Developing a Movie Recommendation System for Enhancing User Experience

Team Members: James Onyejizu, Chibuikem Ezemaduka

1 Introduction

In today's digital age, where the internet is a fundamental aspect of everyday life, individuals are faced with the dilemma of excessive choices. From deciding on a restaurant for dinner to selecting the right educational course, the sheer volume of information online can be daunting. A recommendation system (RS) is a utility designed to aid users by suggesting services or products that they are most likely to find appealing or useful, as outlined by Khanian and Mohd [1]. Recommendation systems utilize three primary methods: collaborative filtering, content-based filtering, and hybrid filtering. Within content-based filtering, the system relies on data provided by the user, either through explicit means like ratings or implicitly through actions such as clicking on links. This information is used to create a unique user profile. Recommendations are generated based on this profile, focusing on items that align with the user's past behaviors and preferences. In contrast, collaborative filtering operates on the principle of shared tastes among users. It identifies users with similar preferences and suggests items that others with comparable tastes have liked, under the assumption that these items will appeal to the user in question [2]. As technological advancements introduce new platforms and modalities, the task of deciphering and predicting user interactions and choices becomes more difficult. Against the backdrop of this increasing digital landscape, where choice overload is a common challenge, the role of an RS becomes even more crucial in guiding user decisions. This need for effective recommendation strategies becomes particularly pronounced in the realm of entertainment, an area where preferences are not just diverse but also dynamically evolving.

Streaming platforms like Netflix are increasingly becoming a staple of entertainment consumption, and personalized content recommendation has emerged as an important factor in user retention and satisfaction. Specifically, a movie recommendation system represents a vital tool in this context. It not only curates content tailored to individual preferences but also enhances the overall user experience, making the platform more intuitive and engaging. These systems actively mitigate the paradox of choice found in extensive content libraries, aiding users in effortlessly and effectively navigating through the assortment of available movies, ensuring the content offered by the platform aligns closely with user preferences.

Today, various recommendation systems utilizing different technologies are prevalent in numerous fields. Jieun Son and Seoung Bum Kim proposed a method in recommendation systems using content-based filtering through multi-attribute networks, incorporating detailed item attributes [3]. Yolanda Blanco-Fernández et al. tackled the issue of overspecialization in intelligent recommendation systems by integrating content-based filtering with semantic reasoning [4]. The foundational concept of collaborative filtering was introduced by Goldberg et al. in 1991, employing the tapestry system, which was designed for smaller user groups and required significant user input [5]. Mehrbakhsh Nilashi et al. enhanced collaborative filtering by utilizing ontology and dimensionality reduction techniques to solve problems of sparsity and scalability [6]. Tianqi Zhou et al. applied the Hadoop programming model to implement an item-based collaborative filtering algorithm, which was tested using the Movielens-10M dataset comprising 1000 rows of ratings [7]. Ningning Yi et al. proposed a movie recommendation

system that utilizes graph databases [8]. In references [9] and [10], model-based methods were explored, which develop user models from their ratings to predict the values of unrated items. M.G. Vozalis and K.G. Margaritis combined Singular Value Decomposition (SVD) with demographic data to enhance both user-based and item-based collaborative filtering [11]. S.M. Ali et al. introduced a hybrid model that merges genomic tags of movies with the concept of content-based filtering [12]. F.Deng et al. developed a method for calculating user preferences using a combination of user-generated features, and image visual features, and transforming these into hybrid feature ratings [13]. C. Yang et al. suggested a hybrid approach based on social similarity and item attributes [14]. Priscila Valdiviezo and J. Bobadilla combined various user ratings with demographic information like age, gender, and occupation into a single matrix model and then applied collaborative filtering to enhance the prediction of ratings [15]. R. Bharti and D. Gupta proposed a system for new users using content-based filtering and for existing users using collaborative filtering, with cosine and pearson similarity measures and Hive for database management [16]. Jeffrey Lund and Yiu-Kai Ng adopted a deep learning approach with an autoencoder-based collaborative filtering system [17]. S. Kumar et al. proposed a sentiment analysis-based hybrid approach for movie recommendation, utilizing tweets from microblogging websites to analyze user sentiments and address the cold start problem [18]. J.K. Leung et al. studied the effect of user mood on recommendation systems, assigning emotional tags to movies through an auto-detected affective attribute model trained on tweet text [19]. T. Singh et al. improved the performance of recommendation systems by using real-time, multilingual tweets, applying sentiment analysis, and classifying the tweets using RNN [20]. Bhavya Ghai, Joydip Dhar, and Anupam Shukla [21] explored multi-level ensemble learning in recommender systems, emphasizing stack generalization and critiquing traditional ensemble learning methods. Lastly, J. Bobadilla et al. proposed a deep learning-based approach to enhance the performance of collaborative filtering [22].

By harnessing the power of advanced recommendation systems using graph neural networks (GNNs) [23], we can significantly boost user enjoyment by providing tailored movie suggestions that resonate with individual preferences. Unlike traditional methods that often rely on simpler, linear correlations, GNNs excel at capturing complex and non-linear relationships within data. Also, by leveraging the relational information inherent in graph data, GNNs can offer more accurate recommendations.GNNs can scale effectively with the size of the dataset, which is crucial given the ever-expanding volume of data in modern recommendation systems. Most importantly, they can handle large, sparse graphs efficiently, making them suitable for real-world applications where scalability is a key concern. This project aims to evaluate GNNs for movie recommendation tasks. In particular, we evaluate two GNN models albeit with slight modifications: the Light Graph Convolution Network (LightGCN) [24] and Neural Graph Collaborative Filtering (NGCF) [25] on the MovieLens 1M dataset and report our results. Our results show that LightGCN generally outperforms the vanilla NGCF in the precision@20 and recall@20 metrics. We also see that we achieve the least RMSE score when we used the least number of layers for both models. This reinforces the idea that GNN model complexity does not guarantee increase in performance due to issues such as oversmoothing. Additionally, a less complex model, like LightGCN, can be more effective due to its focus on leveraging graph structure, despite having fewer learnable parameters.

2 Problem description

The core goal of this project is to evaluate the efficacy of two Graph Neural Network (GNN) models, LightGCN and NGCF, within the scope of movie recommendation systems. This exploration addresses the shortcomings of traditional recommendation algorithms, especially in processing large, sparse datasets and adapting to dynamic user preferences. Central to the project is assessing how these models handle graph-structured data and their ability to capture complex relationships between users and movies. Despite LightGCN and NGCF not being previously evaluated on movie recommen-

dations, this study will employ these models on a subset of the MovieLens dataset[26] to learn user E_u and movie E_i embeddings. The objective is to predict the ratings a user u might give a movie i upon interaction, using a portion of the dataset for training with slight modifications to the standard architectures of LightGCN and NGCF to suit our predictive goals. The study will critically analyze and compare the performance of LightGCN and NGCF, focusing on the accuracy, and relevance of their recommendations, while emphasizing the importance of model simplicity and direct utilization of graph structures for enhanced performance.

3 Methodology

In this section, we delve into the operational intricacies of two GNN models, NGCF and LightGCN, used in this study. GNNs are designed to learn patterns in graph-structured data, integral to various applications. They operate by iteratively passing among nodes, allowing each node to gather and infer information from its multi-hop neighborhood. This approach, central to deep learning, leverages node embeddings - d-dimensional vectors representing latent features. GNNs, adhering to permutation invariance and equivariance, consist of three core functions: message, aggregation, and update, enabling them to capture local and high-order graph structures crucial for recommendation systems. This framework is especially effective for collaborative filtering in recommender systems, surpassing traditional methods like matrix factorization by capturing complex graph structures and relationships.

3.1 Neural Graph Collaborative Filtering (NGCF)

NGCF enhances node embeddings through learnable feature transformations and activation functions at each layer. Its update rules for user and item embeddings are defined as follows:

$$\mathbf{e}_u^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{e}_u^{(k)} + \sum_{i\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\left(\mathbf{W}_1\mathbf{e}_i^{(k)} + \mathbf{W}_2(\mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)})\right)\right), \tag{1}$$

$$\mathbf{e}_i^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{e}_i^{(k)} + \sum_{u\in\mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\left(\mathbf{W}_1\mathbf{e}_u^{(k)} + \mathbf{W}_2(\mathbf{e}_u^{(k)} \odot \mathbf{e}_i^{(k)})\right)\right). \tag{2}$$

In the message function of the NGCF model, each user node receives transformed features from neighboring item nodes, employing matrix $\mathbf{W}_1$ for transformation and $\mathbf{W}_2$ for element-wise multiplication. This process is supplemented by symmetric normalization to maintain the stability of embeddings as graph convolutions increase. Subsequently, the aggregation function compiles these transformed messages for each node. The final step in the process, the update rule, applies a non-linear activation function to each node's embedding, enhancing the detail and depth of user-item interaction representations.

3.2 Light Graph Convolution Network (LightGCN)

LightGCN distinguishes itself by focusing on neighborhood aggregation, a core aspect of its network architecture. The model actively employs only the initial embeddings of users and items as its learnable parameters. It deliberately avoids incorporating complex layer-specific GNN parameters, feature transformations, and nonlinear activations, streamlining its architecture for efficiency. This design choice stems from the understanding that shallow, learnable embeddings are sufficiently expressive for capturing user and item characteristics in collaborative filtering scenarios. By emphasizing the diffusion of node embeddings across the graph, LightGCN efficiently captures necessary information for predicting user preferences. It employs a straightforward update rule where each user node assimilates information from neighboring item nodes directly, fostering effective prediction capabilities. The

update process is mathematically formulated as:

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)}, \tag{3}$$

$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}. \tag{4}$$

To compute the final embedding output for users and items, LightGCN aggregates the intermediate representations from each model layer, applying a weighted summation:

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}, \quad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)}, \tag{5}$$

with the weighting factor α_k typically set uniformly as $1/(K+1)$. This approach not only prevents over-smoothing, a common issue in deeper models but also ensures the capture of diverse semantic levels in user-item interactions.

3.3 Model Training

In this subsection, we explore the training methodologies employed for the LightGCN and NGCF models, specifically focusing on the modifications we make to the loss function and prediction function to enhance recommendation performance.

3.3.1 Loss Function

For the training of our models, we utilize the Mean Squared Error (MSE) as the loss function, appreciating its alignment with supervised learning paradigms:

$$MSE = \frac{1}{N} \sum_{n=1}^{N} (y - \widehat{y})^2, \tag{6}$$

where y is the true rating a user gives to a movie, and $\widehat{y}$ is the predicted rating inferred from the embeddings of the user and movie. This approach differs from the semi-supervised BPR (Bayesian Personalized Ranking) loss that was originally utilized in LightGCN and NGCF models, reflecting a more conventional and direct method for predicting user ratings.

3.3.2 Prediction Function

In the LightGCN model, we depart from the typical prediction function $\widehat{y} = e_u^T e_i$ and instead apply a fully connected layer to the concatenation of user and movie embeddings:

$$\widehat{y} = \text{fully connected layer}(\text{CONCAT}[e_u, e_i]), \tag{7}$$

mapping from a dimension of 128 to a single output to refine the interaction modeling (each node embedding has dim 64). We make this adjustment because, in our trials, we discovered that for this supervised learning task, the LightGCN model was not capable of learning the embeddings when using $\widehat{y} = e_u^T e_i$ at the output. We are unsure of the reason for this but leave its investigation to future work. For NGCF, we conduct experiments using both the original embedding product $\widehat{y} = e_u^T e_i$ and a fully connected layer applied to the concatenated embeddings, to assess the potential benefits of this augmented approach in capturing the intricacies of user-movie interactions.

4 Experiments

We motivate the results of our experiments with a brief description of the datasets and metrics employed.

4.1 Datasets

Our study leverages the MovieLens 1M dataset, comprising 1,000,209 ratings from 6040 users for 3952 movies. Each user has provided ratings for at least 20 movies, ensuring comprehensive connectivity within the dataset. Alongside ratings, the dataset includes rich metadata like user demographics and movie details such as titles, release dates, and genres. We however do not use the metadata for our task. The dataset's structure naturally forms a bipartite graph, making it well-suited for graph machine-learning techniques in movie recommendation tasks. For our experiments, we divided the dataset into 80% training, 10% validation, and 10% test sets to facilitate a thorough evaluation of the model's performance.

4.2 Metrics

In evaluating the performance of LightGCN and NGCF models, we focus on three key metrics: precision@20, recall@20, and the Root Mean Square Error (RMSE). These metrics provide insights into the accuracy and coverage of the recommended items by each model. The precision@20 and recall@20 metrics assesses the proportion of relevant items among the top 20 recommendations (i.e top 20 predicted movie ratings) for each user. Since our task is not directly a binary classification task, we set a threshold rating value of 3.5 to divide the ratings into two classes where ratings ¿= 3.5 are in class 1 and those ¡ 3.5 are in class 0. Recommendations in class 1 are deemed as most desirable to satisfy the user. We follow the common formulas for computing precision and recall where in our case, we denote True positives as those ratings predicted to be in class 1 and whose true value is also in class 1. False positives are those predicted to be in class 1 but whose true values are in class 0. Both False and True negatives follow the same logic. The RMSE is calculated between all predicted ratings and their true values in the test set. These metrics are crucial in understanding the effectiveness of each model in recommending relevant movies to users in the MovieLens 1M dataset.

4.3 Results

This section presents the results of our comprehensive evaluation of the LightGCN and NGCF models. The performance of each model is analyzed under various configurations to understand their behavior in different scenarios. We conduct an ablation study to scrutinize the impact of the number of layers and batch sizes on key performance metrics such as precision, recall, and RMSE. The following subsections provide detailed insights into the outcomes of this analysis, offering a comparative perspective on the models and highlighting the implications of their respective configurations.

4.3.1 Ablation Study

We focus on the impact of the number of layers and batch size on the models' precision, recall, and RMSE (Root Mean Square Error) metrics.The experiments were conducted over 200 epochs with the number of layers varying from 1 to 4 and batch sizes set to three different values: $A = 5120$, $B = 10240$, and $C = 20480$ for both models. Based on validation testing, for all the experiments, we settle on a learning rate of 0.001 and weight decay of 0.01 to prevent over fitting. We use the Adam optimizer and implement our code using the PyTorch library together with the PyG extension which is suited for GNN implementation tasks. We ran our tests on an Nvidia 2070 RX 8gb GPU with intel processor of 2.4Ghz.

Table 1: Performance analysis of LightGCN model with different layer counts and batch sizes. The model was evaluated based on precision@20, recall@20, and RMSE. Each configuration was executed for 200 epochs.

No. of layers		1			2			3			4	
Batch Size	A	B	C	A	B	C	A	B	C	A	B	C
Precision@20	0.70764	0.70596	0.70601	0.70511	0.70414	0.70322	0.69575	0.67804	0.66557	0.69374	0.68088	0.6674
Recall@20	0.70817	0.71257	0.70376	0.7175	0.71494	0.7251	0.75841	0.82134	0.86293	0.76349	0.81807	0.85804
RMSE	1.039	1.0454	1.0479	1.0596	1.0672	1.0719	1.0793	1.089	1.0961	1.0799	1.0899	1.0962

Table 2: Performance analysis of NGCF (with dot product at output) with different layer counts and batch sizes. The model was evaluated based on precision@20, recall@20, and RMSE. Each configuration was executed for 200 epochs.

No. of layers		1			2			3			4	
Batch Size	A	B	C	A	B	C	A	B	C	A	B	C
Precision@20	0.70583	0.70692	0.70701	0.69433	0.67806	0.69185	0.69541	0.69789	0.67601	0.69102	0.68645	0.5717
Recall@20	0.67924	0.69177	0.67097	0.65309	0.54798	0.57612	0.61669	0.59149	0.53214	0.73507	0.55647	0.3984
RMSE	0.9614	0.9556	0.9494	0.9782	0.9817	0.9686	0.98565	0.9815	0.9844	0.9872	0.9856	1.0028

Table 3: Performance analysis of NGCF (with fully connected layer at output) with different layer counts and batch sizes. The model was evaluated based on precision@20, recall@20, and RMSE . Each configuration was executed for 200 epochs.

No. of layers		1			2	
Batch Size	A	B	C	A	B	C
Precision@20	0.70528	0.70748	0.71373	0.70766	0.70414	0.70943
Recall@20	0.73943	0.72735	0.70621	0.71536	0.74105	0.70062
RMSE	0.9542	0.9461	0.9393	0.9653	0.9566	0.9491

4.3.2 Discussion

The performance analysis of the LightGCN model, as delineated in Table 1, reveals a discernible trade-off between precision and recall metrics. Specifically, an increase in the number of layers leads to a decrease in Precision@20, while concurrently enhancing Recall@20. This trend, which also manifests as batch sizes increase, underscores the challenges inherent in achieving a balanced optimization of these metrics within the LightGCN framework. Additionally, the model's RMSE value increases with each additional layer, suggesting that increasing model complexity may inadvertently affect performance negatively. This could be attributed to issues such as over-smoothing or an excessive reliance on global graph topology, rather than local structural nuances.

In contrast, the NGCF model, as shown in Table 2, exhibits a different pattern. With the increase in layer depth, both Precision@20 and Recall@20 generally decline, particularly at larger batch sizes and with four layers. This pronounced decrease in recall scores implies a substantial impact of layer depth on the model's ability to retrieve relevant items effectively.

For both LightGCN and NGCF models, a common observation is the rise in RMSE with an increase in layers and batch size, hinting at potential oversmoothing issues. This trend might be indicative of a model's increased focus on global graph topology at the expense of local structure, a factor that can hamper the efficacy of graph neural networks.

In examining the performance of the NGCF model, distinct patterns emerge when comparing configurations utilizing dot product and fully connected layers at the output, as seen in Tables 2 and 3. Comparing using two layers, we observe that the NGCF model that has a fully connected layer at its output clearly outperforms its vanilla counterpart (inner product at output) in all three metrics. This outcome suggests that learning additional weights through a fully connected layer output in the NGCF model is more powerful at accurately predicting ratings than using the vanilla architecture. However, this uptick in performance comes at a cost of model complexity and training time, as there are additional weights for the model to learn. The performance difference between these two configurations underscore the significant impact of the output layer design on NGCF model performance in terms of all three key metrics.

In conclusion, the ablation study highlights the importance of selecting the optimal number of layers and batch size for each model. While both models exhibit varying trends, the choice of configuration should be guided by the specific requirements of precision, recall, and error minimization in the application context.

5 Conclusion

This study has presented a comparative analysis of two Graph Neural Network (GNN) models, LightGCN and NGCF, repurposed for a supervised learning task in the domain of movie recommendations using the MovieLens 1M dataset. Through careful reformulation of the problem and tailored modifications to the GNN architectures, our experimentation yielded promising results, characterized by a low RMSE and high precision and recall scores under specific configurations. Our findings indicate that the performance of GNN models in recommendation systems is nuanced. Notably, the NGCF model with a single layer (and fully connected layer at output) achieved the most balanced performance, boasting the lowest RMSE of 0.9393 and a precision of 0.7137. Meanwhile, the LightGCN model with three layers excelled in recall, reaching a peak of 0.8623. This suggests that, as mentioned in some areas of literature, increasing the number of layers in GNN models does not invariably enhance model performance, particularly with respect to RMSE in our case. The analysis also reveals that the architectural choice at the output layer has a significant impact on performance. The NGCF model with a fully connected layer consistently outperformed its dot product counterpart, emphasizing the importance of output layer design in the construction of effective GNN-based recommendation systems. In conclusion, our exploration into the application of LightGCN and NGCF models for movie recommendations underscores the critical balance between model complexity and performance. It highlights that careful consideration must be given to the trade-offs presented by different architectural choices, as well as to the distinct behaviors that these models exhibit when applied to supervised learning tasks. Future work should delve deeper into the interplay between model architecture and task formulation, with a focus on identifying the optimal configurations that marry complexity with efficacy, ensuring robust and precise recommendations in various domains of interest.

6 References

[1] Najafabadi, Maryam Khanian, and Mohd Naz'ri Mahrin. "A systematic literature review on the state of research and practice of collaborative filtering technique and implicit feedback." Artificial intelligence review 45.2 (2016): 167-201.

[2] J. Salter and N. Antonopoulos, "CinemaScreen Recommender Agent: Combining Collaborative and Content-Based Filtering", IEEE Intelligent Systems 21(1):35-41, 2006.

[3] J. Son and S. B. Kim, "Content-based filtering for recommendation systems using multiattribute networks", Expert Systems with Applications 89:404-412, 2017.

[4] Y. B. Fernandez, J. J. P. Aris, A. G. Dolla, M. R. Cabrer, and M. L. Nores, "Providing Entertainment by Content-based Filtering and Semantic Reasoning in Intelligent Recommender Systems", IEEE Transactions on Consumer Electronics 54(2):727-735, 2008.

[5] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to Weave an Information tapestry", Communications of the ACM 35(12):61-70, 1992.

[6] M. Nilashi, O. Ibrahim, and K. Bagherifard, "A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques", Expert Systems with Applications 92:507-520, 2018.

[7] T. Zhou, L. Chen, and J. Shen, "Movie Recommendation System Employing the User-Based CF in Cloud Computing", IEEE International Conference on Computational Science and Engineering and IEEE International Conference on Embedded and Ubiquitous Computing , 2:46-50, 2017.

[8] N. Yi, C. Li, M. Shi, and X. Feng, "Design and Implementation of Movie Recommender System Based on Graph Database", IEEE 14th Web Information Systems and Applications Conference , 132-135, 2017.

[9] H. Liu, Z. Hu, A. Mian, H. Tian, and X. Zhu, "A new user similarity model to improve the accuracy of collaborative filtering", Knowledge-Based Systems 56:156-166, 2014.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms", Proceedings of the 10th International Conference on World Wide Web , 285-295, 2001.

[11] M. G. Vozalis and K. G. Margaritis, "Using SVD and demographic data for the enhancement of generalized Collaborative Filtering", Information Sciences 177(15):3017-3037, 2007.

[12] S. M. Ali, G. K. Nayak, R. K. Lenka, and R. K. Barik, "Movie Recommendation System Using Genome Tags and Content-Based Filtering", Springer, Singapore, 85-94, 2018.

[13] F. Deng, P. Ren, Z. Qin, G. Huang, and Z. Qin, "Leveraging Image Visual Features in Content-Based Recommender System", Scientific Programming 2018, 2018.

[14] C. Yang, X. Chen, L. Liu, T. Liu, and S. Geng, "A Hybrid Movie Recommendation Method Based on Social Similarity and Item Attributes", International Conference on Sensing and Imaging. Springer, Cham , 275-285, 2018.

[15] Priscila Valdiviezo D´ıaz and J. Bobadilla, "A Hybrid Approach of Recommendation via Extended Matrix Based on Collaborative Filtering with Demographics Information", International Conference on Technology Trends. Springer, Cham , 384-398, 2018.

[16] R. Bharti and D. Gupta, "Recommending Top N Movies Using Content-Based Filtering and Collaborative Filtering with Hadoop and Hive Framework", Springer, Singapore, 109-118, 2019.

[17] J. Lund and Y. Ng, "Movie Recommendations Using the Deep Learning Approach", IEEE Inter-

national Conference on Information Reuse and Integration, 47-54, Salt Lake City, UT, 2018.

[18] Kumar, Sudhanshu, Kanjar De, and Partha Pratim Roy, "Movie recommendation system using sentiment analysis from microblogging data", IEEE Transactions on Computational Social Systems 2020.

[19] Leung, John Kalung, Igor Griva, and William G. Kennedy, "Making Use of Affective Features from Media Content Metadata for Better Movie Recommendation Making", arXiv preprint arXiv:2007.00636 (2020).

[20] Singh, Tarana, Anand Nayyar, and Arun Solanki, "Multilingual Opinion Mining Movie Recommendation System Using RNN", First International Conference on Computing, Communications, and Cyber-Security, Springer, Singapore, 2020.

[21] Bobadilla, Jesus, Santiago Alonso, and Antonio Hernando, "Deep Learning Architecture for Collaborative Filtering Recommender Systems", Applied Sciences 10(7):24-41, 2020.

[22]Ghai, Bhavya, Joydip Dhar, and Anupam Shukla. "Multi-Level Ensemble Learning based Recommender System." Corpus ID 28239124 (2018).

[23] Scarselli, Franco, et al. "The graph neural network model." IEEE transactions on neural networks 20.1 (2008): 61-80.

[24] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 639-648.

[25] Wang, Xiang, et al. "Neural graph collaborative filtering." Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval. 2019.

[26] Harper, F. Maxwell, and Joseph A. Konstan. "The movielens datasets: History and context." Acm transactions on interactive intelligent systems (tiis) 5.4 (2015): 1-19.