Building a Movie Recommendation System with Simple, Content-Based and Collaborative Filtering Techniques

Satyanarayana Vinay Achanta
Computer Science
Utah State University
Logan, Utah
a02395874@usu.edu

V S S Dheeraj Kotte
Computer Science
Utah State University
Logan, Utah
a02394047@usu.edu

Hari Chandana Kotnani
Computer Science
Utah State University
Logan, Utah
a02396013@usu.edu

***Abstract*—With the increasing popularity of digital streaming services, movie recommendation systems have become crucial for helping users discover new movies that match their interests. In this project, we propose a comprehensive movie recommendation system that utilizes different approaches, including simple genre-based recommendations, content-based recommendations, and collaborative recommendations. A website was developed that demonstrates the three recommendation approaches and provides users with the option to choose their preferred approach and technique for receiving movie recommendations. Our findings showed that the content-based and collaborative recommendation systems provided more personalized and accurate movie recommendations compared to the simple genre-based recommendation system. Furthermore, the collaborative recommendation system performed better for users who had rated more movies in the past, indicating that more user data leads to better recommendations. Our movie recommendation system can help users discover new movies that align with their preferences and enhance their digital streaming experience. This project provides insights into the strengths and weaknesses of different movie recommendation approaches and highlights the importance of personalized recommendations based on user preferences.**

I. Introduction

Movie recommendation systems have become increasingly popular in recent years due to the surge in streaming services and abundantly available content. These systems aim to provide personalized recommendations to users based on their viewing history, preferences, ratings provided, and behavior. The primary motivation behind this project was to build an effective movie recommendation system with a user interface that can suggest relevant and interesting movies to users. In this project, we propose a comprehensive movie recommendation system that utilizes different approaches, including simple genre-based recommendations, content-based recommendations, and collaborative recommendations.

We developed a Simple Recommender system that provides generalized recommendations to all users based on the popularity, gross income, and rating of movies. While this system provides a baseline for recommendation systems, it does not take into account a user's individual preferences.

To address the limitations of the Simple Recommender, we implemented more advanced recommendation systems, such as content-based and collaborative filtering approaches, using Python and several popular libraries, including sci-kit-learn and Pandas. Challenges in building an effective recommendation system included collecting and cleaning data, choosing appropriate features for analysis, as it took a long time to run due to a large amount of movie data, and developing a user-friendly interface.

We developed a website that allows users to input their movie preferences and receive personalized recommendations. The website was built using ReactJS for the front end and Django for the back end.

Our study revealed that advanced recommendation systems, such as content-based and collaborative filtering approaches, outperformed the Simple Recommender in providing accurate and personalized movie recommendations. Particularly, the collaborative recommendation system showed excellent performance for users with more rated movies, suggesting that more user data can lead to better recommendations. These findings emphasize the need to use more sophisticated recommendation techniques and collect sufficient user data to personalize movie recommendations.

Our project showcases the potential of movie recommendation systems and demonstrates their practical implementation, with a user-friendly interface. It also emphasizes the importance of personalized recommendations based on user preferences and the possibility of generating accurate and relevant movie recommendations.

II. Related Work

Several research works have explored different approaches to recommendation systems. Collaborative filtering has been one of the most popular methods used in recommendation systems. One memory-based approach [1] utilizes user-item ratings to find similarities between users and recommend items based on those similarities. The models also address challenges of collaborative filtering such as sparsity and scalability. Another model-based approach [2] involves matrix factorization to reduce the dimensionality of the user-item rating matrix and find latent factors [3] representing user preferences and item

characteristics. Researchers have proposed extensions to these basic models, including implicit feedback and multiple rating scales.

On the other hand, content-based filtering methods have also been investigated. Several studies [4] have proposed various approaches to content representation and similarity calculation and suggested solutions to common issues such as cold start and diversity. Some studies [5] use Flickr photo tags to learn users' interests and preferences. The proposed approach [6] extracts semantic features from the photo tags using Latent Dirichlet Allocation (LDA) and uses the features to build user profiles. Researchers have also proposed hybrid approaches that combine both content-based and collaborative filtering methods to enhance the performance of recommendation systems.

In information retrieval, [7] proposed a relevance feedback approach that uses tf-idf weights to represent documents and queries. The authors demonstrate that the approach can improve retrieval performance by updating the query based on the feedback provided by the user using a Rocchio algorithm. In text summarization, [8] proposed a graph-based ranking algorithm that uses tf-idf weights to represent vertices in the graph. The authors show that their approach outperforms other summarization techniques, such as centroid-based and sentence extraction-based methods. These studies highlight the importance and effectiveness of tf-idf in various natural language processing tasks.

Some researchers studied [9] CountVectorizer for spam filtering, while [10] uses it for keyword extraction. In both papers, CountVectorizer is used to convert text into a bag-of-words representation, where each word is considered a separate feature, and the occurrence of each word is counted. The resulting feature vectors can be used for various applications, such as classification and information retrieval. A comprehensive survey of the field [11]. The authors then discuss recent research on multi-objective recommendation algorithms, including Pareto-based methods, weighted methods, and matrix factorization-based methods and some researchers introduced TensorFlow Recommenders (TFRS) as a potential framework for implementing multi-objective recommendation systems.

III. Methods

A. Dataset Collection and Exploration

Our data collection process involved several techniques to gather diverse movie-related data for our recommendation systems. First, we utilized web scraping to extract data from the IMDb website using beautiful soup for movies released between 2014 to 2021. We collected information such as release year, rating score, Metascore, total votes, gross USA, movie genre, and director/star of the movies for each year separately. We combined all this data into one dataset, which we used for our simple recommendation system.

However, for our content-based recommendation system to perform better, we needed additional information for the scarped data such as movie descriptions and keywords. To obtain this data, we utilized The Movie Database (TMDb) API, which provided us with a vast amount of movie-related information, including descriptions and keywords. Additionally, we used the TMDb API to get the image links for the movies that we scraped from IMDb, which we used to display movie posters on our website.

Finally, to develop our collaborative filtering recommendation system, we used the movies dataset [13], which contains movie ratings provided by users. This dataset provided us with user-given ratings that we needed for our collaborative filtering algorithm. Overall, our data collection process involved a combination of web scraping, API usage, and publicly available datasets, which allowed us to gather a diverse range of data for building various recommendation systems for our project.

B. Pre-Processing

We performed preprocessing on the data collected from various sources to clean and format it for use in our movie recommendation system.

For the data scraped from the web, we merged all the movie datasets from the years 2014 to 2022 into one merged dataset. We removed any rows that contained Chinese characters as they were not relevant to our English-based system. We also dropped any rows with missing values and converted the datatypes. We then concatenated the Director and Stars columns into a single column called Cast.

For the data extracted from the API, we extracted the names of the keywords and descriptions associated with each movie. Instead of dropping rows with missing values, we kept "Not Available" for the movies that don't have keywords and descriptions in order to minimize data loss.

For the Movie Lens dataset, we combined the rating data with the movie names data, calculated the mean rating and the total number of ratings for each movie, and stored them in a separate data frame. We did this to get high mean ratings and high numbers of ratings which are typically considered more popular and well-regarded by the audience. By calculating these metrics, we can use them as features in our recommendation system to make more informed recommendations to users.

C. Methods

Simple recommender system

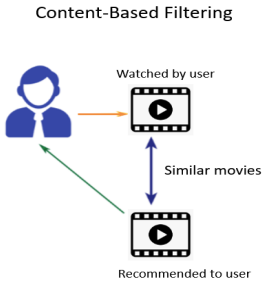
The Simple Recommender is a basic recommendation system that provides generalized recommendations to all users based on the gross, popularity, and rating of movies. It takes into account the average rating of a movie, the number of votes it has received, and the gross income it generated at the box office.

To implement the Simple Recommender, We compute the weighted rating for each movie, which calculates the weighted average rating of a movie by multiplying its rating score by the number of votes it has received and dividing it by the total number of votes for all movies.

Formula

$$W = \frac{\sum_{i=1}^{n} w_i X_i}{\sum_{i=1}^{n} w_i}$$

W = weighted average
n = number of terms to be averaged
w_i = weights applied to x values
X_i = data values to be averaged



We sort the movies by gross income and weighted rating to provide a list of the top recommended movies, then filter the movies by the specified genre, and then sort them based on their gross income and weighted rating to provide a list of the top recommended movies for that genre.

When a user opens our website, we display genres and provide a list of the top movies in each genre using our Simple Recommender system. This feature is prominently displayed on our home page, making it easy for users to find popular movies in their preferred genre.

While the Simple Recommender is easy to implement and provides a baseline for recommendation systems, it does not take into account a user's individual preferences or provide personalized recommendations. To address these limitations, we implemented more advanced recommendation systems, such as collaborative filtering and content-based filtering, which provide more personalized recommendations based on a user's past viewing history or preferences. These systems analyze user behavior and provide more relevant and personalized recommendations.

Content based filtering

Our content-based filtering approach in recommender systems suggests movies to users based on their features' similarities. In this approach, the system examines the content or features of movies selected by a user to recommend other similar movies. The idea is that if a user has shown interest in certain movies, they are likely to be interested in other movies that have similar features. Content-based filtering can be used to provide movie recommendations to users based on their individual preferences.

For our implementation, we used our scraped dataset of movies released from 2014 to 2022 from the IMDB website and extracted features such as movie title, genre, cast, and rating score to create movie profiles for content-based filtering. We also utilized keywords and descriptions of the movie extracted from TMDB API to obtain better recommendations.

In our content-based filtering approach, we used four different techniques to recommend similar movies to users based on their individual preferences. Each approach involves different methods of feature extraction, similarity calculation, and recommendation. By trying multiple approaches, we can compare determine which approach works best for our dataset.

Approach 1: TF-IDF-Based Content-Based Filtering Approach

In this approach, we created a movie recommendation system that uses TF-IDF (Term Frequency-Inverse Document Frequency) for feature extraction. We first cleaned and formatted the relevant movie information (such as cast, genre, and overview) from the dataset. Then, we created a TF-IDF matrix for all the movies based on their features. This matrix measures how often a word appears in a movie description compared to how often it appears in all the descriptions.

Using cosine similarity, we computed the similarity scores between movies based on their features. We recommended similar movies to users who had previously expressed interest in a movie by defining a function that takes a movie title as input and returns a list of recommended movies with their corresponding rating scores.

Approach 2: Count Vectorizer-Based Content-Based Filtering Approach

In the second approach, we used the CountVectorizer method from the sci-kit-learn library for feature extraction. This method counts the number of times a word appears in each movie description, creating a matrix where each row represents a movie, and each column represents a word.

We followed a similar process as the first approach, using cosine similarity to compute similarity scores between movies and recommend similar movies to users based on their individual preferences.

Approach 3: TF-IDF-Based Content-Based Filtering Approach with K-Nearest Neighbors Algorithm

In the third approach, we used the TfidfVectorizer method and the NearestNeighbors algorithm for feature extraction and similarity calculation. We created a TF-IDF matrix for all the movies based on their features, just like in the first approach.

Using the NearestNeighbors algorithm, we computed the similarity scores between movies based on their features. We recommended similar movies to users based on their individual preferences by defining a function that takes a movie title as input and returns a list of recommended movies with their corresponding rating scores.

Approach 4: Count Vectorizer-Based Content-Based Filtering Approach with K-Nearest Neighbors Algorithm

In the fourth and final approach, we used a combination of CountVectorizer for feature extraction and the KNN (k-nearest neighbors) algorithm for finding similar movies. We created

a CountVectorizer matrix for all the movies based on their features, just like in the second approach.

Using the KNN algorithm, we computed the similarity scores between movies based on their features. We recommended similar movies to users based on their individual preferences by defining a function that takes a movie title as input and returns a list of recommended movies with their corresponding rating scores.

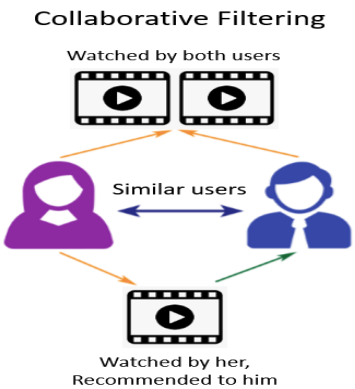
Content-based filtering approaches are known for their effectiveness in providing recommendations based on movie features. However, they may not always be able to capture the diverse and evolving preferences of users. To address this limitation, we implemented a more personalized recommendation system called collaborative filtering. Collaborative filtering takes into account the behavior and preferences of similar users to recommend movies to a user. By analyzing the opinions and preferences of a larger group of people, collaborative filtering can provide recommendations that are more likely to align with the individual user's preferences.

Collaborative filtering

Collaborative filtering is a powerful approach for generating personalized movie recommendations by finding similar users and recommending movies based on their preferences and behavior. Our movie recommendation system utilizes this approach along with the k-nearest neighbors algorithm to recommend movies that are likely to be of interest to a particular user.

The collaborative filtering approach in our system collects data on the past behaviors and preferences of a group of users and utilizes this information to make recommendations for movies. By finding similar users and analyzing their past preferences, the system can recommend movies that are likely to be of interest to a particular user based on their similarities to other users.

Our approach of utilizing collaborative filtering and the k-nearest neighbors algorithm proved to be a powerful way to provide personalized movie recommendations. By analyzing the preferences and behavior of similar users, we can help users discover new and interesting movies that they might not have found otherwise.



Approach 1: Personalized Movie Recommendations using Collaborative Filtering and Item-Based Approach

This approach uses a collaborative filtering technique based on item similarity to suggest movies to users. To provide personalized movie recommendations, an item-based filtering technique analyzes the similarities between movies based on their ratings and recommends highly rated movies that are similar to the ones the user has rated. We found that KNN works best for item-based filtering because it can easily handle sparse data and can find similar items efficiently.

To prepare the data, we merged the movie names and ratings data and grouped the data by movie title to generate a new data frame that contained the mean rating and the total number of ratings for each movie. We pivoted the rating data frame so that each row represented a user and their ratings for all movies in the dataset. This made it easier to calculate similarities between movies.

We used the NearestNeighbors algorithm with the cosine metric parameter to find similar movies based on their ratings efficiently.

To generate movie recommendations on the website, the user inputs ratings for some movies, and the model takes the highest. We calculated the ratings of recommended movies using a weighted average based on the ratings of similar movies. Our model analyzes the similarities between movies based on their ratings and recommends highly rated movies that are similar to the ones the user has rated highest.

Approach 2: Building a User-Based Collaborative Filtering Movie Recommendation System using TensorFlow Recommenders (TFRS)

We have implemented a movie recommendation system using TensorFlow Recommenders (TFRS), a high-level framework specifically designed for building efficient recommender systems. Our model is capable of predicting user preferences based on the behavior of similar users by recommending movies to users based on their previous ratings and the ratings of other users with similar movie preferences.

To achieve this, we preprocessed the datasets and split them into training and testing datasets. Next, we built a TensorFlow model using TFRS, which consisted of two components: a movie model and a user model. Both models were implemented using an embedding layer, which was used to map high-dimensional categorical data into lower-dimensional continuous vectors. The embedding dimension was set to 64 in this implementation.

To predict movie ratings, we implemented a rating model using a multi-layer perceptron with three dense layers. The output of the rating model was a single float value that represented the predicted rating of the user for the given movie. To train the model, we used the Mean Squared Error (MSE) loss function and the Adagrad optimizer.

Finally, the trained model was used to make recommendations for a given user by computing the predicted scores for all movies and returning the top k movies with the highest scores. Overall, our recommendation system is a powerful tool for

helping users discover new movies that match their interests and preferences. With the help of TFRS, we were able to build an efficient recommendation system that can provide accurate and personalized recommendations to users.

D. Experimental settings

Website development

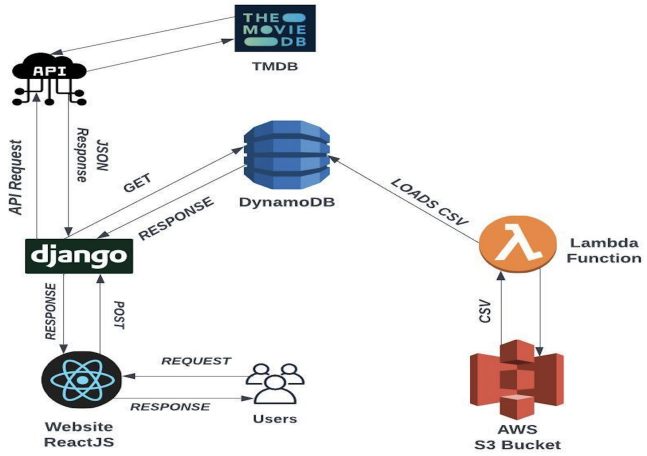
Our website is built using ReactJS for the front end and Django for the back end. The website is designed to provide movie recommendations to users who request movies.

ReactJS is a popular JavaScript library for building user interfaces. We chose to use it because of its ease of use and flexibility in creating dynamic web pages.

The back end of the website is built using Django, a high-level Python web framework. We store the implemented Python code in the Django backend, which enables us to efficiently handle user requests for movie recommendations.

To enhance the user experience, we fetch links to movie posters from The Movie Database (TMDb) and store them in a database hosted on Amazon Web Services (AWS) DynamoDB. When Django has the business logic which fetches data from the DynamoDB database, it receives a response from the database.

In summary, Our website is built using ReactJS and Django, and leverages AWS services such as DynamoDB, S3, and Lambda to provide a seamless user experience. By utilizing these technologies, we are able to efficiently store and retrieve data, handle user requests, and provide visually appealing recommendations.



Architecture

On launching the webpage, users can view a set of movies related to each genre, such as comedy, action, adventure, etc. The movies are recommended following a simple recommendation approach by genre.

Users can view the recommendations based on their preferences. Upon entering input (movie name), users can select between two search options: content-based and collaborative. If the user doesn't select any search option, it defaults to content-based.

On selecting content-based, four options are available: cosine and TF-IDF, cosine and count vec, KNN and TF-IDF, and KNN and count vec. Each approach recommends movies based on the user's input, and for each recommended movie, the movie name, description, and release date can be viewed by the user.

On selecting a collaborative approach, the user's movie preferences are taken as input, which is the rating of certain movies to predict their choices. Six movies are shown on the page to take the user's input. If the user enters a rating for each movie, the system selects the highest-rated movie and recommends movies based on that rating. This involves taking into consideration other movie ratings to predict new ones.

IV. EXPERIMENTAL RESULTS

In this section, we will present the results for each approach and technique that we have implemented and assess their relevance.

Simple recommender system

Based on the simple recommender algorithm, we generated a list of recommended movies for genres. For each genre, we filtered the movies based on their genre and sorted them by their gross income and weighted rating. Here are recommendations for some genres:

Genre		
Action	Mystery	Romance
Star Wars: Episode VII - The Force Awakens	Get Out	Fifty Shades of Grey
Avengers: Endgame	Us	The Fault in Our Stars
Spider-Man: No Way Home	Gone Girl	Trainwreck
Jurassic World	Divergent	Little Women
Star Wars: Episode VIII - The Last Jedi	Murder on the Orient Express	Passengers

The results of this recommendation system appear to be relevant and aligned with the expectations of the respective genres. For the Action genre, the recommended movies are all high-energy and thrilling, while the Mystery genre recommendations are all notable for their unexpected twists and turns. For the Romance genre, the recommended movies all have strong emotional elements and are likely to resonate with fans of the genre. These results suggest that the methodology used for the recommendation system was effective in selecting popular and critically acclaimed movies within each genre, making them great choices for viewers looking for a specific type of movie.

Content-based Approach

To get recommendations for the movie 'Avengers: Age of Ultron', we applied our four content-based approaches: two using cosine similarity (TF-IDF and Count Vectorizer), and two using KNN (TF-IDF and Count Vectorizer).

Cosine similarity (TF-IDF and Count Vectorizer)

Movie Name
Avengers: Endgame
Guardians of the Galaxy
Captain America: The Winter Soldier
Spider-Man: No Way Home
Captain America: Civil War

K- Nearest Neighbors(TF-IDF and Count Vectorizer)

Movie Name
Avengers: Endgame
Guardians of the Galaxy
Captain America: The Winter Soldier
Captain America: Civil War
Thor: Ragnarok

Across all four approaches, 'Avengers: Endgame', 'Guardians of the Galaxy', 'Captain America: The Winter Soldier', and 'Captain America: Civil War' were recommended, indicating that these movies share similar content also they are similar to 'Avengers: Age of Ultron' in terms of genre, characters, and plot.

Moreover, the content-based approach using KNN also recommended 'Spider-Man: No Way Home' and 'Thor: Ragnarok', suggesting that these movies have some similarities with 'Avengers: Age of Ultron' based on their content.

To compare the results of the four content-based approaches, we can observe that they all recommend the same movies. This indicates that these movies share similar content and are likely to be enjoyed by viewers who appreciate the superhero genre.

However, there are some differences in the recommended movies across the four approaches. The cosine similarity approach using TF-IDF and Count Vectorizer recommended 'Spider-Man: No Way Home' as well, while the KNN approach using TF-IDF and Count Vectorizer recommended 'Thor: Ragnarok' instead. This suggests that the KNN approach may be better suited to capture more nuanced similarities between movies based on their content, as it takes into account the proximity of each movie to its neighbors in the feature space.

Collaborative filtering

Item-based

The item-based recommender system is providing recommendations based on collaborative filtering and the K-Nearest Neighbors algorithm. The system calculated the similarity between movies using the cosine distance metric and recommended movies that were most similar to the input movie, 'Toy Story'.

Movie Name
The Ref
Naked
I'll Do anything
Desert Winds
The Celluloid Closet

These recommended movies were identified by the algorithm as those that users with similar preferences to those who liked 'Toy Story' also enjoyed.

The personalized movie recommendations provided by the system are particularly relevant as they are based on the users' previous movie preferences. The collaborative filtering and KNN algorithms allowed the system to suggest movies that the user may not have previously considered but are likely to enjoy based on similar users' histories. Therefore, the system was able to provide a more personalized and accurate recommendation to the users, which enhances their overall movie-watching experience.

User-based

Tensor Flow Recommender Produced Result for one user

Movie Name	Genre
Un long dimanche de fiançailles	Drama
The Greatest Story Ever Told	Drama, History
Un éléphant ça trompe énormément	Comedy
Pitch Black	Thriller, Science Fiction, Action
A Man, a Woman and a Bank	Action, Comedy, Romance

Previously User Rated Films

Movie Name	Genre	Ratings Provided by User
The Wanderers	Drama	4.0
Kurz und schmerzlos	Drama, Thriller	5.0
The Bourne Supremacy	Action, Drama, Thriller	5.0
License to Wed	Comedy	4.0
Anatomie de l'enfer	Drama	4.0

Our movie recommendation system using TensorFlow Recommenders (TFRS) has proven to be an effective tool for providing personalized movie recommendations to users. Based on the user's previously rated movies, our model was able to suggest new movies that match their interests and preferences. The model's performance was evaluated based on the user ratings, and the results indicated that the model was effective in predicting user preferences. The user had previously rated drama films more, and the model reflected on this by suggesting new drama films to watch. Our recommendation system has demonstrated its ability to provide efficient and personalized recommendations, which can enhance the user's movie-watching experience.

V. CONCLUSION

In this paper, we proposed a comprehensive movie recommendation system that utilizes different approaches, including simple genre-based recommendations, content-based recommendations, and collaborative recommendations. We developed a website that demonstrates the three recommendation approaches and provides users with the option to choose their preferred approach and technique for receiving movie recommendations. The experiments conducted showed that the content-based and collaborative recommendation systems provided more personalized and accurate movie recommendations compared to the simple genre-based recommendation system. Furthermore, the collaborative recommendation system performed better for users who had rated more movies in the past, indicating that more user data leads to better recommendations.

We found that the collaborative filtering approach with tensor factorization worked the best, particularly because it takes user input into account. We also learned a lot about developing a website from scratch, understanding how recommendation systems work, and integrating code into user interfaces. Challenges in building an effective recommendation system included collecting and cleaning data, choosing appropriate features for analysis, and developing a user-friendly interface.

Our paper provides valuable insights into the strengths and weaknesses of different movie recommendation approaches and highlights the importance of personalized recommendations based on user data. We plan to expand our dataset and continue experimenting with different techniques for improving the recommendation system.

Based on the experiments conducted, the collaborative filtering approach with tensor factorization worked the best. The content-based recommendation approach was also effective in selecting popular and critically acclaimed movies within each genre, making them great choices for viewers looking for a specific type of movie. The simple genre-based recommendation system may be useful for users who are not as interested in personalized recommendations and simply want to discover popular movies within a particular genre.

In conclusion, we proposed a comprehensive movie recommendation system that utilizes different approaches and techniques and found that personalized recommendations based on user data are the most effective. We provided insights into the strengths and weaknesses of different movie recommendation approaches, and future work involves expanding the dataset and integrating more user input into the recommendation system to further personalize the recommendations for each user.

REFERENCES

[1] Collaborative Filtering for Implicit Feedback Datasets, Yifan Hu, Yehuda Koren, and Chris Volinsky (2008) - https://doi.org/10.1109/ICDM.2008.22

[2] Matrix Factorization Techniques for Recommender Systems, Koren, Y., Bell, R., Volinsky, C. (2009) - https://doi.org/10.1561/1100000009

[3] Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model, Yehuda Koren (2008) - https://dl.acm.org/doi/10.1145/1401890.1401944

[4] Content-based recommendation systems" by Dietmar Jannach et al. (https://link.springer.com/article/10.1007/s10115-016-0987-4)

[5] Learning Semantic User Profiles from Flickr Data for Personalized Content Recommendation" by Daniele Quercia et al.

[6] Tag-Aware Recommender Systems by Fusion of Collaborative Filtering Algorithms" by Tiancheng Li et al.

[7] A feedback approach to information retrieval using tf-idf weights" by Karen Sparck Jones et al.

[8] "TextRank: Bringing Order into Texts" by Rada Mihalcea and Paul Tarau

[9] "Efficient and Effective Spam Filtering and Re-ranking for Large Web Datasets" by Mehrdad Jalali and Gholamreza Nakhaeizade

[10] "Automated Keyword Extraction from Articles using CountVectorizer and Latent Semantic Analysis" by Sanket Gupta et al.

[11] "Multi-objective Recommender Systems: A Survey of the State of the Art" by Robin Burke et al.

[12] "A Novel Movie Recommendation System Based on Collaborative Filtering and Neural Networks.

[13] "Dataset:https://www.kaggle.com/code/rounakbanik/movie-recommender-systems"