Movie Recommendation System

Mayank Raj
Engineering Physics Department
Indian Institute of Technology, Bombay
Mumbai, India
210260031@iitb.ac.in

ABSRACT

In today's era of abundant streaming platforms and endless content choices, users often find themselves overwhelmed when trying to decide what to watch next. This abundance of options can lead to decision fatigue and indecision. However, recommender systems have emerged as invaluable tools to alleviate this burden by providing personalized suggestions tailored to individual preferences and viewing habits. By leveraging advanced algorithms and machine learning techniques, these systems analyse user data, including past viewing history, ratings, and implicit feedback, to generate recommendations that align with the user's tastes and interests. This abstract explores the role of recommender systems in simplifying the content discovery process for users, highlighting their ability to enhance user experience, increase engagement, and drive content consumption on streaming platforms. Through an examination of the underlying technologies and methodologies employed by these systems, as well as their impact on user satisfaction and platform performance, this abstract shed light on the importance of personalized recommendation systems in the modern media landscape.

INTRODUCTION

What is a Recommendation System? An algorithm or program that makes recommendations to users based on their interests and previous interactions is called a recommender system. To improve user experience and engagement, these methods are widely employed across a variety of platforms, including Hulu, Netflix, Amazon Prime, and others.

Why Recommendation Systems? Recommendation Systems are essential for several reasons:

1. Recommendation systems ensure that consumers find products and information that interest them by providing personalized suggestions based on user preferences.
2. Users are more inclined to interact with the platform when personalized recommendations are made, which boosts user retention and happiness.
3. Recommendation algorithms are used by e-commerce platforms to promote products, which increases revenue and sales as users find and buy things they might not have otherwise considered.
4. Recommendation systems broaden users' perspectives and introduce them to things they might not have known about by exposing them to fresh and varied material.
5. Recommendation systems leverage prior behaviour and preferences to assist consumers make well-informed judgments for subjective and complex options, like music, movies, or books.

Types of Recommender Systems:

1. Content-Based Recommender Systems: These programs suggest products based on what a user has enjoyed or engaged with previously. This could apply to performers, directors, plot keywords, or genres in the case of movies.
2. Collaborative Filtering: Collaborative filtering makes product recommendations according on user preferences and behaviour. It is further separated into:

User-Based Collaborative Filtering: Makes suggestions for products based on the tastes of other similar users.
Item-Based Collaborative Filtering: Makes suggestions for items by comparing comparable items.
3. Hybrid Recommender Systems: These systems combine multiple techniques, such as content-based and collaborative filtering, to provide more accurate and diverse recommendations.
Since this project focuses on building a content-based recommender system, we will delve deeper into leveraging movie attributes such as genres, actors, directors, and plot keywords to provide personalized movie recommendations.

DATA DESCRIPTION

Movie dataset has

The dataset used in this project is provided on Kaggle. Ther are 2 csv files which are movies.csv and credits.csv. We merge both csv file to work and use features from both files at one.
It consists of the data for 4803 individuals (rows) and 24 features. Few must know columns are given below:

• Id: a unique identifier for each individual and used to fetch posters from TMBD movie website
• Overview: a short description of movie
• Genres: Type of movie
• Cast: Actors, actresses and director of movie

DATA PRE-PROCESSING

In our data preprocessing phase, we focus solely on extracting essential features to build our movie recommendation system. These features include the movie ID, title, genre, overview, cast, crew, and keywords. By honing in on these key attributes, we streamline the dataset to its most relevant elements, ensuring our recommendation engine operates efficiently and effectively.
Following the extraction of essential features, our next step in data preprocessing involves cleaning the dataset by removing any null values. Null values, or missing data points, can introduce inaccuracies and inconsistencies into our recommendation system, potentially compromising the quality of our suggestions.
With our dataset cleaned of null values, our attention turns to optimizing the representation of certain features, specifically genre and keywords. These attributes are stored as strings containing lists of values, which we need to convert into actual lists to facilitate effective analysis and modelling. By converting genre and keyword strings into lists, we unlock valuable insights into the thematic elements and content descriptors associated with each movie.
Recognizing the abundance of cast members associated with each movie, we implement a strategy to streamline this information while preserving its relevance and significance in our recommendation system. To achieve this, we focus on identifying the top three cast members for each movie from the cast column.
By selecting the top three cast members, we distil the cast information to its most influential and recognizable figures, ensuring that our recommendation engine prioritizes the actors and actresses who have the greatest impact on audience preferences. This approach not only reduces the complexity of the cast data but also enhances the effectiveness of our recommendations by highlighting the key performers associated with each movie.

To further refine our dataset and capture crucial information about the creative leadership behind each movie, we extract the director's name from the crew column.
To consolidate the diverse array of movie attributes into a cohesive and informative feature, we merge the overview, keywords, cast, crew, and genres into a single column named "tag." By combining these elements into a unified tag column, we create a rich and comprehensive representation of each movie's thematic content, creative contributors, and genre classification. This consolidated approach enables our recommendation system to leverage a broader range of information when generating movie suggestions, resulting in more nuanced and contextually relevant recommendations for users.

MODEL BUILDING

Building Model with Count Vectorization:

To construct our recommendation model, we employ count vectorization, a popular technique for converting text data into numerical representations. We utilize a count vectorizer with the following specifications: limiting the features to the top 5000, and excluding common stop words.

1.Count Vectorization: Count vectorization converts textual data into numerical form by creating a
matrix where each row represents a document (in this case, a movie), and each column represents a unique term from the corpus. The cell values indicate the frequency of each term in the respective document. By employing count vectorization, we transform the textual attributes (overview, keywords, cast, crew, and genres) into numerical features that can be utilized by machine learning algorithms.

2.Top 5000 Features: To manage computational complexity and focus on the most informative features, we limit the vocabulary size to the top 5000 terms. This ensures that the model prioritizes the most relevant and discriminative words, phrases, and entities extracted from the movie attributes.

3.Removing Stop Words: Stop words, commonly occurring words such as "the," "is," and "and," are often excluded from text analysis as they provide little discriminatory value. By removing stop words from the corpus, we enhance the model's ability to focus on meaningful content and improve its predictive performance.

By implementing count vectorization with these specifications, we create a numerical representation of the movie attributes that captures the essence of each film while mitigating noise and irrelevant information. This vectorized representation serves as the input for our recommendation model, enabling us to train machine learning algorithms to predict movie preferences and generate personalized recommendations for users.

In our recommendation model, we opt to measure similarity between movies using cosine distance instead of Euclidean distance. Cosine similarity is a widely-used metric for comparing the similarity between two vectors in a multi-dimensional space, particularly in text analysis and recommendation systems.

1. Cosine Similarity:

Cosine similarity measures the cosine of the angle between two vectors, representing the similarity of their directions in the feature space. In the context of our model, each movie is represented as a vector in the high-dimensional feature space derived from count vectorization. By calculating the cosine similarity between pairs of movie vectors, we can quantify how closely aligned their feature representations are, indicating their similarity in terms of content, theme, and other attributes.

2. Advantages over Euclidean Distance:

Unlike Euclidean distance, which measures the geometric distance between vectors in the feature space, cosine similarity is unaffected by the magnitude of the vectors. This makes it particularly suitable for text-based data, where the length of documents can vary significantly. Additionally, cosine similarity is more robust to the presence of irrelevant or noisy features, as it focuses on the relative orientation of vectors rather than their absolute values.

Creating Functions for Movie Recommendations:

1. Fetching Top 5 Similar Movies:

We develop a function that takes a movie as input and returns the top 5 most similar movies based on cosine similarity. This function utilizes the cosine similarity matrix generated from our recommendation model to identify movies with the closest feature representations. Users can input a movie title, and the function will return a list of recommended movies that share similar attributes and themes.

2. Fetching Movie Information from TMDB:

Using the official TMDB (The Movie Database) Developer API, we create a function that retrieves detailed information about a specific movie, such as its title, overview, release date, genres, and ratings. Users can input a movie ID or title, and the function will query the TMDB API to fetch the corresponding movie details.

3. Fetching Movies with Posters:

We implement a function that fetches movie posters from TMDB using the movie IDs obtained from our recommendation model or user input. This function queries the TMDB API to retrieve URLs for movie posters, which can then be displayed within our recommendation platform or application. Users can visualize movie posters alongside their recommendations, enhancing the visual appeal and user engagement of the platform.

Hers our output for recommending movies for Iron Man

By integrating these functions into our recommendation system, we create a comprehensive and user-friendly experience for movie enthusiasts. Users can discover similar movies, explore detailed information about their favourite films, and enjoy visually appealing movie posters—all within a single, cohesive platform powered by the TMDB API and our recommendation model.

CONCLUSION

In conclusion, our movie recommendation project represents a culmination of data preprocessing, model building, and API integration to deliver a personalized and engaging movie discovery experience. By leveraging advanced techniques such as count vectorization, cosine similarity, and TMDB API integration, we have created a robust recommendation system capable of providing tailored movie suggestions to users.

Throughout the project, we focused on enhancing the user experience by streamlining the content discovery process and providing comprehensive movie information. We extracted key features such as genre, keywords, cast, crew, and overview from our dataset, allowing us to capture the essence of each movie and facilitate accurate recommendations.

Utilizing cosine similarity instead of Euclidean distance enabled us to measure similarity between movies more effectively, considering their thematic content and attributes.

Additionally, we developed functions to fetch detailed movie information from the official TMDB website, enriching our recommendations with additional context and insights.

Furthermore, our integration with the TMDB API enabled us to fetch movie posters, enhancing the visual appeal of our recommendation platform and providing users with a more immersive browsing experience.

In essence, our project demonstrates the power of data-driven approaches and API integration in enhancing content discovery and user engagement. Moving forward, there is ample opportunity to further refine and expand our recommendation system, incorporating additional features, improving recommendation accuracy, and catering to the evolving preferences of users in the dynamic landscape of digital entertainment.

ACKNOWLEDGEMENT

We are highly indebted to Prof. Amit Sethi for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project. We would like to express our gratitude towards our TAs for their kind co-operation and encouragement which helped us in completion of this project.

REFERENCES

1. https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
3. https://developer.themoviedb.org/reference/intro/getting-started