RecommendRift: a leap forward in user experience with transfer learning on netflix recommendations

Surabhi Anuradha[1,2], Pothabathula Naga Jyothi[3], Surabhi Sivakumar[4], Martha Sheshikala[5]
[1]School of Computer Science and Artificial Intelligence, SR University, Warangal, India
[2]Department of Computer Science and Engineering (AIML), Keshav Memorial Institute of Technology, Hyderabad, India
[3]Department of Computer Science and Engineering, GITAM School of Technology, GITAM, Visakhapatnam, India
[4]Department of Chemistry, Anil Neerukonda Institute of Technology and Sciences, Visakhapatnam, India
[5]School of Computer Science and Artificial Intelligence, SR University, Warangal, India

Article Info

Article history:

Received Jan 7, 2024
Revised Aug 6, 2024
Accepted Aug 11, 2024

Keywords:

A/B testing
Click-through rate
Cold start problem
Content-based filtering
Cumulative gain
Word2Vec

ABSTRACT

In today’s fast-paced lifestyle, streaming movies and series on platforms like Netflix is a valued recreational activity. However, users often spend considerable time searching for the right content and receive irrelevant recommendations, particularly when facing the “cold start problem” for new users. This challenge arises from existing recommender systems relying on factors like casting, title, and genre, using term frequency-inverse document frequency (TF-IDF) for vectorization, which prioritizes word frequency over semantic meaning. To address this, an innovative recommender system considering not only casting, title, and genre but also the short description of movies or shows is proposed in this study. Leveraging Word2Vec embedding for semantic relationships, this system offers recommendations aligning better with user preferences. Evaluation metrics including precision, mean average precision (MAP), discounted cumulative gain (DCG), and ideal cumulative gain (IDCG) demonstrate the system’s effectiveness, achieving a normalized DCG (NDCG)@10 of 0.956. A/B testing shows an improved click-through rate (CTR) of recommendations, showcasing enhanced streaming experience.

Corresponding Author:

Surabhi Anuradha
School of Computer Science and Artificial Intelligence, SR University
Warangal, India
Email: anuradha@kmit.in

1. INTRODUCTION

Nowadays, entertainment plays a significant role in the day-to-day life of humans. Movies and songs are the source of entertainment. The challenge lies in extracting information from vast amounts of data, given that the consumer wants results that are highly relevant, accurate, and responsive. In order to operate with a high degree of user choice, the model must employ several strategies for choosing pertinent attributes, filtering data, and presenting the top list according to user preferences. The recommender system solves the problem of analysing the content and outcomes of the tailored content to the user [1]. There is a tremendous increase of data on various platforms like YouTube, Spotify, Netflix, and all other e-commerce sites [2]. There are mixed proven results by the recommender system of Amazon, which increased its revenue by 30% and best buy by 23.7%, YouTube by 70 percent on videos and 60 percent in views, Netflix by 6.7% every year with 220.6 million subscribers worldwide by the implementation of recommendation algorithms [3]. More than 71% of e-commerce sites recommend products on their homepage, which has helped them increase engagement, conversions, and revenue. While recommendations contributed to only 7% of visits, they accounted for 26% of income [4].

Netflix has an enormous selection of series and movies, so it might be difficult to decide which to watch next. Building a recommendation system that makes Netflix program recommendations based on user taste simplifies this procedure. The most typical uses of recommendation systems are in the domains like product evaluations, movies, and documents to facilitate content discovery. Movie data from sites like Netflix, MovieLens, and Prime is a significant source of e-content. A total of 4,500 films are produced globally, producing roughly 9,000 hours of content, according to internet movie database (IMDb).

There is demand for the recommendation system to ease access and handle this massive amount of data to facilitate viewers' interests accurately. The recommender system is developed based on essential fundamental aspects, which are content-based and collaborative filtering. Users determine content-based recommendations through their preferences for products, movies, videos, and audio content. If user A likes a product X, the content-based approach will find similar products to X and will be recommended to user A. The collaborative approach suggests that if user A's profile is similar to user B's, when user B likes a movie X, then user A is recommended X, and vice-versa follows [5].

This study investigated the effects of capturing semantic relationships between words in the descriptions of movies and shows for recommendation system. While previous research has investigated the effects of TF-IDF vectorization on attributes such as title, genre, and casting information, it has largely overlooked the descriptions of movies and shows. These studies primarily focused on word frequency within these attributes and did not address the semantic relationships between words in the descriptions. Our research aims to fill this gap by incorporating the description as a key input feature and utilizing Word2Vec embedding techniques for vectorization. The key objectives of our research are:

- To incorporate descriptions as a key input feature: we aim to enhance recommendation accuracy by including the semantic content of movie and show descriptions, which has been largely ignored in previous studies.
- To utilize Word2Vec embedding techniques: our study focuses on applying Word2Vec embeddings to capture the semantic relationships between words in descriptions, addressing the limitations of TF-IDF, which primarily measures word frequency and fails to account for the context and meaning of words.

The rest of the paper is organized into the following sections. Section 2 presents a summary of prior investigations concerning the recommender systems. The pipeline architecture and algorithm of the proposed approach, the data set utilized for the experiments, and the performance evaluation criteria employed are all covered under section 3. In section 4, the experimental results were examined and interpreted, and it was finally wrapped up with future enhancements to the work in section 5.

2. LITERATURE REVIEW

Roy and Dutta [6] reviewed recommender systems across movies, books, and products, addressing limitations such as cold start, sparsity, scalability, and serendipity. They examined content-based, collaborative, and hybrid models, noting content-based systems' quick adaptation but limited feature depth, and collaborative systems' privacy issues but lack of item knowledge. Their review of 60 papers from 2011 to 2021 highlighted the evolution and application of these techniques. Zhang *et al.* [7] developed a personalized movie recommendation system using weighted KM slope-VU, which applies k-means clustering on user profiles and calculates virtual scores for item evaluation. The system, implemented in a web application, uses root mean square error (RMSE) for rating prediction but showed slightly lower performance than singular value decomposition (SVD)++ due to outdated movies and reliance on virtual rather than real-time user data.

Kumar [8] discussed comparative study of movie recommendation systems based on the rating given by the users. The collaborative filtering mechanism is used, and the similarity index is calculated on the feature's importance score on user-user similarity. The model was developed on 13 handcrafted parts and prediction is made using the extreme gradient boosting (XGBoost) regressor. The results are compared between SVD+, k-nearest neighbor (KNN) with SVD and XGBoost regressor machine learning (ML) algorithms. The error function is suggested by the author to calculate the error between the rating given by the user and the global average of all rating biases to another user. L2 regularization is meant to observe different ratings like global average, movie bias, and user bias terms on a large dataset. The pattern for some users may be higher to lower than the global average, or it may be vice versa to minimize the error for accurate results. The results obtained on a partial dataset with fewer dimensional recommenders are attempting to catch the preferences and inclinations.

Havolli *et al.* [9] developed a content-based recommendation system using term frequency-inverse document frequency (TF-IDF) for text vectorization and the Adamic Adar measure to assess movie similarities based on features like actors and directors. While TF-IDF calculates term relevance, the Adamic

Adar measure’s limitation is its focus on undirected graphs. Future work aims to integrate artificial intelligent (AI) and ML to enhance similarity detection in movie recommendations. Tai *et al.* [10] developed an RS based on the KNN ML algorithm; the design created involves candidate generation, content-based filtration, and ranking stages. The collected videos from the YouTube corpus undergo content-based filtration based on the previously watched video. The model recommends the following video. Data usage is framed in a utility matrix w.r.t to significant items and user choices. The degree of preference relationship is calculated based on the user items, the likes and dislikes. The ranking works on module matching. It selects a few things from a list of consumer appeals. The ranking system retrieves ranking objects based on item properties, and user-development needs to be revised to scale for extensive data as mentioned by Suresh *et al.* [11].

Sirisha *et al.* [12] proposed a content-based movie recommendation method using WordNet and topic modeling to categorize films by plot aspects, with precision evaluated on IMDb’s Telugu movies dataset and suggestions for future research. Cagatayli and Celebi [13] explored integrating users’ BIG-five personality scores into movie recommendations but found no significant correlation between these scores and genre preferences, challenging existing assumptions. Rai *et al.* [14] compared algorithms such as KNN, collaborative filtering, and content-based filtering to enhance recommendation accuracy, proposing a hybrid approach for improved efficiency and scalability.

Roy and Shirazi [15] explored matrix factorization (MF) and deep neural network (DNN) models for recommender systems, highlighting the potential of using indirect features, designing systems for unregistered users, and optimizing explicit and implicit feedback integration as future research directions. Falk [16] “practical recommender systems” emphasizes content-based filtering for generating recommendations based on item metadata, rather than subjective opinions. Additionally, Raza and Ding [17], Vu and Le [18] conducted surveys on current trends and methodologies in context-aware recommender systems.

3. METHOD

3.1. Proposed model

Given a user title preference, the proposed algorithm searches for genres and description similarities to discover matching shows or movies. This process involves searching for content that aligns with user perception of similarity and then creating a personalized recommendation approach based on content attributes as described in Figure 1. In our quest to construct a highly efficient Netflix movie recommendation system, our model adopted an innovative strategy by harnessing the power of a pre-trained Word2Vec model. The goal of this technique is to map high-dimensional sparse vectors to low-dimensional dense vectors and calculate user or item similarity. For example, Valcarece suggested the “prefs2vec” model to learn the user and item embedding, which is inspired by the bag-of-words (CBOW) in “Word2Vec” [19], [20]. This method has been shown to be beneficial in lowering computing complexity, resulting in a more efficient and faster recommender system. This cutting-edge approach takes inspiration from the principles of transfer learning, enabling us to delve into and evaluate the intricate interconnections that underlie movie titles, genres, and the comprehensive textual descriptions associated with each movie.

By leveraging the knowledge and semantic understanding embedded within the pre-trained Word2Vec model, trained initially on extensive text corpora, the proposed model is empowered to grasp the nuances of movie-related language and semantics. This sophisticated technique significantly enhances the recommendation process and capitalizes on the depth of information available, providing users with more personalized and context-aware movie suggestions [21], [22].

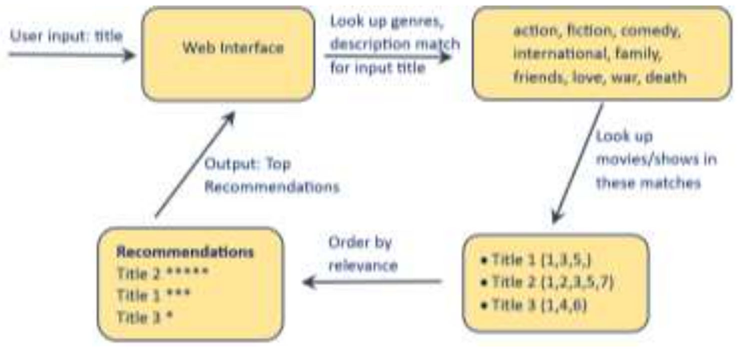
User input: title
Web Interface
Look up genres, description match for input title
action, fiction, comedy, international, family, friends, love, war, death
Look up movies/shows in these matches
• Title 1 (1,3,5,)
• Title 2 (1,2,3,5,7)
• Title 3 (1,4,6)
Order by relevance
Recommendations
Title 2 *****
Title 1 ***
Title 3 *
Output: Top Recommendations

Figure 1. Proposed recommender system pipeline

3.2. Dataset utilized

A dataset of nearly 9,000 unique Netflix titles was used to generate context-aware recommendations for users. The attributes ‘title,’ ‘listed_in,’ and ‘description’ were taken as input features for building the recommender algorithm. These features were tokenized using a transfer learning mechanism during pre-processing phase and then cosine similarity was measured to calculate the similarity scores for making recommendations. Consider Algorithm 1.

Algorithm 1. RecommendRift (title)

```
Input: title
Output: list of titles recommended
Functions:
ExtractVec (title): extracts word vectors for the specified title
CalcSim (v1, v2): calculates cosine similarity score
FilterThrsh (scores, threshold): filters similarity based on given threshold
SortRecomm (filtered_scores): sorts filtered scores to retrieve recommendations
    1. $V_{title}$ ← ExtractVec (title)
    2. scores ← [ ]
    3. for each sample in dataset do:
        a. $v_{sample}$ ← ExtractVec (sample. title)
        b. score ← CalcSim ($v_{title}$, $v_{sample}$)
        c. scores.append(sample, score)
    4. filtered_scores ← FilterThresh (scores, threshold)
    5. recommends ← SortRecomm (filtered_scores)
    6. return recommends
```

3.3. Evaluation metrics

3.3.1. Precision

Precision@K is a metric used to evaluate the quality of recommendations in a content-based recommendation system [16], [23], [24]. It measures the proportion of relevant items in the top K recommendations as defined in (1).

$$Precision@K = \frac{Number\ of\ Relevant\ Items\ in\ Top\ K\ recommendations}{K} \quad (1)$$

3.3.2. Mean average precision

Mean average precision (MAP) is a metric used to assess the quality of a ranking. It involves calculating precision at various cutoff points, starting from 1 and going up to the total number of recommended items, often denoted as ‘k.’ To provide a more comprehensive evaluation, the process continues by averaging the precisions over the first item, then the next two, and so on until it covers all recommended items (up to ‘k’), which is defined in (2).

$$AP = \frac{1}{r}\sum_{k=1}^{K}(Precision@k * Relevant@k) \quad (2)$$
$$rel(k) = \{1, \quad if\ item\ at\ k^{th}\ rank\ is\ relevant$$
$$0, \quad otherwise$$
$$where, r - total\ relevant\ items$$

This procedure is applied to each recommendation individually. To evaluate a recommender system, you can then compute the mean of these individual average precisions across all recommendations as defined in (3).

$$MAP = \frac{\sum AP@k}{Total\ Queries} \quad (3)$$

It evaluates the quality of a ranking by calculating the precision at different levels, starting from the first item and progressively considering more items up to the total number of recommended items (often denoted as ‘k’). To extend this evaluation further, MAP computes the average of these precisions over the first item, then the next two items, and so on until it reaches the last item in the ranking. This provides a comprehensive measure of the ranking’s overall quality.

3.3.3. Discounted cumulative gain and normalized cumulative gain

The MAP merely considers relevance, but the discounted cumulative gain (DCG) considers levels of relevance. To calculate DCG, assign a relevancy score to each item [2]. In the case of a recommender system, it might be the item's expected rating or profit. Relevance is often referred to as benefit in this context. It could also be referred to as the discounted cumulative relevance. The relevance is discounted by the position in the list, and the relevance scores are combined together to calculate the DCG [16] as shown in (4).

$$DCG = \sum_{rank=1}^{k} \frac{Gains}{log_2\,(rank+1)} \tag{4}$$

Where, gains: the true_scores at each rank, which represent the actual relevance or quality of the recommendations. log2 (rank+1): the logarithm base 2 of the rank (adjusted by 1 to avoid division by zero). k is the rank at which we are evaluating the results. Once the result of DCG is obtained, normalized cumulative gain (NDCG), which is the normalized DCG can be computed using the formula presented in the (5).

$$NDCG = \frac{DCG}{IDCG} \tag{5}$$

Where IDCG is the Ideal DCG, which is the DCG that would be achieved if all the True_scores were perfectly ranked in descending order [16], [25], [26].

4. RESULTS AND DISCUSSION

During pre-processing, a pre-trained Word2Vec model is used to transform the processed tokens of input features into vectors, and then a vector similarity method is used to identify the most similar words in the dataset. A sample of the most similar words identified by the Word2Vec model is presented in Table 1. Then in the next phase, given a title, the proposed algorithm filters the input data and creates a matrix of relevant tokens. It then computes similarity scores using the Word2Vec model for category, description, and title attributes against a matrix of all shows. Recommendations with a category similarity above 0.85 are aggregated into a final score, sorted, and the top 10 are returned.

Two test cases were taken for interpreting the results of the proposed algorithm. Table 2 presents a list of top recommendations for the title ‘Black Panther,’ along with their respective similarity scores in various categories: ‘score_title,’ ‘score_category,’ ‘score_description,’ and the final aggregated score. From these results, we can say that the top 1 recommendation ‘Chappie’ stands out with a noteworthy ‘score_title’ of 0.112706, indicating a significant title overlap with ‘Black Panther.’ Furthermore, it achieved a perfect ‘score_category’ of 1.000000, signifying that both titles fall within the same genre or category. The ‘score_description’ of 0.621946 also implies substantial textual similarity in their descriptions. As a result, ‘Chappie’ secured the top position with the highest ‘final_score’ of 1.734651. The table provides a full overview of the remaining entries in this sequence and a varied range of recommendations based on different elements of similarity.

Precision and MAP are the evaluation metrics employed to assess the effectiveness of the recommendations produced. Model performance with two query groups, ‘Black Panther’ and ‘Article 15’ are exhibited in Table 3. MAP is determined by averaging the AP values computed for both queries: MAP=(APBlack Panther+APArticle 15)/2=(0.95+0.9)/2=0.925. The MAP score of 0.925 suggests that, on average, the recommender system is quite effective in returning relevant results for the two queries, “Black Panther” and “Article 15.” A MAP score of 1 indicates perfect performance, so a MAP score of 0.925 is quite good. This means that, for the given queries, most of the relevant documents are ranked highly, and the precision at different cutoffs (P@k) is consistently high. The calculation of DCG, IDCG, and NDCG values for the top 10 ranks was used to evaluate the effectiveness of the suggestions produced by the suggested recommender system. Evaluation results for recommendations across two query groups are presented in Table 4. The true_score and true_ranks were derived as an average of scores gathered from various online sources such as Google, IMDB, Netflix, and RottenTomatoes.

Table 1. Most similar words generated for the token ‘thriller’

Sl. No.	Similar_Word	Similarity_Score
1	Thriller	1.00000
2	Thrillers	0.78241
3	Suspense	0.74405
4	Psychological	0.72550
5	Spy	0.71650
6	Supernatural	0.65463
7	Drama	0.64413
8	Potboiler	0.64344

Table 2. Similarity scores for top 10 recommendations related to 'Black Panther'

Sl. No.	Recommendation	Score_Title	Score_Category	Score_Description	Final_Score
1	Chappie	0.11271	1.00000	0.62195	1.73465
2	Clash of the Titans	0.04708	1.00000	0.67404	1.72112
3	Green Lantern	0.11550	1.00000	0.60476	1.72026
4	Halo: The Fall of Reach	0.05258	1.00000	0.65536	1.70794
5	Stargate	0.10318	1.00000	0.59298	1.69616
6	Illang: The Wolf Brigade	0.21976	0.90278	0.56643	1.68897
7	Men in Black	0.29745	0.88576	0.50308	1.68628
8	DC's Legends of Tomorrow	0.08603	0.90935	0.68981	1.68519
9	Superman returns	0.09047	1.00000	0.59395	1.68442
10	Season of the Witch	0.12699	1.00000	0.55690	1.68389

Table 3. Precision and average precision calculated for the sample queries

	Precision@k				Average precision@k			
Query	P@1	P@3	P@5	P@ 10	AP@1	AP@3	AP@5	AP@ 10
Black Panther	1	1	1	0.8	1	1	1	0.95
Article 15	1	1	1	0.6	1	1	1	0.9

Table 4. Recommendation evaluation with predicted and actual scores and ranks

Sl. No.	Search_Query	Recommendation	Predicted_Score	Predicted_Rank	True_Score	True_Rank
1	Black Panther	Chappie	1	1	0.95	2
2	Black Panther	Clash of the Titans	0.988	2	0.526	7
3	Black Panther	Green Lantern	0.987	3	0.368	9
4	Black Panther	Halo: The Fall of Reach	0.974	4	0.474	8
5	Black Panther	Stargate	0.962	5	1	1
6	Black Panther	Illang: The Wolf Brigade	0.954	6	0.579	6
7	Black Panther	Men in Black	0.952	7	0.842	4
8	Black Panther	DC's Legends of Tomorrow	0.951	8	0.895	3
9	Black Panther	Superman Returns	0.95	9	0.684	5
10	Black Panther	Season of the Witch	0.945	10	0.316	10
11	Article 15	Parmanu: The Story of Pokhran	1	1	1	1
12	Article 15	Kalushi: The Story of Solomon Mahlangu	0.988	2	0.789	4
13	Article 15	Sicilian Ghost Story	0.982	3	0.763	5
14	Article 15	Interrogation	0.97	4	0.947	2
15	Article 15	My Birthday Song	0.954	5	0.368	7
16	Article 15	The Hater	0.947	6	0.842	3
17	Article 15	The Letter Reader	0.915	7	0.789	4
18	Article 15	Metro	0.907	8	0.737	6
19	Article 15	Just Another Love Story	0.892	9	1	1
20	Article 15	Aapla Manus	0.891	10	0.842	3

In order to calculate the DCG at rank 10 (DCG@10) and normalized discounted cumulative gain (NDCG@10), first, the cumulative gains for the top 10 results of each query are computed. Following this, the ideal cumulative gains (IDCG@10) are determined by sorting the true scores in descending order and selecting the top 10 true scores. Subsequently, the DCG@10 values are computed using the actual cumulative gains, while the NDCG@10 is obtained by normalizing the DCG@10 values with their corresponding IDCG@10 counterparts. This normalization ensures that the relevance of the top-ranked results is evaluated relative to the best possible ranking scenario, providing a standardized measure of performance across different queries. The results are shown in Table 5.

The efficacy of the recommender system proposed in this paper was demonstrated by the average NDCG of 0.956 for the two search queries. This method proved to be quite effective in ranking search results, ensuring that users saw the most relevant items first. The recommender system optimized the ranking process and produced nearly perfect NDCG scores by utilizing the proposed sophisticated algorithm and methodology. This performance demonstrated the recommender system's dependability and accuracy, which were crucial for enhancing user engagement and satisfaction.

The model is also run through an A/B test in the context of our content-based recommender system in order to assess its effectiveness and effect on user happiness. Our main goal was to find out if, when compared to the prior recommendation approach, the content-based recommender system significantly improved user engagement and overall satisfaction. While users in the experimental group were exposed to our recently constructed recommender system, individuals in the control group continued to use the current recommendation system. Over the course of a week, users in the experimental group showed a significant rise in click-through rate (CTR), according to our study, which is compelling evidence of our recommender system's efficacy.

Table 5. The DCG, IDCG, and NDCG values at the rank 10 for the two query groups

Group	DCG@10	IDCG@10	NDCG@10
Black Panther	6.97	7.64	0.912
Article 15	7.566	7.568	1.000
Average			0.956

5. CONCLUSION

In this study, we found that capturing semantic relationships between words in movie and show descriptions significantly correlates with enhanced recommendation accuracy. Recent observations suggest that utilizing advanced Word2Vec embeddings alongside content-based filtering improves the system's precision and relevance in recommendations. Our findings provide conclusive evidence that this approach not only improves the accuracy of recommendations but also boosts user engagement, as reflected by a high NDCG of 0.956 and a notable increase in CTR during A/B testing. These results demonstrate the effectiveness of our method in delivering relevant recommendations. Future research could extend these techniques to job recommendation systems, such as LinkedIn, to refine profile matching and recommendation quality.

REFERENCES

[1] N. Mishra, S. Chaturvedi, A. Vij, and S. Tripathi, "Research problems in recommender systems," *Journal of Physics: Conference Series*, vol. 1717, no. 1, p. 012002, Jan. 2021, doi: 10.1088/1742-6596/1717/1/012002.

[2] R. Dridi, L. Tamine, and Y. Slimani, "Context-aware multi-criteria recommendation based on spectral graph partitioning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11707 LNCS, 2019, pp. 211–221.

[3] N. M. Villegas, C. Sánchez, J. Díaz-Cely, and G. Tamura, "Characterizing context-aware recommender systems: a systematic literature review," *Knowledge-Based Systems*, vol. 140, pp. 173–200, Jan. 2018, doi: 10.1016/j.knosys.2017.11.003.

[4] L. Wu, C. Quan, C. Li, Q. Wang, B. Zheng, and X. Luo, "A context-aware user-item representation learning for item recommendation," *ACM Transactions on Information Systems*, vol. 37, no. 2, pp. 1–29, Apr. 2019, doi: 10.1145/3298988.

[5] S. A. Narayan, H. Kumaar, D. S. Narayanan, S. Srikumaran, and S. Veni, "Content-based movie recommender system using keywords and plot overview," in *2022 International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET 2022*, Mar. 2022, pp. 49–53, doi: 10.1109/WiSPNET54241.2022.9767172.

[6] D. Roy and M. Dutta, "A systematic review and research perspective on recommender systems," *Journal of Big Data*, vol. 9, no. 1, p. 59, Dec. 2022, doi: 10.1186/s40537-022-00592-5.

[7] J. Zhang, Y. Wang, Z. Yuan, and Q. Jin, "Personalized real-time movie recommendation system: practical prototype and evaluation," *Tsinghua Science and Technology*, vol. 25, no. 2, pp. 180–191, Apr. 2020, doi: 10.26599/TST.2018.9010118.

[8] D. Kumar, "Comparative study of movie recommendation system using feature engineering and improved error function," in *2022 International Conference on Futuristic Technologies, INCOFT 2022*, Nov. 2022, pp. 1–6, doi: 10.1109/INCOFT55651.2022.10094480.

[9] A. Havolli, A. Maraj, and L. Fetahu, "Building a content-based recommendation engine model using Adamic Adar measure; a Netflix case study," in *2022 11th Mediterranean Conference on Embedded Computing, MECO 2022*, Jun. 2022, pp. 1–8, doi: 10.1109/MECO55406.2022.9797139.

[10] Y. Tai, Z. Sun, and Z. Yao, "Content-based recommendation using machine learning," in *IEEE International Workshop on Machine Learning for Signal Processing, MLSP*, Oct. 2021, vol. 2021-October, pp. 1–4, doi: 10.1109/MLSP52302.2021.9596525.

[11] K. S. Suresh, M. Srivastava, and Mohana, "YouTube and movie recommendation system using machine learning," in *Proceedings of the 2023 2nd International Conference on Electronics and Renewable Systems, ICEARS 2023*, Mar. 2023, pp. 1352–1356, doi: 10.1109/ICEARS56392.2023.10084999.

[12] P. Sirisha, G. L. Devi, and N. Ramesh, "Plot-topic based movie recommendation system using WordNet," in *Proceedings of the 2022 9th International Conference on Computing for Sustainable Global Development, INDIACom 2022*, Mar. 2022, pp. 45–49, doi: 10.23919/INDIACom54597.2022.9763244.

[13] M. Cagatayli and E. Celebi, "Is BIG-5 personality feature set convenient for movie recommender systems," in *2017 25th Signal Processing and Communications Applications Conference, SIU 2017*, May 2017, pp. 1–4, doi: 10.1109/SIU.2017.7960343.

[14] A. Rai, K. Yadav, M. Singh, and S. K. Singh, "Accuracy comparison of various movie recommendation system algorithms," in *Proceedings - 2022 4th International Conference on Advances in Computing, Communication Control and Networking, ICAC3N 2022*, Dec. 2022, pp. 254–260, doi: 10.1109/ICAC3N56670.2022.10074471.

[15] D. Roy and F. Shirazi, "A review on multiple data source based recommendation systems," in *Proceedings - 2021 International Conference on Computational Science and Computational Intelligence, CSCI 2021*, Dec. 2021, pp. 1534–1539, doi: 10.1109/CSCI54926.2021.00298.

[16] K. Falk, "Content-based filtering," in *Practical recommender systems*, 2019, pp. 248–283.

[17] S. Raza and C. Ding, "Progress in context-aware recommender systems - an overview," *Computer Science Review*, vol. 31, pp. 84–97, Feb. 2019, doi: 10.1016/j.cosrev.2019.01.001.

[18] S. L. Vu and Q. H. Le, "A deep learning based approach for context-aware multi-criteria recommender systems," *Computer Systems Science and Engineering*, vol. 44, no. 1, pp. 471–483, 2022, doi: 10.32604/csse.2023.025897.

[19] Q. H. Le, S. L. Vu, T. K. P. Nguyen, and T. X. Le, “A state-of-the-art survey on context-aware recommender systems and applications,” *International Journal of Knowledge and Systems Science*, vol. 12, no. 3, pp. 1–20, Jul. 2021, doi: 10.4018/IJKSS.2021070101.
[20] D. Valcarce, A. Landin, J. Parapar, and Á. Barreiro, “Collaborative filtering embeddings for memory-based recommender systems,” *Engineering Applications of Artificial Intelligence*, vol. 85, pp. 347–356, Oct. 2019, doi: 10.1016/j.engappai.2019.06.020.
[21] M. Hagiwara, “Word and document embeddings,” in *Real-world natural language processing: practical applications with deep learning*, 2023, pp. 49–79, [Online]. Available: https://livebook.manning.com/book/real-world-natural-language-processing/chapter-3/v-1/.
[22] M. Kaneko and D. Bollegala, “Dictionary-based debiasing of pre-trained word embeddings,” in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 212–223, doi: 10.18653/v1/2021.eacl-main.16.
[23] F. Fkih, “Similarity measures for collaborative filtering-based recommender systems: review and experimental comparison,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 7645–7669, Oct. 2022, doi: 10.1016/j.jksuci.2021.09.014.
[24] A. Gunawardana, G. Shani, and S. Yogev, “Evaluating recommender systems,” in *Recommender Systems Handbook: Third Edition*, 2022, pp. 547–601, doi: 10.1007/978-1-0716-2197-4_15.
[25] R. Banik, “Building content-based recommenders,” in *Hands-on recommendation systems with Python: start building powerful and personalized, recommendation engines with Python*, 2018, pp. 87–116.
[26] Y. Wang, L. Wang, Y. Li, D. He, W. Chen, and T. Y. Liu, “A theoretical analysis of NDCG ranking measures,” *Journal of Machine Learning Research*, vol. 30, pp. 25–54, 2013.

BIOGRAPHIES OF AUTHORS

Surabhi Anuradha is an associate professor at Keshav Memorial Institute of Technology in Hyderabad, India, specializing in artificial intelligence, machine learning. deep learning, natural language processing and generative AI. With over two decades of experience in education and administration, she is currently pursuing a Ph.D. in deep learning from SR University, Warangal, India. She is actively involved in different domains of research focusing on generative AI, large language models (LLMs), and visual language models (VLMs). She can be contacted at email: anuradha@kmit.in.

Dr. Pothabathula Naga Jyothi is assistant professor at GITAM School of Technology in Department of Computer and Engineering, GITAM deemed to be University, India. She holds a Ph.D. degree in Computer Engineering with specialization in medical data analysis. Her research areas are artificial Intelligence and machine learning, AI in medical surgery, and network security. She has filed a number of patents and IPRs on her innovative ideas and has been awarded Indian innovation patents publication and filled few IPRs. She can be contacted at email: npothaba@gitam.edu.

Dr. Surabhi Sivakumar is an associate professor at Anil Neerukonda Institute of Technology and Sciences in Visakhapatnam, India, with a Ph.D. in Chemistry. His research primarily focuses on Photocatalysis, Ion exchange materials, and Wastewater treatment. In recent years, he has transitioned into the field of data science and machine learning, where he applies his expertise to analyze chemical and drug-related data using machine learning tools and algorithms. He can be contacted at email: sivakumar.chemistry@anits.edu.in.

Dr. Martha Sheshikala is currently holding the position of head and professor at the School of Computer Science and Artificial Intelligence at SR University in Warangal, India. She earned her Ph.D. in Computer Science and Engineering from K L Educational Foundation, Andhra Pradesh, in March 2018. Her research focuses on areas such as data mining, machine learning, and natural language processing. Her extensive academic contributions include over 50 publications in various national and international journals, conferences, and proceedings. She can be contacted at email: marthakala08@gmail.com.