Rating Formula:***

Graphical user interface, table

Description automatically generated

This is the same as the top 250 films based on IMDB’s formula, the difference is that the users can select their own genre manually and gather the most voted films in descending order. For example, in the left graph shows that the user wanted to see the “Comedy” genre and wanted to see the best films in that. The best film, according to the visualization displayed, is Forrest Gump.

***C.1.3) Couple Genre Films Based On IMDB's Weighted Rating Formula:***

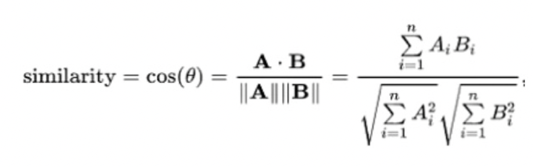
Table

Description automatically generatedThis is also the same as the top 250 films based on IMDB’s formula, the difference is the users can select their own genres and select more than one genre manually and gather the most voted films in descending order. For example, the left graph shows that the user wanted to see the “Comedy” and “Romance” genres and wanted to see the best films on that. The best film, according to the visualization displayed, is still Forrest Gump.

**So, a question for this recommender can be** if we can do the same process to all the data and select the most recommended and liked movies or tv shows, wouldn't it be enough for us to recommend them to everyone? Why do we need other recommenders? The main reason is that every people are different from others. They may not like the same genre, the same type of movies, or even their cultural differences can affect the results and create a biased dataset. To use a more accurate dataset and recommendation engine, the simple recommender stays in the lowest recommender category based on this issue. Even for this metadata example, the voting differences and number of votes changed the direction of the system, and even the more popular movies may not be shown in this system. Also, the data changes from one to another, so it is not possible to clearly say that these are the best movies of all time that everybody will like or love for sure. Contrary, the system shows that there are differences from one to another.

***C.2) Content-based Recommender:***

It suggests similar items based on a particular item. This system uses item metadata, such as genre, director, description, actors, etc., for movies to make these recommendations. The general idea behind these recommender systems is that if a person likes a particular item, he or she will also like an item that is similar to it. And to recommend that it will make use of the user's past item metadata. The main formula is:



The term "cosine similarity" refers to a technique for calculating the difference between two non-zero vectors. In this instance, the movie title and important plot points serve as the movie vector's coordinates. Therefore, if we know the film titles and major plot points of both movies, all we need to do to determine how similar the two movies are is compute the difference between the two movie vectors.

Text

Description automatically generated

For instance, a recommender can examine a film's genre and director to suggest other films with related qualities. To use this method appropriately, cast, crew, keywords, and overview columns are combined and create a column name “soup”. The steps are followed getting the index of the movie that matches the title, getting the pairwise similarity scores of all movies with that movie, sorting the movies based on the similarity scores, getting the scores of the 15 most similar movies, getting the movie indices, get the weighted ratings of the movies, returning the top 15 most similar movies arranged by ratings.

***C.3) Collaborative Filtering Recommender:***

These systems are widely used, and they try to predict the rating or preference that a user would give an item based on past ratings and preferences of other users. Collaborative filters do not require item metadata like their content-based counterparts. This method takes advantage of historical user activity to anticipate products that users may be interested in. It considers the movies a user has already viewed, the numerical ratings provided to those materials, and previously viewed movies by people who are like them.

Table

Description automatically generated

This project aimed to build an item-based collaborative filtering approach. It takes users and their ratings on the movies they watched and generates if a person wants to see a similar movie. The system uses the K-Nearest Neighbor algorithm, which is set to 5 neighbors, by using cosine similarity with using brute as the algorithm to predict what can be the closest to that film that the user can like and recommend them with their distances to the input film. Since it only calculates based on the users’ perspective, it may not recommend any Toy Story film collection even if we want similar movies like Toy Story, as shown in the figure.

***C.4) Hybrid Recommender:***

Table

Description automatically generatedThese systems are widely used, and they try to predict the rating or preference that a user would give an item based on past ratings and preferences of other users. Collaborative filters do not require item metadata like their content-based counterparts. This method essentially combines the two processes mentioned above, with recommendations based on both as well. It calculates their distances while controlling their contents and create a better approach. For this approach, the user-based collaborative filtering system (kNN) is combined with a content-based recommender (cosine similarity). It combines their positive results for the users while discarding their negativities. If a user wants a similar movie like Toy Story, most probably, it will get similar movies inside the collection, but it will also display other users’ positive rated films and recommend them. This system is mostly used on Netflix, Amazon, YouTube, etc., to create a better-performing system for its users.

***C.5) (ANN) Multi-Layer Perceptron Model:***

**This part aimed to predict the IMDB scores based on movie genres. If a user wants to predict what kind of movies have the higher score or a producer wants to create a film and get a high score, they can use this approach to predict the scores based on past movies and rating behaviors. To create a better model, only genres were selected from the Movies dataset, used get\_dummies method to separate the genres, and selected the IMDB score as a target column. Also, for the process, the sequential model is used with two relu activations and linear activation output while using dropout methods to avoid overfitting and underfitting. Model loss on the train set was 1.2753 with mae 0.9284. After that, I defined a function for future use to predict what will be the score if user input is given. Users may select how many genres they want to add and get the IMDB score. The model and score graphs are as follows:**

Text

Description automatically generatedTable

Description automatically generated

***Chart

Description automatically generatedChart, histogram

Description automatically generatedChart

Description automatically generated***

***C.6) CNN Model:***

**This part aimed to predict the IMDB scores based on posters. If a user wants to predict what kind of poster may have the higher score, a producer or a designer wants to create a film and get a high score, they can use this approach to predict the scores based on past posters of movies and rating behaviors. To create a better model, only posters were selected from the Movies dataset and selected the IMDB score as a target column. Also, for the process, the sequential model is used with five relu activations and linear activation output while using flatten method, batch normalization, and dropout method to avoid overfitting and underfitting. Model loss on the train set was 1.4190 with mae 0.9258. After that, I defined a function for future use to predict what will be the score if user input is given. Users may select their own posters in .jpg format and get the related IMDB score. The model and score graphs are as follows:**

Text

Description automatically generatedTable

Description automatically generatedTable

Description automatically generated

Chart, histogram

Description automatically generated**Chart, histogram

Description automatically generated**Chart, histogram

Description automatically generated

***C.7) (MLP + CNN) Multiple Inputs:***

**This part aimed to predict the IMDB scores based on posters and genres. If a user wants to predict what kind of poster may have the higher score, a producer or a designer wants to create a film and get a high score, they can use this approach to predict the scores based on past posters of movies and appropriate genres with rating behaviors. To create a better model, MLP and CNN models are concatenated together and create a base model for multi-input. Model loss on the train set was 0.02 with mae 0.1262. The model and score graphs are as follows:**

**MLP: CNN:**

Table

Description automatically generatedText

Description automatically generatedTable

Description automatically generated

**Multi-Input:**

Table

Description automatically generatedTable

Description automatically generated

*Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generated*

***C.8) Sentiment Analysis:***

**This part aimed to predict the sentiments based on users’ reviews of selected films with the aid of web scraping from IMDB’s website. Most known VADER, Roberta, and Huggingface sentiment analyses are used and selected the Hugginface algorithm, it had better performance among others. The system works by asking the user what movie reviews need to be seen. When the user inputs the name it downloads the data from the website and creates a .csv document with their rating, title, and sentiment. Huggingface analyze the reviews and creates the score prediction that review and adds that score with the predicted sentiment side by side. Users can display the different sentiments available here, visualization of rating comparison with sentiment comparison, or display the true positive, true negative, false positive, and false negative reviews it guessed. This part created a function and was also added inside the chatbot design.**

***C.9) Foxie Chatbot:***

**This part includes all the steps explained above and more with our most known chatbot procedure, which gives information about different subjects while chatting with the user. The reason for using a chatbot in this project was to create a more user-friendly base environment for non-developers to use this project for their personal usage and show how the future is near than we can guess by combining all the methods.**

***D) Baseline:***

The baseline of this project comes from the recommender engines created in content-based and collaborative filtering recommenders. The dataset plays a key role in these recommenders since collaborative filtering includes the user’s input. If there wouldn’t any user dataset available, only half of the code would be applicable in this project. This project aimed to build only using hybrid recommenders with their compliment recommender systems, but after the literature review, an evolution of recommender systems has been seen. People were trying to use sentiments and deep learning approaches. Furthermore, this concluded to change of the project scope to evolve to create part of historical recommender systems to more futuristic approaches.

1. **Implementatıon Detaıls**

This project is created by using Python language on Jupyter Notebook with its’ main packages for Data Science, included as Pandas, NumPy, and Matplotlib. Python is open-source and gives development freedom. Apart from these, there are also different packages needed to be installed to perform a recommendation engine, sentiment analysis, and chatbot. Mainly, nltk and transformers for sentiment and chatbot, selenium for web scraping, sklearn for the content-based recommender, Keras for deep learning approaches, and others like plotly, bar\_chart\_race ones for EDA visualization purposes.

1. **Evaluatıon Detaıls**

The dataset used for recommender systems doesn’t need any train test split or any other methodology to perform the tasks. However, the deep learning part includes both MLP and CNN, so to create a well-performed prediction, MLP, CNN, and Multi-Input approach datasets splitted into three parts as 64% train, 16% validation and 20% test sets, while setting the random state as 1234 to create same performance if the code snippet is repeated. There wasn’t any skewness in the dataset, and not need to normalize the dataset. The performances occurred as expected with low accuracy on all the systems, as explained before. Even though the code itself was really close to overfit, the dropout indices and early stoppings aided in performing better analysis results. The code achieved its’ purpose and created a well-designed process for future use. In the MLP part, genre inputs can give nearly the same IMDB score result, in the CNN part, poster images’ IMDB scores are close to the original score, and even when they are combined together, it gives more accurate results than both of them.

1. **Conclusıon and Recommendatıon**

This project aimed to perform all the recommenders and sentiment analysis inside a chatbot for a more user-friendly approach and usage of non-developer background users. The project presented top ordered 250 movies with and without classifying their genres, content-based and collaborative-based approaches while combining them together with a hybrid approach, sentiment analysis on the selected movie, and finally created a chatbot for a more humanoid approach. The chatbot can gather information about my past EDA visualizations, create sentiment analysis based on scraped data by using new sentiment, and after that, a sentiment the reviews as positive or negative by displaying false positives, false negatives, true positives, and true negatives with their visualizations, recommend movies based on three approaches as simple, content-based and collaborative filtering. I didn't include the multi-input deep learning recommendation and the bar chart race visualization on the EDA part since they wouldn't make any good fit in this chatbot, and it wouldn't be effective to create it. This system provides a more historical pathway from the very beginning until the foreseeable future. However, the datasets gathered may have biased information or be malformed without my knowledge by only one person creating all user ids without gathering information from others. Also, a sentiment analysis-based recommender system is possible, while combining all movie sentiments with respect to the obligation of the original websites can create a better recommender system for the users. However, in this project, it wasn’t possible to create this kind of analysis because of the obligations of websites and only limited to use for one movie to make sentiment analysis; otherwise, due to violation of terms, my account would be banned for disobeying the rules of IMDB. Furthermore, this deep learning part explained how users can create and manipulate the genres and poster images to get better IMDB scores, this would conclude a better performance if combined with all the predictions with respect to the recommenders, and this may also result in giving better user involvement.

**References List**

1) P. Prajapati, A. Thakkar, and A. Ganatra, “A survey and current research challenges in multi-label classification methods,” International Journal of Soft Computing and Engineering (IJSCE), vol. 2, no. 1, pp. 248–252, 2012

2) Han J., Kamber M., “Data Mining: Concepts and Techniques”, Morgan Kaufmann (Elsevier), 2006.

Techniques”, Morgan Kaufmann (Elsevier), 2006.

3) D.K. Yadav, Ankur Singh, Vijay Kr. Gupta, Manoj Kumar, “A Movie Recommender System: MOVREC” International Journal of Computer Applications (0975 – 8887), Volume 124, No.3, 2015

4) Giles C.L., Bollacker K.D., and Lawrence S., “CiteSeer: An automatic citation indexing system,” in Proceedings of the third ACM conference on Digital libraries, 1998, pp. 89–98.

5) Herlocker, e. a. (2004). Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems, 5-53.

6) Konstan, F. M. (2015). The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS).

7) Haykin, Simon S. (1999). Neural Networks: A Comprehensive Foundation. Prentice Hall. ISBN 0132733501

8) P. Bedi., Sharma R. and Kaur H., Recommender System based on collaborative behavior of ants, 2009, Journal of artificial intelligence, ISSN 1994-5450, PP. 40-55

9) P. Baid, A. Gupta, and N. Chaplot, "Sentiment Analysis of Movie Reviews using Machine Learning Techniques," International Journal of Computer Applications, vol. 179, no. 7, pp. 45–49, Dec. 2017, https://doi.org/10.5120/ijca2017916005.

10) Lin, L., D’Haro, L.F., Banchs, R.: "A Web-based platform for the collection of human-chatbot interactions." In: Proceedings of the Fourth International Conference on Human-Agent Interaction, pp. 363–366 (2016)