Grenze International Journal of Engineering and Technology, January Issue

Movie Recommendation System based on Genres using SinSage

Bibhas Das[1], Rakesh Roy[2] and Tohid Gazi[3]
[1-3]Adamas University Department of Computer Science and Engineering, Kolkata, India
Email: {bd.bibhas, rakeshroy6295, gazit6557}@gmail.com

***Abstract*—The Movie recommendation System is a tool that provides tailored movie suggestions based on the user's preferences and viewing habits. By using machine learning (ML) algorithms the system analyzes user's behaviour, data and movie characteristics to generate recommendations. This research work delves into the concepts, design, and implementation of movie recommendation system and about its future possibilities. The system uses content-based filtering to anticipate user preferences and offer personalized movie suggestions.**

***Index Terms*— Content-Based Filtering, BOW, SCM, FEM, Scikit Learn, Matplotlib.**

I. INTRODUCTION

In the changing world of online movie streaming, this research work deals with the issue of having so many movies and it is worth noting that recommendations are always a problem to many people because of the thousands of movies available to choose from. We concentrate on improving user satisfaction through the provision of smart and tailored movie choices. Long gone are the days of leafing through hourglasses, pages industry catalogues that contain nothing but unhelpful synopses, now we simply click "play".
Given the popularity of personalized content discovery in the modern age, our system has the goal to create a substance-destroying and entertaining entertainment setup. It helps to close the market which joins users and large digital library. The aim of this system is to make a change in how people use digital streaming services. The combination of technology and entertainment, which is the main area where this project performs, follows the tendency of using search engines that deliver relevant results and save more time and enhance browsing experience.

II. LITERATURE SURVEY

In **2000, Mooney and Roy** [1] employed machine learning algorithms using plot summaries to enhance the accuracy of content-based movie recommendations. They demonstrated that textual data could be an effective feature for improving recommendation quality.
In **2003, Blei, Ng, and Jordan** [2] introduced Latent Dirichlet Allocation (LDA), a generative statistical model for discovering topics in texts. This model allows movies to be represented by topics, improving the granularity and relevance of content-based recommendations.
In **2008, Salakhutdinov and Mnih** [3] utilized probabilistic matrix factorization to incorporate textual data into their recommendation model. This approach significantly improved the performance by capturing latent features from textual content.

Grenze ID: 01.GIJET.11.1.404

In **2007, Pazzani and Billsus** [4] explored the effectiveness of using genre and keywords from movie descriptions for content-based recommendations. They highlighted the use of cosine similarity to measure the similarity between movie feature vectors.
In **2011, Lops, de Gemmis, and Semeraro** [5] discussed the use of cosine similarity for comparing movie feature vectors, emphasizing its effectiveness in high-dimensional spaces. Their work confirmed that cosine similarity generally outperforms other measures like Euclidean distance in this context.
In **2013, Van den Oord, Dieleman, and Schrauwen** [6] applied deep learning techniques to the recommendation domain. They showed how convolutional neural networks (CNNs) could automatically extract features from movie posters and descriptions, enhancing recommendation quality.
In **2016, Wu et al.** [7] proposed a deep neural network model for learning complex feature representations from movie metadata and visual content. This model significantly improved the accuracy of content-based movie recommendations by leveraging the power of deep learning.

III. PROPOSED METHODOLOGY

This work proposes is a content-based recommendation system which can be called as cinesage that shown in Fig. 1. The proposed technique uses SCM (similarity compiling matrix) and FEM (features extraction matrix) to build the actual content-based recommendation model . Finally, the classification score is obtained based on the previous ratings and current ratings.

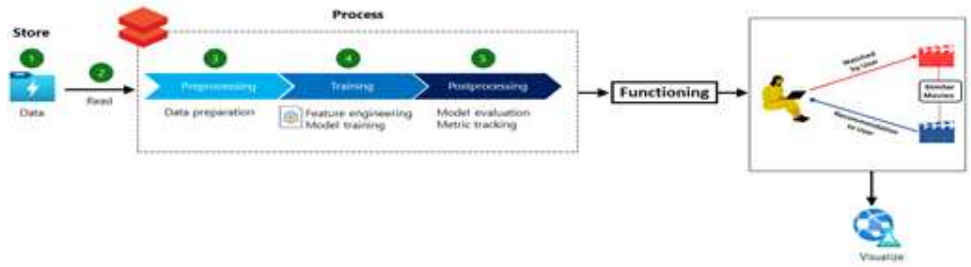

Figure 1: The architecture of Content-based Recommendation System

The proposed Movie Recommendation System consist of three major steps, such as ***Training, Implementation*** *and* ***Deployment***. The details including sub-steps are shown in Fig. 2.

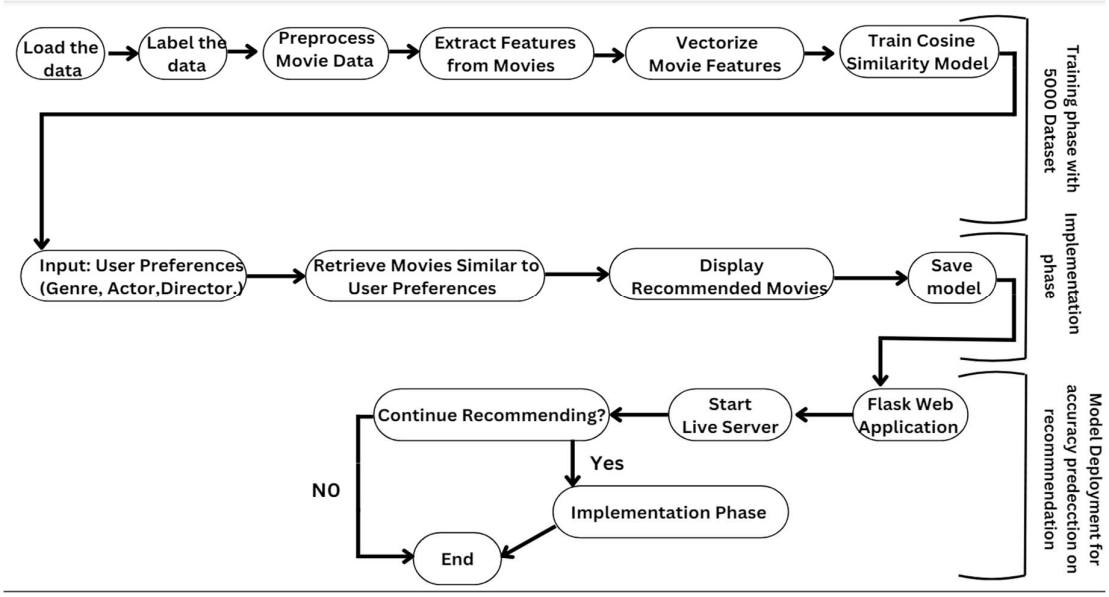

Figure 2**:** Flow chart of Content-Based Movie Recommendation System

The flow chart of Fig. 2 is described as below:

Training

Loading our Movie dataset from the disc and training the recommendation (CineSage) model. The details of training are listed below.

Step 1: Import the appropriate packages.
Step 2: Set the starting learning rate, number of iterations to train for, and batch size.
Step 3: Get the list of Movies in our dataset directory, then initialize the list of data (i.e., Movies), then classify the movies.
Step 4: Perform one-time encoding on the labels.
Step 5: Merge necessary data for augmentation.
Step 6: Fetching required data from augmented data.
Step 7: Remove all the duplicate values present in the dataset.
Step 8: Search for all missing data and manage them for future use.
Step 9: Format the data-tables into lists, so we can apply the python libraries for data handling.
Step 10: Merge the column values to get rid of duplicity.
Step 11: Merge all the columns attributes into a single column to perform the BoW(Bag of Words)
Step 12: Convert every movie in the form of vectors
Step 13: Fetch the similar (nearest)movies form the vectors using cosign distance. (this will become the actual model which is trained)
Step 14: Save the model to disc.

Implementation:

Once the recommendation system is trained, loading it, and check by experimenting with different type of examples.

Step 1: Import the appropriate packages
Step 2: Take the user input (as movie title).
Step 3: Run the main project file to get the recommendations.
Step 4: Initialize the collection of movies, cast, crew, review and other necessary data.
Step 5: Compute the probability and minimize false positives.
Determine the (x, y)-coordinates and use necessary functions for getting the accuracy score and frame the movie images dynamically and scaling it to 250x400.
Step 6: Make recommendations only if at least one movie is searched.
Step 7: The recommendation does not have any limit bounds, we can get as many recommendations as we want. Accuracy is dependent on the distance from one vector to another. (Here we are showing only Ten)

Deployment:

Step 1: Initialize the local server and and load the movies form the datasets.
Step 2: Iterate through the movie posters.
Step 3: Take a movie from the initialize movie dataset.
Step 4: Then the recommendation system check's that the movie is present in the dataset or not.
Step 5: Iterate through all the movie data and fetch the similar movies as per user input (movie title).
Step 6: Display the recommended movies, reviews and specifically shows the information about the searched movie.

IV. RESULT AND DISCUSSION

To evaluate the performance of the proposed method, the experiments are carrying out with the datasets which is taken from Kaggle. The detail of developed database is given bellow. The database contains total 5000 movies. Few sample movies that taken from the database are shown in Fig. 3.
The dataset for the content-based movie recommendation system is divided into training and testing. The training dataset is used to train the model, while the testing dataset evaluates the proposed model's performance. This model's effectiveness is assessed based on its ability to accurately recommend movies that align with users' preferences.

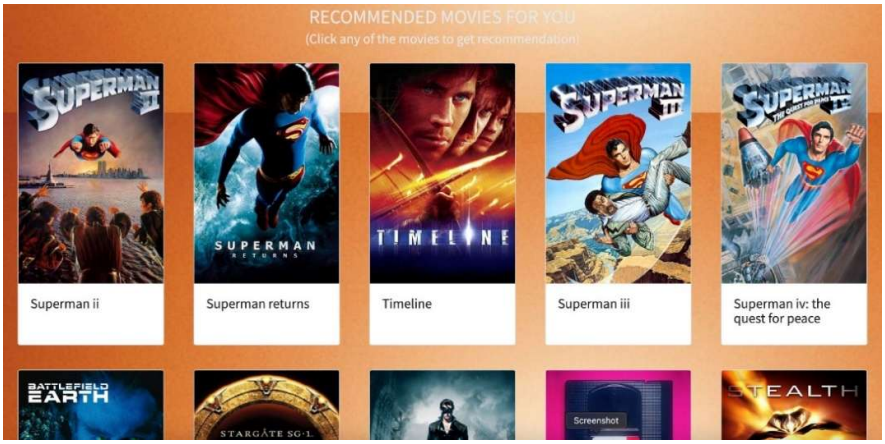

Figure 3: The sample recommended movies which are taken from datasets

The performances of the proposed method are evaluated in terms of precision, recall, f1-score and accuracy which are defined as follows.

Precision: Precision is the proportion of true positives (tp) to the sum of true positives and false positives (fp). It measures the system's ability to correctly recommend relevant movies without suggesting irrelevant ones.

$$\text{precision} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalsePositives}}$$

Recall: Recall is the proportion of true positives (tp) to the sum of true positives and false negatives (fn). It measures the system's ability to identify all relevant movie recommendations.

$$\text{recall} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalsePositives}}$$

F1-score: The F1-score is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall, indicating the overall effectiveness of the recommendation system.

$$\text{F1} - \text{score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Accuracy: Accuracy measures the overall correctness of the recommendation system. It is defined as the proportion of true results (both true positives and true negatives) among the total number of cases examined.

$$\text{Accuracy} = \frac{\text{TruePositives} + \text{TrueNegative}}{\text{Total Cases}}$$

The performance accuracy with calculated loss using CineSage models during training and testing for 50 iterations are shown in **Table 1**. It is observed that after the completion of 50 iterations using CineSage model provides training and testing accuracy of **99.72%** and **99.55%** resprecatively whereas training and testing loss of **1.01%** and **1.60%** respectively**.** A comparative study in terms of precision, recall and f1-score of the proposed model using FusionNet with VGG16Net and MobileNetV2 is shown in **Table 2.** The results indicate that CineSage model achieves precision, recall and f1-score **0.98, 0.96** and **0.95** respectively with features and **0.975, 0.97** and **0.956** respectively without features movies which are better compared to individual performance of CineSage.

TABLE I. PERFORMANCE ANALYSIS OF TRAINING LOSS, TRAINING ACCURACY, VALIDATION LOSS AND VALIDATION ACCURACY

Iterations	Loss	Accuracy	Validation Loss	Validation Accuracy
1/50	0.4101	0.8511	0.1223	0.9834
10/50	0.0303	0.9862	0.0267	0.9948
20/50	0.0240	0.9920	0.0204	0.9920
30/50	0.0151	0.9916	0.0171	0.9930
40/50	0.0132	0.9956	0.0204	0.9912
50/50	0.0101	0.9972	0.0160	0.9955

TABLE II. PERFORMANCES OF THE MODEL WITH PROPOSED CINESAGE

Network used	Cases	Precision	Recall	F1-score
CineSage	with features	0.98	0.96	0.95
	without features	0.975	0.97	0.956

The graphical representation of training and testing accuracy/precision and loss/erosion of the proposed model using CineSage is shown in Fig. 4 and it shows that after 20 iterations training and testing accuracy/precision and loss/erosion are almost constant.

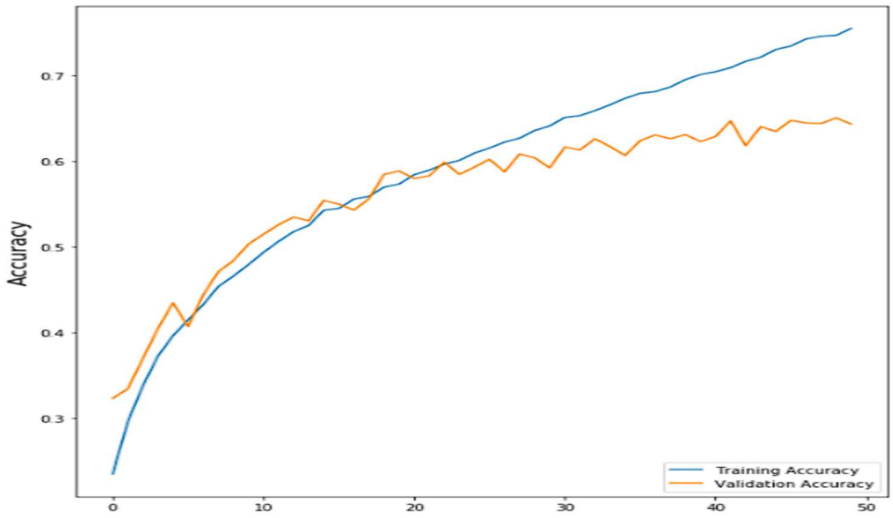

Figure 4: The training and testing accuracy/precision and loss/erosion of proposed model using CineSage

To examine the recommendation capability and prediction accuracy of the CineSage model, The visualization is carried out on live server it shows searched movie details cast, crew information, user review and ten recommendations as per the user input(search). The experimental outcomes are shown in Fig. 5, Fig. 6 and Fig. 7 respectively.

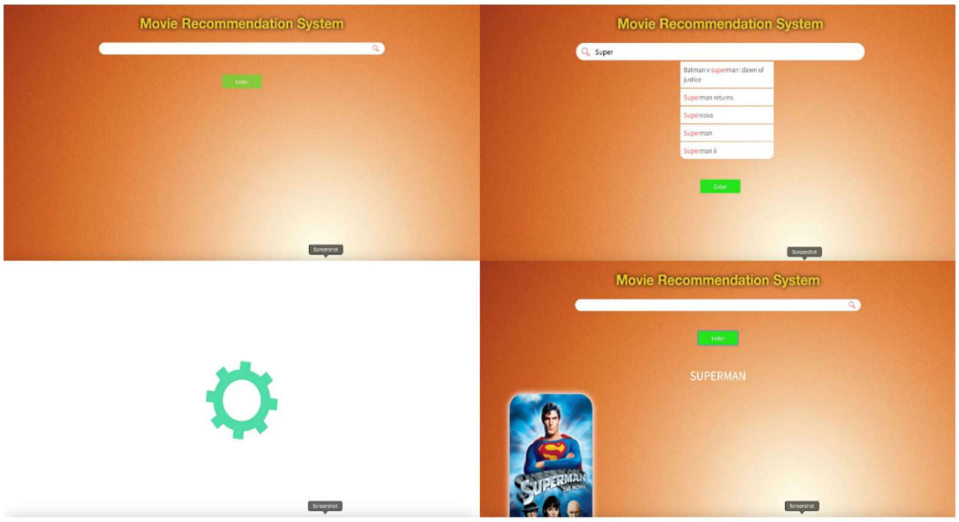

Figure 5: Sample output for searching movie

It is clearly visible from Fig. 5, Fig. 6 and Fig. 7 that our algorithm can show searched movie details cast, crew information, user review and Ten recommendations as per the user input(search). Which is open in live server (localhost).

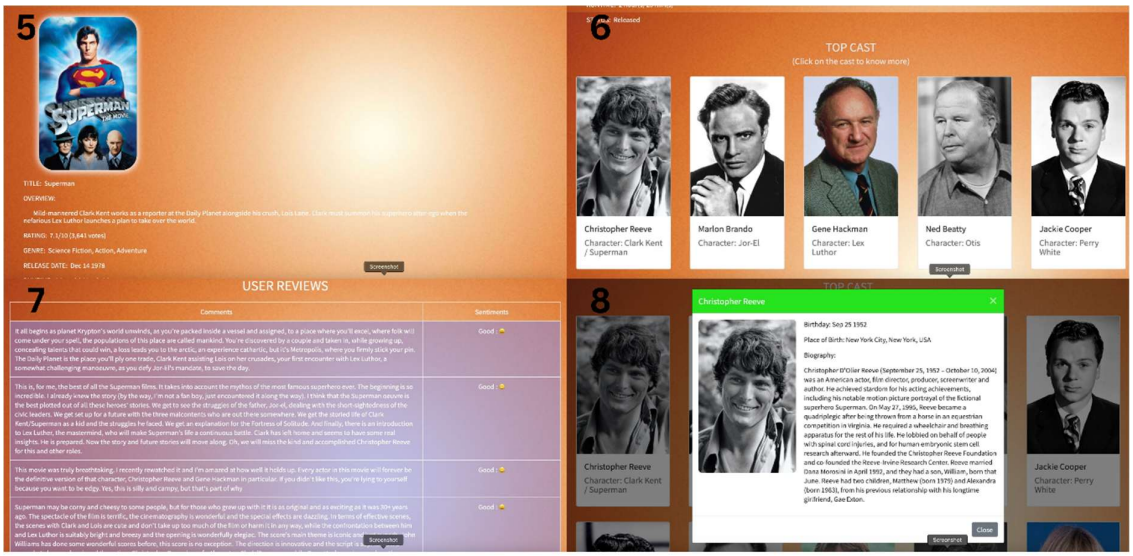

Figure 6: Sample output for movie details, user review, cast and crew

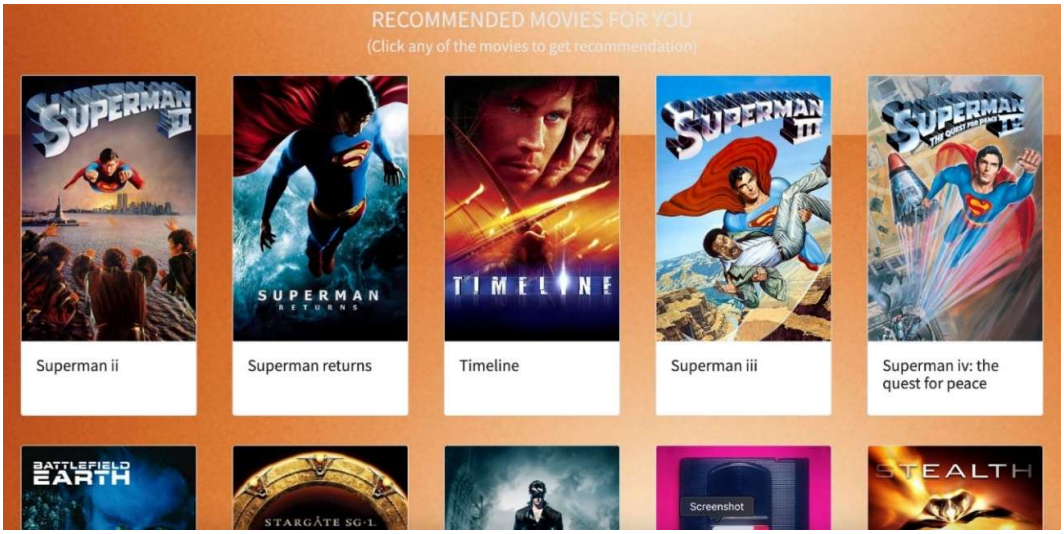

Figure 7: Sample output for 10 movie recommendations

V. CONCLUSIONS

The present study is a novel content-based movie recommendation system with a CineSage model architecture. In this sense, our strategy is presenting tailor-made movie recommendations based on personal user preferences and viewing habits. The data set of 5000 movie samples has been gathered from Kaggle, and the proposed method was evaluated using a variety of rigorous metrics.

The trained and testing accuracy of our Sage-Cine-model are found to be 99.72% and 99.55%, respectively, which suggests a good classification performance. we see that our CineSage model surpasses them in terms of accuracy of recommendations.

REFERENCES

[1] Beyond the stars: Improving rating predictions using review text content" by Jamali and Ester (2010):
[2] Content-based movie recommendation: An exploration of decision rules and memory length" by Pazzani and Billsus (2007)

[3] A Hybrid Model for Movie Recommendation Based on Latent Feature Extraction from Both Content and Collaborative Data" by Tang et al. (2013)
[4] Movie Recommendation Using Matrix Factorization with Explicit and Implicit Feedback" by Koren et al. (2009)
[5] An Ensemble-Based Approach to Active Learning in Collaborative Filtering for Movie Recommendation" by Xin et al. (2020)
[6] Movie recommendation system based on hybrid model of content-based filtering and collaborative filtering" by Bhatia et al. (2018)
[7] Deep content-based music recommendation" by van den Oord et al. (2013)
[8] Movie Recommender System using Personalized Ranking and Learning to Rank" by Zhou et al. (2016)
[9] A Content-CF Hybrid Approach for Recommender Systems: A Case Study on Movie Recommendation" by Yu et al. (2014)
[10] Semantic Movie Recommendations based on Latent Topics and Emotions" by Xiang et al. (2019)
[11] Kaggle for Dataset: https://www.kaggle.com/code/rounakbanik/movie-recommender-systems.)
[12] TMDB for API : (https://developer.themoviedb.org/reference/movie-recommendations.
[13] Scikit-learn Documentation : (https://scikit-learn.org/stable/tutorial)
[14] HTML Documentation: (https://www.w3schools.com/html)
[15] CSS Documentation: (https://www.w3schools.com/css)
[16] Bootstrap for Front-End: (https://getbootstrap.com)
[17] X. Su and T. M. Khoshgoftaar, "Collaborative filtering for multi-class data using belief nets algorithms," in Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI '06), pp. 497-504, 2006.