Computers, Materials & Continua

Tech Science Press

DOI: xxx
Type: xxx

Weighted Forwarding in Graph Convolution Networks for Recommendation Information Systems

Sang-min Lee[1] and Namgi Kim[2, *]

[1,2]Department of Computer Science, Kyonggi University, Korea, Republic of
[*]Corresponding Author: Namgi Kim. Email: ngkim@kyonggi.ac.kr
Received: XX Month 202X; Accepted: XX Month 202X

Abstract: Recommendation Information Systems (RIS) assist users in swiftly locating desired content from the vast amount of information available on the Internet. Graph Convolution Network (GCN) algorithms were employed to implement the RIS efficiently. However, the GCN algorithm faces limitations in terms of performance enhancement owing to the embedding value-vanishing problem that arises during the learning process. To address this issue, we propose a Weighted Forwarding method using the GCN (WF-GCN) algorithm. The proposed method multiplies the different weights for the embedding results generated during graph learning for each hop layer. By applying the WF-GCN algorithm, which multiplies the weights for each hop layer before forwarding to the next layer, nodes with many neighbors attain higher embedding values, leading to the effect of learning more hop layers in the GCN. The efficacy of the WF-GCN was demonstrated through its application to various datasets. In the MovieLens dataset, the implementation of WF-GCN in LightGCN achieved improved performance, with recall and Normalized Discounted Cumulative Gain (NDCG) improving by up to +163.64% and +132.04%, respectively. Similarly, in the Last.FM dataset, LightGCN using WF-GCN substantially improved, with the recall and NDCG surging by up to +174.40% and +169.95%, respectively. Furthermore, the application of WF-GCN to Self-supervised Graph Learning (SGL) and Simple Graph Contrastive Learning (SimGCL) also demonstrated notable improvements in both the recall and NDCG metrics across these datasets.

Keywords: Deep Learning; Graph Neural Network; Graph Convolution Network; Graph Convolution Network model learning method; Recommender Information Systems

1 Introduction

We are surrounded by a lot of information on the Internet and Recommendation Information Systems (RIS) help us quickly find our desired content. RIS, which is employed in various domains such as e-commerce, entertainment, and social media, is used on platforms such as YouTube, Netflix, and Amazon to provide users with desired content. The RIS analyzes user' preferences and behavioral patterns to suggest content tailored to them, offering users new opportunities for discovery, which increases their loyalty to the content platform. Continuously providing users with relevant and intriguing content encourages them to spend more time on the platform, thereby enhancing its profitability.

Various approaches have been employed to effectively construct an RIS. The Collaborative Filtering (CF) approach [1] provides recommendations for between users and items by calculating the similarity between users or items based on past user evaluations. However, the CF-based RIS model, which only uses user-item interaction data, such as user purchase records, for recommendations, has performance limitations. In addition, as the number of users increases, the computation required to determine the similarity between users increases, leading to a scalability problem. Recently, RIS models utilizing Graph Convolution Network (GCN) algorithms have been actively investigated to overcome these problems. However, the GCN algorithm updates the embeddings by combining information from adjacent nodes across multiple layers. As the network continuously averages information from adjacent nodes, the local features become diluted. Consequently, all nodes in the GCN algorithm converge to embeddings with similar features. This causes the learned network not to recognize local differences but to reflect overall features, leading to the embedding value vanishing problem where information becomes generalized and disappears. This reduces the diversity of recommendations and degrades performance [2–9]. The problem of disappearing embedding values occurs in models that use embedding layers in the deep learning training process, such as recommendation systems. These layers convert raw data (e.g. words or user/item IDs) into vectors of real numbers, where each element represents a learned feature. Although these vectors are essential for capturing meaningful relationships between inputs, several problems arise during the learning process. The loss of the gradient can halt the effective updating of the model weights (and hence embeddings). This causes the model to struggle to learn effective embeddings for all items or users, and runs the risk of the model becoming too complex or not capturing enough information. In addition, embeddings from different inputs can converge to similar vectors, thereby losing the ability to distinguish between different data points. To address these issues in this study, we propose Weighted Forwarding method for the GCN (WF-GCN). It multiplies the weights with a embedding values obtained from the learning of the previous layer and forwards them to the next layer during the hop hierarchy learning process of the GCN algorithm. Therefore, the proposed method suppresses the disappearance of embedded values and learns more hop hierarchies, even with fewer hop layers in GCN-based RIS models.

Additionally, mathematical statistics and dimensionality reduction techniques were used to analyze the reasons for the performance improvement of the proposed WF-GCN algorithm, and the stability of various forms of neural networks was analyzed by mathematical methods in studies such as References [10–11].

2 Related Work

Traditional RIS using the CF approach can be broadly divided into two methods: the memory-based (or user-based) method, which identifies a group of users similar to the active user, from the entire user-item database, and the model-based method, applying machine learning algorithms based on Bayesian networks, clustering, and rule-based techniques to create a model of user ratings and then recommends content based on this model. The CF approach generates recommendation content on the premise that new users will also like products that similar users have liked in the past, based on the existing users' database. Approaches based on CF include tapestry [1], Bayesian networks based on decision trees [12], clustering that groups similar users [13], and graph-based techniques that represent users as nodes and similarities between users as edges [14]. However, neither CF nor Matrix Factorization (MF) approaches can integrate and utilize various contextual information, such as user evaluation records of items, user session information, and item category information.

Recent proposals have incorporated Deep Learning (DL) techniques that utilize user experience data to better capture the correlation between users and items, and overcome the limitations of traditional CF and MF approaches. The graph neural network (GNN) algorithm [15] is a commonly used DL model fot RIS. Specifically, a particular form of the GNN algorithm, the GCN, is widely used in RIS owing to its highly suitable architecture for implementing RIS. The GCN succinctly aggregates a graph's node and its surrounding information through convolutional operations, facilitating the learning and optimization of GCN-based RIS models. The GCN algorithm can effectively reflect the complex interactions between users

and items specializing in processing data from graph architectures. To reflect these interactions, the GCN algorithm can integrate additional information about the relationship between users and items (such as user gender, age, and item category), thereby overcoming the limitations of traditional MF and CF approaches, which struggle to fully reflect the intricate interactions between users and items.

The GCN-based RIS model aggregates information from nodes and their surroundings to produce high-dimensional embeddings that effectively reflect the intricate features of users and items. Examining the process step-by-step, the GCN-based RIS model initially represents the interactions between users and items as graphs, showing the different relationships between them. Therefore, the derived graph-based modeling results were learned using the GCN algorithm, which ultimately determined the embedding values for the users and items. The RIS model then calculates preference scores between users and items using the determined user and item embedding values, and based on these scores, it effectively recommends content. The GCN algorithm can easily represent information for new users or items in a graph architecture, alleviating the cold-start problem compared with pure CF or MF approaches. Additionally, with active research on various graph architectures and algorithms, the GCN algorithm possesses flexibility, allowing its application across diverse recommendation scenarios [16–18].

In RIS, algorithms such as LightGCN [19], Self-supervised Graph Learning (SGL) [20], and Simple Graph Contrastive Learning (SimGCL) [21] have demonstrated impressive performance. LightGCN, a simplified version of the CF-based Neural Graph Collaborative Filtering (NGCF) [22], streamlines the GCN by removing feature transformation, non-linear activation, and self-connection from the GCN propagation layer, thereby simplifying the complex weight learning process. LightGCN strengthens the interactions between nodes by utilizing an interaction graph between users and items. Consequently, it generates node embeddings using only the connection information between users and items, thereby enhancing recommendation performance. Instead of the weight learning of the GCN algorithm, LightGCN creates node embeddings using the interaction graph between nodes, thereby significantly reducing the learning and inference times. With its simple and fast architecture, LightGCN can deliver high performance in various recommendation scenarios with minimal learning, making it highly valuable for large-scale RIS.

SGL is an application of the self-supervised learning model to the LightGCN algorithm, utilizing relationships with other nodes in the graph for reconstruction through Structure Reconstruction (SR). Using SR, SGL randomly removes nodes or edges from the graph and reconstructs them, thereby augmenting the learning data. Augmented data can undergo graph learning using self-supervised learning, without separate labeling. Therefore, the SGL has the advantage of learning through unlabeled graphs.

SGL, which augments the graph to increase the learning data, requires significant resources and time for learning. The SimGCL method was proposed to overcome these constraints and achieve efficient learning with fewer resources and less time. SimGCL accurately learns solely through contrastive learning in the learning process of graph neural networks without augmenting the learning data. It uses a contrastive loss function to group similar samples and performs contrastive learning to distinguish between the grouped samples. Consequently, it improves accuracy and diversity by applying only contrastive learning to LightGCN without graph augmentation in the GCN-based RIS model.

These GCN-based RIS models demonstrate impressive performance across various content recommendation services. However, GCN algorithms are used in the RIS face embedding value vanishing problem, in which the embedding value converges to a significantly small value during learning. Therefore, this paper proposes a WF-GCN algorithm that multiplies the embedding value with a weight before the propagation rule and forwards it to the next layer, thereby alleviating the embedding value vanishing problem.

3 Proposed Method

3.1 Learning and inference process for the general GCN algorithm

Fig. 1 illustrates the generalized learning architecture of the general GCN algorithm applied to RIS.

As shown in the figure, the general GCN algorithm initially generates a value through a matrix of user-item behavior records during the early stages of learning. The created initial layer, known as shallow embedding, is represented as $e_u^{(0)}$ and $e_i^{(0)}$ is used for learning. In the subsequent layers, it propagates according to the embedding propagation rule (Eq. (1)). Here, a hop signifies the distance between nodes in a graph, and the layers in the GCN algorithm aggregate the information from surrounding nodes. Therefore, as the propagation rule progresses, after passing through one, two, and N layers, it aggregates information from the 1-hop neighbor of the target node to the 2-hop neighbor and up to the N-hop.

$\mathcal{N}_u$ and $\mathcal{N}_i$ denote the connected nodes of users u and item i, respectively. By normalizing, the symmetric normalization term (=$1/\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}$), ensures that the embedding value does not increase excessively during the convolution operation. $e_u^{(k)}$ is the embedding of user u at the k^{th} layer and $e_i^{(k)}$ is the embedding of item i in the k^{th} layer. The previous embedding values of the neighbors of user u and item i are summed with the normalized weight to obtain the embedding value of the $k+1^{th}$ layer, which includes the k^{th} hop in the graph.

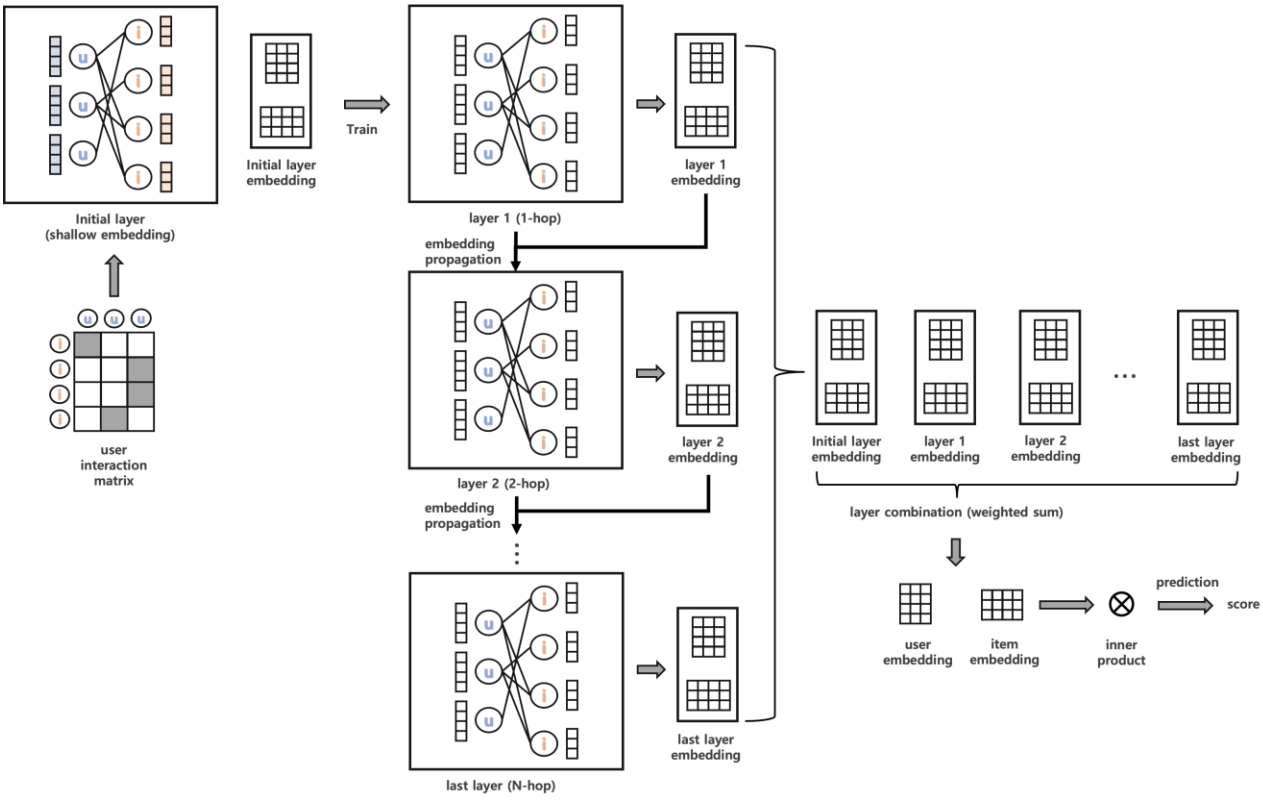

Figure 1: Learning and inference process for a general GCN algorithm

The general GCN algorithm sums the embeddings from each layer according to the propagation rule to obtain the final embedding vector for the users and items (as shown in Eq. (2)). Here, k represents the index of the layer and a_k denotes the importance of embedding from the k^{th} layer. In GCN-based RIS models, which commonly use LightGCN, SGL, and SimGCL, a_k is set to $1/(k+1)$.

$$e_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} e_i^{(k)}; \quad e_i^{(k+1)} = \sum_{i \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} e_u^{(k)} \tag{1}$$

$$e_{\mathrm{u}} = \sum_{k=0}^{n} \alpha_k \mathrm{e}_{\mathrm{u}}^{(k)}; \; e_{\mathrm{i}} = \sum_{k=0}^{n} \alpha_k \mathrm{e}_{\mathrm{i}}^{(k)} \tag{2}$$

$$\hat{y}_{ui} = e_u^T e_{\mathrm{i}} \tag{3}$$

In a general GCN algorithm, the embedding for each layer is calculated according to the propagation rule from the initial layer, and the embedding values of each layer are weighted and summarized to compute the final embedding values e_u for users and e_i for items. The preference score $\hat{y}_{ui}$ between user u and item i is calculated by taking the inner product of e_u and e_i as shown in equation 3. The traditional general GCN algorithm maintains the self-connected effect of graph convolution by summing the embedding values derived from the previous hop layer, thereby inducing a comprehensive representation. However, the general GCN algorithm faces an embedding value-vanishing problem by predicting using only the last layer. The last layer is the sum and division of the embeddings of all the layers, where the embedding value diminishes as it passes through each layer.

3.2 Proposed learning and inference process with the WF-GCN algorithm

In Section 3.1, where the general GCN algorithm is described, a statistical analysis of each layer's embedding value and the final embedding layer value revealed that the average value of the final embedding layer was 4.14×10^{-7} indicating that the overall embedding value was learned locally, leading to the embedding value vanishing problem. Consequently, the traditional general GCN algorithm robustly learns only the item features that users consistently view, undermining the intrinsic purpose of the RIS, which is to discover new and highly relevant content that users may not have encountered through other means, thereby reducing the probability of such discoveries.

Fig. 2 illustrates the architecture of the learning process (WF-GCN algorithm) proposed in this study to overcome the problem of disappearing embedded values. The proposed architecture obtains a more sharpened embedding value ($e'^{(k+1)}_u, e'^{(k+1)}_i$) by transmitting a value, which is the result of multiplying a specific weight to the node embedding value, to the next layer as described in Eq. (4).

Using the embeddings obtained from Eq. (4), the proposed method attempted to overcome the problem of embedding loss by assigning weights to each embedding transfer layer, and the results of the experiment confirmed that the average value of the final embedding layer significantly increased to 3.19×10^{-2} through the WF-GCN algorithm. Using this method, the performance of the existing recommendation information system was improved, and the cause of the increase in recommendation performance was identified through various analyses.

$$e'^{(k+1)}_u = w_k \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} e'^{(k)}_i; \quad e'^{(k+1)}_i = w_k \sum_{i \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} e'^{(k)}_u \tag{4}$$

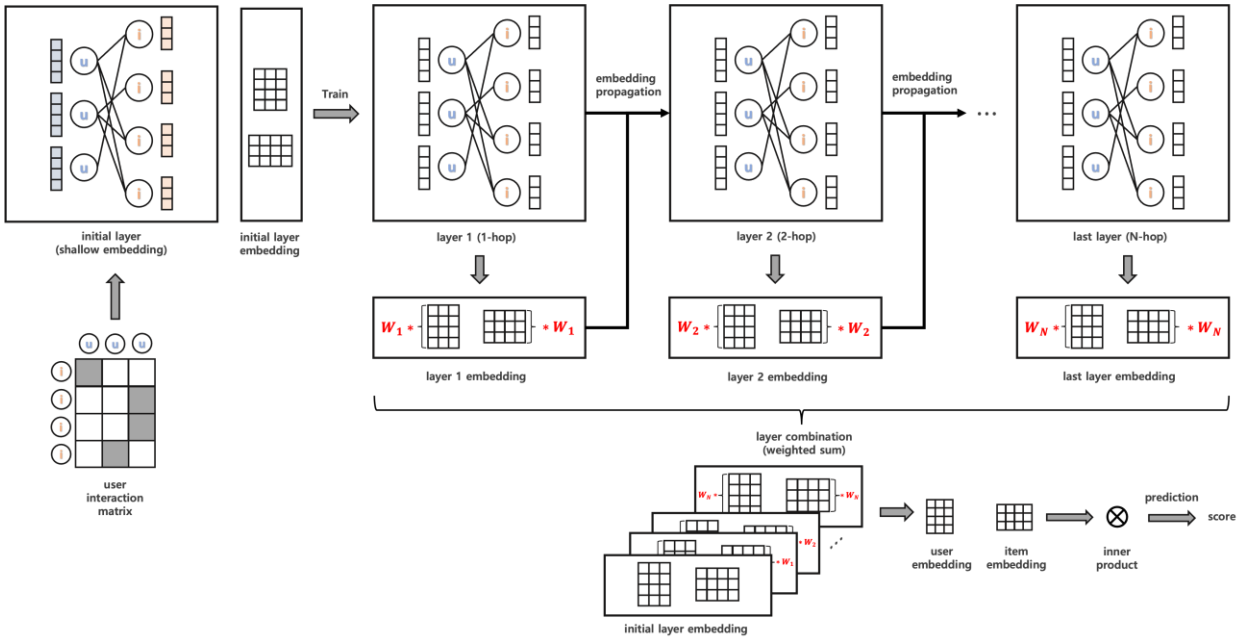

Figure 2: Proposed learning and inference process with WF-GCN algorithm

3.3 Equation Expressions

In this section, we describe the proposed WF-GCN algorithm based on the equations used in the LightGCN algorithm [19].

The user-item interaction matrix R can be represented by Eq. (5). where M and N denote the number of users and items, respectively.

$$R \in \mathbb{R}^{M \times N} \tag{5}$$

The adjacency matrix A of the user-item graph can be represented as in Eq. (6). A diagonal matrix refers to a matrix in which all elements outside the diagonal have values of 0.

$$A = \begin{pmatrix} 0 & R \\ R^T & 0 \end{pmatrix} \tag{6}$$

The embedding matrix of the 0^{th}layer is represented by Eq. (7), where T denotes the embedding size. The embedding value of the $k + 1^{th}$ layer, calculated according to the propagation rule, is expressed by Eq. (8). In this context, D refers to a diagonal matrix with the dimensions M + $(M + N) \times (M + N)$. Eq. (9) is the process of calculating the predicted value from the final embedding of the existing models.

$$E^{(0)} \in \mathbb{R}^{(M \times N) \times T} \tag{7}$$

$$E^{(k+1)} = \left(D^{-\frac{1}{2}} A D^{\frac{1}{2}}\right) E^{(k)} \tag{8}$$

$$E = \sum_{k=0}^{n} \alpha_k E^{(k)} \rightarrow \hat{y}_{ui} \tag{9}$$

Eq. (10) represents the mathematical expression of the proposed WF-GCN algorithm: The WF-GCN algorithm determines the $k + 1^{th}$ embedding value by multiplying weight w_k by $E^{(k)}$, which emerges after

passing through the learning process of the k^{th} layer. Through w_k, different weights can be assigned to each layer, allowing for adjustment of the influence on the embedding.

$$E'^{(k+1)} = w_k \left(D^{-\frac{1}{2}} A D^{\frac{1}{2}} \right) E^{(k)} \tag{10}$$

Eq. (11) represents the embedding obtained after completing the proposed learning process. Here, E' denotes the vector containing the last embedding value for both users and items. Using this last embedding vector, we can obtain the predicted value $\hat{y}_{ui}$ to assess the recommendation performance.

$$E' = \sum_{k=0}^{n} \alpha_k E'^{(k)} \rightarrow \hat{y}_{ui} \tag{11}$$

4 Experiments

In this study, we proposed a WF-GCN algorithm to prevent the embedding value vanishing problem, which hinders the recommendation performance in RIS. LightGCN, SGL, and SimGCL are used to verify the proposed learning process. All of these models are based on LightGCN and share the same architecture for layer combination (weighted sum), making it possible to apply the suggested method uniformly across the three models. The experiment employed RecBole [23], an open-source library designed for GCN-based RIS models. This library, built using PyTorch, incorporates various recommendation algorithms to facilitate RIS research and development. In the experiment [24], we applied the suggested method to the models implemented using RecBole. The data, environment, and parameters used in the experiment are detailed in sections 4.1 and 4.2, while the evaluation functions for the model recommendation performance evaluation are presented in Section 4.3. An analysis of the experimental results is presented in Section 4.4.

4.1 Dataset

In the experiment, we utilized two types of datasets, listed in Table 1. The first is the MovieLens dataset [25], which is frequently used for movie RIS and contains users' ratings and interaction information regarding movies. By leveraging the user context, the interactive relationship between users and movies can be represented as a graph. By learning these interactions, movie recommendations are made, allowing for the assessment of which model performs better on movie recommendation tasks using this dataset. The size of this dataset was divided based on the number of viewing records of the users.

In addition, experiments were conducted to evaluate the performance of the proposed method. The LAST.FM dataset [26] is a large-scale music dataset commonly used in research on music RIS. This includes users' music listening records and artists' information. The interaction information between users and music was utilized for the RIS and music-related analysis.

Table 1: Datasets used in the experiment

	MovieLens-100K	MovieLens-1m	MovieLens-10m	Last.FM
User	944	6,040	69,864	1,886
Item	1,575	3,629	10,599	17,389
Interaction	82,520	836,478	8,242,124	91,779
Sparsity	94.45%	96.18%	98.89%	99.72%

4.2 Experimental Environments

The experiments were conducted under the conditions listed in Table 2. Table 3 presents the variables related to the experimental environment, and Table 4 lists the parameters used in the model. Fig. 3 shows an example of random ordering and ratio-based splitting. If User A was associated with 20 items, they were

randomly extracted based on the set ratio used for the experiment. The data split ratio was set as follows: training: 80%, validation: 10%, and test: 10%. Therefore, of the 20 items, 16 were used for learning, 2 for validation, and 2 for performance evaluation. The validation and test data were not used in the experiment.

Additionally, we investigated the impact of varying the original parameters of the LightGCN model on different datasets (Last.FM, MovieLens-100K, MovieLens-1M, and MovieLens-10M). We found that with a learning rate of 0.001, the performance on the Last.FM dataset decreased for all performance metrics (precision, recall, NDCG, MRR, and hit ratio) compared with a learning rate of 0.002. For the MovieLens-100K dataset, the precision, recall, NDCG, and MRR increased slightly, but the hit values decreased. For the MovieLens-1M dataset, we observed mixed results with decreases in precision, recall, and hits but increases in NDCG and MRR. Finally, the MovieLens-10M dataset exhibited a significant decrease in all performance metrics. The study concluded that changing the learning rate in the LightGCN model did not significantly affect the results or hinder the performance, suggesting that numerical comparisons between parameters may not be important in this context. These results were used to evaluate the performance changes when applying the WF-GCN algorithm, which shares its basic structure with LightGCN, using the same parameters (learning rate = 0.002) in all the experiments.

Table 2: Experimental environments

Resource	Property
CPU	Intel i9-12900k
GPU	RTX 3090 24 GB
OS	Ubuntu 18.04.06 LTS
Epoch	500
Seed	42
Optimizer	Adam

Table 3: Environment variables

Data split	Random Split, Train: 80%, Validation: 10%, Test: 10%
Order	Random Ordering, Ratio-based Splitting
Mode	Full ranking with all item candidates
Epochs	500
Validation metric	MRR@20
Batch size	Train batch size: 32,768, Eval batch size: 4,096,000 (SGL's Train batch size: 16,384)

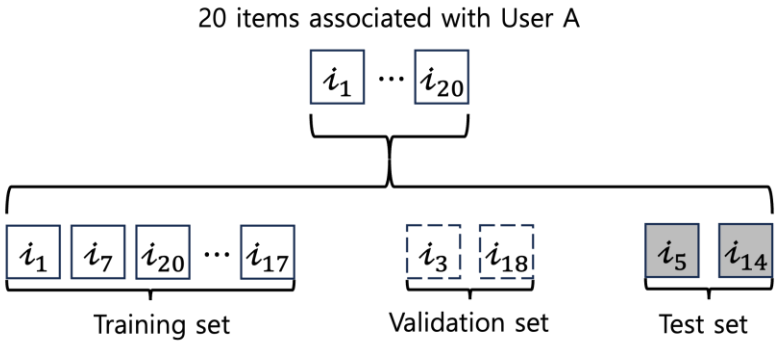

Figure 3: Random ordering, ratio-based splitting's example

Table 4: Model-specific parameters

LightGCN		SGL		SimGCL	
Embedding size	64	Embedding size	64	Embedding size	64
Learning rate	0.002	Learning rate	0.002	Learning rate	0.002
Reg weight	0.0001	Reg weight	0.0001	Reg weight	0.0001
		SSL tau	0.5	Temperature	0.05
		SSL weight	0.005	Lambda	1e-5
		Drop ratio	0.1	Eps	0.1
		Augmentation Type	Edge Drop (ED), Node Drop (ND), Random Walk (RW)		

4.3 Experimental Evaluation

For the evaluation of the model's performance, precision, recall, Normalized Discounted Cumulative Gain (NDCG), Mean Reciprocal Rank (MRR), and hit indicators were used to assess the quality of the model from various perspectives. Precision and recall are used to evaluate the accuracy and diversity of the recommendations. NDCG and MRR assess how accurately the items of interest to the user are recommended in the top ranks. A hit evaluates the success of the RIS by suggesting at least one relevant item for the user.

Precision@K indicates the proportion of items recommended by the RIS that the user is interested in. Similarly, it measures the accuracy with which a user recommends their preferred. High precision suggests that RIS accurately recommends items that are highly relevant to the user. Recall@K represents the proportion of items in which the user is interested, as recommended by RIS. This metric assesses the number of relevant items the RIS has recommended, with a high recall indicating that the RIS suggests many pertinent items without missing any items. NDCG@K evaluates the relevance and ranking of the recommended items, with higher values when the user's preferred items are ranked higher. Thus, a high NDCG value suggests that the RIS accurately recommends items of interest to users with higher rankings. MRR@K is the average rank of the first appearance of an item of interest among the recommended items. A high MRR indicates that the RIS quickly identifies the items preferred by the user. Hit ratio@K represents the proportion of items that the user prefers to be successfully included in the recommendation list. This metric measures whether the RIS recommends at least one item of interest to the user, with a high Hit Rate indicating that the RIS effectively includes items that the user prefers.

In Section 4.4, we summarize the recall and NDCG results, which are the main performance indicators used in previous studies. The results of the experiment are available at this link: https://wandb.ai/d9249/WFGR.

4.4 Experimental Result

A summary of the experimental results is presented in Tables 5, 6, 7 and 8. The [1,1,1] (base, non-weight) entry represents the results of a reproduction experiment to understand the basic performance of each model. The link provides results that compare model performance as the number of layers increases. According to the link, the layers exhibiting the best performance vary according to the data and model. Therefore, in this study, we conducted experiments based on three layers, which is consistent with the results of previous studies. The link displays the comprehensive experimental results not summarized in the tables, and it is evident that the proposed WF-GCN algorithm enhances performance across all indicators.

Table 5 presents the results for the MovieLens-100k dataset, Table 6 for the MovieLens-1m dataset, Table 7 for the MovieLens-10m dataset, and Table 8 for the Last.FM dataset. These tables summarize the

experimental results for the three model datasets obtained when the proposed WF-GCN algorithm was applied. We observed a clear performance improvement in the RIS. However, even if the type of data remains the same, there are differences in the recommendation performance depending on the size of the data, and each model has its own optimal weight. Specifically, LightGCN showed a performance improvement of approximately 200% compared with its original performance, whereas SGL and SimGCL showed performance improvements of 3–8%. These results suggest that the removal of non-linear activation from the NGCF to increase the accuracy of LightGCN overcomes these problems. Moreover, not only can the commonly researched approach of aggregating node information further enhance the recommendation performance, but it can also properly analyze and utilize the model's layer.

Table 5: Experimental results of applying the proposed method to model (layer 3, MovieLens-100k)

	LightGCN		SGL(ED)		SimGCL	
Weight	Recall	NDCG	Recall	NDCG	Recall	NDCG
[1, 1, 1] (base)	0.1474 (0%)	0.1314 (0%)	0.364 (0%)	0.2824 (0%)	0.3781 (0%)	0.2948 (0%)
[2, 2, 2]	0.176 (+19.40%)	0.137 (+4.26%)	0.3743 (+2.83%)	0.2928 (+3.68%)	0.3579 (-5.34%)	0.2846 (-3.46%)
[4, 4, 4]	0.1847 (+25.31%)	0.1431 (+8.90%)	0.3727 (+2.39%)	0.293 (+3.75%)	0.3271 (-13.49%)	0.2433 (-17.47%)
[8, 8, 8]	0.2839 (+92.61%)	0.2173 (+65.37%)	**0.3759 (+3.27%)**	**0.2958 (+4.75%)**	0.2999 (-20.68%)	0.2283 (-22.56%)
[16, 16, 16]	0.3217 (+118.25%)	0.25 (+90.26%)	0.3751 (+3.05%)	0.292 (+3.40%)	0.3466 (-8.33%)	0.2684 (-8.96%)
[32, 32, 32]	0.3598 (+144.10%)	0.2828 (+115.22%)	0.3757 (+3.21%)	0.2934 (+3.90%)	0.3778 (-0.08%)	0.2988 (+1.36%)
[64, 64, 64]	**0.3886 (+163.64%)**	**0.3049 (+132.04%)**	0.3745 (+2.88%)	0.2878 (+1.91%)	**0.3906 (+3.31%)**	**0.3045 (+3.29%)**
[128, 128, 128]	0.3869 (+162.48%)	0.297 (+126.03%)	0.3573 (-1.84%)	0.2846 (+0.78%)	0.3604 (-4.68%)	0.2743 (-6.95%)
[192, 192, 192]	0.3394 (+130.26%)	0.2618 (+99.24%)	0.3369 (-7.45%)	0.2637 (-6.62%)	0.3019 (-20.15%)	0.2268 (-23.07%)
[256, 256, 256]	0.3238 (+119.67%)	0.2393 (+82.12%)	0.3078 (-15.44%)	0.2401 (-14.98%)	0.3028 (-19.92%)	0.2285 (-22.49%)
[512, 512, 512]	0.2712 (+83.99%)	0.2077 (+58.07%)	0.3286 (-9.73%)	0.2539 (-10.09%)	0.2761 (-26.98%)	0.2122 (-28.02%)
[1024,1024,1024]	0.3237 (+119.61%)	0.2432 (+85.08%)	0.3573 (-1.84%)	0.2755 (-2.44%)	0.3326 (-12.03%)	0.2644 (-10.31%)

Table 6: Experimental results of applying the proposed method to model (Layer 3, MovieLens-1m)

	LightGCN		SGL(ED)		SimGCL	
Weight	Recall	NDCG	Recall	NDCG	Recall	NDCG
[1, 1, 1] (base)	0.1233 (0%)	0.108 (0%)	0.2866 (0%)	0.2655 (0%)	0.2853 (0%)	0.2606 (0%)
[2, 2, 2]	0.122 (-1.05%)	0.1166 (+7.96%)	0.2884 (+0.63%)	0.2663 (+0.3%)	0.2819 (-1.19%)	0.2586 (-0.77%)

[4, 4, 4]	0.1288 (+4.46%)	0.1252 (+15.93%)	**0.2914 (+1.67%)**	**0.2696 (+1.54%)**	0.2819 (-1.19%)	0.2541 (-2.49%)
[8, 8, 8]	0.1287 (+4.38%)	0.1247 (+15.46%)	0.2809 (-1.99%)	0.263 (-0.94%)	0.1382 (-51.56%)	0.1358 (-47.89%)
[16, 16, 16]	0.1394 (+13.06%)	0.1353 (+25.28%)	0.287 (-0.49%)	0.267 (+0.56%)	0.1766 (-38.1%)	0.1779 (-31.73%)
[32, 32, 32]	0.1777 (+44.12%)	0.1784 (+65.19%)	0.2906 (+0.76%)	0.2692 (+1.39%)	0.2135 (-25.17%)	0.2121 (-18.61%)
[64, 64, 64]	0.227 (+84.1%)	0.2227 (+106.2%)	0.289 (+0.21%)	0.2671 (+0.6%)	0.2586 (-9.36%)	0.2502 (-3.99%)
[128, 128, 128]	0.2675 (+116.95%)	0.2576 (+138.52%)	0.283 (-1.87%)	0.2636 (+0.72%)	0.29 (+1.65%)	0.2741 (+5.18%)
[192, 192, 192]	0.2869 (+132.68%)	0.2716 (+151.48%)	0.283 (-1.87%)	0.263 (-0.94%)	**0.3016 (+5.71%)**	**0.2805 (+7.64%)**
[256, 256, 256]	**0.3 (+143.31%)**	**0.2814 (+160.56%)**	0.2795 (-3.09%)	0.2598 (-2.15%)	0.2986 (+4.66%)	0.2765 (+6.1%)
[512, 512, 512]	0.2843 (+130.58%)	0.2623 (+142.87%)	0.2615 (-9.33%)	0.2431 (-8.44%)	0.2747 (-3.72%)	0.2531 (-2.88%)
[1024,1024,1024]	0.263 (+113.3%)	0.2475 (+129.17%)	0.25 (-13.31%)	0.238 (-10.36%)	0.2711 (-4.98%)	0.2551 (-2.11%)

Table 7: Experimental results of applying the proposed method to model (layer 3, MovieLens-10m)

	LightGCN		SGL(ED)		SimGCL	
Weight	Recall	NDCG	Recall	NDCG	Recall	NDCG
[1, 1, 1] (base)	0.1511 (0%)	0.1154 (0%)	0.3341 (0%)	0.2894 (0%)	0.3261 (0%)	0.275 (0%)
[2, 2, 2]	0.1623 (+7.41%)	0.1203 (+4.25%)	0.3391 (+1.5%)	0.2922 (+0.97%)	0.3156 (-3.22%)	0.2629 (-4.4%)
[4, 4, 4]	0.1481 (-1.99%)	0.1255 (+8.75%)	0.3443 (+3.05%)	0.2953 (+2.04%)	0.3206 (-1.69%)	0.2693 (-2.07%)
[8, 8, 8]	0.1476 (-2.32%)	0.1251 (+8.41%)	0.3474 (+3.98%)	0.2954 (+2.07%)	0.3237 (-0.74%)	0.2694 (-2.04%)
[16, 16, 16]	0.1474 (-2.45%)	0.1251 (+8.41%)	**0.3508 (+5%)**	**0.2967 (+2.52%)**	0.1534 (-52.96%)	0.1275 (-53.64%)
[32, 32, 32]	0.1953 (+29.25%)	0.158 (+36.92%)	0.3505 (+4.91%)	0.2958 (+2.21%)	0.2051 (-37.11%)	0.1705 (-38%)
[64, 64, 64]	0.2178 (+44.14%)	0.1829 (+58.49%)	0.3455 (+3.41%)	0.2885 (-0.31%)	0.2398 (-26.46%)	0.2041 (-25.78%)
[128, 128, 128]	0.2761 (+82.73%)	0.2339 (+102.69%)	0.3326 (-0.45%)	0.278 (-3.94%)	0.2932 (-10.09%)	0.2523 (-8.25%)
[192, 192, 192]	0.2956 (+95.63%)	0.2509 (+117.42%)	0.3193 (-4.43%)	0.263 (-9.12%)	0.3265 (+0.12%)	0.2821 (+2.58%)
[256, 256, 256]	0.306 (+102.51%)	0.2596 (+124.96%)	0.3143 (-5.93%)	0.2579 (-10.88%)	0.3336 (+2.30%)	0.2863 (+4.11%)

[512, 512, 512]	**0.3227 (+113.57%)**	**0.2729 (+136.48%)**	0.2753 (-17.6%)	0.215 (-35.65%)	**0.3386 (+3.83%)**	**0.287 (+4.36%)**
[1024,1024,1024]	0.298 (+97.22%)	0.2433 (+110.83%)	0.24 (-28.17%)	0.1904 (-34.21%)	0.2978 (-8.68%)	0.2484 (-9.67%)

Table 8: Experimental results of applying the proposed method to model (layer 3, Last.FM)

	LightGCN		SGL(ED)		SimGCL	
Weight	Recall	NDCG	Recall	NDCG	Recall	NDCG
[1, 1, 1] (base)	0.0992 (0%)	0.0812 (0%)	0.2594 (0%)	0.2081 (0%)	0.2626 (0%)	0.2101 (0%)
[2, 2, 2]	0.1038 (+4.64%)	0.0858 (+5.67%)	0.273 (+5.24%)	0.2189 (+5.19%)	0.2612 (-0.53%)	0.2061 (-1.9%)
[4, 4, 4]	0.1132 (+14.11%)	0.0929 (+14.41%)	**0.281 (+8.33%)**	**0.2241 (+7.69%)**	0.1503 (-42.76%)	0.1208 (-42.5%)
[8, 8, 8]	0.169 (+70.36%)	0.136 (+67.49%)	0.2805 (+8.13%)	0.2235 (+7.40%)	0.2082 (-20.72%)	0.1695 (-19.32%)
[16, 16, 16]	0.2058 (+107.46%)	0.1668 (+105.42%)	0.2749 (+7.85%)	0.2206 (+6.01%)	0.2422 (-7.77%)	0.1971 (-6.19%)
[32, 32, 32]	0.2425 (+144.46%)	0.1971 (+142.73%)	0.2725 (+5.05%)	0.2171 (+4.32%)	0.2652 (+0.99%)	0.2144 (+2.05%)
[64, 64, 64]	**0.2722 (+174.40%)**	**0.2192 (+169.95%)**	0.2605 (+0.42%)	0.2059 (-1.06%)	**0.2791 (+6.28%)**	**0.2212 (+5.28%)**
[128, 128, 128]	0.26 (+162.10%)	0.2065 (+154.31%)	0.2347 (-9.52%)	0.1819 (-12.59%)	0.2286 (-12.95%)	0.1821 (-13.33%)
[192, 192, 192]	0.2256 (+127.42%)	0.1764 (+117.24%)	0.2201 (-15.15%)	0.1728 (-16.96%)	0.1986 (-24.37%)	0.1567 (-25.42%)
[256, 256, 256]	0.2054 (+107.06%)	0.1596 (+96.55%)	0.2151 (-17.08%)	0.1652 (-20.62%)	0.2035 (-22.51%)	0.1626 (-22.61%)
[512, 512, 512]	0.2189 (+120.67%)	0.1753 (+115.89%)	0.1975 (-23.86%)	0.1553 (-25.37%)	0.2223 (-15.35%)	0.1761 (-16.18%)
[1024,1024,1024]	0.2147 (+116.43%)	0.1712 (+110.84%)	0.1933 (-25.48%)	0.1523 (-26.81%)	0.2083 (-20.68%)	0.1647 (-37.28%)

To understand the changes in the embedding value of the proposed WF-GCN algorithm, the statistical numerical analysis results of the layer-by-layer embedding of the model without weights are presented in Tables 9 and 10.

In the analysis, the models without assigned weights exhibited a distribution of average values within a narrow range. By contrast, the embedding values of the models with assigned weights generally had a higher average value. The average final embedding with the assigned weights was significantly higher than that without weights, suggesting that the center position of this embedding was higher than that of the other embeddings, as shown in Fig. 4. Moreover, significant increases in the average value and standard deviation indicate that using higher -weight embeddings can lead to a broader distribution of values. The distribution of embedding values in the model with assigned weights is generally spread, suggesting the possibility that the embedding of the model with weights contains diverse information. However, the embeddings of the base model were densely clustered around the average, focusing on specific properties or information. In addition, as the layer deepens, for the distribution and properties of the embedded values tend to change.

Based on this evidence, each layer learns different types of information and properties.

The embedding of the two models exhibited distinct properties and distributions. The model with the assigned weights had data spread widely around the average, encompassing a variety of information. By contrast, the embedding of the base model's embedding has data densely clustered around the average, focusing on specific properties or information. This distinction led to an increase in recommendation accuracy.

Table 9: Statistical analysis of each layer in LightGCN on the MovieLens-1m dataset

Non weight Embeddings	Layer 1	Layer 2	Layer 3	Final
mean	6.29×10^{-7}	3.47×10^{-7}	3.34×10^{-7}	4.14×10^{-7}
standard deviation	3.40×10^{-5}	7.14×10^{-6}	4.67×10^{-6}	1.00×10^{-5}
standard error	3.46×10^{-7}	7.26×10^{-8}	4.75×10^{-8}	1.02×10^{-7}
variance	1.16×10^{-9}	5.10×10^{-11}	2.18×10^{-11}	1.00×10^{-10}
median	1.75×10^{-7}	2.31×10^{-8}	1.38×10^{-8}	7.72×10^{-8}

Table 10: Statistical analysis of each layer of LightGCN (weight=256) on the MovieLens-1m dataset

256 Weight Embeddings	Layer 1	Layer 2	Layer 3	Final
mean	7.62×10^{-5}	4.46×10^{-2}	4.63×10^{-2}	3.19×10^{-2}
standard deviation	1.63×10^{-2}	9.02×10^{-1}	3.92×10^{-1}	3.40×10^{-1}
standard error	1.16×10^{-4}	9.17×10^{-3}	3.99×10^{-3}	3.46×10^{-3}
variance	2.66×10^{-4}	8.13×10^{-1}	1.54×10^{-1}	1.16×10^{-1}
median	6.55×10^{-5}	1.54×10^{-2}	1.13×10^{-2}	9.22×10^{-3}

Figs. 4 and 5 present the analysis results of the embedding values based on Dimensionality Reduction Techniques (DRT) according to weight. For this, we employed two methods: Principal Component Analysis (PCA) and Kernel Density Estimation (KDE) to investigate the distribution and density of the embeddings. For the base case, the PCA results showed data points appearing in multiple clusters, but the boundaries between each cluster were relatively indistinct. The KDE analysis exhibited a similar pattern, with a primary distribution peak present, but distinguishing multiple centroids or clusters was challenging. The embedding with a weight of 256 demonstrated the most outstanding results in all analyses. In PCA, the clustering was the most defined, and almost no outliers were observed. The KDE analysis also displayed the most pronounced peaks and density areas.

In conclusion, embedding the model using a weight of 256 through the proposed WF-GCN algorithm suggests that it captures properties better than the existing model. The performance increase achieved through the suggested method was confirmed using the DRT.

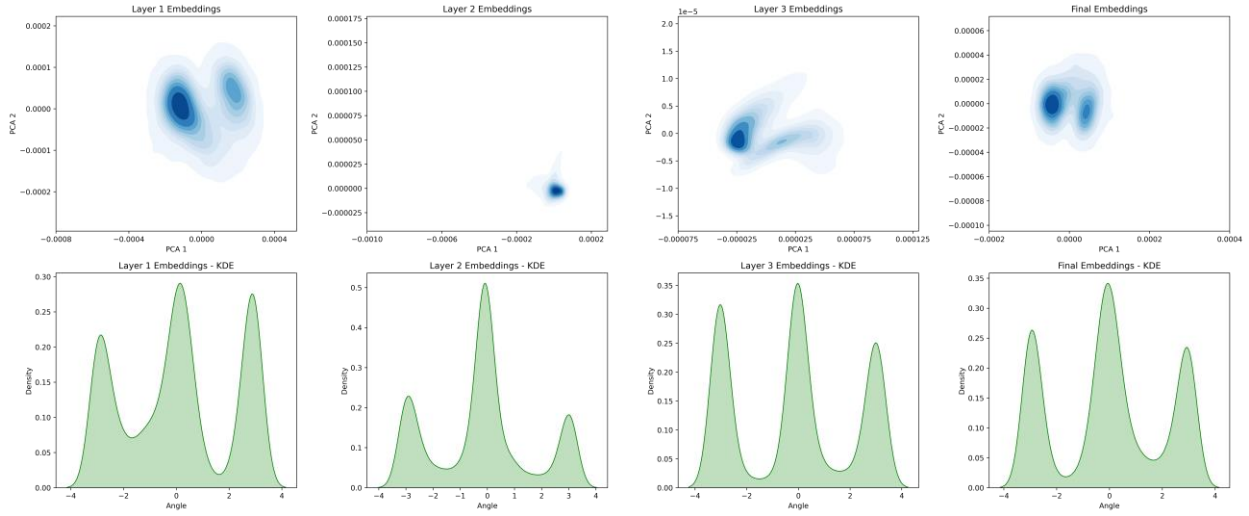

Figure 4: Visualize LightGCN (non weight), PCA and KDE on the MovieLens-1m dataset

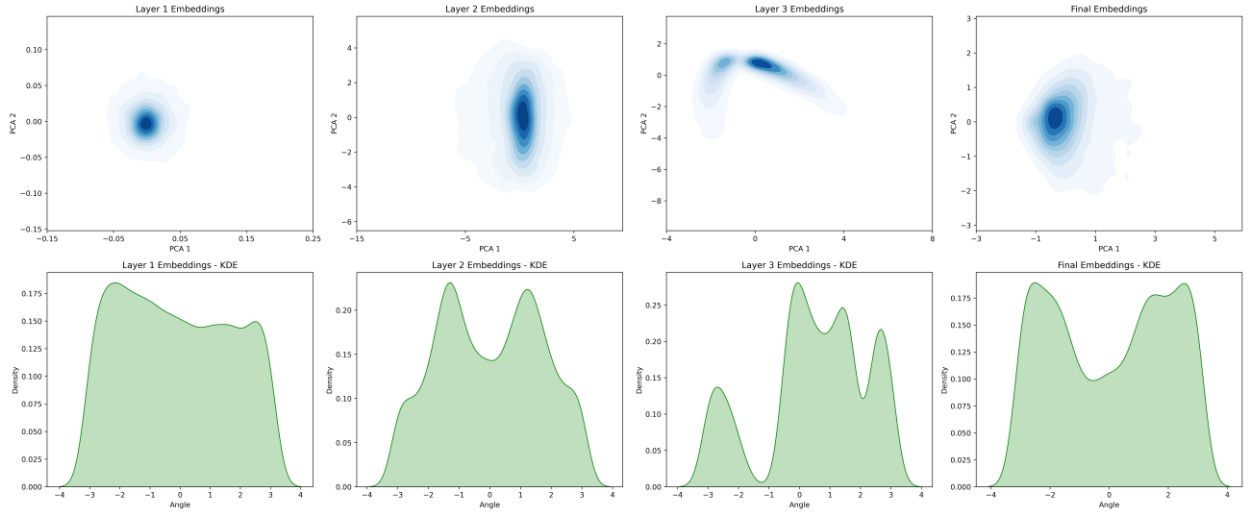

Figure 5: Visualize LightGCN (weight=256), PCA and KDE on the MovieLens-1m dataset

Additionally, we conducted an experiment in which different numbers, rather than common numbers, were inserted into a weight array. The experimental results are listed in Table 11. These results indicate that the performance of RIS varies significantly depending on the value provided by the propagation rule of the initial layer. Moreover, by merely applying an appropriate weight to the initial embedding, the recommendation performance can be drastically improved compared with the previous performance.

Table 11: Experimental results to determine the importance of each layer based on its weight (LightGCN, layer 3, MovieLens-1m)

Weight	Precision	Recall	NDCG	MRR	Hit
[1, 1, 1] (base)	**0.0695** (0%)	**0.1249** (0%)	**0.108** (0%)	**0.1937** (0%)	**0.6062** (0%)
[256, 256, 256] (ours)	**0.1532 (+120.43%)**	**0.3 (+140.19%)**	**0.2814 (+160.56%)**	**0.4652 (+140.17%)**	**0.8562 (+41.24%)**
[256, 1, 1]	0.1505 (+116.55%)	0.2937 (+135.15%)	0.2762 (+155.74%)	0.4609 (+137.95%)	0.8551 (+41.06%)

[1, 256, 1]	0.1232 (+77.27%)	0.2213 (+77.18%)	0.2176 (+101.48%)	0.3967 (+104.80%)	0.7681 (+26.71%)
[1, 1, 256]	0.0999 (+43.74%)	0.1709 (+36.83%)	0.1722 (+59.44%)	0.3332 (+72.02%)	0.6936 (+14.42%)

5 Conclusions

In this study, we propose the WF-GCN algorithm, which is an enhancement of the GCN algorithm used in the RIS model. The WF-GCN algorithm multiplies the embedding value of the previous embedding layer by a weight and forwards it to the next layer when calculating the embedding value of each layer in the GCN algorithm. Consequently, by paying more attention to nodes with many neighbors, the WF-GCN algorithm, which learns fewer hop layers, can achieve the effect of learning deeper hop layers. Thus, the WF-GCN algorithm can infer a deeper relationship between users and items without requiring computations to calculate the depth of the hop layer. Therefore, the WF-GCN algorithm can mitigate the embedding value-vanishing problem of conventional GCN-based algorithms.

To verify the performance of the proposed WF-GCN algorithm, we applied the WF-GCN algorithm to representative GCN algorithms, namely LightGCN, SGL, and SimGCL. The performance was evaluated using the MovieLens and Last.FM datasets. In the MovieLens dataset, LightGCN exhibited performance improvements of up to +163.64% in recall and +132.04% in NDCG. The SGL demonstrated an enhancement of up to +5% in recall and +2.52% in the NDCG, whereas the SimGCL exhibited an enhancement of +5.71% in recall and +7.64% in the NDCG. In addition, on the LAST.FM dataset, LightGCN displayed a performance surge of up to +174.40% in recall and +169.95% in NDCG. SGL showed an increase of up to +8.33% in recall and +7.69% in NDCG, whereas SimGCL showed an enhancement of +6.28% in recall and +5.28% in NDCG.

The proposed method employs a layer combination using the weighted sum of the learning process and the resulting final embedding layer for recommendations in GCN algorithms. Our experimental results show that not all layer embeddings are necessary. This can be intuitively seen in the case of [256, 1, 1], where each embedding is weighted differently. If we treat the weight applied to each embedding as a learning parameter and apply a learned end-to-end approach to the weights, we can find the optimal combination of embeddings for higher recommendation performance. Higher recommendation performance can be achieved by improving the layer combination architecture of the GCN algorithms. Therefore, we plan to conduct future research to apply a layer-combination process and end-to-end learning.

Funding Statement: This work was supported by the Kyonggi University Research Grant 2022.

Conflicts of Interest: The authors declare no conflicts of interest.

References

[1] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using Collaborative Filtering to Weave an Information Tapestry,” *Communications of the ACM*, vol. 35, no.12, pp. 61–70, 1992.

[2] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, “Item-Based Collaborative Filtering Recommendation Algorithms,” in *Proc. WWW*, Hong Kong, pp. 285–295, 2001.

[3] K. C. Park and S. Lee, “Investigating Consumer Innovativeness for New Media Infusion: Role of Literacy in the Context of OTT Services in Korea,” *KSII Transactions on Internet and Information Systems*, vol. 16, no.6, pp. 1935-1952, 2022.

[4] M. Goyani and N. Chaurasiya, “A review of movie recommendation system: Limitations, Survey and Challenges,” *ELCVIA: electronic letters on computer vision and image analysis*, vol. 19, no.3, pp. 18–37, 2020.

[5] H. Wang, Z. Le and X. Gong, "Recommendation System Based on Heterogeneous Feature: A Survey," *IEEE Access*, vol. 8, pp. 170779–170793, 2020.

[6] T. L. Ho, A. C. Le and D. H. Vu "Enhancing Recommender Systems by Fusing Diverse Information Sources through Data Transformation and Feature Selection," *KSII Transactions on Internet and Information Systems*, vol. 17, no.5, pp. 1413–1432, 2023.

[7] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton *et al.*, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," in *Proc. KDD*, London, United Kingdom, pp. 974–983, 2018.

[8] S. Zhang, H. Tong, J. Xu and R. Maciejewski, "Graph convolutional networks: a comprehensive review." *Computational Social Networks*, vol. 6, 2019.

[9] P. Chen, J. Zhao and X. Yu, "LighterKGCN: A Recommender System Model Based on Bi-layer Graph Convolutional Networks," *Journal of Internet Technology*, vol. 23, no.3, pp. 621–629, 2022.

[10] G. Rajchakit, P. Agarwal, S. Ramalingam, "Stability Analysis of Neural Networks," in *Springer*, Singapore, 2021.

[11] N. Boonsatit, G. Rajchakit, R. Sriraman *et al.*, "Finite-/fixed-time synchronization of delayed Clifford-valued recurrent neural networks," in *Advances in Difference Equations*, 276, 2021.

[12] N. Friedman, D. Geiger and M. Goldszmidt, "Bayesian Network Classifiers," *Machine Learning*, vol. 29, pp. 131–163, 1997.

[13] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: an open architecture for collaborative filtering of netnews," in *Proc. CSCW*, Chapel Hill, NC, USA, pp. 175–186, 1994.

[14] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An Algorithmic Framework for Performing Collaborative Filtering," in *Proc. SIGIR*, Berkley, CA, USA, pp. 230–237, 1999.

[15] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, "The Graph Neural Network Model," *IEEE Transactions on Neural Networks*, vol. 20, no.1, pp. 61–80, Jan. 2009.

[16] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang *et al.*, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no.1, pp. 4–24, 2021.

[17] K. Xu, W. Hu, J. Leskovec and S. Jegelka, "How Powerful are Graph Neural Networks?" in *Proc. ICLR*, New Orleans, Louisiana, United States, 2019.

[18] J. You, J. Leskovec, K. He, S. Xie, "Graph Structure of Neural Networks," in *Proc. ICML*, Vienna, Austria, pp. 10881–10891, 2020.

[19] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang *et al.*, "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation," in *Proc. SIGIR*, China, pp. 639–648, 2020.

[20] J. Wu, X. Wang, F. Feng, X. He, L. Chen *et al.*, "Self-supervised Graph Learning for Recommendation," in *Proc. SIGIR*, Canada, pp. 726–735, 2021.

[21] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui *et al.*, "Are Graph Augmentations Necessary?: Simple Graph Contrastive Learning for Recommendation," in *Proc. SIGIR*, Madrid, Spain, pp. 1294–1303, 2022.

[22] X. Wang, X. He, M. Wang, F. Feng and T. S. Chua, "Neural Graph Collaborative Filtering," in *Proc. SIGIR*, Paris, France, pp. 165–174, 2019.

[23] W. X. Zhao, Y. Hou, X. Pan, C. Yang, Z. Zhang *et al.*, "RecBole 2.0: Towards a More Up-to-Date Recommendation Library," in *Proc. CIKM*, Woodstock, NY, pp. 4722–4726, 2022.

[24] W. X. Zhao, J. Chen, P. Wang, Q. Gu and J. Wen "Revisiting Alternative Experimental Settings for Evaluating Top-*N* Item Recommendation Algorithms," in *Proc. CIKM*, Ireland, pp. 2329–2332, 2020.

[25] F. M. Harper and J. A. Konstan, "The MovieLens Datasets: History and Context," *ACM Transactions on Interactive Intelligent Systems*, vol 5, no.19. pp. 1–19, 2015.

[26] T. B. Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset," in *Proc. ISMIR*, Miami, Florida, pp. 591–596, 2011.