Section A -Research paper

MOVIES RECOMMENDER SYSTEM USING MACHINE LEARNING ALGORITHM

Kundan*, Kundan Singh[+], Tathagat [°] and R. K. Yadav[☆]
Programming Lab
Dept. of Computer Science, Delhi Technological University
Delhi, India
Email: kundansingh_2k19co202@dtu.ac.in, [+] kundan_2k19co201@dtu.ac.in, [°] tathagat_2k19co408@dtu.ac.in, [☆]rkyadav@dtu.ac.in

doi: 10.48047/ecb/2023.12.si4.921

***Abstract* - The purpose of this project is to develop a movie recommendation system using Python and machine learning algorithms, specifically cosine similarity. The system aims to provide personalized movie recommendations to users based on their preferences and similarities to other users. The cosine similarity algorithm will be used to measure the similarity between movies and users, allowing for effective recommendation generation. The project involves data collection and pre-processing, where a dataset of movie ratings and user information will be gathered. Feature extraction techniques will then be applied to extract relevant information from the dataset, such as genre, director, and actors. The cosine similarity algorithm will be implemented to compute the similarity scores between movies and users based on their shared features. Evaluation metrics will be employed to assess the performance of the recommendation system, such as precision, recall, and accuracy. The experimental setup will involve splitting the dataset into training and testing sets, ensuring robustness of the system. Results and analysis will be presented to showcase the effectiveness of the system in providing accurate and personalized recommendations. In conclusion, this project aims to develop a movie recommendation system using Python and the cosine similarity algorithm, providing users with personalized movie suggestions based on their preferences. The implementation of this system has the potential to enhance the movie-watching experience and facilitate movie discovery for users. Future work may involve incorporating additional machine learning algorithms and enhancing the system's scalability**

Index Terms – Machine Learning, Content Based Filtering, Cosine Similarity, API.

I. INTRODUCTION

Collaborative filtering, content-based filtering (also known as the personality-based approach), and other systems like knowledge-based systems are employed in the majority of recommendation systems.

In collaborative filtering, a model is created based on a user's prior actions (things previously purchased or selected, as well as the numerical ratings given to those items). Then, this model is used to forecast the ratings for things that the consumer could find interesting.

In order to recommend other objects with almost same qualities, content-based filtering approaches use a collection of distinctive, previously occurred features of an element. A hybrid system in today's recommender systems often combines one or more methodologies.

The distinctions among two filtering functions one of which is based on contest and other is cooperative can be shown by contrasting two frameworks which are very first to recommended for music system - Last.fm and Pandora Radio.

- Last.fm creates a "station" of suggested melodies by comparing the client's listening behaviour to the groups and individual tracks that they have consistently paid attention to. Last.fm will play songs that are frequently played by other users who are interested in the same things but are not in the user's library. Because it takes advantage of user behaviour, this strategy is an illustration of a collaborative filtering strategy.
- Using a portion of the 400 criteria provided by the Music Genome Project, Pandora develops a "station" that plays music with comparable quality. When a user "likes" a song, their input is taken into account by the station's results, but when they "dislike" a song, other variables are given more importance. This method illustrates a content-based strategy.

Both positives and negatives can be said for each system type. Last.fm requires a lot of information about the person in the preceding scenario in order to provide reliable recommendations. This is a typical manifestation of the cold start issue in collaborative filtering systems. While Pandora only requires a small amount of information to begin, its capabilities are significantly more restricted (for instance, it can only make suggestions that are similar to the initial seed).

Recommender systems are a beneficial alternative to search algorithms since they help users find products that they might not have found otherwise. It should be mentioned that the deployment of recommender systems typically makes use of search engines that index non-traditional data.

The Grundy system, a computer-based librarian that recommended books to users, was the first application of the RS concept. It debuted in 1979. This was followed by the introduction of Tapestry, the first commercial RS, in the early 1990s. In the early 1990s, Group Lens, a research lab at the University of Minnesota in the United States, produced another RS implementation to assist users in finding their chosen articles. Group Lens Recommender System is the term given to the system in honour of the group. This system asserts to be inspired by the same spirit as Tapestry, Ringo, Bell Core, and Jester. In the late 1990s, the installation of Amazon Collaborative Filtering contributed to the further development of RSs, one of the most popular RS innovations. Collaborative filtering-based RSs have grown in popularity since this time and are now being used by numerous internet and e-commerce platforms. It covered the problems with offline assessments and offered surveys of the current difficulties and available research paper recommender systems in the literature.

Digital recommendation systems are among the consumer products that are becoming more and more prevalent, such as books, music, apparel, movies, news items, locations, utilities, and so on. These systems collect user data in

order to improve future suggestions. It is possible to save the most recent technologies, user actions, and private data to social networks or e-commerce websites. For recommender systems, this kind of technology makes it easier to analyse user settings. Community based networks have the same level of complexity as Hui et al.'s. distinguish k-coterie networks among understudies and utilize this data to plan a powerful approach to communicating for portable organizations.

.

A. Problem Statement

The overwhelming number of movies available to users, leading to decision paralysis and difficulty in discovering relevant and enjoyable movies. The goal is to develop an efficient and personalized movie recommendation system that can accurately suggest movies to users based on their preferences and similarities to other users. The challenges include handling large datasets of movies and user ratings, extracting meaningful features from the data, and implementing an algorithm that can effectively measure the similarity between movies and users. The system should also consider diverse user preferences, genre preferences, and other factors like director, actors, and ratings. The project aims to solve the problem by applying machine learning techniques, specifically the cosine similarity algorithm, to calculate the similarity scores between movies and users. By analyzing user ratings and movie features, the system will generate personalized movie recommendations that align with the user's interests.

B. Methodology

Firstly, A comprehensive dataset of movie ratings and user information is collected from reliable sources or APIs. This dataset is then pre-processed by handling missing values, removing duplicates, and standardizing the data format. Subsequently, the dataset is split into training and testing sets for evaluating the system's performance. Feature extraction is performed to extract relevant features from the dataset, such as movie genres, directors, actors, and user preferences. Categorical features are represented numerically using techniques like one-hot encoding or vectorization, while numerical features are normalized for consistency. To evaluate the recommendation system, appropriate metrics such as precision, recall, and accuracy are selected. These metrics are used to assess the system's performance by comparing the recommended movies with the actual user ratings. the methodology encompasses data collection and pre-processing, feature extraction, implementation of the cosine similarity algorithm, evaluation metrics, experimental setup, results and analysis, user interface design, system implementation, and testing/validation.

C. Scope of the Project

The scope of the movie recommendation system project using the cosine similarity machine learning algorithm includes developing a system that provides personalized movie recommendations to users based on their preferences and similarities to other users. The project involves collecting a comprehensive dataset of movie ratings and user information, pre-processing the data, and extracting relevant features such as genres, directors, actors, and user preferences. The cosine similarity algorithm will be implemented to calculate similarity scores between movies and users, enabling the identification of movies that align with individual user interests. The system's performance will be evaluated using metrics like precision, recall, and accuracy, comparing recommended movies with actual user

ratings. The project encompasses an experimental setup, user interface design, and system implementation. Testing and validation will ensure the reliability and accuracy of the recommendation system. The goal is to deliver a functional and accurate movie recommendation system that enhances the movie-watching experience and facilitates movie discovery for users.

II. RELATED WORK

Movie recommendation systems have been extensively researched, leading to advancements in various areas. Collaborative filtering techniques, such as user-based and item-based approaches, have been utilized to generate recommendations based on user-item ratings and similarities. Content-based filtering methods have focused on extracting meaningful features from movie metadata and employing machine learning algorithms for recommendation models. Deep learning and neural networks, including convolutional neural networks and recurrent neural networks, have been applied to enhance content-based recommendations and capture temporal patterns in user-item interactions. Context-aware and personalized recommendations have considered additional contextual information to deliver more relevant suggestions. Social network analysis has incorporated social influence, user communities, and influence propagation for improved recommendations. Evaluation metrics and techniques, such as precision, recall, and online user studies, have been employed to assess recommendation system performance. The research aims to develop accurate, personalized, and user-centric movie recommendation systems that enhance the movie-watching experience

III. OVERVIEW OF MOVIE RECOMMENDATION SYSTEM

A movie recommendation system is a technology that provides personalized movie suggestions to users based on their preferences and interests. It aims to enhance the movie-watching experience by offering relevant and diverse movie recommendations. The system analyses various factors, such as user ratings, movie features (genre, director, actors), and user behaviour, to generate recommendations that align with individual tastes.

There are different approaches used in movie recommendation systems:

Collaborative Filtering: This technique analyses user-item interactions and similarities between users or items to generate recommendations. It recommends movies that similar users have enjoyed or movies that are similar to those previously rated positively by a user.

Content-Based Filtering: This approach considers the characteristics and features of movies, such as genre, director, and actors, to recommend movies with similar attributes to the ones a user has liked in the past.

Hybrid Methods: These methods combine collaborative filtering and content-based filtering to leverage the strengths of both approaches and provide more accurate recommendations. They take into account both user preferences and movie attributes.

Deep Learning: With the advancement of deep learning techniques, movie recommendation systems have employed neural networks, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to extract meaningful features from movie images or capture sequential user behaviour for better recommendations.

The evaluation of movie recommendation systems involves assessing the accuracy and effectiveness of the recommendations. Metrics such as precision, recall, mean average precision, and user satisfaction are commonly used to measure the performance of the system.

Overall, movie recommendation systems aim to assist users in discovering new movies based on their preferences and improve their movie-watching experience by providing personalized and relevant recommendations

IV. PREDICTION MODEL

A prediction model using machine learning is a type of model that uses historical data and a set of predefined algorithms to make predictions about future events or outcomes. This type of model can be used for a wide variety of applications, such as forecasting sales, predicting stock prices, or identifying potential fraud. To create a prediction model using machine learning, you would typically start by collecting and cleaning a dataset, and then selecting and training a machine learning algorithm on that dataset. The trained model can then be used to make predictions on new, unseen data.

V. MACHINE LEARNING

Machine learning is increasingly being used in intrusion detection systems (IDS) to improve their accuracy and effectiveness. Machine learning algorithms can analyze large amounts of network traffic data and identify patterns that indicate an attack, even if the attack is not immediately recognized by traditional signature-based IDS. [14]

There are several types of machine learning algorithms that can be used in intrusion detection, including:

Anomaly detection: This involves identifying abnormal behaviour that deviates from normal patterns of network activity. This can be useful in detecting zero-day attacks where there is no known signature.

Supervised learning: This involves training an algorithm on a dataset of known attacks and normal activity. Once trained, the algorithm can then classify new network activity as either normal or malicious.

Unsupervised learning: This involves clustering normal and abnormal network traffic data together to find patterns in the data. This can be used to identify new types of attacks.

Machine learning-based IDS can be more effective than traditional signature-based IDS because it can detect unknown threats and adapt to new types of attacks. However, they also require a lot of data to train and can have a higher rate of false positives.

VI. RECOMMENDATION SYSTEM

Intrusion All of us are familiar with vectors, which might be 2D, 3D, or any other type of D. For the sake of clarity, let's concentrate on 2D for a bit, and first go through what the dot product is. The speck object that occurs between the two vectors is comparable to the projection of one vector onto the other. Consequently, the squared module of two vectors whose components are identical is the same as their dot product; However, if the vectors are perpendicular, which indicates that they do not share any directions, the dot product is zero. For n-dimensional vectors, the method shown below can typically be used to calculate the dot product.

$$u \cdot v = [u_1\ u_2\ \dots\ u_n] \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = u_1 v_1 + u_2 v_2 + \dots + u_n v_n = \sum_{i=1}^{n} u_i v_i$$

Fig1. Dot product

Dot product.

Because it is inextricably linked to the similarity, the dot product plays a crucial role in its definition. The ratio of the dot products of the two vectors u and v to the sum of their magnitudes is the definition of similarity.

$$similarity = cos(\theta) = \frac{u \cdot v}{\|u\|\|v\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}}$$

Fig2. Cosine Similarity

Similarity.

This will actually equal 1 if the two vectors are identical and 0 if they are orthogonal if the similarity definition is utilised. To put it another way, the similarity is a number between 0 and 1 that indicates the degree to which the two vectors are alike. It's not hard at all!

VI. DATA EXPLORATION

To carry out the implementation, we have collected data from approximately 5,000 films. The data has been loaded into the jupyter notebook data frame from the csv file. In order to carry out the implementation of the recommendation system, more than 20 attributes of a movie have been inserted into the data in each row. These attributes include the movie's id, name, genre, budget, popularity, and production company. Movies will be chosen based on cosine similarity, which only takes into account a small number of attributes because it's hard to process a lot of data. These characteristics include the film's genre, cast, director, keyword, and tagline.

Fig 3. Data Preprocessing

In order to use textual data for predictive modelling, the text must be parsed to remove certain words – this process is called tokenization. These words need to then be encoded as integers, or floating-point values, for use as inputs in machine learning algorithms. This process is called feature extraction (or vectorization).

Count Vectorizer is used to convert a collection of text documents to a vector of term/token counts. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.

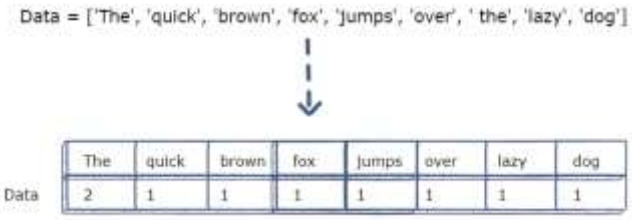

Fig 4. Count Vectorizer

```
In [38]: from sklearn.feature_extraction.text import CountVectorizer
         cv = CountVectorizer(max_features=5000,stop_words='english')

In [39]: vectors = cv.fit_transform(new_df['tags']).toarray()

In [40]: vectors

Out[40]: array([[0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                ...,
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

Fig 5. Training

API

An application programming interface, or API, enables companies to open up their applications' data and functionality to external third-party developers, business partners, and internal departments within their companies.

This allows services and products to communicate with each other and leverage each other's data and functionality through a documented interface. Developers don't need to know how an API is implemented; they simply use the interface to communicate with other products and services. API use has surged over the past decade, to the degree that many of the most popular web applications today would not be possible without APIs.

Fig 6. API fetching command

Results:-

Fig 7. Code for checking movie recommender system

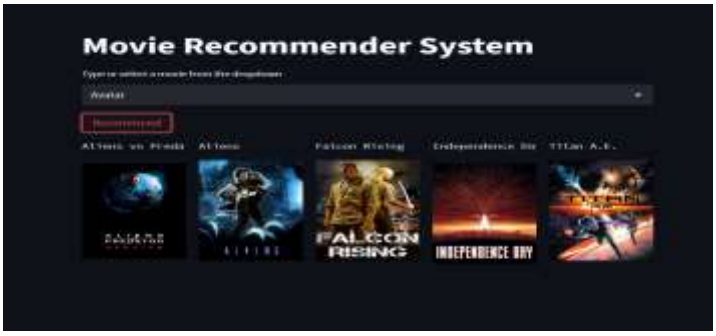

Fig 8. Result

VII. CONCLUSION

APIs. The theory underlying the most widely used recommendation algorithms, including collaborative filtering along with content-based filtering and hybrid methods, is explained in this report. The purpose of this study was to learn about the advantages and disadvantages of each algorithm and choose the one that best suited the dataset. Building a system that gets good recommendations from new users or from a cold start is hard. It may be necessary to count with more information, not only about the user's profile but also about the movies, in order to produce a model that produces results that are acceptable. This may make it possible for us to apply other techniques, such as content-based filtering and hybrid filtering, and it may also lead us to results that are more significant.

VIII.REFERENCES

[1] Isinkaye, F.O., Y.O. Folajimi, and B.A. Ojokoh (2015). "Recommendation systems: Principles, methods and evaluation". In: Egyptian Informatics Journal 16.3, pp. 261 –273. ISSN: 1110-8665.

[2] Liang, Xijun et al. (2016). "Measure prediction capability of data for collaborative filtering". English. In: Knowledge and Information Systems 49.3. Copyright - SpringerVerlag London 2016; Last updated - 2016-11-03; CODEN - KISNCR, pp. 975– 1004.

[3] Karlgren, Jussi (October 2017)A digital bookshelf: original work on recommender systems". Retrieved 27 October 2017.

[4] D.H. Wang, Y.C. Liang, D.Xu, X.Y. Feng, R.C. Guan (2018), "A content-based recommender system for computer science publications", Knowledge-Based Systems, 157: 1-9

[5] Lee D (2015) Personalizing information using users' online social networks: a case study of CiteULike. J Inf Process Syst 11:1–21

[6] Ricci F, Rokach L, Shapira B (2011) Introduction to recommender systems handbook, recommender systems handbook. Springer, Berlin, pp 1–35

[7] Kumar Manoj, D.K. Yadav, Singh Ankur and Kr Vijay, "A Movie Recommender System: MOVREC", 2015 International Journal of Computer Applications, vol. 124, pp. 7-11.

[8] A. Jain and S. K. Vishwakarma, "Collaborative Filtering for Movie Recommendation using RapidMiner", International Journal of Computer Applications, vol. 169, no. 6, pp. 0975-8887, July 2017.

[9] Y. Zhou, D. Wilkinson, R. Schreiber, R. Pan. "Large-Scale Parallel Collaborative Filtering for the Netflix Prize", AAIM 2008: 337-348.

[10] B. Sarwar, G. Karypis, J. Konstan and John Riedl, "Item-Based Collaborative Filtering Recommendation Algorithms", Proceedings of the 10th international conference on World Wide Web 2001: 285-295.