Enhancing the quality of recommendation lists using Graph Convolutional Networks

Le Thi Vinh Thanh[1], Le Manh Thanh[1], and Nguyen Van Long[2]

[1]University of Sciences, Hue University, Hue, Vietnam
[2] University of Transport and Communications, Ha Noi, Viet Nam
lethivinhthanh.hcm@gmail.com,lmthanh@hueuni.edu.vn,
nvlongdt@utc.edu.vn

Abstract. In the realm of recommender systems, enhancing the quality of recommendation lists has become a focal point for researchers. This paper presents a novel approach integrating clustering structures with Graph Convolutional Network (GCN) techniques to improve recommendation quality. Initially, we employ a hierarchical tree structure to cluster similar users and items based on energy-based similarity measures. This allows for a more accurate modeling of user and product groups. We then construct graphs representing user relationships (SU-Graph) and item relationships (SI-Graph) based on these clusters, as well as a graph derived from the user-item rating matrix. Utilizing this framework, we train a GCN to predict user ratings for previously unseen items, significantly enhancing the accuracy of recommendations. Finally, we refine the recommendation lists by balancing precision and diversity, ensuring users receive suggestions that are both relevant and varied. Experimental results on the MovieLens dataset validate the effectiveness of our proposed approach, demonstrating substantial improvements over traditional methods.
Keywords: Recommender Systems, Clustering, Diversity, Graph Convolutional Network.

1 Introduction

Recommendation systems are an essential tool in various fields such as e-commerce, social media, and online entertainment. They help users find the most suitable products or content based on personal preferences, habits, and activity history [1, 2]. Providing accurate and personalized suggestions not only enhances user experience but also contributes to increased revenue and overall satisfaction. Traditional recommendation systems typically rely on two main methods: content-based filtering and collaborative filtering [3]. Content-based filtering uses the features of products and user information to suggest items similar to those the user has previously shown interest in. This is achieved by analyzing data related to preferences, product description keywords, or other detailed attributes to create an appropriate predictive model. However, this method has limitations when faced with scenarios where there is a lack of user or product data, which can reduce the effectiveness of the recommendations. Conversely, collaborative filtering primarily relies on user behavior data to make

suggestions. This method uses data from various users to identify trends and common preferences, thereby building connections between users and products based on similarities in behavior. A significant advantage of this method is its ability to make recommendations regardless of product content, as it does not require a detailed analysis of product attributes [4].

Recent studies have focused on applying deep learning techniques based on graphs, including GNN (Graph Neural Networks), GCN, and other variations, to enhance collaborative filtering systems. These models leverage graph structures to represent complex relationships between users and products. Instead of relying solely on simple matrix calculations, using GCN allows the model to learn more complex and profound features, such as influencing factors across multiple connection layers in the user-product network [5-8]. Systems based on GNN and GCN not only help improve the accuracy of recommendations but also expand the system's adaptability to large and diverse data. Graph structures also aid in identifying clusters of users or products with similar characteristics, thereby enhancing the ability to detect potential connections that traditional methods may find difficult to recognize. This is particularly important in the context of today's large and complex data, where information is not just linear but intertwined through multiple multi-dimensional relationships [9, 10]. The quality of recommendation lists has become an essential focus in the research of recommender systems. This paper proposes an advanced method that integrates complex clustering structures with GCN techniques to comprehensively enhance recommendation quality.

The contributions of the paper include: (1) utilizing a hierarchical tree structure to group users and products based on energy-based similarity measures, enabling accurate modeling of user and product groups; (2) constructing graphs representing user relationships (SU-Graph) based on similar user clusters, item relationships (SI-Graph) based on similar item clusters, and a graph representing the relationships between users and items based on the rating matrix; (3) training the GCN to predict user ratings for unseen products, thereby improving the accuracy and relevance of recommendations; (4) refining the final recommendation list through a balance between precision and diversity, ensuring that users receive suggestions that are both highly relevant and varied.

2 Related Work

In the rapidly evolving field of information technology, improving recommendation systems has become increasingly vital due to the growing volume of user data. Traditional methods often struggle with the complex relationships between users and products, leading recent studies to explore Graph Neural Networks (GNN) and Graph Convolutional Networks (GCN) as effective solutions. These approaches leverage the graph structure of user and product data to uncover hidden patterns, enhancing the accuracy and personalization of recommendations. By integrating information from neighboring nodes, GCN has been shown to improve recommendation relevance,

while the combination of GNN with traditional collaborative filtering methods optimizes user experience by providing diverse and relevant suggestions. Overall, these advancements offer valuable insights for developing more effective recommendation systems.

Edoardo et al. introduced an advanced model called Item Graph Convolution Collaborative Filtering (IGCCF) aimed at handling dynamic graphs and leveraging information from user-item graphs through graph convolutional networks. This method facilitates the learning of latent item features, enhancing the accuracy of recommendations and prediction capabilities for new users without requiring a complex retraining process. Experimental results on various real-world datasets show that IGCCF outperforms previous graph-based models in terms of recommendation accuracy and performance. However, one drawback that needs to be addressed is the high computational complexity when processing large graphs, which impacts training time and resources. Jiani Zhang et al. introduced a new architecture aimed at improving the performance of recommendation systems, particularly in cold start scenarios. This method combines stacked GCN encoder-decoder blocks with intermediate supervision, enhancing prediction accuracy [11]. However, some drawbacks may include the complexity of handling large graphs due to the need to learn parameters for each block separately. Ultimately, STAR-GCN has achieved state-of-the-art performance across multiple benchmark datasets, demonstrating its significant potential in addressing recommendation-related challenges. LeWu et al. introduced the Adaptive Graph Convolutional Network (AGCN) for improving item recommendations and inferring user/item attributes in scenarios where data is incomplete. The key strength of AGCN lies in its ability to iteratively refine both the graph embeddings and attribute estimates, leading to significant performance improvements across multiple tasks [12]. However, the approach may face challenges with high computational complexity and sensitivity to the quality of initial attribute estimates. Experimental results demonstrate that AGCN achieves superior performance compared to state-of-the-art methods, particularly in the context of incomplete data. Xiang Wang et al. proposed a new method in recommendation systems called Neural Graph Collaborative Filtering (NGCF). This method utilizes the user-item graph structure to enhance the quality of user and item embeddings by propagating information through high-order connections. Experimental results demonstrate that NGCF significantly outperforms current models such as HOP-Rec and Collaborative Memory Network, thanks to its effective exploitation of collaborative signals [7]. However, the study still faces some challenges, including the integration of attention mechanisms to improve prediction accuracy. The findings from this work open up new research directions in understanding user behavior through more complex networks. Chong Li et al. presented a novel approach for learning representations in bipartite graphs. This method utilizes a hierarchical framework to effectively capture the relationships between two distinct sets of nodes, enhancing the quality of the learned embeddings. The results demonstrate significant performance improvements compared to existing models, showcasing the advantages of this hierarchical approach in terms of capturing complex interdependencies. However, the method may face challenges in scalability when applied to large datasets due to increased computational complexity. Overall, the proposed technique offers a

promising direction for further research in representation learning for graph-based data.

In light of recent advancements in recommendation systems, my study seeks to enhance this evolving field by addressing key aspects that improve user experience. By utilizing advanced modeling techniques and specialized graph structures, this research aims to predict user ratings for unseen products while balancing precision and diversity in recommendations. This approach builds upon existing models' strengths and seeks to overcome their limitations, paving the way for more effective recommendation systems.

3 Recommender system Model

In this paper, the recommendation system is constructed with a modular structure to provide users with a diverse and accurate recommendation list. First, the system clusters of users and items based on common characteristics from the rating matrix of the MovieLens dataset. This clustering module employs a tree structure and energy distance measure to enhance the accuracy and efficiency in grouping users and items. Next, the system builds a user-item graph, where nodes represent users and items, and edges indicate similarity or interaction relationships between the nodes. This graph structure is then used as input for the GCN network to predict ratings between users and items that have not yet been rated, thereby increasing the system's recommendation accuracy. The GCN module consists of three main parts: the user graph, the item graph, and the user-item graph, allowing the system to deeply exploit the features of both users and items. Finally, the recommendation list diversification algorithm integrates accuracy and diversity factors, creating a highly personalized and rich recommendation list. Each module works together to optimize the user experience, ensuring that recommendations are not only tailored to preferences but also offer significant diversity, allowing users to discover more new options.

3.1 User-item clustering

In this study, we focus on clustering similar users and items from the MovieLens dataset, which comprises a comprehensive user-item rating matrix detailing how users have rated various films. To facilitate the clustering process, we adopt a tree structure that is inherited from previous research [13], leveraging its efficiency in organizing and grouping data. However, rather than relying on the traditional Euclidean distance for calculating the similarity between users and items, we employ an Energy distance measure [14]. This approach allows us to better capture the nuanced relationships within the data, enhancing the effectiveness of the clustering framework we construct. By doing so, we aim to achieve more meaningful clusters that reflect true similarities in user preferences and item characteristics.

Energy Distance is a powerful tool for multivariate analysis [2, 15, 16]. It is used to test for independence, and multivariate normality, and to perform non-parametric analysis of complex structured data. Energy distance is applied to random vectors of any size in Euclidean space.

Suppose we have two sets of independent random vectors $I = \{I_1, I_2, \dots I_p\}$ and $I = \{J_1, J, \dots J_q\}$. The distance between I and J is defined as:

$$D^2(I,J) = 2\sum_{i=1}^{p}\sum_{j=1}^{q}\|I - J\| - \sum_{i=1}^{p}\sum_{j=1}^{p}\|I - I'\| - \sum_{i=1}^{q}\sum_{j=1}^{q}\|J - J'\|$$

where: $I'$and J' are independent random copies, distributed identically to I and J, respectively.
This formula defines the "potential energy" of the independent random variables I and J, denoted as $\varepsilon_{p,q}(I,J)$:

$$\varepsilon_{p,q}(I,J) = 2E[\delta(I,J)] - E[\delta(I,I')] - E[\delta(J,J')]$$

where:

$$E[\delta(I,J)] = \frac{1}{pq}\sum_{i=1}^{p}\sum_{j=1}^{q}\|I_i - J_j\|$$

$$E[\delta(I,I')] = \frac{1}{p^2}\sum_{i=1}^{p}\sum_{j=1}^{p}\|I_i - I'_j\|$$

$$[\delta(J,J')] = \frac{1}{q^2}\sum_{i=1}^{q}\sum_{j=1}^{q}\|J_i - J'_j\|$$

Properties: The energy distance $\varepsilon_{p,q}(I,J)$ is always non-negative and equals zero if and only if I and J have the same distribution.

The step of clustering similar users and items serves as the foundation for constructing the user-item similarity graph in the next phase of the study. This grouping creates a clear structure of relationships within the data, enhancing our understanding of user behavior. The similarity graph will allow us to leverage these relationships for more accurate recommendations. The ultimate goal is to improve the effectiveness of the recommendation system.

3.2 Graph construction

In this section, we construct graph structures to represent relationships among users, items, and between users and items. The structure of the graphs is created with nodes representing users and items, while the edges signify the relationships among them. Edges are connected between users if they exhibit a high degree of similarity in ratings or behavior, helping to identify user groups with common preferences. Similarly, items are connected if they share common characteristics, supporting the discovery of relationships among items. Notably, edges between users and items are created when users have rated an item, reflecting the connection between users and the items they have interacted with. The goal of this graph structure is to serve as input for GCN (Graph Convolutional Network) to predict user ratings for unknown items, thereby improving the accuracy of the recommendation system.

To support accurate and diverse recommendations, constructing appropriate data structures is crucial. In this study, we design three specific types of graphs to model the relationships between users and items. These graphs not only help capture the structure and correlations within the data but also facilitate the efficient training of GCNs. The structure of the graphs is defined as follows:

User Graph (SU-Graph):

Definition 1. The $SU\_Graph = (V_u, E_u)$ is a graph representing the relationships between users, where the vertex set $V_u = \{u_i \in U\}$, $\mathrm{U} = \{\mathrm{u}_1, \mathrm{u}_2, \ldots, \mathrm{u}_n\}$ is the set of users in the system; the edge set $E_u = \{e_{u,v}\}$, $u, v \in C_k$, with C_k is the cluster of similar users and $v \in KNN(u)$, with $KNN(u)$is the set of K nearest neighbors of user u within C_k.

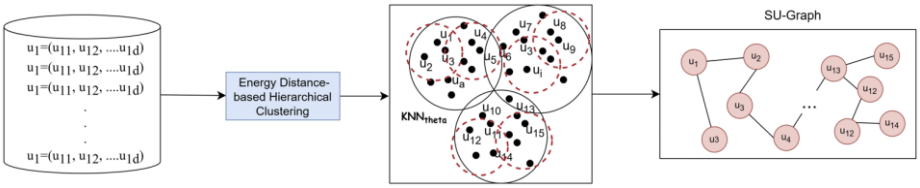

Fig. 1. Graph representing the relationships between users.

Item Graph (SI-Graph):

Definition 2. The $SI\_Graph = (V_i, E_i)$ is a graph representing the relationships between items, where the vertex set $V_i = \{i_i \in I\}$, $\mathrm{I} = \{\mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_m\}$ is the set of items available in the system; the edge set $E_i = \{e_{i,j}\}$, $i, \mathrm{i} \in C_t$, with C_t is the cluster of similar items and $i \in KNN(j)$, with $KNN(i)$ is the set of K nearest neighbors of item i within C_t.

SI-Graph is constructed similarly to the SU-Graph.

User-Item Graph:

Definition 3. The $UI\_Graph = (V, E)$ is a graph describes the relationships between users and items based on the rating matrix, where the vertex set $V = U \cup I$; the edge set $E = \{(u, i)\}$, where user u has rated item i.

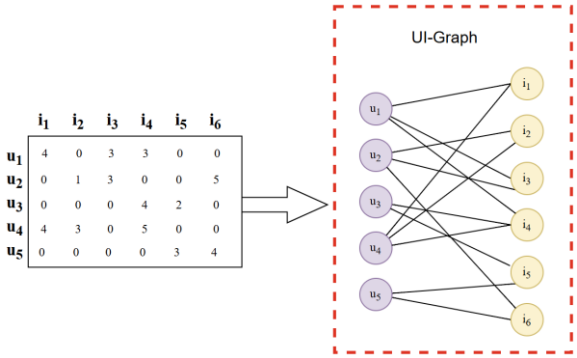

Fig. 2. The experimental results on the MovieLens -10M dataset.

After constructing the graph structure that represents the relationships between users and items, we will use Graph Convolutional Networks (GCN) to leverage these interactions. GCN allows us to effectively model the complex dependencies within the graph, thereby enhancing prediction and recommendation capabilities for users. The following section will detail the implementation and training of GCN on the created graph, aimed at providing accurate suggestions based on user preferences and item characteristics.

3.3 Rating prediction using Graph Convolutional Networks

Graph Convolutional Networks are designed to process and leverage information from complex graph structures. In the context of recommendation systems, GCN helps to capture the interaction relationships between users and items through the features of nodes and edges in the graph. This allows GCN to effectively model user behavior and provide more accurate recommendations based on the relationships between items that users have interacted with. Here are some key benefits that GCN brings in enhancing the effectiveness of recommendation systems.

GCN models view the recommendation system as a bipartite graph consisting of two sets of nodes, users, and items, with edges representing rating values from users to items. The goal is to predict ratings for items that users have not yet rated, based on a small set of known ratings. There are two types of tasks: transductive rating prediction (using data available in the training set) and inductive rating prediction (applied to new nodes that appear only during testing).

While traditional methods struggle to solve the inductive task without retraining, models like CDL [17] and DropoutNet [18] use neural networks to learn content features but depend on this information. STAR-GCN [11] leverages both the content and structural information of the graph to learn embeddings for new nodes, effectively solving the cold-start problem even in the absence of content information, making it superior to previous methods. Our GCN model training process is based on the work of [11]. We add user and item feature information aggregated from similarity graphs to enhance the user-item graph's input features for the GCN, aiming to improve prediction accuracy.

In this section, we use three GCN networks based on three different types of input graphs: the user graph, the item graph, and the user-item graph. The GCN for the user graph updates user embeddings based on relationships and information from neighboring nodes. Similarly, the GCN for the item graph generates item embeddings by propagating information between related items. Then, the GCN for the user-item graph combines both user and item embeddings to create comprehensive feature vectors. These embeddings integrate information from both sides, enabling accurate predictions of the match between users and items.

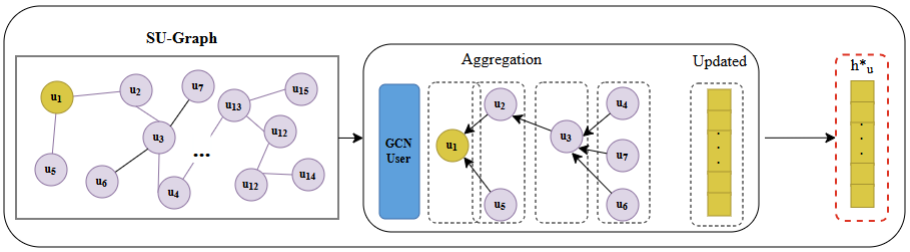

Fig. 3. Training the GCN model for users.

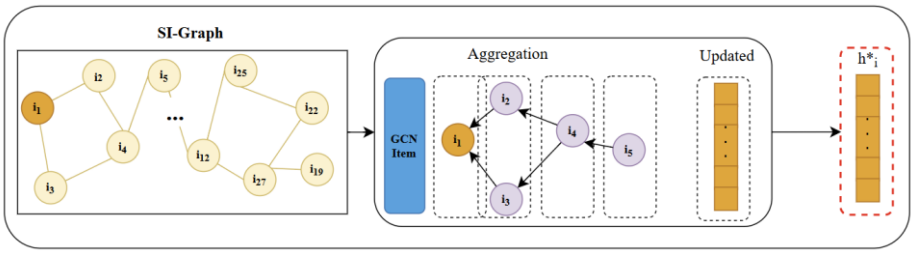

Fig. 4. Training the GCN model for items.

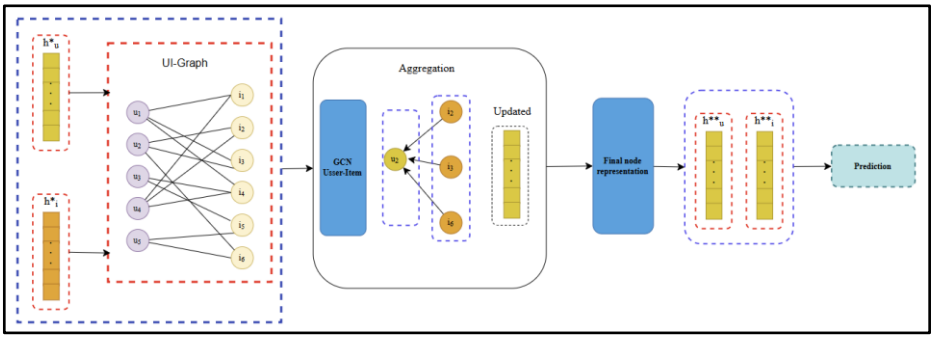

Fig. 5. Training the GCN model to predict user ratings for items.

The detailed steps are presented below.

Step 1. Training GCN on the SU-Graph

Use the user similarity graph as input to train the GCN, connecting users with similar attributes or interactions. Initialize and propagate through GCN layers to extract user embeddings.

Step 2. Training GCN on the SI-Graph

Use the item similarity graph as input for the GCN, connecting items with similar attributes or interactions. Initialize and propagate through GCN layers to obtain item embeddings.

Step 3. Create features for the UI-Graph

Combine user and item embeddings into a single vector, assign it to the edges of the user-item graph as input for the next GCN network.

Step 4. Training GCN on the UI-Graph

The GCN propagates and aggregates information from the neighbors of each node. The features of user and item nodes are updated to new embeddings.

Step 5. Predict user rating for items

Combine the final embeddings of users and items into a single vector, passing it through hidden layers of a neural network to explore relationships between features. The neural network generates 5 possible rating levels, which are converted to probabilities via Softmax. The rating with the highest probability is selected, indicating the item's suitability for the user.

3.4 Re-ranking items

In recent years, numerous studies have focused on how to increase the diversity of recommendation lists while maintaining a certain level of accuracy. The traditional approach to generating highly diverse recommendation lists involves a two-stage process as follows:

Stage 1: Generate the initial candidate list. The system applies the CF method (or possibly a hybrid method) to create an initial recommendation list $L_N(u)$ consisting of N candidate items focused on accuracy (the Top_N items with the highest predicted rating values).

Stage 2: Refine the final list. The system refines the initial recommendation list $L_N(u)$) from Stage 1 by re-ranking or removing similar items, resulting in a final recommendation list $L_M(u)$ consisting of only M items ($M \leq N$) with higher diversity compared to the initial list $L_N(u)$.

However, a significant limitation of this traditional two-stage approach is that, in some cases, Stage 2 cannot significantly change the diversity level of the recommendation list $L_N(u)$ from Stage 1. Specifically, if $L_N(u)$ contains N items with high accuracy but too much similarity, Stage 2 will only filter out a list $L_M(u)$ that still contains many similar items, resulting in a low diversity level for $L_M(u)$. In other words, if the initial recommendation list $L_N(u)$ already consists of highly similar items, no matter how the system filters in Stage 2, the final recommendation list $L_M(u)$ will still include similar items, offering limited diversity and fewer choices for the user.

To address this limitation, the "Diversify the recommendation list" algorithm simultaneously integrates both factors: the predicted rating value (focused on accuracy) and the distance $d(u,i)$ (focused on diversity) from the beginning, as expressed in the following formula:

$score(u,i) = k \times \hat{r}_{u,i} + \gamma \times (1-k) \times d(u,i); k \in [0.1]$

where $d(u,i)$ is the distance between item I and the profile of user u:

$$d(u,i) = 1 - sim(u,i)$$

, and

$$sim(x,y) = \cos(\vec{x},\vec{y}) = \frac{\vec{x}.\vec{y}}{\|\vec{x}\|_2 \times \|\vec{y}\|_2} = \frac{\sum_{s\in S_{xy}} r_{x,s}.r_{y,s}}{\sqrt{\sum_{s\in S_{xy}} r_{x,s}^{\ 2}}\sqrt{\sum_{s\in S_{xy}} r_{y,s}^{\ 2}}}$$

Since the predicted rating score lies in the range [1, 5] and $d(u,i)$ is between [0, 1], the algorithm scales $d(u,i)$ to the same range by multiplying it by 5. The pseudo-code is illustrated in Algorithm 2.

Algorithm 2. Diversify the recommendation list

Input: user profile of user u, I ={ items (content)}, $ListR_u = \{\hat{r}(u,i), i \in I\}$
Threshold TH , $TopM$ items to recommend

Output: $TopM$ recommendation list $L_M(u)$.

Begin

Foreach (r in $ListR_u$)
{
Compute d(u, i) ;
Compute $score(u,i) = k * \hat{r}(u,i) + (1-k) * \gamma * d(u,i)$;
Sc[i] = $score(u,i)$;
}
SSc = SortItem(I*, Sc[i]);
//Function SortItem sort the items $i \in I^*$ in descending order of scores Sc[i];
$L_M(u)$ = Filter_TopM(SSc);
// Function *Filter_TopM* selects the $TopM$ items $L_M(u)$ recommend for user u;
Return $L_M(u)$;

end

Thus, after calculating $score(u,i)$ for all items, the Algorithm 1 selects the $topM$ items with the highest scores for the final recommendation list $L_M(u)$ for user u, without needing to generate an initial candidate list $L_M(u)$.

4 Experimental Procedure

The preparation process for the experiments in the paper includes: preparing the dataset, setting parameter values, selecting comparison algorithms, and choosing measures to evaluate the effectiveness of the algorithms.

4.1 Experimental data

The experiments in this paper are conducted using datasets including: MovieLens 100k (ML-100K), MovieLens 1M (ML-1M), and MovieLens 10M (ML-10M). These are considered some of the standard datasets for testing in the field of recommendation systems. The dataset ML-100K contains 100,000 ratings provided by 943 users on 1,682 movies across 19 different genres. The ML-10M dataset consists of 10 million movie ratings provided by users, along with various attributes of the items. Each movie is categorized into multiple genres, such as action, science fiction, and others. To obtain reliable results, we utilizes the pre-split training and testing datasets from

MovieLens, with 80% of the data allocated for training and 20% for testing prediction quality.

4.2 Methodology

The effectiveness of the WH algorithm is analyzed based on the coefficient k. Regarding the parameters of the algorithms, the study will select "popular" and unbiased values to ensure the objectivity of the experimental results:

- The size of $L_N(u)$ consists of the initial candidate items: N = 50 (Top_N items with the highest predicted scores)
- The effectiveness or quality of the algorithms will be analyzed based on the size of $L_M(u)$: Top_M = {10, 20, 30, 40, 50} ($M \leq N$)
- The k coefficient of the WH algorithm is chosen with a value of 0.5 (the average value)

Additionally, to ensure the reliability of the results, we only considers users who have provided a sufficient number of ratings:

- Minimum number of ratings in the testing set: $\alpha \geq 30$
- Minimum number of ratings in the training set: $\beta \geq 20$

To evaluate the effectiveness of the algorithms, the study is based on accuracy and three diversity measures. There are many measures that can be applied for the accuracy of the recommendation list, and we uses the most common measure, MAE (Mean Absolute Error), as shown in the following formula:

$$MAE = \frac{1}{n} \sum_{i \in L_M(u)} \frac{| \hat{r}_{u,i} - r_{u,i} |}{l}$$

where the denominator is l, representing the rating scale, and:

- $L_M(u)$: the Top_M list of recommended items for user u
- n: the size of $L_M(u)$
- $\hat{r}_{u,i}$: the predicted rating of user u for item i
- $r_{u,i}$: the actual rating of user u for item i

The lower the *MAE* value, the higher the accuracy of the algorithm. Therefore, the value of (1 – *MAE*) can be considered an indication of the algorithm's accuracy.

For diversity, three measures: *IntraDistance*, *AggDivNum*, and *IntraDistanceProfile* is used.

The diversity of the recommendation list LM(u) is considered as the degree of difference between the items in the LM(u) list. This diversity is often defined as the average distance between two items in the recommendation list LM(u) and is calculated using the following formula:

$$IntraDistance\big(L_M(u)\big) = \frac{2}{n(n-)} \sum_{i \in L_M(u)} d(i, i')$$

where, $d(i, i')$ is the energy distance between item i and item i'; n is the size of $L_M(u)$

Note that the value of $IntraDistance\big(L_M(u)\big)$ is higher when the diversity of $L_M(u)$ is higher. Additionally, $IntraDistance\big(L_M(u)\big)$ is considered personal and will be referred to as individual diversity, as it depends on the recommendation lists for each individual user.

Aggregate diversity is defined as the total number of items that the system has recommended to all users as shown in the following formula:

$$AggDivNum = \left| \bigcup_{u \in U} L_M(u) \right|$$

where, U is the set of users of the system

The diversity of the recommendation list $L_M(u)$ is defined as the average distance of all items in the recommendation list to the user's profile.

$$IntraDistanceProfile\left(L_M(u)\right) = \frac{1}{n} \sum_{i \in L_M(u)} d(u,i)$$

where, $d(u,i)$ is the energy distance between the profile of user u and item i; n is the size of $L_M(u)$. Note that the value of $IntraDistanceProfile(L_M(u))$ is higher when the diversity of $L_M(u)$ is higher.

Additionally, accuracy and diversity are generally opposing (increasing the value of one measure will decrease the value of the other and vice versa), so the paper also employs the $F\_Measure$ to balance these two measures as follows:

$$F\_Measure = \frac{2 \times (1 - MAE) \times \text{IntraDistanceProfile}}{(1 - MAE) + \text{IntraDistanceProfile}}$$

The experiments were conducted on a PC with the following specifications: CPU 13th Gen Intel(R) Core(TM) i9-13900H 2.60 GHz, RAM 32.0 GB (31.6 GB usable), 64-bit operating system, x64-based processor.

Subsequent paragraphs, however, are indented.

4.3 Results and experimental analysis

Below are the experimental results of the recommendation system evaluated using various metrics, applied to recommendation lists from Top 10 to Top 50. These figures illustrate the changes in accuracy, diversity, and overall performance as the recommendation range is expanded.

Table 1. The experimental results on the MovieLens -100K dataset.

Measures	**Top 10**	**Top 20**	**Top 30**	**Top 40**	**Top 50**
1-MAE	0.829	0.806	0.797	0.735	0.701
IntraDistance	0.758	0.734	0.721	0.695	0.674
AggDivNum	532	608	713	825	924
IntraDistanceProfile	0.749	0.706	0.674	0.649	0.628
F_Measure	0.787	0.753	0.731	0.689	0.663

Table 2. The experimental results on the MovieLens -1M dataset.

Measures	**Top 10**	**Top 20**	**Top 30**	**Top 40**	**Top 50**
1-MAE	0.834	0.801	0.795	0.775	0.731

IntraDistance	0.851	0.804	0.765	0.710	0.711
AggDivNum	612	711	824	898	1044
IntraDistanceProfile	0.822	0.816	0.805	0.715	0.699
F_Measure	0.828	0.808	0.800	0.744	0.715

Table 3. The experimental results on the MovieLens -10M dataset.

Measures	**Top 10**	**Top 20**	**Top 30**	**Top 40**	**Top 50**
1-MAE	0.846	0.821	0.814	0.775	0.751
IntraDistance	0.879	0.846	0.818	0.755	0.716
AggDivNum	646	724	773	876	1286
IntraDistanceProfile	0.884	0.836	0.802	0.785	0.736
F_Measure	0.865	0.828	0.808	0.780	0.743

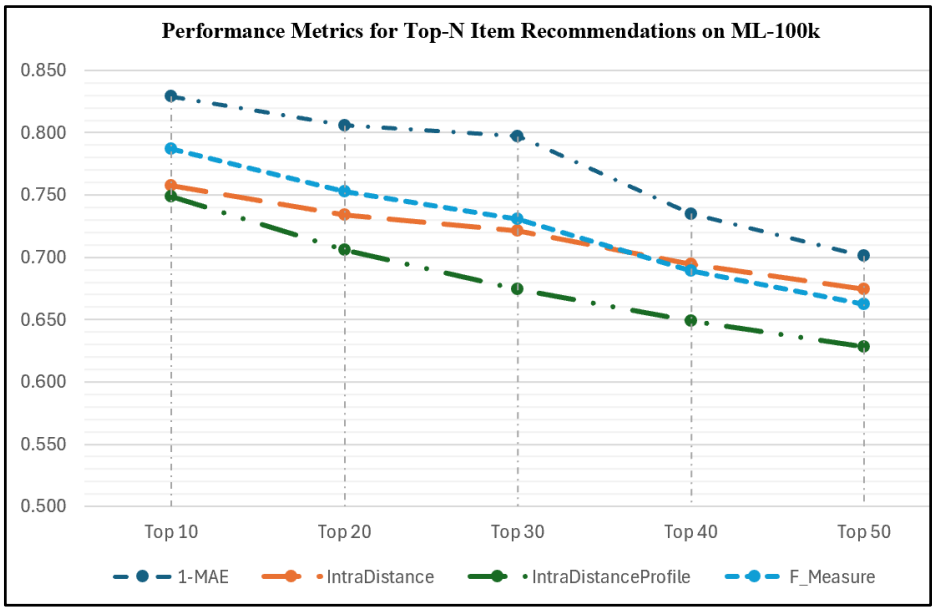

Fig. 6. The experimental results on the MovieLens -100K dataset.

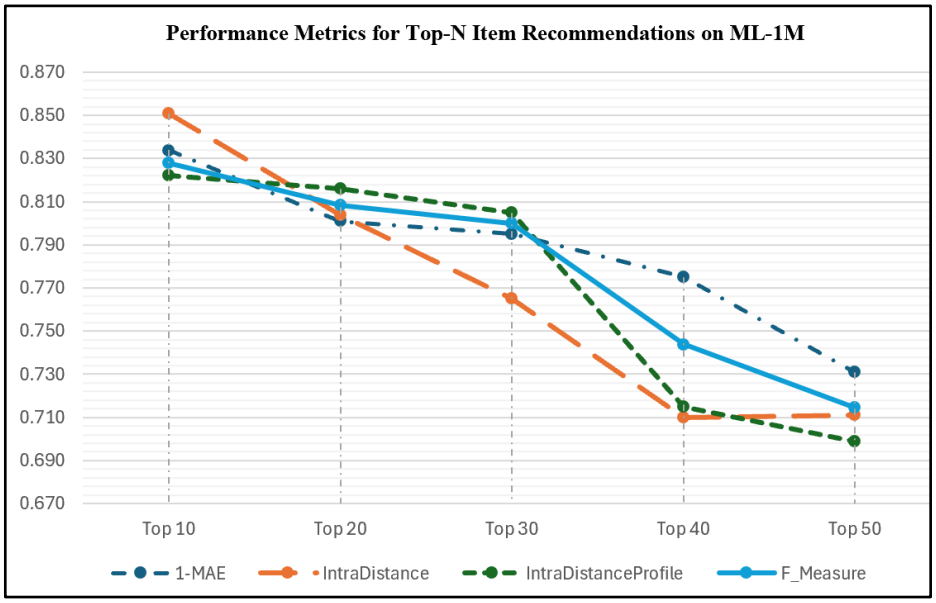

Fig. 7. The experimental results on the MovieLens -1M dataset.

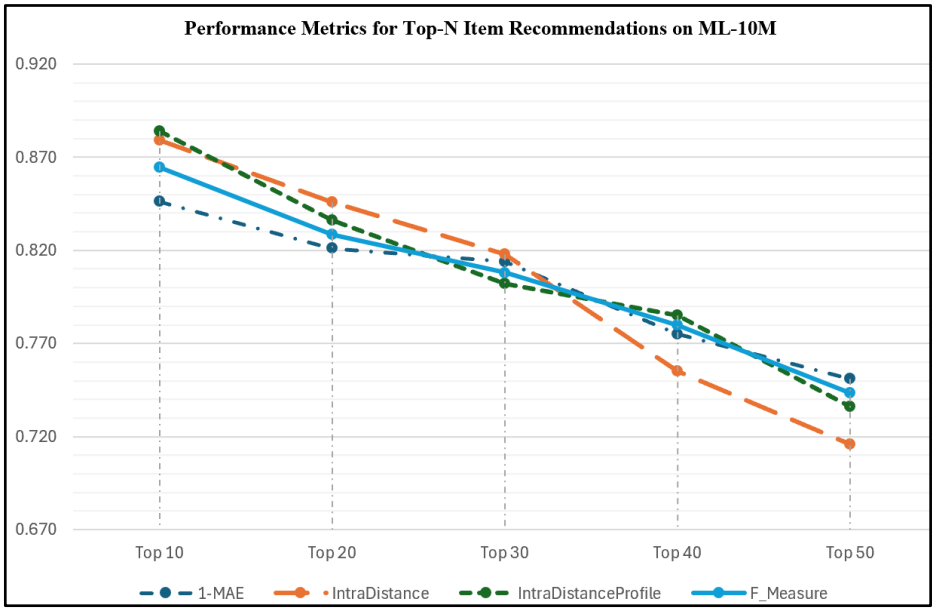

Fig. 8. The experimental results on the MovieLens -10M dataset.

The experimental results from the MovieLens datasets provide important insights into the performance of recommendation models. The Mean Absolute Error (*1-MAE*) shows a decreasing trend in error rates as the number of recommendations increases,

with the MovieLens -10M dataset reaching the highest MAE of 0.846 for the top 10 recommendations, indicating improved accuracy with more recommendations. *Intra-Distance* measures reveal that larger datasets yield higher similarity among recommended items, particularly in the top 10 (0.879) and top 20 (0.846) categories. The Aggregate Diversity (AggDivNum) significantly increases, with MovieLens -10M achieving a diversity score of 1286 for the top 50 recommendations, showcasing greater variety in suggestions. Additionally, *IntraDistanceProfile* shows lower distance values for larger datasets, suggesting tighter item clustering. Finally, *F_Measure* values peak at 0.865 for the top 10 recommendations in the MovieLens -10M dataset, highlighting the model's effectiveness in balancing precision and diversity.

Below is a comparison table of the performance of different methods on the ML-100K, ML-1M, and ML-10M datasets based on the MAE metric. The results help assess the accuracy of each method and demonstrate the competitiveness of the proposed method compared to previous approaches.

Methods	**ML-100K**	**ML-1M**	**ML-10M**
GRALS [Rao et al., 2015] [19]	0.945	-	-
CF-NADE [Zheng et al., 2016] [20]	-	**0.829**	0.771
Factorized EAE [Hartford et al., 2018] [21]	0.910	0.860	-
GC-MC [Berg et al., 2017] [22]	0.910	0.832	0.777
STAR-GCN [Jiani Zhang et al., 2019] [11]			
Proposed method	**0.854**	**0.831**	**0.769**

The performance comparison table shows that the proposed method achieves the lowest MAE on ML-100K with 0.854, indicating better accuracy compared to other methods such as GRALS (0.945) and GC-MC (0.910). On ML-1M, the proposed method has an MAE of 0.831, slightly higher than CF-NADE (0.829) but better than other methods like GC-MC (0.832). For ML-10M, the proposed method also demonstrates relatively good performance with an MAE of 0.769, lower than CF-NADE (0.771) and GC-MC (0.777), showcasing its competitiveness on large datasets.

The system achieves impressive performance by utilizing a hierarchical tree structure to group users and products based on energy-based similarity, enabling accurate modeling of preferences. The construction of SU-Graph and SI-Graph for user and product relationships allows for a deeper understanding of the data, leading to more accurate predictions. Training the Graph Convolutional Network (GCN) to predict ratings for unseen products enhances the accuracy and personalization of recommendations. Finally, refining the recommendation list through a balance of precision and diversity ensures that users receive relevant and varied suggestions, increasing overall satisfaction.

5 Conclusion

This paper proposes an improved method for enhancing the quality of recommendation lists by combining clustering structures with Graph Convolutional Networks. Key contributions include utilizing a hierarchical tree to group users and products based on energy-based similarity measures, constructing graphs representing relationships between users and products, and training the GCN to predict ratings for unseen products. Experimental results on the MovieLens dataset indicate that the accuracy of predictions improves as the number of recommendations increases, with the highest MAE value of 0.846 for the top 10 recommendations in the MovieLens -10M dataset. Additionally, the IntraDistance measures show higher similarity among recommended items, while the Aggregate Diversity Number reflects greater variety in suggestions. Finally, F_Measure values peak at 0.865, highlighting the model's effectiveness in balancing precision and diversity, thereby providing a better user experience.

References

1. Lin, W., et al., *Transformer-empowered content-aware collaborative filtering.* arXiv preprint arXiv:2204.00849, 2022.
2. Tran, T.C.T., L.P. Phan, and H.X. Huynh, *Energy-based collaborative filtering recommendation.* International Journal of Advanced Computer Science and Applications, 2022. **13**(7).
3. Mouhiha, M., O.A. Oualhaj, and A. Mabrouk. *Combining Collaborative Filtering and Content Based Filtering for Recommendation Systems*. in *2024 11th International Conference on Wireless Networks and Mobile Communications (WINCOM)*. 2024. IEEE.
4. Glauber, R. and A. Loula, *Collaborative filtering vs. content-based filtering: differences and similarities.* arXiv preprint arXiv:1912.08932, 2019.
5. He, X., et al. *Neural collaborative filtering*. in *Proceedings of the 26th international conference on world wide web*. 2017.
6. Li, C., et al. *Hierarchical Representation Learning for Bipartite Graphs*. in *IJCAI*. 2019.
7. Wang, X., et al. *Neural graph collaborative filtering*. in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 2019.
8. Tan, Q., et al. *Learning to hash with graph neural networks for recommender systems*. in *Proceedings of The Web Conference 2020*. 2020.
9. Sun, J., et al. *Multi-graph convolution collaborative filtering*. in *2019 IEEE International Conference on Data Mining (ICDM)*. 2019. IEEE.
10. Sun, J., et al. *Neighbor interaction aware graph convolution networks for recommendation*. in *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 2020.

11. Zhang, J., et al., *Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems.* arXiv preprint arXiv:1905.13129, 2019.
12. Wu, L., et al. *Joint item recommendation and attribute inference: An adaptive graph convolutional network approach*. in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 2020.
13. Le, T.M. *AN IMPROVEMENT OF R-TREE FOR CONTENT-BASED IMAGE RETRIEVAL*. in *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae. Sectio Computatorica*. 2022.
14. Rizzo, M.L. and G.J. Székely, *Energy distance.* wiley interdisciplinary reviews: Computational statistics, 2016. **8**(1): p. 27-38.
15. Edelmann, D., T.F. Móri, and G.J. Székely, *On relationships between the Pearson and the distance correlation coefficients.* Statistics & probability letters, 2021. **169**: p. 108960.
16. Tran, T.C.T., L.P. Phan, and H.X. Huynh, *Approach of Item-Based Collaborative Filtering Recommendation Using Energy Distance.* Journal of Advances in Information Technology, 2024. **15**(1).
17. Wang, H., N. Wang, and D.-Y. Yeung. *Collaborative deep learning for recommender systems*. in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015.
18. Volkovs, M., G. Yu, and T. Poutanen, *Dropoutnet: Addressing cold start in recommender systems.* Advances in neural information processing systems, 2017. **30**.
19. Rao, N., et al., *Collaborative filtering with graph information: Consistency and scalable methods.* Advances in neural information processing systems, 2015. **28**.
20. Zheng, Y., et al. *A neural autoregressive approach to collaborative filtering*. in *International Conference on Machine Learning*. 2016. PMLR.
21. Hartford, J., et al. *Deep models of interactions across sets*. in *International Conference on Machine Learning*. 2018. PMLR.
22. Berg, R.v.d., T.N. Kipf, and M. Welling, *Graph convolutional matrix completion.* arXiv preprint arXiv:1706.02263, 2017.