Movie Recommendation Technique Using Cosine Similarity

Gowri Venkat N , Keerthevasan T S

Abstract—Over the past years, the internet has broadened the horizon of various domains to interact and share meaningful information. As it is said that everything has its pros and cons therefore, along with the expansion of domain comes information overload and difficulty in extraction of data. To overcome this problem the recommendation system plays a vital role. It is used to enhance the user experience by giving fast and coherent suggestions. This paper describes an approach which offers generalized recommendations to every user, based on movie popularity and/or genre. Content-Based Recommender System is implemented using various deep learning approaches. This paper also gives an insight into problems which are faced in content-based recommendation system and we have made an effort to rectify them.

Keywords: Recommendation System, Content-Based Recommender System, Deep learning

I. INTRODUCTION

Advancement in technology is reaching new heights every day and due to which we can see enormous growth in information. To deal with such large data we use machine learning that automates analytical model building [1]. The early classification of machine learning is divided into three broad categories: Supervised learning, Unsupervised learning and Reinforcement learning [2]. We use computers to make predictions to help us achieve better results using various computational statistics. Tasks can be performed without being explicitly programmed to do so [3]. It becomes a tedious task to extract the relevant information. Search engines solve the problem to some extent but it does not solve the personalization problem. Recommendation System framework plays a vital role in today's internet surfing, be it buying a product from an e-commerce site or watching a movie on some video-on-demand service [4]. In our everyday life, we depend on recommendations given by other people either by word of mouth or reviews of general surveys. People often use recommender systems over the web to make decisions for the items related to their choice. Recommendation systems are software tools and techniques whose goal is to make useful and sensible recommendations to a collection of users for items or products that might interest them [5]. In other words, the recommender system or recommendation systems belongs to a class of information filtering system that aims at predicting the 'preference' or 'rating' given to an item. Recommendation systems are primarily using three approaches [6]. In content-based filtering, we do profiling based on what type of content any user is interested in and using the collected information, it recommends items. Another one is collaborative filtering, where we make clusters of similar users and use that information to make recommendations. Hybrid systems are the one which takes into account both above stated approaches to deal with operational data more concisely [7]. Our goal is to provide accurate recommendations with less computational complexity.

II. RELATED WORKS

Some of the common approaches of recommender system are:

1. Content-based filtering

2. Collaborative filtering

3. Hybrid filtering

A. Content Based Filtering

This approach filters the items based on the likings of the user. It gives result based on what the user has rated earlier. The method to model this approach is the Vector Space Model (VSM). It derives the similarity of the item from its description and introduces the concept of TF-

The similarity between item vectors can be computed by three methods:

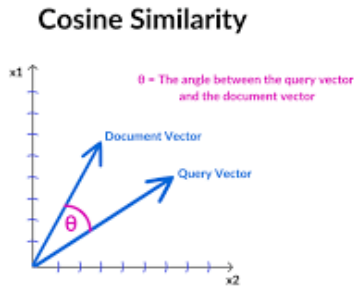
1. Cosine similarity

2. Euclidean distance

3. Pearson's correlation

COSINE SIMILARITY

Cosine similarity among two objects measures the angle of cosine between the two objects. It compares two documents on a normalized scale. It can be done by finding the dot product between the two identities.



[25] As the above diagram shows, the angle between document vector and query vector is. Lesser the angle between the two vectors the more is the similarity. It means if the angle between two vectors is small, they are almost alike each other and if the angle between the two vectors is large then the vectors are very different from each other.

B. Collaborative Filtering

It depends upon the users who have similar interests and gives the result based on all the users. User-based: In user-based collaborative filtering, it is considered that a user will like the items that are liked by users with whom have comparable taste.

Item-based: Item-based collaborative-filtering is different, it expects the users to like items that are related to items that he has liked earlier.

C. Hybrid Filtering

Hybrid filtering can be known as a combination of collaborative filtering and content-based filtering. It is the most common and popular technique today. It avoids the weakness of every single recommender technique. There are some problems related to the recommendation system. They are:

Cold-start problem: When a user registers for the first time, he has not watched any movie. So, the recommendation system does not have any movie based on which it can give results [8]. This is called the cold-start problem [9].Recommendation system goes through this problem as a result of no previous record. This happens with every new user once.

Data sparsity problem: This problem occurs when the user has rated very few items based on which it is difficult for the recommendation system to give accurate results. In this problem, the results given are not much similar to the expected result. Sometimes, the system fails to give successful results and generates weak recommendations [10]. Also, data sparsity always leads to coverage problems.

Scalability: Another problem associated with recommendation is scalability. In this, the encoding goes linearly on items. The system works efficiently when the data set is of limited size [11]. As the data set increases, it becomes difficult for the recommendation system to give accurate results based on varying genres of movies. The scalability problem can be solved by dimensionality reduction technique such as singular value decomposition (SVD). It also speeds up the generation of result and produces a reliable and efficient result. Synonym: When many words have a similar meaning then they are known as synonyms. In this problem, the system sometimes fails to understand the difference between two similar words and cannot produce the desired output. E.g.: movie and film have the same meaning but the recommendation system considers them as different words and because of this it fails to give the accurate output. Singular Value Decomposition (SVD), especially Latent Semantic Indexing is capable of solving the synonymy problem [12]. In other recommendation systems, recommendations are based on the ratings and genres given by other users due to which it became difficult to give accurate results. This is called collaborative filtering. This is a common method for a recommendation system. To make things easier for people and overcome this drawback, we have used another method for the recommendation which is content-based filtering. This method uses ratings and genres given by the user itself. It gives recommendations based on the movies watched by the user earlier. This helps us in giving accurate suggestions because the ratings and genres are given by the user only and depend entirely on the user's choice, not on any other person's choice. Previous research lacks the accuracy of true recommendations due to their dependency on other users. Here it is important to mention that recommendations can be based on a particular user and they assure to give recommendations on the mark as they depend on the likes and dislikes of a particular user. The proposed system is capable of storing a large amount of data and give efficient results. Our recommendation system searches for

the best movies that are similar to the movie that we have watched based on genre and gives the result.

III. METHODOLOGY

The proposed recommendation system aims to enhance user experience on streaming platforms by providing highly personalized movie suggestions using a hybrid approach that combines **cosine similarity** and the **K-Nearest Neighbor (KNN)** algorithm. The methodology is divided into several stages, each addressing key challenges in traditional recommendation systems, such as cold start, data sparsity, and scalability.

3.1 Data Collection and Preprocessing

The initial step involved collecting movie data from reliable sources, such as **IMDb** and **TMDb**. The system required a diverse dataset to include information on movie titles, genres, cast members, plot summaries, and user ratings. This data was essential for developing a content-based filtering system using cosine similarity and clustering algorithms. Data was collected using API integration, ensuring that the dataset was up-to-date and accurate.

3.2 Feature Engineering and Data Transformation

To optimize the recommendation algorithms, the data underwent extensive preprocessing. The attributes, such as genres and cast, were one-hot encoded, while plot summaries were transformed using TF-IDF (Term Frequency-Inverse Document Frequency) for text vectorization. This step ensured that the feature vectors were structured and ready for similarity calculations. Additionally, user ratings were normalized to bring them into a comparable range, ensuring consistency across the dataset.

3.3 Cosine Similarity Calculation

The cosine similarity algorithm was integrated to measure the degree of similarity between movies based on their feature vectors. Each movie was represented in a multi-dimensional space, allowing the system to compute similarity scores between pairs of movies efficiently. By using Python's scikit-learn library, the similarity matrix was generated to rank movies based on their relevance to the user's profile.

3.4 KNN Clustering for Efficient Recommendations

To enhance the recommendation process, a K-Nearest Neighbor (KNN) algorithm was used for clustering similar movies. The system grouped movies into clusters based on their cosine similarity scores, optimizing the search space for generating recommendations. The clustering step ensured that users received diverse yet relevant movie suggestions, improving recommendation accuracy.

3.5 Recommendation Engine Development

The recommendation engine was developed to integrate the outputs from the cosine similarity and KNN modules. By analyzing user profiles and movie clusters, the engine provided personalized movie suggestions. The recommendation logic was fine-tuned to filter out previously watched movies and prioritize new content. The engine was implemented using Python, leveraging efficient data structures for real-time performance.

3.6 Feedback Integration and Continuous Learning

The system was designed to incorporate a feedback loop, allowing it to adapt to user preferences over time. User interactions, such as likes, dislikes, and skips, were tracked to adjust similarity scores dynamically. This continuous learning process ensured that the recommendations remained relevant and aligned with evolving user tastes.

3.7 System Deployment and Testing

The fully developed recommendation system was implemented using Flask for the web interface and Django for the back-end. The system was tested extensively, including unit tests for the recommendation algorithms and integration tests for the overall platform functionality. The deployment utilized Heroku for hosting, with MongoDB for scalable data storage.

IV. RESULTS AND DISCUSSION

In the deep learning framework recommendation system, we have used Cosine Similarity and Content-based filtering to predict our result and recommend a movie to the user by running the code in python using NumPy and panda libraries.

A. Experiment Result

The formula used to measure how similar the movies are based on their similarities of different properties. Mathematically, it shows the cosine of the angle of two vectors projected in a multidimensional space. The cosine similarity is very beneficial since it helps in finding similarobjects.

$$similarity(A, B) = cos(\theta) = \frac{A \cdot B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sum_{i=1}^{n} B_i^2}.$$

Fig: Formula of cosine similarity for the recommendation system

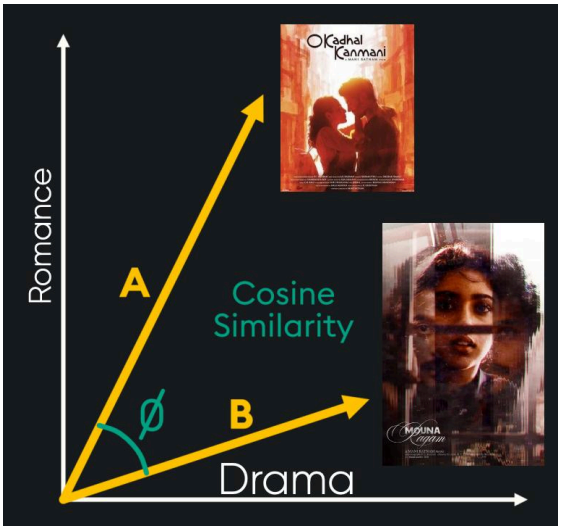


Fig: To implement Cosine similarity we take an example of 2 movies of different genre romance and drama

The angle theta between the two movies will determine the similarity between the two movies. The theta ranges from 0- 1. If the value of the theta is near 1 then it is most similar and if it's near to 0 then it is least similar. The movie will be recommended if it is close to 1 otherwise there would be no similarity between them. It will recommend the best movies to the user according to the Cosine similarity. After the cosine similarity, we have used a normalised popular score through which we get our function of computing distance. Then by using the KNN functionality, we have found the nearest neighbour which will be recommended to the user.

movie_id	title	genres
19995	Avatar	[Action, Adventure, Fantasy, ScienceFiction]
285	Pirates of the Caribbean: At World's End	[Adventure, Fantasy, Action]
206647	Spectre	[Action, Adventure, Crime]
49026	The Dark Knight Rises	[Action, Crime, Drama, Thriller]
49529	John Carter	[Action, Adventure, ScienceFiction]
...	...	...
9367	El Mariachi	[Action, Crime, Thriller]
72766	Newlyweds	[Comedy, Romance]
231617	Signed, Sealed, Delivered	[Comedy, Drama, Romance, TVMovie]
126186	Shanghai Calling	[]

V. CONCLUSION AND FUTURE WORKS

To summarize, the hybrid movie recommendation system successfully achieves its goal of providing personalized content suggestions by leveraging a combination of cosine similarity and K-Nearest Neighbor (KNN) algorithms. The system effectively addresses challenges found in traditional recommendation methods, such as the cold start problem, data sparsity, and scalability issues, by incorporating both content-based filtering and clustering techniques. Users benefit from tailored recommendations that enhance content discovery, improve engagement, and increase satisfaction on streaming platforms. By integrating continuous user feedback, the system remains adaptive to changing user preferences, ensuring a dynamic recommendation experience**.**

5.1 Constraint and Limitation

One of the primary constraints encountered during the development of the recommendation system was managing the computational complexity associated with calculating cosine similarity across a large dataset. Although the use of clustering reduced the search space, optimizing performance for real-time recommendations remains a challenge, particularly as the volume of data grows. Another limitation was the integration of user feedback into the recommendation model. While the system captures explicit feedback (likes, dislikes), implicit feedback, such as viewing duration, requires further refinement for accurate preference prediction. Additionally, maintaining data quality from external sources, such as IMDb and TMDb, proved to be challenging due to inconsistencies in API responses.

5.2 Future Works

In the future, the movie recommendation system will undergo several enhancements to improve its accuracy and scalability. One key focus will be the integration of deep learning techniques, such as neural collaborative filtering, to capture more complex user-item interactions and deliver more accurate recommendations. Additionally, the system will incorporate advanced feedback mechanisms, using sentiment analysis to interpret implicit feedback like user engagement patterns.

Real-time personalization capabilities will be developed to allow the system to instantly adapt to user interactions, thereby improving responsiveness. To handle an expanding dataset, optimization of the existing algorithms will be prioritized, ensuring seamless scalability as the platform grows. Another area of enhancement involves integrating social media data to enrich user profiles and generate more holistic recommendations based on social interests and current trends. Finally, by expanding content features to include attributes like release dates, language preferences, and regional popularity, the system aims to provide a more diverse set of recommendations, catering to a broader range of user preferences. Continuous user feedback and rigorous testing will guide these improvements, ensuring that the system remains adaptive and effective in meeting user needs.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to the development team and all the collaborators who contributed to the success of this project. Special thanks go to the data science experts and machine learning researchers who provided valuable insights into the design and implementation of the recommendation algorithms. We are also grateful to the beta testers who participated in the evaluation phase, offering critical feedback that helped refine the system's functionality. Additionally, we acknowledge the support from platforms such as IMDb and TMDb for providing access to extensive movie datasets, which were integral to the project. Finally, we extend our appreciation to the OpenAI technology, which played a crucial role in shaping the research and development of this work.

REFERENCES

[1] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web (WWW)*, Hong Kong, 2001, pp. 285-295.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30-37, Aug. 2009.

[3] R. Burke, "Hybrid web recommender systems," in *The Adaptive Web*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Germany: Springer, 2007, pp. 377-408.

[4] A. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. New York, NY, USA: Springer, 2011, pp. 73-105.

[5] C. Zhang, X. Yao, S. Zeng, and W. Zhang, "Deep learning based collaborative filtering for personalized recommendations," in *Proc. 2019 Int. Joint Conf. Neural Networks (IJCNN)*, Budapest, Hungary, 2019, pp. 1-8.

[6] H. Aggarwal, *Data Mining: The Textbook*. Cham, Switzerland: Springer, 2015.

[7] C. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008.

[8] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109-132, July 2013.

[9] A. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: Scalable online collaborative filtering," in *Proc. 16th Int. Conf. World Wide Web (WWW)*, Banff, AB, Canada, 2007, pp. 271-280.

[10] F. Furtado and A. Singh, "Latent factor models in recommender systems," in *IEEE Trans. Knowledge and Data Engineering*, vol. 25, no. 2, pp. 236-250, Feb. 2013.