WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

A Personalized Movie Recommendation System Using Content-Based Filtering and User Sentiment

Nagaraj M. Lutimath[1], Guvvala Harshitha[2] ,Dharmavaram Poojitha[3] ,Jananya R Reddy[4] ,Ganga[5]

[1]nagarajlutimath@gmail.com, Computer Science and Engineering, Dayananda Sagar Academy Of Technology and Management, Bengaluru, India
[2]harshithareddyguvvala@gmail.com, omputer Science and Engineering,Dayananda Sagar Academy Of Technology and Management,Bengaluru,India
[3]dpoojitha308@gmail.com, Computer Science and Engineering , Dayananda Sagar Academy Of Technology and Management, Bengaluru,India
[4]jananyaravi81@gmail.com,Computer Science and Engineering,Dayananda Sagar Academy Of Technology and Management, Bengaluru,India
[5]gangabandrawad@gmail.com, Computer Science and Engineering, Dayananda Sagar Academy Of Technology and Management,Bengalru,India

ABSTRACT:

This paper presents an improved approach to building a better content-based movie recommendation system by integrating sentiment analysis. The goal is to offer users more personalized, relevant, and meaningful movie suggestions. To achieve this, the system first gathers essential movie details — like the title, genre, runtime, ratings, and poster — using the TMDb API. It also pulls in user reviews from IMDb through web scraping, using the Beautiful Soup library, which helps collect this data effectively. Once the reviews are collected, sentiment analysis is applied to understand whether the feedback is positive, negative, or neutral, based on the emotions expressed by users.

For generating recommendations, the system uses cosine similarity, a powerful method that measures the similarity between text data. By comparing the angles between vector representations of movie features and user preferences, it ensures that the recommendations are not just based on the volume of data, but on meaningful connections between different movie attributes. This results in accurate and trustworthy suggestions for the user.

Keywords: Content based filtering, Sentiment Analysis, Cosine Similarity, TMDb API, IMDb Web scraping.

I. INTRODUCTION:

1. A Comprehensive Overview of Recommender Systems
•Recommender systems are designed to recommend the appropriate items to offer like movies, books, or any type of products. For movie recommendation, these systems analyse user preferences and recommend the similar films in order to enhance the user's satisfaction.
2. Content-Based Filtering
Content-based filtering gives much importance to the critical elaboration on item attributes that may take to include:
o Titles of films
o Genres
o Running Time
Advantage- Organic fit for a new user or a niche product that doesn't generate enough other users' data.

II. LITERATURE REVIEW

1.Authors: P. Deshmukh et al.
Published: 2021, ICAIS
Proposed Solution: Combines collaborative and content-based filtering using cosine similarity and KNN classifier on the IMDb dataset. The system calculates similarity between user preferences and movies to predict recommendations. Merits: Personalized suggestions, handles large datasets well. Demerits: Cold-start issues, less accurate with limited user history.[1]

2.Authors: A. Thakur et al.
Published: 2021, ISPCC
Proposed Solution: A hybrid system that integrates collaborative filtering and content-based filtering using user ratings and genre tags to reduce sparsity and improve accuracy. Merits: Better accuracy, reduces individual method flaws. Demerits: High complexity, depends on input data quality.[2]

3.Authors: K. Jayasankar et al.
Published: 2022, ICCCIS

Proposed Solution: Uses a hybrid approach combining TF-IDF, user similarity, and the MovieLens dataset to improve accuracy and diversity of movie suggestions. Merits: Accurate suggestions, handles sparsity. Demerits: Complex design, needs good user data.[3]

4.Authors: Salton, G., Wong, A., & Yang, C. S.
Published:1975, Communications of the ACM
Proposed Solution: The authors proposed the vector space model for automatic indexing, which represents documents and queries as vectors in a multi dimensional space. The similarity between documents and queries is measured using metrics such as cosine similarity. Merits: It provided an efficient way of calculating relevance measures between documents and user queries that would eventually serve as the foundation for modern text retrieval systems. The method introduced the concept of term weighting and dimensionality reduction for accuracy in retrieval. Demerits: Semantic relationships between terms are not captured in the model. It depends on term frequency and vector space representation. Hence, it may face serious problems with large-scale datasets since the computational complexity is inherent. [4].

5.Authors: Karypis, G.
Published: 2001, Tenth International Conference on Information and Knowledge Management
Proposed Solution: The paper introduced item-based collaborative filtering algorithms for recommendation systems, which analyze the relationships between items rather than the users. The algorithm generates item-item similarity matrices and then predicts user preferences based on previously rated items. Merits: Item-based collaborative filtering is highly efficient for sparse datasets and scales well with a growing number of users. It provides stable recommendations, especially in environments with a large volume of users and relatively fewer items. Demerits: Cold-start problems still persist since recommendations are based on historical data for new users or items. In addition, the accuracy of the algorithm may decrease in domains where user preferences are very diverse. [5]

6.Authors: The Movie Database (TMDb)
Published: Ongoing, API Documentation.
Proposed Solution: TMDb offers an API to access millions of movies with metadata, user ratings, and reviews that can be used to construct movie recommendation systems. Benefits: TMDb provides comprehensive, current information to improve the quality and significance of recommendations. The fact that it is a community updated service information on time. guarantees developers get Drawbacks: This increases issues of API limits for TMDb, chances of data unavailability and dependence on third-party service providers for the working of the application.[6]

III. METHODOLOGY

General Movie Recommender Systems:

Recommender systems are of great significance to the entertainment industry by making the viewing experience rich by providing users with specific films based on individual tastes and choices.
Content-Based Filtering: It is a process where movies are recommended using features like genre, name, running time, and casts.
Collaborative Filtering: Movies are suggested by understanding the behavior of users, thus finding the patterns and patterns related to users having similar choices.
Hybrid Systems: Combines content-based and collaborative filtering to overcome the limitations of standalone approaches

Movie Recommender System Development:

-API Integration: External APIs such as TMDb provide detailed movie data, allowing for accurate and scalable recommendations. IMDb)
-User Sentiment Analysis: Mining opinions from reviews (e.g., adds recommendations. emotional nuance to recommendations.
-Machine Learning Methods: Implements algorithms such as KNN and matrix factorization in order to further enhance accuracy in predictions.
-NLP Methods: Employs cosine similarity as well as word embeddings for boosting similarity measures in the text form.

The Challenges While Developing Movie Recommendation Systems:

-The cold start problem: Where the availability of data would be nil for new users or even movies. Solution: Hybrid model or scrapping of some external data.
-Data Sparsity: The system performance is weakened by the sparse data. Solution: Uses matrix completion and collaborative filtering.
-Recommendation Bias: System is biased toward popular genres or movies. Solution: Makes use of fairness-aware algorithms along with re-ranking mechanisms. -Lack of integration of sentiment analysis in realtime recommender systems.

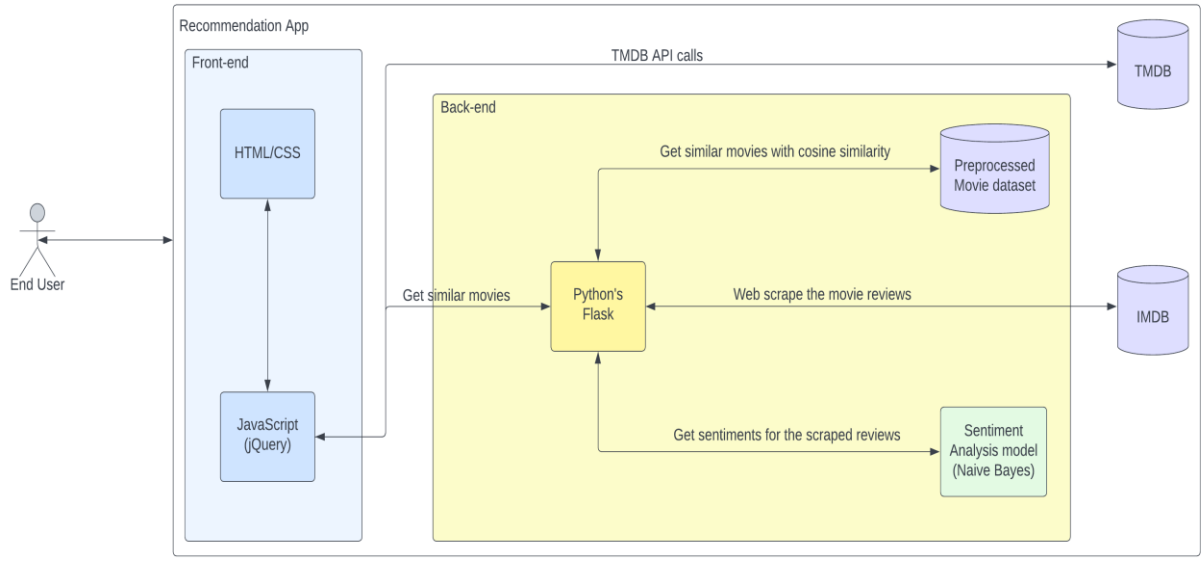

Fig: Methodology of Movie Recommendation System

IV. REQUIRED SPECIFICATIONS

Front-End Specifications

- Technologies: HTML, CSS, JavaScript (jQuery).
- Features: User-friendly input forms for choosing favorite films.
- Sentiment analysis output visualized as interactive charts.
- Responsive design to ensure compatibility on devices.
- Dynamic display of movie suggestions with posters and descriptions.
- User-friendly

Back-End Specifications

- Framework: Flask (Python).
- Movie Recommendation Logic: Applies cosine similarity to calculate movie features and suggest similar movies.
- Web Scraping: Scrapes IMDB reviews for sentiment-related information
- Sentiment Analysis: Applies Naive Bayes classification to classify reviews as positive, negative, or neutral.
- Integration with TMDB API: Retrieves current movie information, such as movie metadata and details.
- Database Handling: Stores processed movie information for easy access during recommendation and sentiment analysis.

Hardware Specifications

- Processor: Minimum Intel i5 (8th Gen) or equivalent.
- RAM: 8GB or higher.
- Storage: 50GB SSD (minimum).
- Graphics Card: Optional (NVIDIA GTX 1050 Ti or higher for deep learning operations).

Software Specifications

- Operating System: Windows 10 or Linux (Ubuntu for development environment).
- Python: Version 3.6+.
- Framework: Flask for backend development.
- Libraries: NumPy, Pandas, Scikit-learn, BeautifulSoup, Scrapy, etc
- API Integration: TMDB API for movie metadata.

V. IMPLEMENTATION

- **Data Sourcing:** Utilize TMDB API to gather extensive movie data such as genres, cast, crew, posters, and popularity.
- **Keyword Extraction :**Tokenize and extract meaningful textual features like overview, tagline, genres, and keywords from the dataset.
- **Feature Vectorization:** Employ TF-IDF to transform textual features into numerical vectors denoting word importance.

- **Similarity Matrix Generation**: Calculate cosine similarity scores between all movies to generate a similarity matrix for recommendation logic.
- **Web Scraping Integration** :Create a web scraper with Beautiful Soup to obtain real time IMDB reviews of chosen movies.
- **Model Training for Sentiment Analysis:** Train a Multinomial Naive Bayes model based on a labelled dataset of IMDB reviews to predict sentiment classification
- **Frontend Development:** Create a Streamlit-based UI featuring a movie selector, recommendation section, and review analysis plot**.**
- **Top-N Recommendation Logic :**With user input, fetch similar top 5 movies without duplicating the chosen movie**.**
- **Sentiment Visualization :**Use Matplotlib or Plotly to create a pie chart of scraped review sentiment distribution.
- **Error Handling and Logging System :**Use try-except blocks with logging for scraping errors, missing data, or API problems to make it robust.

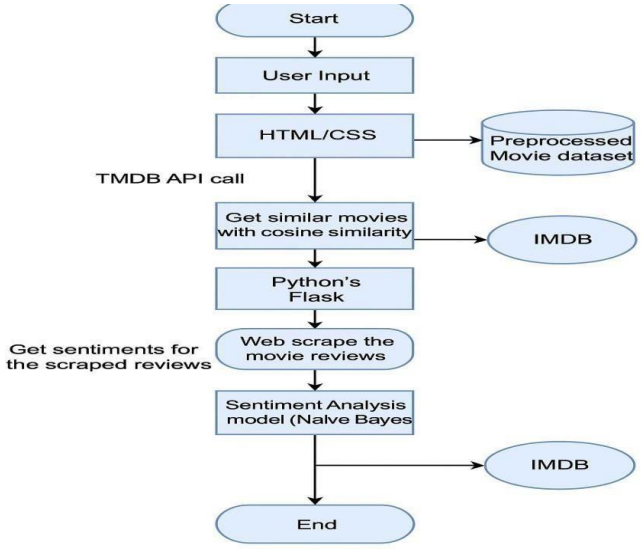

Fig: Flowchart of Implementation

VI. RESULTS

After performing various test cases, it was found that the system is working as expected. The movie recommendation results were accurate, IMDb reviews were fetched and classified correctly, and the interface was user-friendly and responsive. Proper error messages were shown for invalid inputs. Overall, the system passed all major test scenarios.

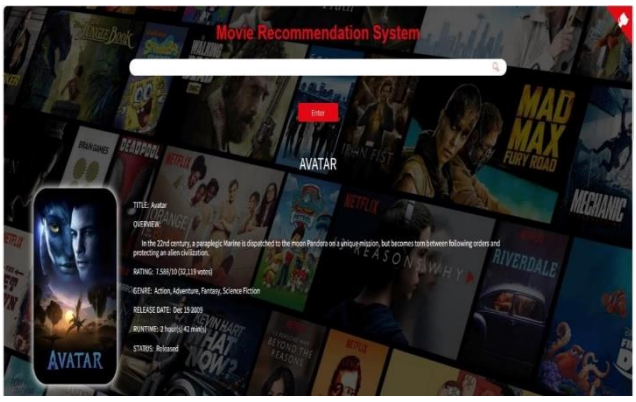

Fig: Searching a Movie

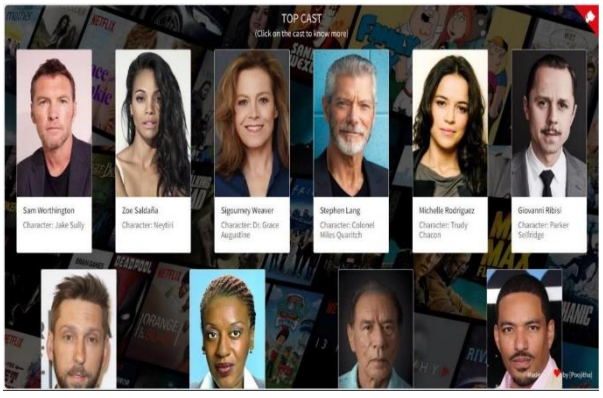

Fig: Top cast in the movie

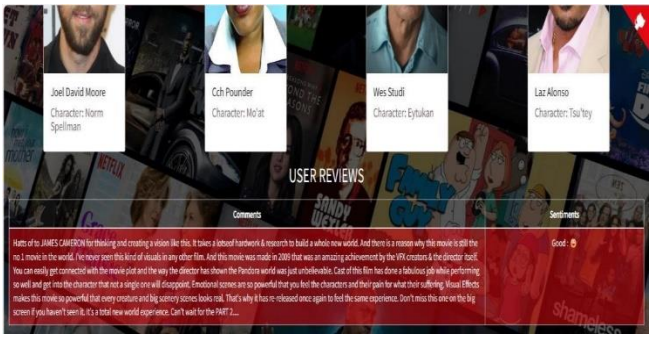

Fig: User Reviews

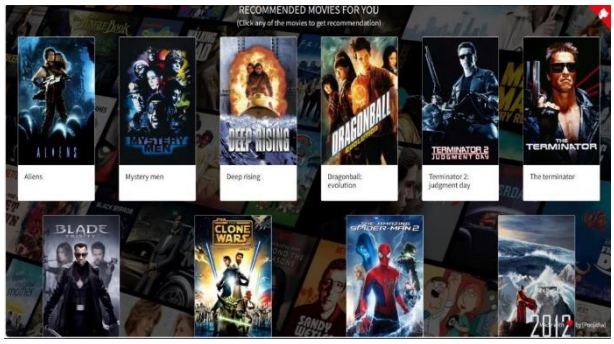

Fig: Recommended Movies based on Searched Movie

VII.CONCLUSION:

The development of sophisticated content-based movie recommender system that integrates with the process of sentiment analysis is a considerable and major advancement over the realm of personalized user experiences. Since the TMDb API is a fully comprehensive source of movie data and also due to the aid of advanced web scraping techniques that carefully extract user reviews from the IMDb dataset, the suggested movie recommendation system can provide dynamic, realtime, and rather precise recommendations based on the user's preferences. Cosine similarity will introduce relevance guarantees for the suggested results. It does this through proper assessment of the textual similarities that exist between the movies. This kind of sentiment analysis highly and considerably enhances the ability of such a system in terms of effectively capturing and interpreting emotional feedback expressed through user reviews, bringing with it a most important subjective dimension in the recommendations provided

VIII. FUTURE ENHANCEMENTS

- Add Movie Trailers or Posters - Embed YouTube trailers or movie posters using IMDb API or TMDb API.
- User Interface Improvement - Implement `Streamlit`, `Flask`, or `Tkinter` to create a basic GUI for user interaction.
- Genre-Based Filtering - Include dropdown or checkbox to filter movies based on genres (action, comedy, drama, etc.).
- Show Similar Movies in Grid View - Show results with images in grid or carousel view for a better appearance.
- Sort Results by IMDB Rating or Release Year - Allow users to decide how to sort recommended films.
- Watchlist Feature - Enable users to add films to a "Watch Later" list in a CSV or miniature database.
- Autocomplete Search Bar - Implement a search bar that auto-completes movie titles as the user inputs.
- Hybrid Recommendation System - Blend content-based with collaborative filtering with sophisticated models (`Surprise`, `Light FM`, etc.).
- Semantic Similarity Using NLP - Employ TF-IDF, BERT, or Word2Vec to compare movie synopses and match on meaning, rather than keywords.
- Content-Based User Profiles - Construct content vectors for users from several ratings and preferences.

IX. REFERENCES:

[1] Deshmukh, P., et al., “Movie Recommendation System using Machine Learning”, ICAIS Conference Proceedings, 2021.
[2] Thakur, A., et al., “Recommendation System using Machine Learning”, International Conference on Smart Power and Clean Energy (ISPCC), 2021.
[3] Jayasankar, K., et al., “Movie Recommender System Using Content-based and Collaborative Filtering”, International Conference on Computer Communication and Informatics (ICCCIS), 2022.
[4] Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18 (11), 613-620. Karypis, G. (2001). In the conference proceedings of the tenth international conference dedicated to Information and knowledge management, an author describes item-based collaborative filtering algorithms with their efficiency and effectiveness in many applications.
[5] Karypis, G. (2001). In the conference proceedings of the tenth international conference dedicated to Information and knowledge management, an author describes item-based collaborative filtering algorithms with their efficiency and effectiveness in many applications.
[6] The Movie Database (TMDb). API Documentation. Available from: https://www.themoviedb.org/documentation/a pi.
[7] Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In Recommender Systems Handbook (pp. 1-35).
[8] Ricci, F., and Sebastiani, F. (2011). In their seminal work, they provide a very detailed survey of recommender systems, exploring the different algorithms used and the wide range of applications for which these systems can be used. This important contribution is published by Springer Science & Business Media.
[9] Nagaraj M. Lutimath, Chandra Mouli, B. K. Byre Gowda, K. Sunitha, “Prediction of Heart Disease Using Hybrid Machine Learning Technique”, Springer Book Series "Transactions on Computer Systems and Networks”, with title "Paradigms of Smart and Intelligent Communication, 5G and Beyond", Springer Nature, Singapore,2023, pp 277– 293.

[10] Nagaraj M. Lutimath, Divya H. N, Ravi Gatti, Vasudeva G, Byregowda B. K, "Prediction of Heart Disease Using Hybrid Gaussian Naïve Bayes Technique", Elsevier SSRN Series, Available at SSRN: https://ssrn.com/abstract=4602168, Oct 2023.

[11] Reddy, S.S. and Nandini, C. " A comprehensive Review of Machine Learning Approaches in Livestock Health Monitoring. Journal of Big data technology and Business Analytics e ISSN: 25837834, vol 3, Issue 3 (Sep – Dec,2024)pp11-19)

[12] In the year 2007, Bell, R. M., and Koren, Y. presented a seminal paper where scalable collaborative filtering techniques were used through joint matrix factorization. It was specifically designed to be used in recommender systems. This work appeared in the proceedings of the major 2007 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, where it appeared on pages 43 to 52.

[13] Aggarwal, C. C. (2016). Content-based recommender systems. In Recommender Systems (pp. 139-166). Springer.

[14] Zhao, T., Zhang, Y., & Tang, J. (2020). Featurelevel deeper attention network for sequential recommendation. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), 2020. Introduces a deep learning framework improving sequential recommendations using feature-level attention.

[15] Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., & Jiang, P. (2020). AutoInt: Automatic feature interaction learning via self-attentive neural networks. Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM), 2020. Proposes a self-attentive neural network to automatically learn feature interactions in recommender systems.