Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 14, No. 1, 2026, 83-94.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Accuracy Improvement of Collaborative Recommender System Using Deep Learning

Maryam Baghi, Kourosh Kiani* and Razieh Rastgoo

Electrical and Computer Engineering Department, Semnan University, Semnan, Iran.

Article Info

Article History:
Received 28 April 2025
Revised 17 June 2025
Accepted 21 August 2025

DOI:10.22044/jadm.2025.16129.2731

Keywords:
Recommender System, Collaborative Filtering, Recommendation, Clustering, Graph Convolutional Network.

Corresponding author: Kourosh.kiani@semnan.ac.ir (K. Kiani).

Abstract

With rapid advancements in information and communication technology, recommender systems have become vital tools across a wide range of online activities and e-commerce processes. Collaborative recommender systems, which utilize user data and contributions to provide suggestions, represent a significant innovation in this field. In this paper, we conduct an analysis of collaborative recommender systems and evaluate their impact on enhancing the efficiency and accuracy of recommendations. To this end, we propose a deep learning approach using a Graph Convolutional Network (GCN), as a special type of Graph Neural Network (GNN). By assigning weights to edges between nodes, scores are calculated for these edges. The importance of the edges varies based on the number of neighboring nodes and their proximity to the target node. The higher the edge score, the more significant the path. To calculate edge weights, we leverage metrics such as Jaccard similarity, cosine similarity, LHN index, and Salton cosine similarity. This approach improves the identification of relationships between nodes and enhances the accuracy of the recommender system. For implementation, we utilized two well-known datasets: MovieLens and Yelp2018. Ultimately, users were clustered into the clusters, with a large number of nodes within each cluster. By clustering users, we increased the number and diversity of recommendations. This significantly improved the performance of the recommender system, yielding promising results.

1. Introduction

Recommender systems, as powerful tools for providing personalized suggestions to users, are widely used in various domains such as e-commerce, social media, and online streaming platforms [1]. One of the most common techniques in these systems is collaborative filtering, which generates recommendations by analyzing patterns of similarities between users and items [2]. This technique can be broadly categorized into user-based and item-based filtering methods, both of which rely on historical interactions to infer preferences [3]. However, traditional collaborative filtering methods suffer from scalability issues and sparsity in user-item interactions, rendering them less effective for large-scale applications [1]. To overcome these limitations, researchers have explored various machine learning and deep learning approaches, enabling more sophisticated and accurate recommendations [1,4]. Recent advancements in Graph Neural Networks (GNNs) [2] have further enhanced recommender systems by modeling user-item interactions as graph structures, allowing for more nuanced and dynamic representations.

In recent years, Artificial Neural Networks (ANNs) have extensively used in different research areas [5-22]. Specifically, GNNs have emerged as powerful tools for learning graph representations and have been effectively applied to recommender systems. These models enhance user and item representations by employing message passing on graph nodes, whereby each node gathers information from its neighboring nodes [1]. Unlike traditional matrix factorization and deep learning-based methods, GNNs can capture higher-order connectivity and implicit relationships, making them particularly effective for handling sparse data. However, despite their advantages, recent studies indicate that the message-passing mechanisms in GNN models may inadvertently collect irrelevant or even harmful collaborative signals from user-item interactions. These unintended signals can lead to over-smoothing, where the learned representations of nodes become indistinguishable, ultimately degrading recommendation performance [23]. Additionally, noise in user behavior, such as random clicks or exploratory interactions, can further distort the quality of aggregated messages [24]. Addressing these challenges requires novel methodologies to refine message-passing processes and selectively filter out misleading signals to ensure optimal performance.

To tackle these issues, we analyze the message-passing mechanisms in GNN-based recommender systems and build upon an existing approach to enhance its effectiveness. In this work, we adopt the Common Interaction Ratio (CIR) and the Collaboration-Aware Graph Convolution (CAGC) from the base paper and integrate them into our proposed model. CIR quantifies the significance of messages received from neighboring nodes, helping to identify and prioritize relevant collaborative signals while filtering out noisy interactions. Leveraging this metric, CAGC selectively aggregates information from neighbors based on their relevance to the user's actual preferences [2]. Our model extends this foundation by further refining the integration of these techniques, ensuring that only meaningful interactions contribute to the final recommendation. Extensive experiments on real-world datasets demonstrate that our model significantly outperforms existing state-of-the-art approaches, achieving higher recommendation accuracy and robustness.

The remainder of this paper is structured as follows. Section 2 reviews related work on recommender systems and GNN-based approaches. Section 3 elaborates on the proposed methodology in detail. In Section 4, we present our experimental setup, dataset, and evaluation metrics. Section 5 reports the results and performance evaluation. Finally, Section 6 concludes the paper and outlines potential future research directions.

2. Literature review

Recommender systems, born from remarkable advances in cognitive sciences, approximation theory, information retrieval, and prediction theory, have evolved swiftly since their early development, playing a crucial role in shaping today's commercial landscape regardless [4]. Such systems eliminate non-essential data from vast data volumes, selecting items that can fulfill users' latent needs. Traditional recommender systems typically focus on whether users have shown interest in or rated a particular item [23]. By analyzing historical interaction data, they uncover users' latent needs and offer effective recommendations. In 2008, Ma et al. took an innovative step by integrating users' social interaction data with historical item rating data. This pioneering effort introduced the Social Recommendation (SoRec) algorithm, which was based on probabilistic matrix factorization (PMF) [3]. Recognizing the growing importance of social influence on recommendations, studies in this domain experienced exponential growth. In 2010, Jamali further explored how trust spreads among users and their social connections, skillfully combining matrix factorization methods with dynamic trust modeling to construct the Social MF model. GNNs collectively refer to models that apply neural networks to graphs, exhibiting strong capabilities in learning from structured graph data [2,25]. A graph consists of nodes and their connecting edges, representing inherently non-Euclidean spatial data structures. One of the most common examples of graph-based data is social network data. Utilizing GNNs, social network graph models can be seamlessly integrated into recommender systems [2]. Consequently, social recommendation systems have increasingly attracted attention [26]. Leveraging deep learning to merge insights from social networks allows for a more comprehensive understanding of user and item characteristics, ultimately improving recommendation quality and user satisfaction.

Building on this concept, Graph Sample and Aggregate (GraphSAGE) serves as an inductive learning model designed to effectively create embedding functions for unseen vertices using their attribute information [27]. This addresses the issue of models' inability to generalize to new, unseen nodes that were not present during the training phase. Expanding on the GraphSAGE framework, Ying et al. proposed a powerful Graph Convolution Network (GCN) method that integrates random walk strategies with graph convolution operations to learn node embeddings. The model's key advantage is its capability to extend from limited samples to more complex, unfamiliar patterns, making it well-suited for real-world applications where graph structures are continuously evolving [28].

Graph representation learning has been extensively explored in recent years to enhance collaborative filtering (CF) models by capturing high-order collaborative relations [25]. GNNs have proven effective in modeling user-item relationships for recommender systems. These models leverage recursive message passing on user-item graphs to generate embeddings, capturing intricate dependencies. Early approaches, such as Stacked and reconstructed graph convolutional networks (Star-GCN) [29], applied GCNs for mapping interactions into latent embeddings [8]. To simplify message propagation, methods like Lightweight Graph Convolutional Network (LightGCN) [30] and Graph Convolutional Collaborative Filtering (GCCF) [31] removed non-linear transformations, making graph-based recommendations more efficient. Additionally, hyperbolic space representation, as used in Hyperbolic Graph Collaborative Filtering (HGCF) [32], has been introduced to improve embedding quality for graph-based CF.

One study leverages a graph attention-based collaborative filtering model that jointly learns user-item interactions and knowledge graph embeddings via deep neural networks, improving recommendation specificity and explainability [33,34]. A complementary work introduces a deep learning knowledge graph neural network framework that models user, item, and item-attribute relationships to deliver informed and interpretable event recommendations [35]. Another study proposes a multimodal movie recommendation system using a graph convolutional neural network that captures high-order and cross-modal signals to enhance rating prediction [36]. These efforts align with a novel knowledge graph recommendation algorithm based on graph convolution networks aimed at boosting collaborative filtering via structurally rich KG embeddings [37]. Together, these works point toward a growing trend: embedding knowledge graphs into deep learning architectures, whether via attention or convolution, to improve personalization, interpretability, and performance in recommendation tasks. However, most of the graph-based models proved less effective primarily because they treat all neighboring nodes with equal importance when aggregating information. This uniform weighting can lead to suboptimal representations, especially in recommendation scenarios where some interactions or connections are inherently more significant than others. As a result, the model may fail to capture the nuanced patterns of user preferences and item relationships. Recent research has explored self-supervised learning (SSL) to mitigate issues related to data sparsity and noise in recommender systems. A notable SSL paradigm is contrastive learning, where positive and negative sample pairs are strategically aligned and contrasted. Various data augmentation techniques have been employed in this domain, including random graph perturbations, as seen in Self-supervised Graph Learning (SGL) [38], and node embedding modifications. Other works focus on pre-defined alignment methods for structured contrastive learning, such as hypergraph construction in HCCF [39] and user clustering in Neighborhood-enriched Contrastive Learning (NCL) [40]. These advances demonstrate that SSL can effectively enhance recommendation models by leveraging additional structural insights, improving robustness and generalization in real-world scenarios.

3. Proposed model

In this section, details of the proposed model are presented in two main blocks: base model architecture and user clustering. The flowchart of the proposed model is illustrated in Figure 1.

3.1. Model Architecture

In the baseline model, CAGCN (Collaboration-Aware Graph Convolutional Network) is utilized [2]. This model is essentially a GCN with the weighted edges between nodes [41]. To calculate these edge weights, similarity formulas such as Jaccard similarity [42], Salton cosine similarity [43], and the Leicht-Holme-Nerman (LHN) index [44] are employed. The formulas for these metrics are provided below. In all the following formulas, N_i^1 denote the set of first-order neighbors (i.e., directly connected nodes) of node i, and similarly, N_j^1 denotes set of first-order neighbors of node j. These neighbor sets are used to compute structural similarity between nodes.

Jaccard Similarity (JC): The Jaccard similarity measures how similar two nodes are in terms of their neighborhoods by taking the ratio of the intersection of their neighbors to the union of all their neighbors. A higher value indicates a greater degree of similarity, meaning that a larger fraction of their neighbors is shared. It is defined as:

$$Jc(i,j)=\frac{\left|N_i^1 \cap N_j^1\right|}{\left|N_i^1 \cup N_j^1\right|} \tag{1}$$

Salton Cosine Similarity (SC): This metric measures the cosine of the angle between two binary vectors representing the neighborhoods of nodes i and j. A value close to 1 indicates that the nodes share a proportionally large number of neighbors relative to their total number of neighbors. It is calculated as:

$$Sc(i,j)=\frac{\left|N_i^1 \cap N_j^1\right|}{\sqrt{\left|N_i^1 \cup N_j^1\right|}} \tag{2}$$

Leicht-Holme-Nerman (LHN): The LHN index tends to penalize pairs of nodes with high degrees even if they share several neighbors because the actual overlap is significant compared to what might be expected by chance. It is defined as:

$$LHN(i,j)=\frac{\left|N_i^1 \cap N_j^1\right|}{\left|N_i^1\right| \cdot \left|N_j^1\right|} \tag{3}$$

Dice Similarity: A higher Dice score indicates a stronger similarity between nodes based on their shared local connections. It is computed as:

$$Dice(i,j)=\frac{2*\left|N_i^1 \cap N_j^1\right|}{\left|N_i^1\right| + \left|N_j^1\right|} \tag{4}$$

Cosine Similarity: This metric is effective for capturing similarity in terms of direction rather than magnitude, making it suitable for comparing the structure. It is defined as:

$$\text{Cosine}(i,j)=\frac{\sum_{i=1}^{m} N_i N_j}{\sqrt{\sum_{i=1}^{m} N_i^2 \sum_{j=1}^{m} N_j^2}} \tag{5}$$

The expected number of common neighbors is higher in such cases. Therefore, a high LHN index signals that when calculating edge weights, the number of neighbors for each node is considered as the nodes directly connected to the target node. These relationships can have various meanings in the dataset, such as purchasing, rating, liking, commenting, sharing, and other forms of interaction that define the type of connection. This approach of weighting edges based on the specific type of interaction and the similarity between nodes helps capture richer relationships, improving the model's accuracy and relevance in providing recommendations. This architecture operates by calculating the embeddings of users and items while considering the relationships between them. To differentiate the connections between nodes, the aforementioned formulas are utilized. By applying these formulas, weights are assigned to the edges. The higher the similarity between the nodes, the greater the weight of the edge connecting them [45]. This weighted approach allows the model to better capture the significance of the relationships between nodes, ensuring that more meaningful connections contribute more significantly to the embedding process and ultimately improve the quality of recommendations. Let L_hops denote the number of edges linking a node to its neighboring nodes. In the base paper, L_hops was set to 2, meaning that up to 2 edges from the neighborhood of each node were considered. However, in this paper, we extend this to 3 edges to capture a broader range of connections. In Figure 2, a sample relationship between users and items is illustrated. Here, u represents the target user, v is a non-target user connected to items associated with the target user, and J represents the items linked to the target user. $p^2_{j_1 j_2}$ demonstrates that L_hops indicating paths of 2 edges between items j_1 and j_2. In Figure 2, the computation of the Common Interacted Ratio (CIR) between the target user (u) and items connected to the target user is presented [2]. This metric is used to calculate edge weights. The higher the CIR value, the more significant the corresponding path. Here, we demonstrate that by configuring different functions f and parameters $\hat{L}$, the general formula $\emptyset_u^{\hat{L}}$ (j) can represent many well-known graph similarity metrics. The overall form of this equation remains the same and is expressed as follows:

$$\Phi_u^{\hat{L}}(j)=\frac{1}{\left[N_u^1\right]}\sum_{i\in N_u^1}\sum_{L=1}^{\hat{L}}\beta^{2L}\sum_{p_{ji}^{2L}\in\rho_{ji}^{2L}}\frac{1}{f\left(N_k^1 \mid K\in p_{ji}^{2L}\right)} \tag{6}$$

Let L=1 and set $f(\{N_k^1 | K \in p_{ji}^{2L}\}) = |N_i^1 \cup N_j^1|$ then, the following formula calculates the CIR for the target user using Jaccard similarity:

$$\Phi_u^{\hat{L}}(j)=\frac{1}{\left[N_u^1\right]}\sum_{i\in N_u^1}\beta^2\sum_{p_{ji}^{2L}\in\rho_{ji}^{2L}}\frac{1}{\left|N_i^1\cup N_j^1\right|}=\frac{\beta^2}{\left|N_u^1\right|}\sum_{i\in N_u^1}\frac{\left|N_i^1\cap N_j^1\right|}{\left|N_i^1\cup N_j^1\right|}=\frac{\beta^2}{\left|N_u^1\right|}\sum_{i\in N_u^1}Jc(i,j) \tag{7}$$

where:

- $\emptyset_u^1$: Denote the first-order CIR-based similarity score between node u and node j.
- β: The returned weight in the GNN.

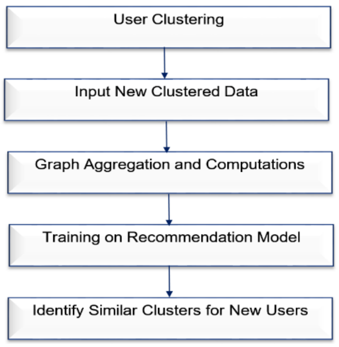

User Clustering
Input New Clustered Data
Graph Aggregation and Computations
Training on Recommendation Model
Identify Similar Clusters for New Users

Figure 1. Flowchart of the proposed model.

- N_u^1: The number of items in the direct neighborhood of the user.
- ρ_{ji}^2: The set of 2-edge paths from j to i.
- jc(i,j): Jaccard similarity between nodes i and j.

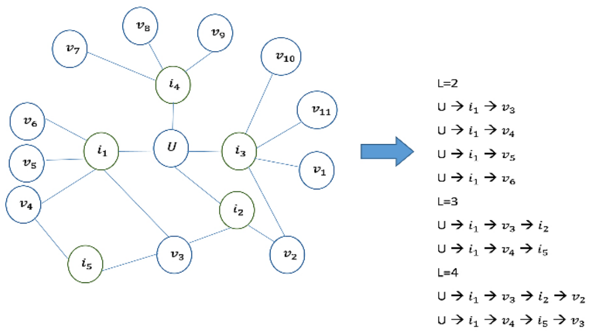

v_7
v_8
v_9
v_{10}
i_4
v_6
v_{11}
v_5
i_1
U
i_3
v_1
v_4
i_2
i_5
v_3
v_2
L=2
$U \rightarrow i_1 \rightarrow v_3$
$U \rightarrow i_1 \rightarrow v_4$
$U \rightarrow i_1 \rightarrow v_5$
$U \rightarrow i_1 \rightarrow v_6$
L=3
$U \rightarrow i_1 \rightarrow v_3 \rightarrow i_2$
$U \rightarrow i_1 \rightarrow v_4 \rightarrow i_5$
L=4
$U \rightarrow i_1 \rightarrow v_3 \rightarrow i_2 \rightarrow v_2$
$U \rightarrow i_1 \rightarrow v_4 \rightarrow i_5 \rightarrow v_3$

Figure 2. Sample graph.

3.2 Improvements in the Proposed Model

In comparison with the CAGCN model, the proposed approach introduces improvements in two main aspects:

Replacing GCNConv with GATConv: Instead of the GCNConv layer, the GATConv layer is used to enhance performance by leveraging attention mechanisms. Specifically,

$$h_i^{l+1} = \sigma\left(\sum_{j \in N(i)} \alpha_{ij} W^l h_j^l\right) \tag{8}$$

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(a^T\left[W h_i^l \| W h_j^l\right]\right)\right)}{\sum_{k \in N(i)} \exp\left(\text{LeakyReLU}\left(a^T\left[W h_i^l \| W h_k^l\right]\right)\right)} \tag{9}$$

where σ(·) is a non-linear activation function; W^l is a shared weight matrix; h_j^l represents the input feature vector of neighboring node j at layer l; $N(i)$ is the set of neighboring nodes of node; LeakyReLU(·) indicates the leaky rectified linear unit activation function with a small negative slope; α_{ij} represents the attention coefficient indicating the importance of node j's features to node i; a is a learnable weight vector for computing attention scores; and $[\cdot \,||\, \cdot]$ denotes vector concatenation. This formulation allows the network to dynamically learn edge importance based on node feature interactions, enabling context-sensitive aggregation rather than fixed weights, and making the model more expressive and adaptive throughout training.

User Clustering: The core idea of this paper is to cluster users in order to reduce the overall complexity of the graph. By grouping users into clusters, the number of distinct user nodes is significantly decreased. Rather than operating on individual users, the model utilizes the aggregated clusters, effectively reducing the interaction matrix to the representative users. Consequently, item scores are computed only for the cluster representatives. The clustered user embedding, $h_u^{clustered}$, is computed as:

$$h_u^{clustered} = \sum_{c=1}^{c} c\,\mu_{uc} \tag{10}$$

where c denotes the number of clusters and μ_{uc} represents the degree of user u's association with cluster c.

The fuzzy clustering method is employed for user clustering [34]. Based on prior research conducted on the same dataset, we compared several clustering methods, including K-means, SOM, C-means & Max, and C-means & COG. Consistent with those findings, we selected the fuzzy C-means approach due to its superior performance in capturing user similarity and reducing error. Fuzzy clustering, also known as the Fuzzy C-means algorithm, is a variation of the K-means algorithm. Unlike K-means, which assigns data points exclusively to a single cluster, fuzzy clustering uses degree-based assignment. In this approach, each data point can belong to multiple clusters to varying extents, meaning that every data point is associated with a specific degree of membership in each cluster. Based on a review of previous studies conducted on this dataset, we identified the optimal number of clusters [46,47,48]. Clustering was performed based on the ratings users assigned to movies. The clustered data were then input into our model, which reduced the size of the interaction matrix. After calculating edge weights in the graph and determining the CIR, the item-user rating matrix was generated. For new users, their similarity to existing clusters was identified using cosine similarity [49]. Once a cluster match was determined, new items that the target user had not seen could be recommended. This approach led to an increase in both the number and diversity of recommendations. Cosine similarity was used to calculate the similarity between a user and other users as follows:

$$Sim\left(u_i, u_k\right) = \frac{r_i \cdot r_k}{\|r_i\| * \|r_k\|} = \frac{\sum_{j=1}^{m} r_{ij} r_{ik}}{\sqrt{\sum_{j=1}^{m} r_{ij}^2 \sum_{j=1}^{m} r_{kj}^2}} \tag{11}$$

In the formula, u_i represents the i-th user, u_k represents the k-th user, r indicates the rating, and m is the number of items. Once the new user's similarity to a specific cluster is determined, recommendations can be made. In this study, 10 recommendations were generated for each user.

4. Experimental settings

In this section, we first outline the experimental settings, which strictly follow [2]. Then, we conduct a comparative analysis of our proposed method against other graph neural network-based approaches. The proposed method was trained using the Adam optimizer with a learning rate of 0.0001. The learning rate was adaptively decreased based on changes in the network's loss function, guiding the model toward an optimal state. L2-regularization with a coefficient of $1e^{-3}$ and γ=1 was applied to help prevent overfitting. The training process was conducted over 1000 epochs with a batch size of 1024. The proposed method was implemented on a system with 16 GB P100 graphics processing unit (GPU) and 29 GB RAM, with experiments conducted over 1000 iterations.

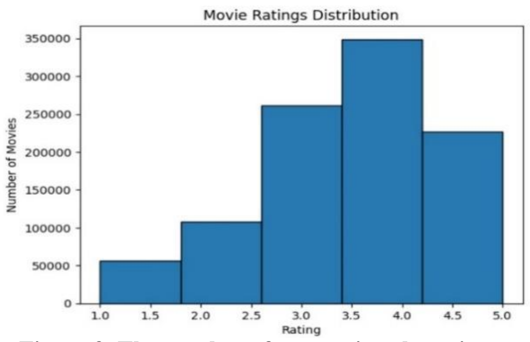


Figure 3. The number of connections by points.

4.1. Cost Function

The cost function measures how much the model's predictions deviate from the true values and serves as a guide for improving its accuracy. This function directs the optimization of the network's weights to minimize errors. The performance of the cost function is critical during the model training phase. In this study, we employed the Bayesian Personalized Ranking (BPR) loss [50]. BPR is specifically designed for collaborative recommendation systems, aiming to rank items for a particular user such that items of interest to the user are positioned higher in the ranking. The goal of this function is to maximize the likelihood that a positive item the user has interacted with is ranked above a negative item, which the user has not interacted with. In other words, the function seeks to guide the model toward accurately predicting user preferences.

The formula for calculating the BPR loss is shown below, where σ is the sigmoid function. y_{ui} represents the scores of items that are connected to and interacted with by the target user, calculated as $y_{ui} = e_u^T e_i$, y_{ui^-} represents the scores of items not connected to the target user, calculated as $y_{ui^-} = e_u^T e_i^-$.

$$L_{BPR} = \sum_{(u,i,i^-)\in O} -\ln\sigma\left(y_{ui} - y_{ui^-}\right) \tag{12}$$

4.2. Dataset

For this research, we relied on the MovieLens dataset [http://grouplens.org/datasets/movielens], curated by the GroupLens Research Group at the University of Minnesota. This dataset is widely recognized as a reliable benchmark for evaluating recommendation algorithms. The MovieLens dataset contains 6040 users and 3900 items, with a total of 1000209 interactions, where interactions represent user ratings of movies. This number reflects the total connections between users and the items they interacted with. A detailed analysis of the dataset is provided below. In Figure 3, the horizontal axis represents the rating values ranging from 1 to 5, while the chart shows the number of interactions for each rating category.

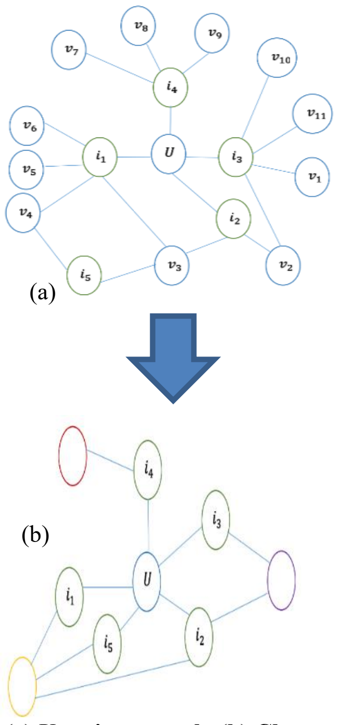


Figure 4. (a) User-item graph. (b) Clustered user graph.

The MovieLens dataset is represented as a 2 Dimensions NumPy array with the following format: $A_{ij}(A_i$ act in place of the movies, and the values are centered on zero by subtracting the mean from the respective elements) [51].

The Yelp2018 dataset is derived from the 2018 edition of the Yelp Challenge. It contains comprehensive information on user reviews and local businesses, such as restaurants and bars, which are treated as items in recommendation settings. To ensure the reliability and quality of the data used in experiments, the dataset follows a 10-core setting, where each user and each item has at least 10 interactions. This setup helps filter out sparse user-item interactions and is widely used in recommendation research for consistent benchmarking.

5. Results and discussion

The section presents the experimental results and provides a detailed discussion of their implications. The statistical details of the datasets are presented in Table 2. Specifically, this section offers an in-depth analysis to address the following research questions:

- How does user clustering affect recommender systems?
- Which similarity measurement formula between nodes is more effective for weighting edges?

Figure 4(a) depicts a standard graph in which **U** represents the target user, **i** refers to the movies connected to the target user, and **v** denotes other users who have rated these movies. In contrast, Figure 4(b) shows a simplified version of this graph, where users have been grouped into clusters, making the structure more organized.

To assess the impact of different similarity measurement formulas on edge weighting, we analyzed the results of two evaluation metrics: Recall and NDCG [52]. The detailed findings are presented below.

Recall measures the proportion of true positive samples correctly identified by the system or model. It indicates the proportion of relevant items that have been successfully retrieved by the model. Normalized Discounted Cumulative Gain (NDCG) is one of the most common metrics used to evaluate the quality of recommender systems and search engines. This metric assesses the relevance of items recommended or ranked by a system, prioritizing more relevant items at the top of the list. In NDCG, items ranked higher should have greater value. Higher-ranked, more relevant items contribute to a better score—the higher the score. NDCG is scale-independent and optimized for measuring qualitative performance [31]. To calculate NDCG, we first compute the Discounted Cumulative Gain (DCG), which accumulates relevance scores with decreasing weight as the rank position increases. Then, we calculate the Ideal Discounted Cumulative Gain (IDCG), representing the best possible scenario where all items are perfectly ranked in descending order of relevance. The formulas for DCG and IDCG are as follows:

$$DCG_k = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)} \tag{13}$$

$$DCG_k = \sum_{i=1}^{k} \frac{rel_i^{(ideal)}}{\log_2(i+1)} \tag{14}$$

The table below compares the performance of the CAGCN model with GCN and GAT layers. The results demonstrate that the proposed model incorporating GAT layers exhibits superior accuracy.

While recent models such as UltraGCN [53], SGL [54], and HCCF [55] have made significant progress in graph-based recommendation by introducing simplified propagation schemes, self-supervised objectives, and hypergraph structures, they still face challenges related to graph noise, scalability, and embedding over-smoothing. The proposed method outperforms these baselines by explicitly addressing these limitations through the integration of fuzzy clustering and attention mechanisms. Furthermore, we argue that models like UltraGCN, SGL, and HCCF could benefit from incorporating clustering techniques as a preprocessing step. By grouping similar users or items, clustering not only reduces graph size and complexity but also mitigates noisy or redundant connections, ultimately enhancing the effectiveness and efficiency of these models without deviating from their core methodologies.

In Table 2, the accuracy of the proposed model was examined alongside other models that are structurally similar to the proposed model. Moreover, for a detailed comparison and to clearly illustrate the contribution of each component individually and in combination, Table 3 presents the ablation study results on the model components. This table shows the performance of the model with only GATConv, only clustering, and both components together. As the results indicate, the effect of clustering alone was greater than GATConv alone, and combining both components achieved the highest overall improvement in performance.

In Table 4, we compared the proposed model before and after user clustering. The results show that by clustering users, the complexity of the model decreases and, as a result, the accuracy increases.

Table 5 reports the final performance results of the proposed model on two benchmark datasets, MovieLens and Yelp. In addition to standard performance metrics, significance testing was conducted to evaluate the robustness of the observed improvements. Specifically, paired t-tests were performed between our method and each baseline across multiple experimental runs. The resulting p-values for Recall and NDCG are included in the table. These results indicate that the improvements achieved by our model are statistically significant ($p < 0.05$) in most cases, thereby reinforcing the effectiveness of the proposed approach.

In the proposed model, 6,022 users are grouped into 18 clusters, effectively reducing the user-item interaction matrix to 18 representative users. Consequently, item scores are computed only for these cluster representatives. To determine the optimal number of clusters, the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) values across different cluster counts are analyzed, as illustrated in Figure 5. Both error metrics show a significant decline up to approximately 18 clusters, beyond which additional clusters yield only marginal improvements. Therefore, 18 clusters are selected as the optimal balance between prediction accuracy and model complexity.

Table 1. Comparison of types of similarity criteria.

With GCN Layer					
Model(similarity)	CAGCN(JC)	CAGCN(LHN)	CAGCN(SC)	CAGCN(DICE)	CAGCN(COSINE)
Recall	62.72	62.61	63.08	62.20	63.05
NDCG	62.52	62.75	63.54	62.59	63.12
With GAT Layer					
Model(similarity)	CAGCN(JC)	CAGCN(LHN)	CAGCN(SC)	CAGCN(DICE)	CAGCN(COSINE)
Recall	62.98	62.83	63.59	62.93	63.49
NDCG	63.03	62.92	63.96	62.71	64.07

Table 2. Evaluation of the proposed method and previous model.

Model(similarity)	SASRec [36]	BERT4Rec [37]	BPRMF [38]	CAGCN (JC)	CAGCN (LHN)	CAGCN (SC)	CAGCN (DICE)	CAGCN (COSINE)
Recall	-	-	12.95	62.98	62.83	63.59	62.93	63.49
NDCG	59.05	48.18	61.02	63.03	62.92	63.96	62.71	64.07

Table 3. Comparison of impact of GATConv and clustering on performance on the MovieLens dataset.

Model	Recall	NDCG
Baseline CAGCN(JC)	62.72	62.52
CAGCN(JC) + GATConv	62.98	63.03
CAGCN(JC) + Clustering	63.43	64.12
CAGCN(JC) + GATConv + Clustering	66.76	66.01

Table 4. Comparison of the results obtained before clustering and after clustering of users.

With GAT Layer					
Model(similarity)	CAGCN(JC)	CAGCN(LHN)	CAGCN(SC)	CAGCN(DICE)	CAGCN(COSINE)
Recall	62.98	62.83	63.59	62.93	63.49
NDCG	63.03	62.92	63.96	62.71	64.07
Clustered Data					
Model(similarity)	CAGCN(JC)	CAGCN(LHN)	CAGCN(SC)	CAGCN(DICE)	CAGCN(COSINE)
Recall	66.76	66.67	67.33	65.99	67.21
NDCG	67.01	65.75	68.15	66.43	68.00

Table 5. Performance comparison of CAGCN variants with different similarity metrics on MovieLens and Yelp2018. datasets.

Dataset	Model	CAGCN(JC)	CAGCN(LHN)	CAGCN(SC)	CAGCN(DICE)	CAGCN(COSINE)	p-value
MovieLens	Recall	66.76	66.67	67.33	65.99	67.21	3.34e-4
	NDCG	67.01	65.75	68.15	66.43	68.00	3.78e-5
Yelp2018	Recall	70.21	69.54	69.11	68.42	69.02	5.84e-7
	NDCG	70.69	66.38	69.91	66.75	70.87	4.97e-2

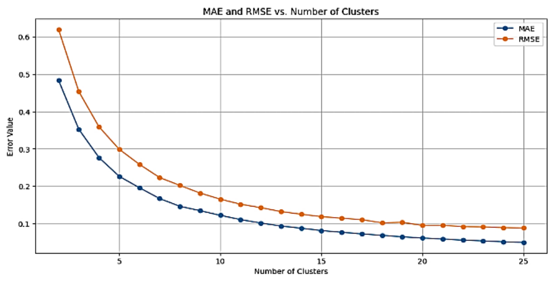
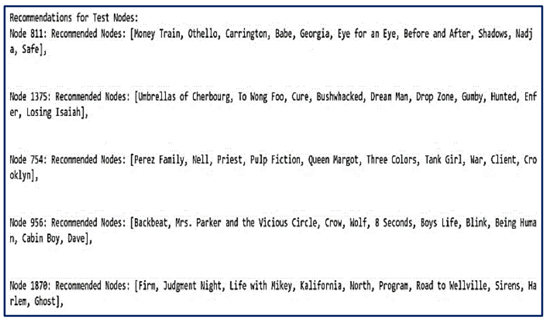


Figure 5. MAE and RMSE values across different numbers of clusters.



```
Recommendations for Test Nodes:
Node 811: Recommended Nodes: [Money Train, Othello, Carrington, Babe, Georgia, Eye for an Eye, Before and After, Shadows, Nadj
a, Safe],


Node 1375: Recommended Nodes: [Umbrellas of Cherbourg, To Wong Foo, Cure, Bushwhacked, Dream Man, Drop Zone, Gumby, Hunted, Enf
er, Losing Isaiah],


Node 754: Recommended Nodes: [Perez Family, Nell, Priest, Pulp Fiction, Queen Margot, Three Colors, Tank Girl, War, Client, Cro
oklyn],


Node 956: Recommended Nodes: [Backbeat, Mrs. Parker and the Vicious Circle, Crow, Wolf, 8 Seconds, Boys Life, Blink, Being Huma
n, Cabin Boy, Dave],


Node 1870: Recommended Nodes: [Firm, Judgment Night, Life with Mikey, Kalifornia, North, Program, Road to Wellville, Sirens, Ha
rlem, Ghost],
```

Figure 6. Sample output results for each user.

In Figure 6, for each user identified by ID number, 10 offers are shown in the form of ID numbers. The sample is the final output display.

In terms of computational complexity, our model performs fuzzy clustering as a one-time preprocessing step with a time complexity of $O(N.C.d)$, where N is the number of users, C is the number of clusters (18 in our experiments), and d is the feature dimensionality. During training, each GATConv layer incurs a time complexity of approximately $O(F.|E|)$, where $|E|$ represents the number of edges and F the embedding size. While GAT layers introduce a higher per-edge computational cost compared to the baseline CAGCN, due to the dynamic computation of attention coefficients, this overhead is substantially mitigated by the clustering step, which reduces the effective graph size.

Finally, we have calculated and compared the number of learnable parameters and the overall model complexity of our approach relative to the baseline CAGCN configurations. In the original CAGCN models, the total number of parameters primarily depends on the embedding size F, the number of layers L, and the choice of similarity weighting functions (e.g., JC, SC, or LHN). For instance, in CAGCN-JC on the ML-1M dataset, parameter counts are configured using $\gamma = 2$ and an L2-regularization factor of 1×10^{-3}. Similarly, CAGCN-SC and CAGCN-LHN typically utilize embedding sizes between 64 and 128 with one or two layers, resulting in approximately 1.2 to 1.5 million parameters in total. In the proposed model, the integration of GATConv layers increases the parameter count, as each attention head introduces additional sets of learnable weight vectors and projection matrices. Specifically, with a hidden embedding size of $F = 64$, a single GAT layer with four attention heads contributes approximately $4F + F$ parameters when combined with user and item embeddings. Consequently, our model contains around 1.8 million learnable parameters—representing a 25–35% increase compared to the original CAGCN baselines. Although this leads to a modest increase in model complexity and per-epoch training time (as previously discussed), it remains well within the capabilities of a single modern GPU. Importantly, this added capacity contributes directly to improved accuracy and more effective modeling of user–item interactions, as evidenced by the results in our experiments.

Various studies have demonstrated that employing advanced techniques, such as graph-based models and deep learning, significantly enhances the performance of recommender systems. Specifically, leveraging user clustering and intelligently integrating information from neighbors enables the modeling of complex interactions between users and items. This approach not only reduces graph complexity and increases its density but also facilitates the generation of more diverse and accurate recommendations, particularly for new users or those with limited data. By identifying clusters for new users with insufficient information, it becomes possible to recommend new items effectively. Experiments conducted on the MovieLens dataset demonstrate that fuzzy clustering methods achieve higher Recall and NDCG scores than traditional approaches like K-means and SOM [56]. Moreover, recent advancements in deep learning and the adoption of deep neural network-based models [57] promise a bright future for recommender systems. These techniques, with their capabilities for extracting deep features, modeling nonlinear relationships, and providing advanced predictions, have successfully addressed the challenges faced by traditional systems. Overall, recent research underscores that integrating graph-based techniques, deep learning, and advanced clustering methods can significantly enhance the accuracy and performance of recommender systems [58].

While clustering reduces graph complexity, improves computational efficiency, and increases the diversity of recommendations, it also has some potential downsides. In particular, grouping users into clusters may blur fine-grained personalization by merging distinct individual preferences into broader profiles. This can lead to information loss and occasionally reduce the accuracy of recommendations for users whose behaviors do not closely match the dominant patterns in their cluster. Additionally, static clustering does not adapt to changes in user behavior over time, which may cause outdated or less relevant recommendations. To address these limitations, future work will investigate adaptive clustering strategies that update cluster assignments dynamically, as well as hybrid methods that combine global clustering with personalized fine-tuning for each user.

6. Conclusion and Future Work

In this study, we explored the capabilities of collaborative recommender systems and assessed their effectiveness in enhancing recommendation accuracy and efficiency. We proposed a deep learning-based framework leveraging a GCN, a subclass of GNNs, to model and analyze user-item interactions.

By assigning adaptive edge weights based on similarity metrics such as Jaccard similarity, cosine similarity, the LHN index, and Salton cosine similarity, the model effectively captures the relative importance of relationships within the graph. This weighting mechanism enables more accurate edge scoring and better identification of relevant user-item paths. To validate our approach, we conducted extensive experiments on two benchmark datasets—MovieLens and Yelp2018.

Furthermore, we introduced a user clustering strategy to reduce graph complexity and enhance diversity in recommendations. Grouping users into clusters significantly improved both computational efficiency and recommendation quality. Overall, the proposed model demonstrated robust performance, underscoring its potential for advancing collaborative recommendation systems.

The following directions are proposed for future work, aiming to further improve the effectiveness and adaptability of recommender systems:

1. Incorporating real-time user feedback: enhancing and fine-tuning the model dynamically using direct user feedback.
2. Simultaneous clustering of users and items: exploring methods for clustering both users and items together to improve recommendations.
3. Updating training data to reflect evolving user preferences: regularly updating training data to account for changes in user preferences over time.

References

[1] J. Lu, D. Shuangwu, M. S. Mao, W. Wang, and G. Q. Zhang, "Recommender system applications: A survey of systems decision support," Decision Support Systems, vol. 74, pp. 12–32, 2015.

[2] Y. Wang, Y. Zhao, Y. Zhang, and T. Derr, "Collaboration-Aware Graph Convolutional Network for Recommender Systems," arXiv:2207.06221, 2023.

[3] H. Ma, H. Yang, M. R. Lyu, and I. King, "SoRec: Social recommendation using probabilistic matrix factorization," in Proceedings of the 17th ACM Conference on Information and Knowledge Management, USA, Oct. 2008, pp. 931–940.

[4] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning-based recommender system: A survey and new perspectives," ACM Computing Surveys (CSUR), vol. 52, no. 1, pp. 1–38, 2019.

[5] H. Koohi, K. Kiani, "A new method to find neighbor users that improves the performance of collaborative filtering," Expert Systems with Applications, vol. 83, pp. 30-39, 2017.

[6] K. Kiani, R. Rastgoo, A. Chaji, S. Escalera, "Image Inpainting Enhancement by Replacing the Original Mask with a Self-attended Region from the Input Image," *Journal of AI and Data Mining*, vol. 13, no. 3, pp. 379-391, 2025.

[7] N. Esfandiari, K. Kiani, R. Rastgoo, "Development of a Persian Mobile Sales Chatbot based on LLMs and Transformer," *Journal of AI and Data Mining*, vol. 12, no. 4, pp. 465-472, 2024.

[8] N. Esfandiari, K. Kiani, R. Rastgoo, "Transformer-based Generative Chatbot Using Reinforcement Learning," *Journal of AI and Data Mining*, vol. 12, no. 3, pp. 349-358, 2024.

[9] A.M. Ahmadi, K. Kiani, R. Rastgoo, "A Transformer-based model for abnormal activity recognition in video," *Journal of Modeling in Engineering*, vol. 22, no. 76, pp. 213-221, 2024.

[10] F. Bagherzadeh, R. Rastgoo, "Deepfake image detection using a deep hybrid convolutional neural network," *Journal of Modeling in Engineering*, vol. 21, no. 75, pp. 19-28, 2023.

[11] M. Talebian, K. Kiani, R. Rastgoo, "A Deep Learning-based Model for Fingerprint Verification," *Journal of AI and Data Mining*, vol. 12, no. 2, pp. 241-248, 2024.

[12] H. Zaferani, K. Kiani, R. Rastgoo, "Real-time face verification on mobile devices using margin distillation," *Multimedia Tools and Applications*, vol. 82, no. 28, pp. 44155-44173, 2023.

[13] S. Zarbafi, K. Kiani, R. Rastgoo, "Spoken Persian digits recognition using deep learning," *Journal of Modeling in Engineering*, vol. 21, no. 74, pp. 163-172, 2023.

[14] N. Esfandiari, K. Kiani, R. Rastgoo, “A conditional generative chatbot using transformer model,” Journal of Modeling in Engineering, vol. 23, no. 82, pp. 99-113, 2025.

[15] N. Majidi, K. Kiani, R. Rastgoo, “A deep model for super-resolution enhancement from a single image,” *Journal of AI and Data Mining*, vol. 8, no. 4, pp. 451-460, 2020.

[16] R. Rastgoo, K. Kiani, “Face recognition using fine-tuning of Deep Convolutional Neural Network and transfer learning,” *Journal of Modeling in Engineering*, vol. 17, no. 58, pp. 103-111, 2019.

[17] R. Rastgoo, V. Sattari-Naeini, “Gsomcr: Multi-constraint genetic-optimized qos-aware routing protocol for smart grids,” *Iranian Journal of Science and Technology, Transactions of Electrical, Engineering*, vol. 42, no. 2, pp. 185-194, 2018.

[18] R. Rastgoo, V. Sattari-Naeini, “Tuning parameters of the QoS-aware routing protocol for smart grids using genetic algorithm,” *Applied Artificial Intelligence*, vol. 30, no. 1, pp. 52-76, 2016.

[19] R. Rastgoo, V. Sattari Naeini, “A neurofuzzy QoS-aware routing protocol for smart grids,” *22nd Iranian Conference on Electrical Engineering (ICEE)*, pp. 1080-1084, 2014.

[20] F. Alinezhad, K. Kiani, R. Rastgoo, “A Deep Learning-based Model for Gender Recognition in Mobile Devices,” *Journal of AI and Data Mining*, vol. 11, no. 2, pp. 229-236, 2023.

[21] Havva Askari, Razieh Rastgoo, Kourosh Kiani, “Accuracy Improvement of Real-Time Driver Drowsiness Detection Using Transformer Model,” *Journal of AI and Data Mining*, vol. 13, no. 4, pp. 481-490, 2025.

[22] M.S. Tavallali F. Bordbar, R. Rastgoo, M.A. Askarzadeh, “Prediction of Residential Natural Gas Consumption Using Artificial Neural Network,” The 9th International Chemical Engineering Congress & Exhibition (IChEC 2015), 2015.

[23] K. Zou, A. Sun, X. Jiang, Y. Ji, H. Zhang, and J. Wang, "Hesitation and Tolerance in Recommender Systems," arXiv: 2412.09950, 2024.

[24] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, May 2015, pp. 111–112.

[25] R. Devooght and H. Bersini, "Long and short-term recommendations with recurrent neural networks," in Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, Bratislava, Slovakia, Jul. 2017, pp. 13–21.

[26] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: A survey," arXiv:2011.02260, 2020.

[27] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," arXiv preprint, arXiv:1706.02216, 2017.

[28] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, UK, Aug. 2018, pp. 974–983.

[29] Zhang, J., Shi, X., Zhao, S., and King, I., “Star-GCN: Stacked and reconstructed graph convolutional networks for recommender systems,” arXiv:1905.13129, 2019.

[30] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in Proc. Int. Conf. Research and Development in Information Retrieval (SIGIR), 2020, pp. 639–648.

[31] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph-based collaborative filtering: A linear residual graph convolutional network approach," in Proc. AAAI Conf. Artificial Intelligence, vol. 34, 2020, pp. 27–34.

[32] J. Sun, Z. Cheng, S. Zuberi, F. Pérez, and M. Volkovs, "HGCF: Hyperbolic graph convolution networks for collaborative filtering," in Proc. The Web Conf. (WWW), 2021, pp. 593–601.

[33] S. Peng, S. Siet, I. Sadriddinov, D.Y. Kim, K. Park, D.S. Park, " Integration of Federated Learning and Graph Convolutional Networks for Movie Recommendation Systems," Computers, Materials and Continua, vol. 83, Issue 2, pp. 2041-2057, 2025.

[34] E. Elahi, Z. Halim, "Graph attention-based collaborative filtering for user-specific recommender system using knowledge graph and deep neural networks," Knowledge and Information Systems, vol. 64, Issue 9, pp. 2457 – 2480, 2022.

[35] G. Kaur, F. Liu, Y.P. Phoebe Chen, "A deep learning knowledge graph neural network for recommender systems," Machine Learning with Applications, vol. 14, 100507, 2023.

[36] P. Mondal, D. Chakder, S. Raj, S. Saha, and N. Onoe, “Graph Convolutional Neural Network for Multimodal Movie Recommendation,” in Proc. 38th ACM/SIGAPP Symp. Applied Computing (SAC), Tallinn, Estonia, 2023.

[37] H. Guo, C. Yang, L. Zhou, S. Wei, "A novel Knowledge Graph recommendation algorithm based on Graph Convolutional Network," Connection Science, vol. 36, Issue 1, Article: 2327441, 2024.

[38] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in Proc. Int. Conf. Research and Development in Information Retrieval (SIGIR), 2021, pp. 726–735.

[39] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. Huang, "Hypergraph contrastive collaborative filtering," in Proc. Int. Conf. SIGIR, 2022, pp. 70–79.

[40] Z. Lin, C. Tian, Y. Hou, and W. X. Zhao, "Improving graph collaborative filtering with neighborhood-enriched contrastive learning," in Proc. The Web Conf. (WWW), 2022, pp. 2320–2329.

[41] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," arXiv:1609.02907, Feb. 2017.

[42] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," Journal of the American Society for Information Science and Technology, vol. 58, no. 7, pp. 1019–1031, 2007.

[43] G. Salton, Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Reading, MA: Addison-Wesley, 1989.

[44] E. Leicht, P. Holme, and M. E. J. Newman, "Vertex similarity in networks," Physical Review E, vol. 73, no. 2, 026120 ,2006.

[45] Z. Yang, M. Ding, X. Zou, J. Tang, B. Xu, C. Zhou, and H. Yang, "Region or global: a principle for negative sampling in graph-based recommendation," IEEE Trans. Knowl. Data Eng., vol. 35, pp. 6264–6277, 2022.

[46] H. Koohi and K. Kiani, "User based collaborative filtering using fuzzy C-means," Measurement, vol. 91, pp. 134-139, 2016.

[47] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," Knowledge-Based Systems, vol. 46, pp. 109–132, 2013.

[48] N. Yadav, S. Pal, A. K. Singh, and K. Singh, "Clus-DR: Cluster-based pre-trained model for diverse recommendation generation," Expert Systems with Applications, vol. 34, no. 8, pt. 1, pp. 6385–6399, Sept. 2022.

[49] A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," in Proc. CITSM, Indonesia, Apr. 2016, pp. 1–6.

[50] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian Personalized Ranking from Implicit Feedback," arXiv:1205.2618, 2012.

[51] Redwane Nesmaouia, Mouad Louhichia, Mohamed Lazaara, "A Collaborative Filtering Movies Recommendation System Based on Graph Neural Network," Procedia Computer Science, vol. 220, pp. 456–461, 2023.

[52] H. Jung, S. Kim, and H. Park, "Dual Policy Learning for Aggregation Optimization in Graph Neural Network-based Recommender Systems," arXiv:2302.10567, 2023.

[53] K. Mao, J. Zhu, X. Xiao, B. Lu, Z. Wang, and X. He, "UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation," arXiv:2110.15114, 2023.

[54] W.-C. Kang and J. McAuley, "Self-Attentive Sequential Recommendation," in Proc. IEEE Int. Conf. on Data Mining (ICDM'18), 2018.

[55] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang Alibaba Group, Beijing, China, "BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer", arXiv:1904.06690, Aug. 2019.

[56] W.-T. Chu and Y.-L. Tsai, "A hybrid recommendation system considering visual information for predicting favorite restaurants," World Wide Web, vol. 20, no. 6, pp.1213-1231, Nov. 2017.

[57] R. Rastgoo, K. Kiani, and S. Escalera, "A non-anatomical graph structure for boundary detection in continuous sign language," Scientific Reports, vol. 15, no. 1, 25683, 2025.

[58] A. Bilge and H. Polat, "A comparison of clustering-based privacy-preserving collaborative filtering schemes," Applied Soft Computing, vol. 13, pp. 2478–2489, 2013.

بهبود دقت سیستم توصیه‌گر مشارکتی با استفاده از یادگیری عمیق

مریم باغی، کوروش کیانی* و راضیه راستگو

دانشکده مهندسی برق و کامپیوتر، دانشگاه سمنان، سمنان، ایران.

ارسال ۲۰۲۵/۰۴/۲۸؛ بازنگری ۲۰۲۵/۰۶/۱۷؛ پذیرش ۲۰۲۵/۰۸/۲۱

چکیده:

با پیشرفت‌های سریع در فناوری اطلاعات و ارتباطات، سیستم‌های توصیه‌گر به ابزارهایی حیاتی در طیف گسترده‌ای از فعالیت‌های آنلاین و فرایندهای تجارت الکترونیک تبدیل شده‌اند. سیستم‌های توصیه‌گر مشارکتی، که از داده‌ها و تعاملات کاربران برای ارائه پیشنهادات استفاده می‌کنند، یکی از نوآوری‌های مهم در این حوزه به شمار می‌روند. در این مقاله، ما به تحلیل سیستم‌های توصیه‌گر مشارکتی پرداخته و تأثیر آن‌ها را بر بهبود کارایی و دقت توصیه‌ها ارزیابی می‌کنیم. برای این منظور، یک رویکرد یادگیری عمیق مبتنی بر شبکهٔ کانولوشن گراف (GCN) که نوعی از شبکه‌های عصبی گراف (GNN) است، پیشنهاد می‌دهیم. با تخصیص وزن به یال‌های بین گره‌ها، امتیازهایی برای این یال‌ها محاسبه می‌شود. میزان اهمیت یال‌ها بسته به تعداد گره‌های همسایه و میزان نزدیکی آن‌ها به گره هدف متفاوت است. هرچه امتیاز یال بالاتر باشد، مسیر مرتبط‌تر و مهم‌تر خواهد بود. برای محاسبه وزن یال‌ها، از معیارهایی مانند شباهت جاکارد، شباهت کسینوسی، شاخص LHN و شباهت سالتون استفاده می‌کنیم. این رویکرد موجب بهبود شناسایی روابط بین گره‌ها و افزایش دقت سیستم توصیه‌گر می‌شود. برای پیاده‌سازی، از دو مجموعه داده معروف MovieLens و Yelp2018 استفاده شده است. در نهایت، کاربران در قالب خوشه‌هایی گروه‌بندی شدند که هر خوشه شامل تعداد زیادی از گره‌ها بود. با این خوشه‌بندی، تعداد و تنوع توصیه‌ها افزایش یافت و عملکرد سیستم توصیه‌گر به‌طور قابل توجهی بهبود پیدا کرد و نتایج امیدبخشی حاصل شد.

کلمات کلیدی: سیستم توصیه‌گر، فیلترینگ مشارکتی، توصیه، خوشه‌بندی، شبکه‌های کانولوشن گراف.