HYPERPARAMETER TUNING OF GRAPH CONVOLUTION NETWORKS BASED COLLABORATIVE RECOMMENDERSYSTEMS - A COMPARATIVE STUDY

V. Lakshmi Chetana [1], [1]AssistantProfessor,DVR&Dr.HSMIC Collegeof Technology,India
D. Prasad[22]AssociateProfessor,DVR&Dr.HSMIC Collegeof Technology,IndiSa
,**Sk.Shareena Bhanu**[3] DepartmentofComputerScienceandEngineering,
DVR&Dr.S MIC Collegeof Technology
,**N.Geetha Sree Priya**[4] DepartmentofComputerScienceandEngineering,
VR&Dr.HS MIC Collegeof Technology,India
,**B.BajiAvula Raju** [5] DepartmentofComputerScienceandEngineering,
DVR&Dr.HS MIC Collegeof Technology,India

ABSTRACT*:*
With the rapid development of e-commerce and social media platforms, recommender systems have become indispensable tools for many business organizations. They are used in various applications like product suggestions on online e-commerce websites or playlist generators for video and music services. It has the ability to predict whether a particular user would prefer an item or not based on the user's past preferences and explore what they are interested in. Recommendation systems are divided into two types: Collaborative filtering and Content-Based recommendation Systems. Recently, deep learning models are used in recommender systems because of their ability to capture the non-trivial relationship between user and item. Graph Neural Networks (GNN) are a class of deep learning methods designed to perform inference on data described by graphs. NGCF and LightGCN are variants of GNN. These frameworks perform user recommendations with deep learning instead of the traditional matrix factorization.They measure the similarity between the user and item, therefore allowing us to understand how likely it is for the user to like the movie. In this paper, we compare two methods Light GCN and Neural Graph Collaborative Filtering(NGCF) to capture the collaborative signal between users/items.
Keywords: Collaborative filtering, Data Sparsity, Graph Neural Networks, High-orderconnectivity, Recommendation Systems.

INTRODUCTION

Rating prediction is an important task which aims at predicting the user's rating for the items which are not yet rated by the user. Collaborative filtering is a popular recommendation technique in various domains. It predicts the unknown ratings based on the ratings of the similar users and historical data. Sparsity due to cold start problem, where the user/item does not appear during the training process of generating recommendations, is a severe problem of collaborative filtering. To reduce this sparsity and to improve the recommendation accuracy, matrix factorization methods are used. Matrix factorization [1] is an efficient model-based collaborative filtering approach applied in the recommendation systems to address the sparsity problem. It decomposes the rating matrix (user-movie interaction matrix) into two low dimensional rectangular matrices, and a simple dot product is applied to predict the unknown ratings. The conventional matrix factorization methods are not considered to be efficient as they do not capture the user-item interactions entirely as it uses a simple dot product while predicting the rating. Next, various matrix factorization methods [2-7] were proposed to learn the user –iteminteractions by incorporating side information such as user content (like demographic, social relations, trust/untrust etc.,) and item content (like genre, categories, topics etc.,). Deep learning is an advanced learning technique which gained popularity in various domains [8]. Nowadays, lot of research was reported on deep learning-based recommendation systems to overcome the problems of traditional recommendation algorithms because they capture the hidden, non-linear, and non-trivial interactions between the user and item, which helps in a better prediction rate.The two important components of any deep learning based collaborative filtering models –1) embedding, represents the user and items as vector, and 2) interaction, reconstructs the interactions between the user and items based on embeddings [9]. In matrix factorization, the users and items are embedded as vectors and the interaction matrix is constructed by the dot product of these vectors. In deep learning-based m

 DOI10.36893.JK.2023.V13I04N16.00168-00177

[10-13], the user and item embeddings are merged and applied to the non-linearity of the neural networks. Graph Neural Networks (GNN) are a special case of deep neural networks where data is represented in the form of a bipartite graph. Nowadays, GNN based recommender systems have been dominant both in academic and industrial applications. The interaction matrix of the recommender systems is represented in the form of a graph. The users and items in the matrix are represented as nodes and the observed ratings are represented as links in the graph. In this work, we consider the unobserved rating prediction as link prediction on graphs. The main idea of GNN is to find the neighbors of the users/items and allow high-quality learning of user and item representations, which consequently improves the performance. Even though, existing methods are considered to be effective Xiang Wang et al. [9] believed that these methods are alone not sufficient to generate embeddings that capture the collaborative signal, which is hidden in the interactions between the users and items. To be more precise, most of the deep learning-based methods generated embedded vectors using descriptive features (like UserID, MovieID, and other attributes) and ignore hidden user-item interactions (a.k.a. collaborative signal), which are solely used to create the objective function for model training [15,16]. This collaborative signal helps to reveal the similarity between the users/items. As integration of user-item interactions into embedding vectors is practically impossible due to presence of large number of users and items, Xiang Wang et al. [9] proposed the concept of high – order connectivity to capture collaborative signal from the user-item interaction graph.To find the high-order connectivity information from the embedded vectors, a neural network is designed from the bipartite interaction graph to propagate the graph embeddings recursively.

The bipartite graph for the user-item matrix is represented in the figure1.

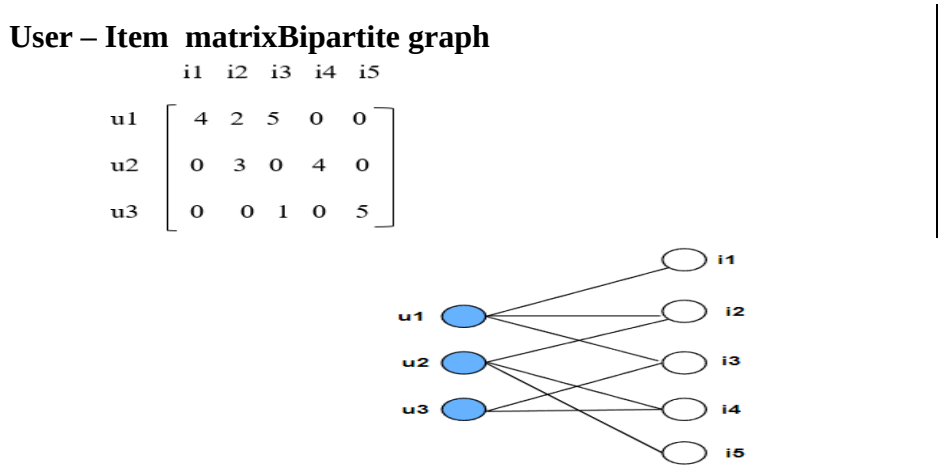

Figure 1: User-Item Bipartite graph

In this paper, we compare two methods LightGCN [14] and Neural Graph Collaborative Filtering(NGCF)[9] to capture the collaborative signal between users/items.The rest of the paper is organized as follows: section 2 deals with the previous works done in this domain, section 3 explains the two GNN based frameworks NGCF andLightGCN,ExperimentalSetup and Comparisons.

1. RELATED WORKS:

We review existing work on model-based CF, graph-based CF, and graph neural network-based methods, which are most relevant with this work.

2.1 SimplifyingGraphConvolutionNetworks:

Historically, the development of machine learning Graph Convolutional Networks and their variants became prominent because of their significant methods for learning graph representations.GCNs are primarily inspired from recent deep learning approaches, as a result, it may inherit unnecessary complexity and redundant computation. Hence, In SGC we reduce this complexity through successively removing the nonlinearities and collapsing weight matrices between consecutive layers.

 DOI10.36893.JK.2023.V13I04N16.00168-00177

We empirically show that the final linear model exhibits comparable or even superior performance to GCNs on avariety of tasks while being computationally more efficient and fitting significantly fewer parameters. We theoretically analyze the linear model and show that it corresponds to a fixed low-pass filter followed by a linear classifier. Notably, our experimental evaluation demonstrates that these simplifications do not negatively impact accuracy in many downstream applications. Moreover, the resulting model efficiently scales larger datasets and yields g up to two orders of magnitude speed over FastGCN.

GCNN FOR WEB SCALE RECOMMENDER SYSTEMS:

Recent advancements in deep neural networks for graph-structured data have revolutionized the performance of recommender systems. However, making them practical and scalable to web scale recommendation tasks with billions of items and trillions of users always remains a challenge. Here we present a highly scalable GCN framework that we have developed and deployed in production at Pinterest. Our framework, a random-walk-based GCN named Pin Sage, operates on a massive graph with 3 billion nodes and 18 billion edges - a graph that is 10, 000 times larger than typical applications of GCNs. Pin Sage drastically improves the scalability of GCNs by combining efficient random walks and graph convolutions to generate embeddings of nodes (i.e., items) that incorporate both graph structure as well as node feature information. According to offline metrics and user studies, Pin Sage generates higher quality recommendations than other graph-based alternatives. GCNs are more prominent as they use both content information as well as graph structure. Hence, We proposed Pin Sage, a random-walk highly-scalable graph convolutional network (GCN) algorithm which is capable of learning embeddings for nodes in web-scale graphs containing billions of objects. In addition to new techniques that ensure scalability, we introduced the use of importance pooling and curriculum training that drastically improved embedding performance. We deployed Pin Sage at Pinterest and comprehensively evaluated the quality of the learned embeddings on a no. of recommendation tasks, with offline metrics and user studies all demonstrating a substantial improvement in recommendation performance. Our work demonstrates the impact that graph convolutional methods can have in a production recommender system, and we believe that Pin Sage can be further extended in the future to tackle other graph representation learning problems at large scale, including knowledge graph reasoning and graph clustering.

GRAPH CONVOLUTIONAL MATRIX COMPLETION:

Graph Convolutional Matrix Completion (GCMC) is a technique for matrix completion in the context of graph-structured data. It aims to predict missing entries in a matrix based on the information available in a graph. The approach involves constructing a graph where the rows and columns of the matrix are represented as nodes, and the observed entries are represented as edges. A graph convolutional network (GCN) is then used to learn a low-dimensional representation of the nodes in the graph, which captures the underlying structure of the data. The GCN is trained to predict the missing entries in the matrix, based on the observed entries and the learned node representations. The loss function used to train the model typically involves a combination of reconstruction error and regularization terms. One of the key advantages of GCMC is that it can handle missing entries in the graph as well as missing entries in the matrix. This makes it particularly useful for applications where data is incomplete or sparse. GCMC has been applied in a variety of domains, including recommender systems, drug discovery, and social network analysis. Here, we view matrix completion as a link prediction problem on graphs where the interaction data in collaborative filtering can be represented by a bipartite graph between user and item nodes, with observed ratings/purchases represented by links. Content information can naturally be included in this framework in the form of node features. Further predicting ratings, then reduces to predicting labeled links in the bipartite user-item graph. The graph convolutional matrix completion (GC-MC): a graph-based auto-encoder framework for matrix completion, builds on recent progress in deep learning on graphs. The auto-encoder produces latent features of user and item nodes through a form of message passing on the bipartite interaction graph. These representations are used to reconstruct

 DOI10.36893.JK.2023.V13I04N16.00168-00177

the rating links using a bilinear decoder. The advantage of formulating matrix completion as a link prediction task on a bipartite graph becomes especially apparent when recommender graphs are accompanied with structured external information such as social networks. Combining such external information with interaction data can alleviate performance bottlenecks related to the cold start problem.

2. METHODOLOGY:

3.1 NGCF:

Neural Graph Collaborative Filtering (NGCF) is a recommendation algorithm that utilizes graph neural networks to performcollaborative filtering. In NGCF, the user-item interaction data is represented as a bipartite graph, where the nodes represent the users and items, and the edges represent the interactions between them. The graph neural network is then used to learn the embeddings of the users and items by propagating information along the graph. These embeddings are used to predict the user-item interactions and provide recommendations to the users. NGCF has shown promising results in terms of recommendation accuracy and scalability, and it has been applied in various real-world applications.

ARCHITECTURE OF NGCF:

The architecture of NGCF consists of 3 components: Embedding Layer - It initializes the user and item vectors, Multiple Embedding Propagation Layer – Embeddings are refined by introducing high order connectivity relations and Prediction Layer – It predicts the final score given by a user to an item by combining the redefined embeddings from different propagation layers.

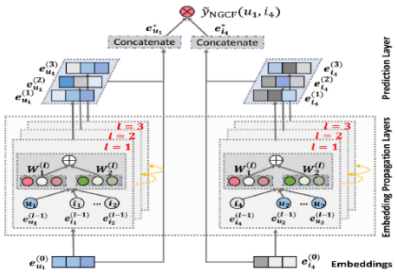

Figure a : Architecture of NGCF[15]

3.1.1EMBEDDING LAYER:

The embedding layer initializes user embeddings and item embeddings, where each user or item is represented by an embedding vector. Thus, an embedding look-up table can be built by combining all user anditem embeddings. These embeddings are then taken to the graph, where message passing happens in the second component of the framework.The

1-D matrix of embeddings formulated as:

$$E = [e_{u1}, \ldots, e_{uN}, e_{i1}, \ldots, e_{iM}] \quad (1)$$

MULTIPLE EMBEDDING PROPAGATION LAYER:

In this layer webuild message-passing architecture of GNN to capturecollaborative filtering signal along the graph structure and refine the embeddings of users and items. We first illustrate the design of one-layer propagation and generalize it to multiple successive layers. Here there are two types of propagation:

1) First-order Propagation
2) High-order Propagation

1)First-order Propagation:

The user's preference is calculatedbased on user - item interactions hence, the users that consume an item can be treated as the item's features and used to measure the collaborative similarity of two items. We build upon this basis to perform embedding propagation between the connected users and items, by performing two major operations: Message construction and Message aggregation.

- **Message Construction**:

 DOI10.36893.JK.2023.V13I04N16.00168-00177

ISSN: 2278-4632
Vol-13, Issue-04, No.06, April : 2023

The embedding propagation component consists of propagationlayers, where messages are passed from node to node on the graph. These messages are used to update a node with information from its neighbors.For each connected user-item pair *(u, i)*, the message from i to u is defined as:

$$m_{u\leftarrow i} = \frac{1}{\sqrt{|Nu||Ni|}}(W_1 e_i + W_2(e_i \odot e_u)) \quad (2)$$

Where, N_u, N_i are the first hop neighborsof *u* and *i*, W_1, W_2 are trainable weight matrices, and e_u, e_i are the embeddings of the item and user.

To make training easier, the normalizing constant in message construction, can actually be modeled as an adjacency matrix. This makes it more efficient to do batch processing throughout the propagation layers.

- **Message aggregation:**

The messages are aggregated to update the embeddings of users and items. The representation of user *u* is updated as:

$$e_u^{(1)} = \text{LeakyReLU}(m_{u\leftarrow u} + \sum_{i \in N_u} m_{u\leftarrow i})$$

2)High-order Propagation:

From previous representations created by first-order connectivity, we can stack more embedding propagation layers to explore the high-order connectivity information to encode the collaborative signal to estimate the relevance score between a user and item.By stacking the layers, a user and an item is can receive
tneighbours.

$$e_u^{(l)} = \text{LeakyReLU}(m_{u\leftarrow u}^{(l)} + \sum m_{u\leftarrow i}^{(l)})$$

3.1.3 MODEL PREDICTION:

Representations obtained from different layers are concatenated to form the final embedding for user or item.

$$e_u^* = e_u^{(0)} \| \cdots \| e_u^{(L)}, \quad e_i^* = e_i^{(0)} \| \cdots \| e_i^{(L)}$$

3.1.4 OUTPUT LAYER:

Finally, Users are recommended items with high preference by using the below formula:

$$\hat{y}_{NGCF}(u, i) = e_u^{*T} . e_i^*$$

3.2 LIGHTGCN:

LightGCN is a graph convolutional network (GCN) model designed for collaborative filtering tasks on sparse and large-scale recommendation graphs.The basic idea of LightGCN is to learn user item embeddings by linearly propagating themon user item interaction graph and uses the weighted sum of embeddings learned from all layer's final embedding. To Achieve this, it performs graph convolution iteratively, i.e., aggregating the features of neighbours as the new representation of a target node.

Architecture of LightGCN:

 DOI10.36893.JK.2023.V13I04N16.00168-00177

ISSN: 2278-4632
Vol-13, Issue-04, No.06, April : 2023

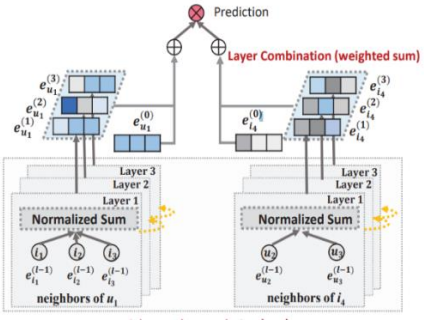

Figure b : Architecture of LightGCN[16]

In LightGCN architecture, only the normalized sum of neighbour embeddings is performed towards next layer; the other operations like self-connection, feature - transformation, and nonlinear activation are all removed,

which simplifies GCNs. In Layer Combination, we sum over the embeddings at each layer to obtain the final representations. It has twolayers:

1)Intralayer neighbourhood aggregation

2)Interlayer combination and model prediction

3.2.1 INTRALAYER NEIGHBOURHOOD AGGREGATION:

In LightGCN, we use simple weighted sum aggregator method and abandon the use of feature transformation and nonlinear activation. The graph convolution operation or propagation rule [39]) in LightGCN is defined as:

$$e_u^{(k+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{|Nu||Ni|}} e_i^{(k)}$$

$$e_i^{(k+1)} = \sum_{u \in N_u} \frac{1}{\sqrt{|Nu||Ni|}} e_u^{(k)}$$

Where, $e_u^{(k)}$ and $e_i^{(k)}$ - user and item node embeddings at the k-th layer.$|N_u|$ and $|N_i|$- the user and item nodes' number of neighbors**.**

3.2.2 INTERLAYER COMBINATION AND MODEL PREDICTION:

- **LAYER COMBINATION:**In LightGCN, the only trainable model parameters are the embeddings at the 0-th layer, i.e., $e_u^{(0)}$ forall users and $e_i^{(0)}$ for all items. Whenthey are given, the embeddings at higher layers can be computed via LGC defined in above Equation.After K layers LGC, we further combine the embeddings obtained at each layer and the 0^{th} layer embeddings to form the final representation of a user (an item) and item are combined using below equation:

$$e_u = \sum_{k=0}^{K} \alpha_k e_u^{(k)} \quad ; \quad e_i = \sum_{k=0}^{K} \alpha_k e_i^{(k)}$$

where, α_k -is hyperparameter and in our experiments, we find that setting αk uniformly as 1/(K+1) leads to good performance in general. There are 3 main reasons for performing layer combination.

With the increasing of the no. of layers, the embeddings will be smoothed .

The embeddings at different layers capture different semantics. E.g., the first layer smoothens the users and items thathave interactions, the second layer smooths usersthat overlap on interacted items, and higher-layerscapture higher-order. proximity. Thus, combining them will give effective representation.

 DOI10.36893.JK.2023.V13I04N16.00168-00177

F1-score = 2*Precision*Recall / Precision + Recall 12)

Combining embeddings at different layers with weighted sum captures the effect of graph convolution

Table 1: Description of Movielens Dataset

Datasets	#Users	#Items	#Ratings	Rating Levels	# Ratings > 3	Rating Distribution		
						Rating 3	Rating 4	Rating 5
Movie lens 100K	943	1682	100000	1,2,3,4,5	82520	27145	34174	21201

with self-connections, an important trick in GCNs.

- **MODEL PREDICTION:**

The model prediction is defined as the inner product of final user and item final representations embeddings:

$$\hat{y}_{ui} = e_u^T . e_i$$

RESULTS AND DISCUSSIONS:

a) DATASET DESCRIPTION: The dataset we used to run experiments is Movielens 100K dataset. It consists of 1,00,000 ratings given by 943 users for 1682 movies. The detailed analysis of MovieLens dataset is given in the following table.

b) EXPERIMENTAL SETUP:

The Software Requirements are Operating System – Windows 11 (64 bit), Language –Python 3.7 and Hardware Requirements are RAM – 8GB, Graphic card – NVIDIA GeForce GTX 1650 and Processor – AMD Ryzen 5 4600H.

c) EVALUATION METRICS:

We compare the performance of LightGCN and NGCF with the help of evaluation metrics like Precision and Recall. This reveals the effectiveness of LightGCN

- **Precision:** The proportion of true positive predictions over the total number of positive predictions.

Precision@K=TruePositive/TruePositive+FalsePositive

Recall@K=TruePositive/TruePositive+FalseNegative (11

- **Recall:** the proportion of true positive predictions over the total number of actual positive cases.

- **F1-score:** a weighted average of precision and recall that balances the trade-off between them.

COMPARISONS

We conducteda detailed comparison of LightGCN with NGCF using evaluation metrics like Precision, Recall and F-Score.For each epoch we calculated the Precision, Recall and F-Score for both NGCF and LightGCN in the following table.

 DOI10.36893.JK.2023.V13I04N16.00168-00177

ISSN: 2278-4632
Vol-13, Issue-04, No.06, April : 2023

Table 2: Comparison of NGCF and LightGCN

Latent factors=64, # layer = 3, Batch size=1024, Decay = 0.0001, LR = 0.005, Dataset=Movie Lens 100K

	# Epochs	*Precision@20*	*Recall@20*	*F-score@20*	*Computational Time (sec/iteration)*
NGCF	10	0.2078	0.3025	0.2464	3.61
	20	0.212	0.3175	0.2542	3.51
	30	0.2126	0.3162	0.2543	3.55
	40	0.2144	0.3143	0.2549	3.51
	50	0.2123	0.3134	0.2531	3.63
LightGCN	10	0.2063	0.3121	0.2484	2.93
	20	0.2211	0.3339	0.2660	2.55
	30	0.2292	0.3411	0.2742	2.50
	40	0.2343	0.3492	0.2804	2.68
	50	0.2406	0.3587	0.2880	2.70

Latent factors=128, # layer= 3, Batch size=1024, Decay = 0.0001, LR = 0.005, Dataset =MovieLens100K

	# Epochs	*Precision@20*	*Recall@20*	*F-score@20*	*Computational Time (sec/iteration)*
NGCF	10	0.1975	0.2956	0.2368	5.46
	20	0.1997	0.2956	0.2384	5.59
	30	0.2025	0.3024	0.2426	5.12
	40	0.2003	0.2997	0.2401	5.18
	50	0.2071	0.2996	0.2449	5.20
LightGCN	10	0.2144	0.3242	0.2581	3.76
	20	0.228	0.3391	0.2727	3.13
	30	0.238	0.3557	0.2852	3.07
	40	0.2409	0.2997	0.2671	3.14
	50	0.2397	0.3564	0.2866	3.15

Latent factors=256, # layer=3, Batch size=1024, Decay = 0.0001, LR = 0.005, Dataset=Movie Lens 100K

	# Epochs	*Precision@20*	*Recall@20*	*F-score@20*	*Computational Time (sec/iteration)*
NGCF	10	0.189	0.2749	0.2240	11.16
	20	0.1904	0.2798	0.2266	10.90
	30	0.1913	0.2813	0.2277	10.90
	40	0.1918	0.2844	0.2291	10.89
	50	0.1677	0.2583	0.2034	10.88

DOI10.36893.JK.2023.V13I04N16.00168-00177

LightGCN	10	0.2195	0.3283	0.2631	4.13
	20	0.2382	0.3536	0.2846	3.50
	30	0.2376	0.3558	0.2849	3.53
	40	0.2398	0.3594	0.2877	3.51
	50	0.2313	0.3589	0.2813	3.54

CONCLUSION:

In this work, we described the complexity in design of NGCF for collaborative filtering.We proposed an effective framework LightGCN which consists of two essential components - light graph convolution and layer combination. In light graph convolution, we discard feature transformation and nonlinear activation - two standard operations in GNNs but inevitably increase the training difficulty. In layer combination, we construct a node's final embedding as the weighted sum of its embeddings on all layers. This improves the performance of LightGCN . We conducted experiments to demonstrate the strengths of LightGCN in being simple, easier to be trained, better generalization ability, and more effective.We believe the insights of LightGCN are inspirational to future developments of recommender models.

REFERENCES

[1] Y. Koran and R. B. and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *IEEE Comput. Soc.*, no. August, pp. 42–49, 2009.

[2] R. Ravanifard, · Wray Buntine, and · Abdolreza Mirzaei, "Recommending content using side information," *Appl. Intell.*, vol. 51, pp. 3353–3374, 2021.

[3] Barjasteh, I. "Matrix completion with side information for effective recommendation" [Michigan State University]. In *ProQuest Dissertations and Theses*, 2016. https://search.proquest.com/docview/1864680320?accountid=14169

[4] Mei, J., de Castro, Y., Goude, Y., Azaïs, J. M., &Hébrail, G, "Nonnegative Matrix Factorization with Side Information for Time Series Recovery and Prediction," *IEEE Transactions on Knowledge and Data Engineering*, *31*(3),2019 https://doi.org/10.1109/TKDE.2018.2839678

[5] Symeonidis, P., &Malakoudis, D, "Multi-modal matrix factorization with side information for recommending massive open online courses,"*Expert Systems with Applications*, *118*,*2019*. ("Recent Developments in Recommender Systems | SpringerLink") https://doi.org/10.1016/j.eswa.2018.09.053

[6] Zhao, H., Yao, Q., Song, Y., Kwok, J. T., & Lee, D. L," Side Information Fusion for Recommender Systems over Heterogeneous Information Network,"*ACM Transactions on Knowledge Discovery from Data*, *15*(4), 2021. https://doi.org/10.1145/3441446

[7] Babkin, A. "Incorporating side information into Robust Matrix Factorization with Bayesian

[8] Quantile Regression," *Statistics and Probability Letters*, *165*. 2020. https://doi.org/10.1016/j.spl.2020.108847

[9] Behera, G., & Nain, N, "Handling data sparsity via item metadata embedding into deep collaboative recommender system," *Journal of King Saud University - Computer and Information Sciences*, 2022. https://doi.org/10.1016/j.jksuci.2021.12.021

[10] X. Wang, X. He, M. Wang, F. Feng, and T. S. Chua, "Neural graph collaborative filtering," *SIGIR 2019 - Proc. 42nd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 165–174, 2019.

[11] Huang, T., Zhang, D. and Bi, L., "Neural embedding collaborative filtering for recommender systems", Neural Computing and Applications, vol.32, pp.17043-17057, 2020.

[12] Sun, X., Zhang, H., Wang, M., Yu, M., Yin, M. and Zhang, B., "Deep Plot-Aware Generalized Matrix Factorization for Collaborative Filtering", Neural Processing Letters, vol.52, no.3, pp.1983-1995, 2020.

[13] [13] M. Fu, H. Qu, Z. Yi, L. Lu, and Y. Liu, "A novel deep learning-based collaborative filtering model for a recommendation system," *IEEE transactions on cybernetics*, vol. 49, no. 3, pp. 1084-1096, 2018.

[14] [14] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.S. Chua, "Neural collaborative filtering," In *Proceedings of the 26th international conference on world wide web*, pp. 173-182, 2017.
[15] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, Tat-Seng Chua, " Neural Graph Collaborative Filtering".
[16] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, Meng Wang, "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation".
[17] V. S. Rao, V. Mounika, N. R. Sai and G. S. C. Kumar, "Usage of Saliency Prior Maps for Detection of Salient Object Features," 2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2021, pp. 819-825, doi: 10.1109/I-SMAC52330.2021.9640684