Optimizing E-commerce Recommendations Using TDGCN-L: Integrating Explicit Feedback for Better User Satisfaction

Abstract

Recommender systems are essential in real-world applications, especially on e-commerce platforms where both accuracy and diversity impact user satisfaction. While Graph Convolutional Networks (GCNs) have advanced diversity-aware recommendations, many existing methods overlook explicit feedback such as ratings and reviews, resulting in reduced accuracy. To address this, we propose TDGCN-L, a Trade-off GCN enhanced with Large Language Models (LLMs). The model integrates structural, behavioral, and semantic signals to balance diversity and accuracy. First, we decompose the dataset into a user-item interaction matrix and a rating matrix to represent implicit and explicit feedback, respectively. Then, GCN layers learn embeddings from interaction data, while LLM-derived semantic embeddings are extracted from item and user content. These embeddings are fused and fed into a Fully Connected Neural Network (FCNN), supervised by Root Mean Square Error (RMSE) loss. A similarity matrix is computed from the final embeddings, followed by k-means clustering to group similar users. Lastly, a similarity-aware filtering algorithm adjusts Top-N recommendations based on cluster-level preferences. Experiments on five public datasets show that TDGCN-L improves diversity—achieving an ILD (Intra-List Diversity) gain of 0.1—while maintaining competitive accuracy against state-of-the-art baselines.

Keywords: E-commerce recommender system, Graph convolutional network, Large Language Models, Diversity-accuracy trade-off, User satisfaction

1. Introduction

Recommender systems have become indispensable in e-commerce, where personalized product suggestions can enhance user experience and drive business success(Sarwar et al., 2001; Dai et al., 2019; Liu et al., 2022). By mitigating information overload, these systems enable users to navigate extensive product catalogs more efficiently (Burke et al., 2011; Sun et al., 2015; Hazrati and Ricci, 2022). For many years, research in this field focused primarily on improving accuracy—often using explicit user ratings or reviews—yet higher accuracy alone does not necessarily yield higher user satisfaction (Aytekin and Karakaya, 2014; Costa and Roda, 2011; Ahmadian et al., 2023).

Indeed, an overemphasis on accuracy tends to generate recommendations for popular or closely related items that users might already know, weakening the perceived benefit of these systems (Zuo et al., 2023). As a result, diversity has emerged as a key factor for improving user engagement and satisfaction (Wang et al., 2019; Yin et al., 2023). Without diversity, users may see repetitive or overly homogeneous recommendations, diminishing interest and negatively affecting sales. For example, an e-commerce platform might repeatedly suggest workout equipment to a fitness enthusiast, neglecting items such as nutritional supplements or fitness apparel and thus failing to capture the user's broader interests.

To address these limitations, a growing body of work has explored graph-based recommender systems for enhanced diversity. Since user-item interactions can be represented as bipartite graphs, graph algorithms are naturally suited for capturing complex relationships (Tang et al., 2021). Early studies such as Lee and Lee (2015) employed an undirected graph of positively rated items to optimize novelty and relevance via entropy-based techniques, yet they provided limited solutions for challenges such as cold-start or high data sparsity in e-commerce.

Recent advances incorporate Graph Neural Networks (GNNs) to harness relational information more effectively (Lathia et al., 2010; Beel et al., 2013). For example,Zheng et al. (2021) embedded diversity objectives directly into a GCN's matching phase, while He et al. (2020) refined GCN via improved neighborhood aggregation strategies. Further improvements emerged from Yang et al. (2023) and Su et al. (2024), who introduced sophisticated embedding procedures and contrastive learning frameworks to bolster representation quality.

Meanwhile, with the emergence of large language models (LLMs), such as

GPT and BERT-style architectures, the integration of pretrained semantic knowledge into recommendation systems has shown great promise (Acharya et al., 2023; Fan, 2024). LLMs can enhance item and user representation by extracting rich semantic features from textual descriptions, reviews, and metadata—particularly useful in domains where content understanding is critical. However, current GNN-based models seldom fully utilize this potential, and many rely solely on structural or interaction signals, overlooking the high-value semantic information embedded in textual contexts.

Despite these developments, many GNN-based methods largely focus on implicit interactions (clicks, views) or domain-specific tasks. For instance, Rao et al. (2024) proposed Adaptive Graph Contrastive Learning (AGCL) to model multi-faceted dependencies among Points of Interest (POIs) using contrastive learning. While AGCL excels at capturing transition patterns in location-based recommendations, it overlooks the richness of explicit rating data—critical in many e-commerce platforms—reducing its effectiveness where such feedback is abundant.

In e-commerce, explicit ratings and textual reviews often play a decisive role in defining user preferences (Wei et al., 2020; Xu et al., 2024). Ignoring these signals forfeits fine-grained semantic cues that are essential for nuanced personalization. When combined with implicit behaviors—such as browsing logs and purchase histories—explicit feedback offers a comprehensive view of user intent. Recent advances in LLMs (Zou et al., 2025), such as GPT and BERT-style architectures, have further opened the possibility of extracting rich semantic embeddings from unstructured text. LLMs can serve as powerful tools to distill context-aware representations from item descriptions, reviews, and query histories. However, few existing recommender systems effectively integrate LLM-generated semantic embeddings into graph-based learning pipelines, leaving a critical opportunity unexploited.

Two main challenges persist in e-commerce recommender systems:

- Maximizing utilization of rating datasets: While many graph-based algorithms increase diversity, they often neglect explicit feedback and textual semantics that are readily available in e-commerce environments. A unified model that incorporates both user behavior and LLM-enhanced semantic information is vital for capturing fine-grained user preferences.

- Achieving an accuracy–diversity trade-off: While many graph-based algorithms increase diversity, they often neglect explicit feedback and textual semantics that are readily available in e-commerce environments. A unified model that incorporates both user behavior and LLM-enhanced semantic information is vital for capturing fine-grained user preferences.

Our goal is to address these gaps by fully leveraging both explicit ratings and semantic textual information in a graph-based framework to enhance recommendation diversity without incurring prohibitive accuracy losses. By integrating implicit behaviors, explicit feedback, and LLMs–based semantics, we aim to provide a scalable and comprehensive solution tailored for modern e-commerce platforms. Specifically, this study presents TDGCN-L (Trade-off Diversity Graph Convolutional Network(GCN)), a novel recommendation model that combines GCNs with Fully Connected Neural Networks(FCNNs) and pretrained language models to address the limitations of current approaches. Our method enhances the utility of rating datasets and semantic metadata while effectively balancing diversity and accuracy, ultimately improving user satisfaction and engagement.

To tackle the first challenge, we decompose the e-commerce dataset into a user-item interaction matrix and a rating matrix. The interaction matrix reflects implicit behavioral signals such as clicks and purchases, while the rating matrix contains explicit numerical ratings that indicate user preferences. In parallel, we extract textual information—such as user profiles, item descriptions, and review texts—and use a pretrained LLMs to derive semantic embeddings. We then apply a GCN to the interaction graph to learn structural embeddings, and fuse them with the LLM-based semantic embeddings. These fused representations are input into a fully connected network, trained with Root Mean Square Error (RMSE) loss against the rating matrix, resulting in refined embeddings that jointly encode behavioral, structural, and semantic signals.

For the second challenge, we compute a similarity matrix from the learned user embeddings and apply k-means clustering to group users into smaller, more homogeneous clusters based on their preferences. This user segmentation enables fine-grained personalization by accounting for local similarity structures in the embedding space. Additionally, we introduce a similarity-aware filtering function that adjusts the top-N recommendation list to fine-tune the trade-off between diversity and accuracy. This mechanism ensures that the generated recommendations remain both varied and relevant, catering to the nuanced and evolving needs of individual users.

Our key contributions are as follows:

1. We bridge the gap between graph structure and semantic information by introducing a unified embedding strategy that integrates user-item interaction signals (via GCN) and textual semantics (via pretrained LLMs). Unlike prior graph-based models that rely solely on structural data or ignore rich textual content, our approach captures both behavioral patterns and semantic preferences—offering a more expressive and informative representation space.
2. We propose TDGCN-L, a new recommendation architecture that integrates GCNs with FCNNs. This model leverages the strengths of both components: GCNs capture structural relationships among users and items, while FCNNs enhance the accuracy of preference modeling based on explicit feedback. By unifying these perspectives, TDGCN-L improves diversity without substantially compromising accuracy.
3. We introduce a similarity-aware filtering mechanism to address the challenge of balancing accuracy and diversity. By clustering users based on their learned embeddings and adjusting filtering strategies dynamically, our method provides a flexible filtering strategy to optimize recommendation performance according to user-specific preference patterns.
4. Extensive experiments on five real-world rating datasets demonstrate the effectiveness of our approach. TDGCN-L consistently achieves about 10% improvements in diversity metrics while competing admirably with accuracy-based models.

The remainder of this paper is organized as follows. In Section 2, we introduce the methodology of our proposed model. Section 3 presents experimental evaluations on various e-commerce datasets, including comparative and ablation studies. Finally, we conclude the paper in Section 4.

2. Related Work

In this section, we review the literature on accuracy-centric recommendation systems, graph-based diversified recommendations and diversity-accuracy tradeoff recommendation.

2.1. *Accuracy-centric recommendation systems*

Early research in recommender systems predominantly focused on improving the accuracy of recommendations, often measured by metrics such as Precision, Recall, and RMSE. These approaches typically rely on collaborative filtering (CF) techniques, which include memory-based and model-based methods. Memory-based CF, such as user-based and item-based methods, calculates similarity using heuristic measures like cosine similarity or Pearson correlation (Valcarce et al., 2019). Model-based CF, including matrix factorization (MF) (Mnih and Salakhutdinov, 2007) and probabilistic latent semantic analysis (PLSA) (Huang et al., 2019; Xu et al., 2005), learns latent representations of users and items from historical interaction data. Among them, MF and its variants, such as SVD++ (Koren, 2008) and NMF (Luo et al., 2014), have shown remarkable performance in capturing user preferences and item characteristics. Moreover, deep learning-based models, such as autoencoders and neural collaborative filtering (NCF) (He et al., 2017), have been introduced to better capture complex and nonlinear patterns in user-item interactions.

Despite the impressive accuracy achieved by these models, researchers have pointed out their tendency to overfit popular items and ignore niche interests. This often leads to recommendation redundancy and filter bubbles, undermining the diversity of results and user satisfaction (Yang et al., 2023).

2.2. *Graph-based diversified recommendation systems*

To address the limitations of accuracy-centric models, recent research has turned to graph-based methods for enhancing diversity in recommendation systems. In these methods, user-item interactions are modeled as a bipartite graph, where nodes represent users and items, and edges denote interactions. The graph structure allows for better representation of relational information and contextual dependencies.

Traditional graph-based diversity models, such as those proposed by Lee and Lee (2015), construct item-item graphs based on positively rated items and apply entropy measures to select novel yet relevant recommendations. These approaches successfully capture item correlations, but suffer from data sparsity and cold start issues due to their reliance on positive interactions.

With the development of Graph Neural Networks (GNNs), especially GCNs, graph-based recommendations have seen significant progress. GCNs enable layer-wise aggregation of neighbor information, thus capturing higher-order connectivity in the user-item graph. Zheng et al. (2021) proposed a GCN-based diversity model that integrates diversity constraints into the matching process, yielding more varied and user-relevant results. Building on this, He et al. (2020) found that nonlinear transformations in GCNs might not contribute significantly and proposed LightGCN, which uses simplified neighborhood aggregation to retain essential collaborative signals while reducing complexity. Furthermore, Yang et al. (2023) attempted to diversify recommendations by refining the embedding learning process within GNN frameworks, emphasizing the embedding's structural diversity.

In addition, contrastive learning has been employed to further enhance diversity in graph-based recommendations. Su et al. (2024) proposed DCL, a contrastive learning framework that introduces category-level supervision into GNNs. By aligning embeddings of semantically related items, DCL encourages structural diversity within the recommendation space. Rao et al. (2024) designed AdaGCL for POI recommendation, incorporating adaptive graphs and multi-faceted contrastive learning. Although these models yield promising results in specific tasks, they often neglect the explicit feedback data such as ratings, thus limiting their generalizability to rating-based datasets.

2.3. Diversity-accuracy tradeoff recommendation systems

While diversity enhances user satisfaction by mitigating redundancy and promoting content exploration, excessive diversity can lead to irrelevant or less personalized recommendations, highlighting the need to balance accuracy and diversity. Several studies have attempted to address this tradeoff by integrating diversity-promoting objectives into the recommendation process.

One common approach is to introduce diversity-aware loss functions or post-processing re-ranking strategies that penalize recommendation redundancy while maintaining relevance. For instance, methods such as Maximal Marginal Relevance (MMR) (Wu et al., 2023) and Determinantal Point Processes (DPP) (Gan et al., 2020) aim to optimize for both diversity and accuracy during recommendation ranking. However, these techniques are often applied as an afterthought to base recommendation models, limiting their ability to jointly learn user preferences and diversity-aware representations.

Graph-based models offer a promising avenue for balancing accuracy and diversity. Recent works have begun embedding diversity constraints directly into the GNN framework to optimize both objectives simultaneously. For example, to mitigate the aforementioned limitations, Yang et al. (2023) proposed DGRec, a GNN-based model that incorporates submodular neighbor selection and layer-wise attention to promote long-tail item exposure while maintaining personalization. Meanwhile, building on its contrastive foundation, DCL further incorporates user–category and item–category bipartite graphs to balance diversity with relevance. This multi-graph design enables the model to uncover underexplored content while mitigating noise from sparse interactions (Su et al., 2024).

To achieve an effective accuracy-diversity tradeoff, it is essential to leverage both implicit and explicit user feedback. Rating-based datasets, which contain detailed information on user preferences, offer an opportunity to bridge this gap. However, most current models underexploit the full richness of rating-based datasets, which include both implicit interactions and explicit preference signals. This gap motivates our hybrid approach that decomposes the rating data into interaction and rating matrices and leverages a GCN-FCNN architecture to optimize both accuracy and diversity.

3. Preliminaries

In this section, we provide a concise overview of some background methods, including GNN (Zheng et al., 2021) and LightGCN(He et al., 2020), to establish context.

3.1. GNNs

Most data in recommender systems is inherently structured as a graph. For instance, interaction data can be represented by a bipartite graph connecting user and item nodes, where observed interactions are denoted by edges. Graph-based representations are advantageous when integrating structured external information, such as social relationships among users or knowledge graphs about items. GNNs provide a unified framework for effectively modeling such data.

Given a graph, the core concept of GNNs is to iteratively aggregate feature information from neighboring nodes and integrate this aggregated information with the current node's representation during the propagation process. Architecturally, GNNs consist of multiple propagation layers, each containing aggregation and update operations. The propagation process is formulated as follows:

$$\text{Aggregation: } n_v^{(l)} = f_{\text{aggregate}}\left(\{h_u^{(l)} \mid u \in \mathcal{N}_v\}\right), \tag{1}$$

$$\text{Update: } h_v^{(l+1)} = f_{\text{update}}\left(h_v^{(l)}, n_v^{(l)}\right), \tag{2}$$

where $h_u^{(l)}$ denotes the representation of node u at layer l, $\mathcal{N}_v$ represents the set of neighboring nodes of v, and $f_{\text{aggregate}}(\cdot)$ and $f_{\text{update}}(\cdot)$ are the aggregation and update functions, respectively.

After L layers of propagation, we obtain $L+1$ different representations for node v, namely $h_v^{(0)}, h_v^{(1)}, \ldots, h_v^{(L)}$. The final representation of each user and item is generated by concatenating these embeddings:

$$h_v = f_{\text{concat}}\left([h_v^{(0)}, h_v^{(1)}, \ldots, h_v^{(L)}]\right). \tag{3}$$

3.2. LightGCNs

LightGCN utilizes linear propagation on the user-item interaction graph to learn embeddings for users and items. It aggregates embeddings learned across all layers using a weighted sum to generate the final embedding. The propagation rules in LightGCN are defined as:

$$e_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} e_i^{(k)}, \tag{4}$$

$$e_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} e_u^{(k)}, \tag{5}$$

where $e_u^{(k)}$ and $e_i^{(k)}$ represent the embeddings of user u and item i at layer k, respectively. $\mathcal{N}_u$ denotes the set of items interacted by user u, and $\mathcal{N}_i$ denotes the set of users who have interacted with item i. The symmetric normalization term $\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}$ prevents the scale of embeddings from increasing during the convolution operations.

After K layers of propagation, we aggregate the embeddings from each layer to construct the final representation of a user or an item:

$$e_u = \sum_{k=0}^{K} \alpha_k e_u^{(k)}, \quad e_i = \sum_{k=0}^{K} \alpha_k e_i^{(k)}, \tag{6}$$

where $\alpha_k \geq 0$ signifies the importance of the k-th layer embedding in composing the final embedding.

The model predicts the interaction between user u and item i by calculating the inner product of their final representations:

$$\hat{y}_{ui} = e_u^\top e_i. \tag{7}$$

4. Methodology

In this section, we provide a detailed introduction to our model, covering aspects such as the user-item interaction, the recommendation process, the loss function, and the time complexity analyses of TDGCN-L. Most of the symbols used are standard in related literature; for better readability, we summarize the main notations in Table 1 for easy reference.

Table 1: **Main notations used in this paper.**

Notation	Description
u	A user
i	An item
N_u	Set of items that are interacted by user u
N_i	Set of users that interact with item i
R	User-item rating matrix
A	Adjacency matrix for user-item graph
e_u, e_i	Embeddings of users and items
L	Number of GCN layers
k	Number of clusters
R_{ui}	Rating of user u on item i
$\hat{y}_{ui}$	Predicted rating of user u on item i
λ	The regularization parameter
w_{ui}	Binary weights of user u on item i
$simM$	Similarity matrix

4.1. Overview of Our Approach

Our proposed TDGCN-L model is illustrated in Figure 1. We begin by partitioning the collected data into rating data and interaction data, as elaborated in Section 4.2. We transform the interaction data into a bipartite graph. We then employ LightGCN (He et al., 2020) augmented with fully connected layers to generate user and item embedding vectors. Subsequently, we compute a similarity matrix based on the dot product of these vectors. Utilizing this matrix, we cluster users to achieve the final recommendations.

Our TDGCN-L model includes the following key components:

1. Data preprocessing: We begin by preprocessing the original rating dataset to extract both the user-item interaction matrix and the ex-

plicit rating matrix. The interaction matrix is then converted into a bipartite graph structure suitable for LightGCN.

2. User-item representation learning: We model user-item structural interactions using a multi-layer LightGCN. To incorporate explicit feedback, the embeddings produced by GCN are passed through a FCNN, trained with supervision from the rating matrix using RMSE loss. In parallel, semantic embeddings are extracted from a pretrained LLM and linearly projected to the same dimensionality for fusion with the GCN outputs.
3. User clustering: We employ k-means clustering to group users based on the similarity matrix generated from the user and item embeddings. We analyze how varying the number of clusters impacts the recommendation results.
4. The balance of diversity and accuracy for top-N recommendation: To achieve a better balance between accuracy and diversity, we introduce a filter function. This function allows us to modify the filtering strategy based on the inclusion of similar or dissimilar users/items, thereby supporting both accuracy-prioritized and diversity-prioritized scenarios.

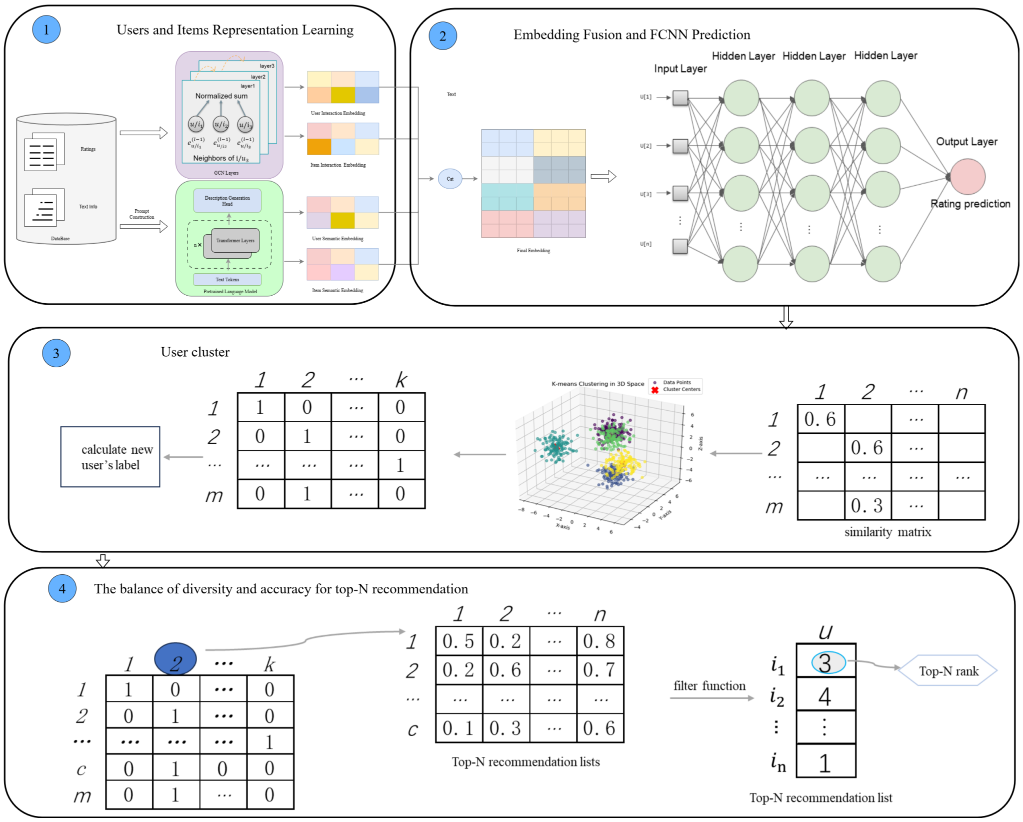


Figure 1: The architecture of the proposed TDGCN-L model.

4.2. Data Preprocessing

Before training, we preprocess the data by splitting the rating data into two initial matrices: the user-item interaction matrix and the user-item rating matrix, as shown in Figure 2. Explicit zero ratings—where users assign a numerical rating of zero—are excluded from model training to avoid introducing noise. This is because such ratings may reflect factors unrelated to the user's true preferences, such as accidental clicks. In contrast, implicit interactions (e.g., clicks or views) are retained using binary weights. Specifically, we apply a binary weighting strategy for implicit interactions: a value of 1 indicates the presence of an interaction, and 0 indicates its absence.

$$R_{ui} = \begin{cases} 1, & \text{if an interaction exists} \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

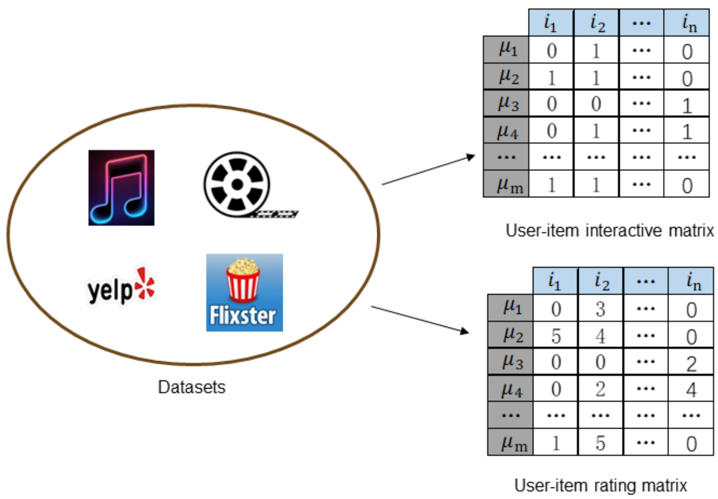


Figure 2: Illustration of decomposing the dataset into the user–item interaction matrix and the user–item rating matrix.

We adopt the matrix form from LightGCN for ease of implementation. Let $R \in \mathbb{R}^{M \times N}$ represent the user-item interaction matrix, where M and N denote the number of users and items, respectively. Each entry R_{ui} is 1 if user u has interacted with item i, and 0 otherwise. We then construct the adjacency matrix A of the user-item graph as follows:

$$A = \begin{pmatrix} 0 & R \\ R^\top & 0 \end{pmatrix}. \tag{9}$$

This adjacency matrix captures the connections between users and items in the bipartite graph.

4.3. Users and Items Representation Learning

We represent user-item interaction data as a bipartite graph G_{ui}, and generate embeddings using two encoders: LightGCN for structural patterns and a GPT-based model for semantic content. These embeddings are fused and passed to a FCNN, which is trained on explicit ratings. This hybrid approach allows us to enhance recommendation diversity while maintaining accuracy.

4.3.1. GCN Layers

Users and items interact in a way that can be effectively depicted as a heterogeneous graph, more specifically as a bipartite graph comprising users and items. Many graph-based recommendation algorithms have been proposed that leverage this representation, ranging from simple random walk (Jiang et al., 2018) methods to more complex approaches like GCN (Wu et al., 2019; Ying et al., 2018). In the user-item bipartite graph, a user's higher-order neighbors are likely to encompass a broader range of diverse items. This is because these neighbors not only include items interacted with by the user but also items preferred by other similar users. To make better use of neighbor information, Zheng et al. (2021) enhance diversification at the upstream candidate generation stage using GCN. Their experiments achieved very good results. However, He et al. (2020) proved that their model was too redundant and simplified it based on theirs, namely LightGCN (Light Graph Convolutional Network), as introduced in Section 3.2.

4.3.2. LLM-based Semantic Embedding

While user-item interaction graphs provide valuable structural information, many e-commerce platforms contain rich textual content that reflects user preferences and item characteristics more explicitly. To fully exploit such semantic information, we incorporate pretrained LLMs to generate contextual embeddings for both users and items. This approach complements the graph-based structural embeddings produced by the GCN layer and enhances the representational capacity of our model.

Specifically, we utilize textual sources such as product descriptions, user-generated reviews, and item attributes as inputs to a pretrained transformer-based language model. For each item, we aggregate its associated textual content and feed it into the LLM to extract a high-dimensional semantic vector. Similarly, for each user, we encode all available reviews or textual interactions authored by the user. In cases where direct textual data for users is unavailable, we adopt an aggregate representation based on the semantic embeddings of items the user has interacted with.

Formally, let T_i denote the aggregated textual input for item i, and let $\mathrm{LLM}(\cdot)$ denote the semantic encoder derived from a pretrained language model (e.g., BERT or GPT-based variants). The semantic embedding $e_i^{(\mathrm{sem})} \in \mathbb{R}^d$ for item i is computed as:

$$e_i^{(\mathrm{sem})} = \mathrm{LLM}(T_i) \tag{10}$$

Likewise, the user semantic embedding $e_u^{(\text{sem})}$ can be computed as:

$$e_u^{(\text{sem})} = \frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} e_i^{(\text{sem})} \tag{11}$$

where $\mathcal{I}_u$ denotes the set of items user u has interacted with.

These semantic embeddings are then fused with the structural embeddings learned from the GCN layer (see Section 4.3.1). The fused representation is subsequently passed to the FCNN for supervised learning, as detailed in Section 4.3.3. This joint representation enables the model to capture both topological and semantic signals, which is especially beneficial in cold-start and sparse-data scenarios.

By integrating LLM-based embeddings, our model is able to leverage rich linguistic knowledge and domain-specific context, thereby enhancing the overall quality and diversity of recommendations.

4.4. Embedding Fusion and FCNN Prediction

To fully leverage both graph structural information and semantic signals from textual metadata, we fuse the embeddings generated by the GCN and the pretrained LLMs, and feed the combined representations into a FCNN with three hidden layers for rating prediction, as illustrated in Figure 3. This design allows the model to jointly capture user-item interaction topology and semantic preferences derived from content.

Let $e_u^{(\text{gcn})}$ and $e_v^{(\text{gcn})}$ denote the structural embeddings of user u and item v learned via LightGCN layers 3.2, and let $e_u^{(\text{sem})}$, $e_v^{(\text{sem})}$ represent their semantic embeddings generated via LLM 4.3.2. The final input representation for each user-item pair is constructed by concatenating their respective embeddings:

$$x_{uv} = \left[e_u^{(\text{gcn})} \;\middle|\; e_v^{(\text{gcn})} \;\middle|\; e_u^{(\text{sem})} \;\middle|\; e_v^{(\text{sem})} \right] \tag{12}$$

This fused vector $x_{uv} \in \mathbb{R}^m$ is then passed through a three-layer FCNN to predict the corresponding rating $\widehat{r_{uv}}$. The FCNN architecture consists of:

$$\text{Layer 1:} \quad x_1 = \text{ReLU}(W_1 x_{uv} + b_1) \in \mathbb{R}^{64} \tag{13}$$

$$\text{Layer 2:} \quad x_2 = \text{ReLU}(W_2 x_1 + b_2) \in \mathbb{R}^{32} \tag{14}$$

$$\text{Output:} \quad \widehat{r_{uv}} = W_3 x_2 + b_3 \in \mathbb{R} \tag{15}$$

The model is optimized using Root Mean Square Error (RMSE) loss between predicted and actual ratings, along with ℓ_2 regularization to prevent overfitting:

$$\mathcal{L} = \frac{1}{m} \sum_{(u,v,r_{uv})} (\widehat{r_{uv}} - r_{uv})^2 + \lambda \sum_{\theta \in \Theta} |\theta|^2 \tag{16}$$

The entire training process is outlined in Algorithm 1, where the embeddings are computed via LightGCN layers, semantic vectors are retrieved from pretrained LLM encoders, and both are concatenated and passed through the FCNN for final prediction. Parameters are updated via backpropagation using gradient descent.

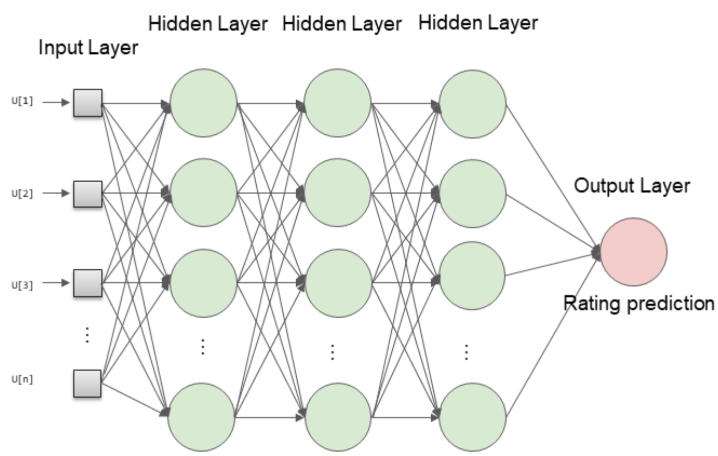


Figure 3: Architecture of the FCNN with three hidden layers.

Algorithm 1: Joint training of LightGCN and LLM with FCNN

Input: Training dataset $\mathcal{D}$; user-item interaction graph G_{ui}; initial model parameters Θ; learning rate α; regularization coefficient λ; number of epochs E; batch size m

Output: Trained model parameters Θ^*

Construct Laplacian matrix L_G from G_{ui};
Randomly initialize model parameters Θ;
Divide $\mathcal{D}$ into batches $\{B_k\}_{k=1}^{|\mathcal{D}|/m}$;
for $t = 1$ **to** E **do**
  $L_{\text{total}} \leftarrow 0$; `// Reset batch loss`
  for *each batch* $B_k = \{(u_j, i_k, r_{jk})\}$ **do**
    Compute LightGCN embeddings:: $e^{(l)} \leftarrow \text{ReLU}(L_G \cdot e^{(l-1)})$ for $l \in \{1, \ldots, L\}$;
    Compute semantic embeddings via LLM: $e_u^{(sem)} \leftarrow \text{LLM}(u_j)$, $e_i^{(sem)} \leftarrow \text{LLM}(i_k)$
  ;
  Concatenate: $e_{\text{final}} \leftarrow \text{concat}(e_u^{(gcn)}, e_i^{(gcn)}, e_u^{(sem)}, e_i^{(sem)})$
;
Predict: $\hat{r}_{jk} \leftarrow \text{FCNN}(\text{concat}(e_u, e_i))$;
Compute loss: $L \leftarrow \frac{1}{m} \sum (\hat{r}_{jk} - r_{jk})^2 + \lambda \sum_{\theta \in \Theta} \|\theta\|^2$;
Clear gradients: $\nabla_\Theta \leftarrow 0$;
Compute gradients: $\nabla_\Theta \leftarrow \frac{\partial L}{\partial \Theta}$;
Update parameters: $\Theta \leftarrow \Theta - \alpha \cdot \nabla_\Theta$;
Accumulate loss: $L_{\text{total}} \leftarrow L_{\text{total}} + L$;
return $\Theta^* \leftarrow \Theta$;

4.5. User Clustering

We generate a similarity matrix based on the user and item embeddings obtained from the previous steps. Specifically, we compute the similarity matrix simM by taking the dot product of the user embedding matrix $E_u \in \mathbb{R}^{M \times K}$ and the item embedding matrix $E_i \in \mathbb{R}^{K \times N}$:

$$\text{simM} = E_u E_i. \tag{17}$$

We then apply the k-means clustering algorithm to group users based on this similarity matrix. The k-means algorithm minimizes the sum of squared Euclidean distances between data points and their corresponding cluster centroids (Wang et al., 2012):

$$d = \sum_{k=1}^{K} \sum_{i=1}^{n} \|x_i - \mu_k\|^2, \tag{18}$$

where K is the number of clusters, x_i is the i-th data point, and μ_k is the centroid of the k-th cluster.

We utilize methods such as the Elbow Method and the Average Silhouette Score to determine the optimal number of clusters K:

- Elbow method: We plot the within-cluster sum of squares (WCSS) against different values of K and look for the "elbow" point where adding more clusters does not significantly reduce WCSS.
- Average Silhouette Score: We calculate the silhouette coefficient for each data point and average them. A higher average silhouette score indicates better-defined clusters.

4.6. The balance of diversity and accuracy for top-N recommendation

To balance recommendation diversity and accuracy, we employ k-means clustering combined with a filtering function. For each user u, we identify the cluster C_u to which u belongs and calculate the average similarity avg_sim between u and other users v in the cluster:

$$\text{avg\_sim}_u = \frac{1}{|C_u| - 1} \sum_{\substack{v \in C_u \\ v \neq u}} \text{sim}(u, v) \tag{19}$$

We set a pivot value as the filtering threshold:

$$\text{pivot}_u = p \times \text{avg\_sim}_u, \tag{20}$$

where p is a tunable parameter controlling the filtering strength. This pivot serves as the basis for two filtering strategies: (a) diversity-oriented filtering (filter out users with similarity lower than pivot_u to promote diversity), or (b) accuracy-oriented filtering (filter out users with similarity higher than pivot_u to prioritize accuracy).

We define the filtering function $f(u_i)$ as:

$$f(u_i) = \begin{cases} 1, & \text{if similarity}(u, u_i) \geq \text{pivot}, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

Here, u_i represents other users in the same cluster as u. If $f(u_i) = 1$, the items associated with u_i are added to u's recommendation list. By adjusting p and the filtering criteria, we can prioritize diversity or accuracy as needed. More precisely, we adjust filtering parameters for different user segments by dynamically scaling the threshold based on cluster density (see Algorithm 2). In dense clusters, where the average similarity ave_sim_u is high, a larger pivot_u naturally emerges. This leads to stricter filtering and promotes greater diversity. For example, when $\text{avg\_sim}_u = 0.8$ and $p = 0.8$, the resulting threshold $\text{pivot}_u = 0.64$ filters out more users, thereby broadening the recommendation scope. Conversely, in sparse clusters where similarity values are lower, the threshold remains relatively lenient. This preserves more high-similarity users, thereby maintaining recommendation accuracy. In summary, the parameter p provides global control over this trade-off, enabling flexible and systematic adjustment to accommodate different recommendation goals. Finally, we select the top-N items with the highest predicted ratings as the final recommendations.

Algorithm 2: Adaptive threshold calculation and filtering

Input: target user u; clusters $\{C_1, C_2, \ldots, C_K\}$; parameter p; top-N items

Output: Recommendation list R_u

Assign user u to a cluster C_u via k-means clustering;
for *each user* $v \in C_u$*, where* $v \neq u$ **do**
  Compute similarity: $\text{sim}(u, v) \leftarrow \text{cosine\_similarity}(u, v)$;
Compute cluster-specific average similarity:
  $\text{average\_sim}_u \leftarrow \frac{1}{|C_u|-1} \sum_{\substack{v \in C_u \\ v \neq u}} \text{sim}(u, v)$;
Set adaptive threshold: $\text{pivot}_u \leftarrow p \times \text{average\_sim}_u$;
if *Diversity-Priority Mode* **then**
  Define filter: $f(v) = \begin{cases} 1, & \text{if } \text{sim}(u, v) < \text{pivot}_u \\ 0, & \text{otherwise} \end{cases}$;
else if *Accuracy-Priority Mode* **then**
  Define filter: $f(v) = \begin{cases} 1, & \text{if } \text{sim}(u, v) > \text{pivot}_u \\ 0, & \text{otherwise} \end{cases}$;
Generate candidate pool: collect all users v where $f(v) = 1$;
Rank items from candidate users by similarity and select top-N:;
$R_u \leftarrow \text{argsort}_{v \in \text{candidates}}(\text{sim}(u, v))[: N]$;
return R_u;

4.7. Time Complexity Analysis

We analyze the time complexity of TDGCN-L, assuming the following:

- M and N are the numbers of edges and nodes in the user-item graph, respectively.
- L and L' are the numbers of layers in the GCN and FCNN, respectively.
- F and F' are the input and output feature dimensions of the GCN layers.
- H is the output dimension of the FCNN.
- K is the number of clusters.
- D is the dimension of each data point.

- I is the number of iterations in the k-means algorithm.
- T is the token length of item descriptions or metadata input to the LLM.
- d is the hidden size of the LLM (e.g., 768 for BERT-base).

The total time complexity of TDGCN-L consists of four parts:

1. **Data preprocessing**: Generating the user-item bipartite graph has a complexity of $O(M)$.
2. **LLM-based semantic embedding**:
    - We assume the LLM (e.g., BERT) is used to generate embeddings for N items or users.
    - The complexity for each item is approximately $O(Td^2)$, due to the self-attention mechanism in transformer blocks.
    - Thus, the total complexity is $O(NTd^2)$. Since this process can be done offline, it is treated as a one-time cost.
3. **Embedding generation**:
    - *GCN*: Each layer involves:
        - Feature-weight multiplication: $O(NFF')$
        - Adjacency-feature multiplication: $O(MF')$

      Total for L layers: $O\left(L\left(MF' + NFF'\right)\right)$
    - *FCNN*: The FCNN prediction layer has a complexity of $O\left(L'NF'H\right)$

   Total embedding generation cost:
   $$O\left(L\left(MF' + NFF'\right) + L'NF'H\right)$$
4. **User recommendation and post-processing**:
    - *User Clustering (k-means)*: $O\left(INKD\right)$
    - *Similarity filtering* adds negligible overhead compared to k-means and can be amortized.

Therefore, the total time complexity of TDGCN-L is:

$$O\left(M + NTd^2 + L\left(MF' + NFF'\right) + L'NF'H + INKD\right) \tag{22}$$

This comprehensive analysis demonstrates that TDGCN-L is computationally feasible for practical applications in e-commerce recommender systems.

5. Experiments

In this section, we evaluate the performance of our proposed TDGCN-L model through a series of experiments. We begin by introducing the datasets used, the evaluation metrics, and the experimental settings. We then present the results, including comparisons with baseline methods and an ablation study to demonstrate the effectiveness of each component of our model.

5.1. Datasets

We utilize five widely-used real-world benchmark datasets in recommender systems to evaluate our model:

- MovieLens-100K (Harper and Konstan, 2015): Contains 100,000 ratings from 943 users on 1,682 movies.
- MovieLens-1M (Harper and Konstan, 2015): Contains 1,000,209 ratings from 6,040 users on 3,706 movies.
- Yelp2018 (Asghar, 2016): A dataset from Yelp containing 1,561,406 ratings from 31,668 users on 38,048 businesses.
- YahooMusic (Dror et al., 2012): Contains 5,335 ratings from 3,000 users on 3,000 songs, known for its high sparsity.
- Flixster (Monti et al., 2017): Contains 26,173 ratings from 3,000 users on 3,000 movies, with ratings ranging from 0.5 to 5 in 0.5 increments.

The statistics of these datasets are summarized in Table 2.

Table 2: Statistics of the datasets.

Dataset	**#Users**	**#Items**	**#Ratings**	**Density (%)**
MovieLens-100K	943	1,682	100,000	6.30
MovieLens-1M	6,040	3,706	1,000,209	4.47
Yelp2018	31,668	38,048	1,561,406	0.13
YahooMusic	3,000	3,000	5,335	0.06
Flixster	3,000	3,000	26,173	0.29

We randomly split each dataset into training (80%), validation (10%), and test (10%) sets. The validation set is used for hyperparameter tuning, while the test set is reserved for evaluating the final performance metrics.

5.2. Evaluation Metrics

To assess the effectiveness and diversity of our top-K recommendations, we employ five widely-used evaluation metrics:

- Precision@K: Measures the proportion of relevant items in the top-K recommendations.
- Recall@K: Measures the proportion of relevant items retrieved in the top-K recommendations out of all relevant items.
- F1-Score@K: The harmonic mean of Precision@K and Recall@K.
- RMSE): Evaluates the accuracy of predicted ratings compared to the ground truth.
- Intra-List Dissimilarity (ILD) (Han and Yamana, 2017): Measures the diversity among recommended items.

The formulas for Precision, Recall, and F1-Score are as follows:

$$\text{Precision@K} = \frac{|N(u) \cap M(u)|}{|N(u)|}, \tag{23}$$

$$\text{Recall@K} = \frac{|N(u) \cap M(u)|}{|M(u)|}, \tag{24}$$

$$\text{F1-Score@K} = 2 \times \frac{\text{Precision@K} \times \text{Recall@K}}{\text{Precision@K} + \text{Recall@K}}, \tag{25}$$

where $N(u)$ is the set of top-K recommended items for user u, and $M(u)$ is the set of relevant items for user u.

The RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{(u,i)\in\Omega} (y_{ui} - \hat{y}_{ui})^2}, \tag{26}$$

where n is the number of ratings, y_{ui} is the ground truth rating, and $\hat{y}_{ui}$ is the predicted rating.

The ILD is calculated as:

$$\text{ILD} = \frac{1}{|R| \times (|R| - 1)} \sum_{r_1 \in R} \sum_{r_2 \in R, r_2 \neq r_1} \text{dissim}(r_1, r_2), \tag{27}$$

where R is the recommendation list for a user, and $\text{dissim}(r_1, r_2)$ quantifies the dissimilarity between items r_1 and r_2.

By default, we set $K = 300$ and report the average metrics across all users in the test set. Because this study primarily focuses on accuracy and diversity, we do not present a direct runtime comparison. However, a detailed complexity analysis of our algorithm appears in Section4.7.

5.3. Experimental Settings

Our model is implemented using PyTorch, leveraging its efficient computation capabilities. We set the embedding size to 32 and employ three GCN layers in the architecture. The Adam optimizer is used for training, with a batch size fixed at 2048 except for MovieLens-1M, where a larger batch size of 10240 is applied due to dataset scale. The learning rate is fine-tuned over the set $\{0.0001, 0.0005, 0.001, 0.005\}$, and the ℓ_2 regularization coefficient is selected from $\{0.0001, 0.001, 0.01, 0.1, 1\}$. Unless otherwise specified, we construct 5 links for each user and item in the generated graphs. Model parameters are initialized via the Kaiming initializer.

For the semantic embedding component, we utilize a pretrained GPT-based language model (MiniGPT or a comparable lightweight variant) to extract item-level semantic representations. Textual inputs—comprising movie titles, genres, and, when available, summaries—are tokenized and truncated to a maximum sequence length of 128 tokens. The output embedding dimension of the LLM is 768, which is further projected to 32 dimensions through a learnable linear transformation before being fused with the GCN-based embeddings.

5.4. Effect of the Number of Clusters (K)

We investigate the impact of varying the number of clusters, K, on the recommendation results using the MovieLens-100K dataset. We vary K from 5 to 36 and present the results in Table 3.

As shown in Table 3 and Figure 4, when K is small, the recommendation accuracy (Precision, Recall, and F1-Score) is higher, but the diversity (ILD) is lower. As K increases, the diversity improves significantly at the expense of a slight decrease in accuracy. Specifically, at $K = 34$, the ILD reaches its maximum value of approximately 1.7574, with only a marginal drop in precision (from 13.35% to 11.85%).

Table 3: Performance metrics of TDGCN-L with different values of K on MovieLens-100K.

K	**Prec.** (%)	**Rec.** (%)	**F1** (%)	**ILD**	K	**Prec.** (%)	**Rec.** (%)	**F1** (%)	**ILD**
5	13.35	51.48	21.20	0.4633	21	12.33	48.76	19.68	1.1830
6	13.32	51.74	21.19	0.5732	22	12.27	48.70	19.60	1.3934
7	13.20	51.17	20.98	0.7582	23	12.19	48.12	19.45	1.3772
8	13.24	51.93	21.10	0.7473	24	12.11	48.06	19.35	1.4404
9	13.04	51.63	20.82	0.8265	25	12.22	48.18	19.50	1.5804
10	13.01	51.53	20.77	0.8872	26	12.14	47.47	19.34	1.6430
11	13.03	51.49	20.79	0.9732	27	12.10	47.54	19.29	1.5056
12	12.92	51.38	20.65	1.0215	28	12.07	47.35	19.23	1.5247
13	12.81	50.88	20.47	1.0534	29	11.97	46.76	19.07	1.5342
14	12.77	50.81	20.41	1.0826	30	12.02	47.14	19.15	1.5738
15	12.60	50.19	20.15	1.0463	31	11.90	46.65	18.96	1.3651
16	12.68	50.43	20.26	1.0999	32	11.90	46.46	18.95	1.6925
17	12.66	50.22	20.22	1.2383	33	11.87	46.22	18.89	1.6795
18	12.51	49.74	20.00	1.0878	34	11.85	46.25	18.86	1.7574
19	12.47	49.36	19.91	1.2937	35	11.75	45.65	18.69	1.6152
20	12.30	48.82	19.66	1.3106	36	11.79	46.11	18.78	1.7021

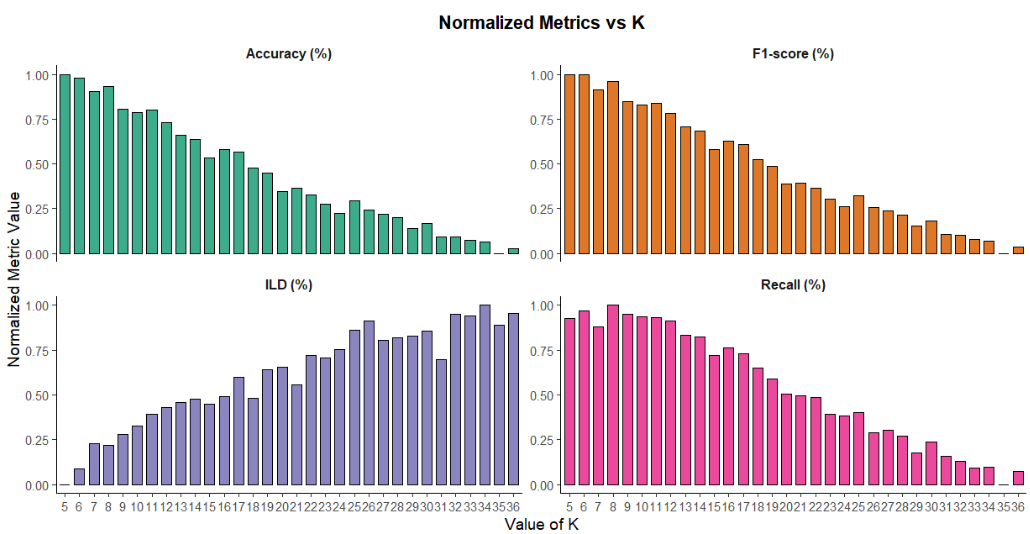


Figure 4: Performance metrics of TDGCN-L with varying K on MovieLens-100K.

These results indicate that adjusting the number of clusters allows us to balance between recommendation accuracy and diversity effectively.

5.5. Effect of the Pivot Parameter (p)

To further balance accuracy and diversity, we introduce a parameter p in our filtering function (see Section 4.6). We quantitatively analyze the impact of adjusting p on recommendation performance. As shown in Table 4, increasing p from 0.7 to 1.0 results in only a slight decrease in accuracy ($0.1228 \rightarrow 0.1185$, $\Delta = 0.0043$), while significantly improving ILD by 27.6% ($1.3771 \rightarrow 1.7574$). These findings confirm that an appropriate trade-off can effectively enhance diversity while maintaining relatively high accuracy.

Table 4: Performance metrics of TDGCN-L with different values of p on MovieLens-100K.

p	**Precision (%)**	**Recall (%)**	**F1-Score (%)**	**ILD**
0.70	12.28	48.43	19.59	1.3771
0.72	12.25	48.29	19.55	1.3957
0.74	12.23	48.14	19.50	1.4112
0.76	12.20	48.02	19.46	1.4286
0.78	12.18	47.91	19.42	1.4436
0.80	12.14	47.70	19.36	1.4589
0.82	12.12	47.61	19.33	1.4796
0.84	12.09	47.46	19.28	1.5046
0.86	12.07	47.33	19.23	1.5287
0.88	12.02	47.15	19.16	1.5570
0.90	11.99	46.98	19.11	1.5864
0.92	11.97	46.84	19.07	1.6103
0.94	11.94	46.76	19.03	1.6431
0.96	11.91	46.56	18.97	1.6759
0.98	11.88	46.40	18.92	1.7112
1.00	11.85	46.25	18.86	1.7574

Figure 5 illustrates that increasing p results in a significant improvement in ILD, indicating enhanced diversity, with only a slight decrease in accuracy metrics. This demonstrates that the pivot parameter effectively controls the trade-off between accuracy and diversity.

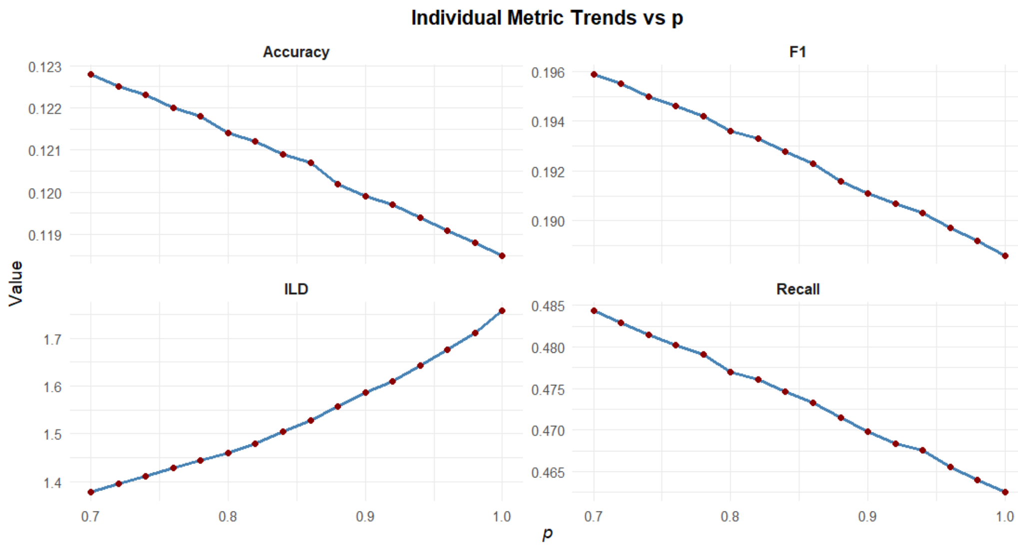
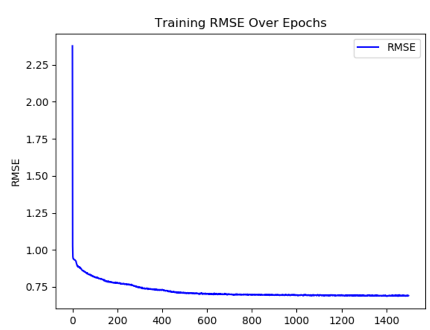


Figure 5: Performance metrics of TDGCN-L with varying p on MovieLens-100K.

5.6. *Results and Comparison with State-of-the-art Methods*

We demonstrate the variation of loss as the number of training batches increases, depicted in Figure 6, The loss undergoes a change of approximately 0.1% upon reaching 1400 batches, marking the conclusion of training, with the loss stabilizing at approximately 0.421.

(a) Training loss



(b) Training RMSE

Figure 6: Training loss and RMSE over epochs on MovieLens-100K.

To demonstrate the effectiveness of our proposed framework, we compared TDGCN-L with baselines, including two traditional models based on matrix factorization and four state-of-the-art models based on GNN. The specific details of these baselines are provided below:

- DeepFM (Guo et al., 2017): Combines factorization machines and deep neural networks for feature learning and interaction modeling. This integration significantly enhances click-through rate (CTR) prediction performance. The framework simultaneously captures both low-level feature interactions, as modeled by traditional FM, and high-level feature relationships, characteristic of deep neural networks. By enabling a comprehensive end-to-end learning process, DeepFM eliminates the need for manual feature engineering while effectively modeling complex feature interdependencies.

- DGCN (Zheng et al., 2021): Enhances recommendation diversity through GCN-based neighbor discovery and adversarial learning. It simulates collaborative filtering by applying GCN to sampled subgraphs, propagating node features between users and items, and refining negative sampling to favor similar items. DGCN achieves diversified recommendations while maintaining relevance.

- LightGCN (He et al., 2020): Simplifies GCN for collaborative filtering by removing feature transformation and nonlinear activation.

- DGRec (Yang et al., 2023): Achieves diversified recommendations using submodular neighbor selection and layer attention in GNN. DGRec enhances recommendation diversity through a tripartite module architecture designed to optimize GNN performance: a submodular selection mechanism for curating heterogeneous neighbor nodes, an adaptive layer-weighting system for capturing hierarchical feature importance, and a loss recalibration strategy that emphasizes learning from underrepresented categories.

- AdaGCL (Jiang et al., 2023): AdaGCL employs a self-supervised learning approach within collaborative filtering, integrating two adaptive view generation mechanisms—a graph construction model and a noise reduction model—to dynamically create contrasting perspectives for improved representation learning.

- DCRLRec (Bai et al., 2024): DCRLRec introduces a dual-domain contrastive reinforcement framework for recommendation by leveraging

LLMs and graph neural networks. It consists of a collaborative domain feature perception module and a semantic graph domain reinforcement module to extract user/item preferences from text and graph data, respectively. These embeddings are then aligned via a cross-domain contrastive learning strategy, enabling more comprehensive and semantically enriched user-item representations for improved recommendation performance.

We fine-tune all baseline models on the validation set and report their performance on the test set. The results are presented in Table 5.

Table 5: Overall comparison on five datasets. The best and second-best results are bolded and underlined, respectively.

Dataset	Metrics	DeepFM	DGCN	LGCN	DGRec	AdaGCL	DCRLRec	TDGCN-L
MovieLens-100K	Precision@300	**0.4321**	0.0341	0.0567	0.0424	0.2839	0.3212	0.2253
	Recall@300	0.1201	0.4282	**0.8127**	0.6342	0.5202	0.1018	0.2401
	F@300	0.1880	0.0632	0.1060	0.0795	**0.3673**	0.1546	0.2325
	ILD@300	1.4165	1.5242	1.5294	1.6261	1.4876	1.4021	**1.7264**
MovieLens-1M	Precision@300	**0.3552**	0.0360	0.0482	0.0438	0.3032	0.3012	0.2163
	Recall@300	0.0832	0.3837	0.5125	**0.5846**	0.4602	0.2725	0.2418
	F@300	0.1348	0.0658	0.0881	0.0815	**0.3656**	0.2861	0.2283
	ILD@300	1.4463	1.5451	1.5547	1.5703	1.4476	1.2865	**1.6804**
Yelp2018	Precision@300	**0.4930**	0.0401	0.0421	0.0413	0.3204	0.3392	0.1920
	Recall@300	0.0554	0.4793	0.4822	**0.4943**	0.2931	0.2532	0.2387
	F@300	0.0996	0.0740	0.0774	0.0762	**0.3061**	0.2900	0.2128
	ILD@300	1.2545	1.5432	1.5824	1.6078	1.4476	1.4874	**1.6513**
YahooMusic	Precision@300	**0.1243**	0.0224	0.0213	0.0267	0.0932	0.1034	0.1126
	Recall@300	0.0175	0.2400	**0.2683**	0.2634	0.0210	0.0263	0.1445
	F@300	0.0307	0.0410	0.0426	0.0485	0.0343	0.0419	**0.1265**
	ILD@300	0.4543	0.8012	0.8247	0.8221	0.5926	0.6423	**0.8721**
Flixster	Precision@300	**0.3321**	0.0364	0.0336	0.0402	0.2232	0.2454	0.2287
	Recall@300	0.0443	0.3925	0.4424	**0.4942**	0.1832	0.1440	0.2123
	F@300	0.0782	0.0666	0.0625	0.0744	**0.2496**	0.1815	0.2202
	ILD@300	1.1013	1.4650	1.4722	1.5121	1.3406	1.2780	**1.6840**

As shown in Table 5, TDGCN-L achieves consistently strong performance across all five datasets, particularly excelling in Intra-List Diversity (ILD) while maintaining competitive accuracy metrics such as Precision@300 and F@300. This demonstrates its effectiveness in addressing the long-standing accuracy–diversity trade-off.

On relatively dense datasets like MovieLens-100K and MovieLens-1M, TDGCN-L achieves substantially higher ILD scores (1.7264 and 1.6804, respectively), surpassing all baseline models, including diversity-enhancing methods like AdaGCL and DCRLRec. Simultaneously, its F@300 scores (0.2325 and 0.2283) remain comparable to or better than models focused on accuracy, such as DeepFM and DGCN. This suggests that our model achieves balanced recommendations without overfitting to popular items.

In sparser datasets such as Yelp2018 and YahooMusic, TDGCN-L maintains its advantage in ILD (1.6513 and 0.8721) and notably improves F@300, outperforming DGRec and AdaGCL by a large margin. For instance, on YahooMusic, TDGCN-L achieves four times higher F@300 (0.1265) compared to DGRec (0.0485), while also registering the highest diversity among all methods. These results underscore the robustness of our model in cold-start or sparse interaction settings, where semantic embedding from LLMs becomes especially beneficial. Compared with DGRec, TDGCN-L outperforms across all four metrics in every dataset. In MovieLens-100K, for example, it achieves a higher F@300 (0.2325 vs. 0.0795) and ILD@300 (1.7264 vs. 1.6261), with only a slight decrease in Recall@300. This confirms that our model provides a better trade-off by integrating semantic representations and similarity-aware filtering.

The overall accuracy–diversity trade-off is further visualized in Figure 7, which plots Precision@300 against ILD@300 on the five datasets. Models closer to the top-right corner indicate better trade-off performance. The brown star representing TDGCN-L is clearly positioned at the top-right, signifying its superior balance between diversity and precision. Compared to AdaGCL and DCRLRec, TDGCN-L improves ILD by 0.1 while maintaining competitive precision. Compared to DGRec, it significantly improves F@300 (0.2325 vs. 0.0795) with only a minor recall reduction.

5.7. Ablation Study

To investigate the contributions of individual components in TDGCN-L, we conduct an ablation study on the five datasets. Specifically, we evaluate the impact of (1) the FCNN layer for explicit preference modeling, (2) the semantic embeddings extracted via pretrained LLMs, and (3) the similarity-aware clustering and filtering mechanism. All methods are evaluated under consistent settings using Precision@300, Recall@300, F@300, and ILD@300. The results are summarized in Table 6, Table 7 and Table 8.

5.7.1. Impact of FCNN

To assess the role of the FCNN in bridging structural embeddings and rating predictions, we compare three variants:

- **GCN only:** LightGCN without FCNN or supervision.

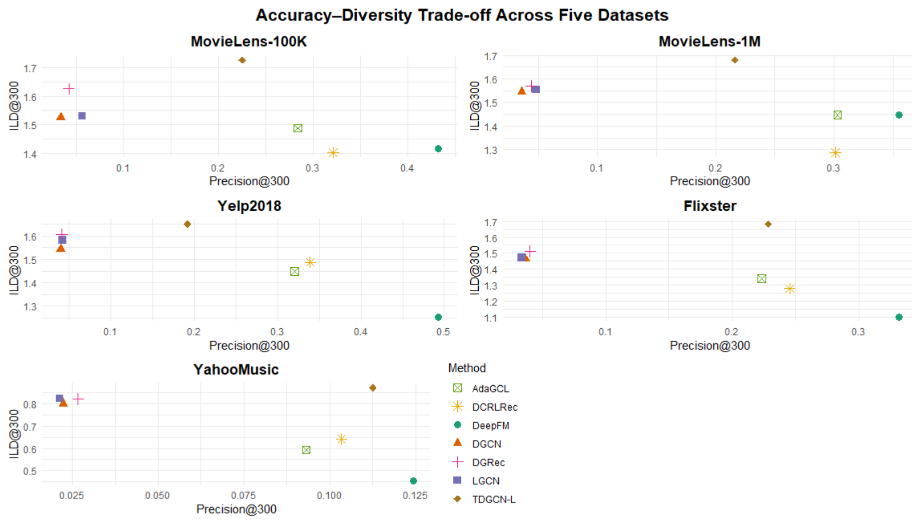


Figure 7: Accuracy–diversity trade-off comparison on five datasets.

- **GCN + FCNN:** Adds FCNN trained on rating data without semantic fusion.
- **TDGCN-L:** Full model with FCNN and semantic-aware fusion.

Table 6: Ablation study on MovieLens-100K dataset (Impact of FCNN)

Method	**Precision@300**	**Recall@300**	**F@300**	**ILD@300**
GCN	0.0567	0.8127	0.1060	1.5294
GCN + FCNN	0.2021	0.1020	0.1356	1.4832
TDGCN-L	0.2253	0.2401	0.2325	1.7264

The FCNN module significantly improves precision by capturing explicit user ratings, albeit with some loss in recall. When combined with semantic embeddings and structural signals (TDGCN-L), both accuracy and diversity improve, demonstrating the importance of joint modeling.

5.7.2. Impact of Semantic Embedding

We evaluate the contribution of LLM-based semantic representations by comparing the following:

- **TDGCN-L w/o LLM:** Removes semantic features, using only GCN and FCNN.
- **TDGCN-L (full):** Incorporates semantic embeddings from pretrained LLMs into FCNN fusion.

Table 7: Ablation study on MovieLens-100K dataset (Impact of LLM)

Method	Precision@300	Recall@300	F@300	ILD@300
TDGCN-L w/o LLM	0.1983	0.2034	0.2008	1.6123
TDGCN-L (full)	0.2253	0.2401	0.2325	1.7264

Adding semantic embeddings improves both F@300 (by +0.03) and ILD@300, indicating that language-model-derived representations help uncover latent item relationships and enhance personalization.

5.7.3. Impact of Clustering and Filtering

To assess the effect of the similarity-aware filtering mechanism, we compare:

- **TDGCN-L w/o Filtering:** Directly recommends top-scoring items without similarity-based re-ranking.
- **TDGCN-L (full):** Applies filtering based on k-means clustering and diversity-aware adjustment.

Table 8: Ablation study on MovieLens-100K dataset (Impact of Filtering)

Method	Precision@300	Recall@300	F@300	ILD@300
TDGCN-L w/o Filtering	0.2416	0.2422	0.2419	1.5981
TDGCN-L (full)	0.2253	0.2401	0.2325	1.7264

The filtering module slightly reduces precision but significantly enhances diversity (+0.13 in ILD), aligning with the model's goal of optimizing the accuracy–diversity tradeoff.

Our ablation study confirms that each module—FCNN, semantic embeddings, and similarity-aware filtering—contributes uniquely to model performance. FCNN improves precision via supervised learning, semantic embeddings enhance representation quality, and filtering ensures diverse yet personalized recommendations.

6. Conclusion

Balancing recommendation accuracy and diversity remains a fundamental challenge in the design of recommender systems. Conventional approaches often prioritize accuracy, which can inadvertently reduce the diversity of recommended items, ultimately leading to user fatigue and reduced satisfaction.

In this work, we propose TDGCN-L, a novel recommendation framework that integrates GCN, FCNN, and semantic features derived from LLMs to jointly model explicit and implicit feedback. Our model first enhances user and item representations via GCN-based neighborhood aggregation. Next, FCNN layers are applied to align learned embeddings with rating signals, minimizing prediction error. To further balance the diversity–accuracy trade-off, we introduce a similarity-aware filtering mechanism based on k-means clustering, enabling fine-grained personalization through post-processing.

Extensive experiments conducted on five public datasets demonstrate that TDGCN-L not only achieves competitive accuracy but also significantly improves diversity, outperforming state-of-the-art baselines in ILD@300 while maintaining strong F-score performance. The results validate the model's ability to jointly optimize structural interactions and rating-level supervision under a unified framework.

From a theoretical standpoint, TDGCN-L addresses the common limitation of underutilized explicit feedback in graph-based recommenders. Practically, its modular design and tunable filtering mechanism allow deployment in dynamic recommendation environments, where platform-specific goals or user preferences require adjustable emphasis on accuracy or diversity.

Nonetheless, this study has some limitations. while the proposed similarity-aware filtering mechanism offers tunable control over the accuracy–diversity trade-off, it relies on static clustering and may not fully capture dynamic user preference shifts over time.

References

B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, ACM, Hong Kong, Hong Kong, 2001, pp. 285–295. doi:10.1145/371920.372071.

W. Dai, Q. Zhang, W. Pan, Z. Ming, Transfer to rank for top-n recommendation, IEEE Transactions on Big Data 6 (2019) 770–779. doi:10.1109/TBDATA.2019.2915813.

D. Liu, J. Li, J. Wu, B. Du, J. Chang, X. Li, Interest evolution-driven gated neighborhood aggregation representation for dynamic recommendation in e-commerce, Information Processing & Management 59 (2022) 102982. doi:10.1016/j.ipm.2022.102982.

R. Burke, A. Felfernig, M. Göker, Recommender systems an overview, AI Magazine 32 (2011) 13–18. doi:10.1609/aimag.v32i3.2387.

Z. Sun, L. Han, W. Huang, X. Wang, X. Zeng, M. Wang, H. Yan, Recommender systems based on social networks, Journal of Systems and Software 99 (2015) 109–119. doi:10.1016/j.jss.2014.09.040.

N. Hazrati, F. Ricci, Recommender systems effect on the evolution of users' choices distribution, Information Processing & Management 59 (2022) 102766. doi:10.1016/j.ipm.2021.102766.

T. Aytekin, M. Karakaya, Clustering-based diversity improvement in top-n recommendation, Journal of Intelligent Information Systems 42 (2014) 1–18. doi:10.1007/s10844-013-0252-9.

A. Costa, F. Roda, Recommender systems by means of information retrieval, in: Proceedings of the International Conference on Web Intelligence, Mining and Semantics (WIMS '11), ACM, Sogndal, Norway, 2011, pp. 1–5. doi:10.1145/1988688.1988755.

M. Ahmadian, S. Ahmadian, M. Ahmadi, RDERL reliable deep ensemble reinforcement learning-based recommender system, Knowledge-Based Systems 263 (2023) 110289. doi:10.1016/j.knosys.2023.110289.

Y. Zuo, Y. Zhou, S. Liu, Y. Liu, A tag-aware recommendation algorithm based on deep learning and multi-objective optimization, in: 2023 International Conference on Pattern Recognition, Machine Vision and Intelligent Algorithms (PRMVIA), IEEE, 2023, pp. 42–46.

X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, Paris, France, 2019, pp. 165–174. doi:10.1145/3331184.3331267.

P. Yin, D. Ji, H. Yan, H. Gan, J. Zhang, Multimodal deep collaborative filtering recommendation based on dual attention, Neural Computing and Applications 35 (2023) 8693–8706. doi:10.1007/s00521-022-07756-7.

H. Tang, G. Zhao, X. Bu, et al., Dynamic evolution of multi-graph based collaborative filtering for recommendation systems, Knowledge-Based Systems 228 (2021) 107251.

K. Lee, K. Lee, Escaping your comfort zone a graph-based recommender system for finding novel recommendations among relevant items, Expert Syst. Appl. (2015) 4851–4858.

N. Lathia, S. Hailes, L. Capra, X. Amatriain, Temporal diversity in recommender systems, in: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, Geneva, Switzerland, 2010, pp. 210–217. doi:10.1145/1835449.1835486.

J. Beel, S. Langer, M. Genzmehr, B. Gipp, C. Breitinger, A. Nürnberger, Research paper recommender system evaluation a quantitative literature survey, in: Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation, ACM, Hong Kong, China, 2013, pp. 15–22. doi:10.1145/2532508.2532512.

Y. Zheng, C. Gao, L. Chen, D. Jin, Y. Li, Dgcn: Diversified recommendation with graph convolutional networks, in: Proceedings of the Web Conference 2021, 2021, pp. 401–412.

X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn simplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, Virtual Event, China, 2020, pp. 639–648. doi:10.1145/3397271.3401063.

L. Yang, S. Wang, Y. Tao, J. Sun, X. Liu, P. Yu, T. Wang, Dgrec: Graph neural network for recommendation with diversified embedding generation, in: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, 2023, pp. 661–669.

D. Su, B. Fan, Z. Zhang, H. Fu, Z. Qin, Dcl diversified graph recommendation with contrastive learning, IEEE Trans. Comput. Soc. Syst. 11 (2024) 4114–4126.

A. Acharya, B. Singh, N. Onoe, Llm based generation of item-description for recommendation system, Proceedings of the 17th ACM Conference on Recommender Systems (2023) 1204–1207. URL: https://dl.acm.org/doi/10.1145/3604915.3610647. doi:10.1145/3604915.3610647.

W. Fan, Recommender systems in the era of large language models (llms), IEEE Transactions on Knowledge and Data Engineering (2024) 1–20.

X. Rao, R. Jiang, S. Shang, L. Chen, P. Han, B. Yao, P. Kalnis, Next point-of-interest recommendation with adaptive graph contrastive learning, IEEE Transactions on Knowledge and Data Engineering (2024).

Y. Wei, X. Wang, L. Nie, X. He, T. Chua, Graph-refined convolutional network for multimedia recommendation with implicit feedback, in: Proceedings of the 28th ACM International Conference on Multimedia, ACM, Seattle, WA, USA, 2020, pp. 3541–3549. doi:10.1145/3394171.3413538.

E. Xu, K. Zhao, Z. Yu, Y. Zhang, B. Guo, L. Yao, Limits of predictability in top-n recommendation, Information Processing & Management 61 (2024) 103731. doi:10.1016/j.ipm.2024.103731.

G. Zou, P. Li, S. Yang, S. Hu, S. Pang, Y. Gan, Llm-enhanced service semantic representation and category co-occurrence feature augmentation for web api recommendation, Information Processing & Management 62 (2025) 104219. Publisher: Elsevier.

D. Valcarce, A. Landin, J. Parapar, Á. Barreiro, Collaborative filtering embeddings for memory-based recommender systems, Eng. Appl. Artif. Intell. 85 (2019) 347–356.

A. Mnih, R. Salakhutdinov, Probabilistic matrix factorization, Adv. Neural Inf. Process. Syst. 20 (2007).

L. Huang, W. Tan, Y. Sun, Collaborative recommendation algorithm based on probabilistic matrix factorization in probabilistic latent semantic analysis, Multimed. Tools Appl. 78 (2019) 8711–8722. doi:10.1007/s11042-018-6232-x.

G. Xu, Y. Zhang, X. Zhou, A web recommendation technique based on probabilistic latent semantic analysis, in: Proceedings of the International Conference on Web Information Systems Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 15–28.

Y. Koren, Factorization meets the neighborhood a multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, Las Vegas, Nevada, USA, 2008, pp. 426–434. doi:10.1145/1401890.1401944.

X. Luo, M. Zhou, Y. Xia, Q. Zhu, An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems, IEEE Trans. Ind. Inform. 10 (2014) 1273–1284.

X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, Perth, Australia, 2017, pp. 173–182. doi:10.1145/3038912.3052569.

C. Wu, Y. Wang, J. Ma, Maximal marginal relevance-based recommendation for product customisation, Enterp. Inf. Syst. 17 (2023) 1992018. doi:10.1080/17517575.2021.1992018.

L. Gan, D. Nurbakova, L. Laporte, S. Calabretto, Enhancing recommendation diversity using determinantal point processes on knowledge graphs, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, Virtual Event, China, 2020, pp. 2001–2004. doi:10.1145/3397271.3401213.

Z. Jiang, H. Liu, B. Fu, Z. Wu, T. Zhang, Recommendation in heterogeneous information networks based on generalized random walk model and bayesian personalized ranking, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, ACM, Marina Del Rey, CA, USA, 2018, pp. 288–296. doi:10.1145/3159652.3159715.

S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, T. Tan, Session-based recommendation with graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI Press, 2019, pp. 346–353.

R. Ying, R. He, K. Chen, P. Eksombatchai, W. Hamilton, 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, 2018, pp. 974–983.

Q. Wang, C. Wang, Z. Feng, J. Ye, Review of k-means clustering algorithm, Electronic Design Engineering 20 (2012) 21–24.

F. Harper, J. Konstan, The movielens datasets history and context, ACM Transactions on Interactive Intelligent Systems (TIIS) 5 (2015) 1–19. doi:10.1145/2827872.

N. Asghar, Yelp dataset challenge review rating prediction, 2016. `https//doi.org/10.48550/arXiv.1605.05362`.

G. Dror, N. Koenigstein, Y. Koren, M. Weimer, The yahoo! music dataset and kdd-cup'11, in: G. Dror, Y. Koren, M. Weimer (Eds.), Proceedings of KDD Cup 2011, volume 18 of *Proceedings of Machine Learning Research*, PMLR, 2012, pp. 3–18. URL: `https//proceedings.mlr.press/v18/dror12a.html`.

F. Monti, M. Bronstein, X. Bresson, Geometric matrix completion with recurrent multi-graph neural networks, in: Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017.

J. Han, H. Yamana, A survey on recommendation methods beyond accuracy, IEICE Transactions on Information and Systems 100 (2017) 2931–2944. doi:10.1587/transinf.2017EDP7079.

H. Guo, R. Tang, Y. Ye, Z. Li, X. He, Deepfm a factorization-machine based neural network for ctr prediction, 2017. https//doi.org/10.48550/arXiv.1703.04247.

Y. Jiang, C. Huang, L. Huang, Adaptive graph contrastive learning for recommendation, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, ACM, Long Beach, CA, USA, 2023, pp. 4252–4261. doi:10.1145/3580305.3599249.

T. Bai, X. Wang, Z. Zhang, W. Song, B. Wu, J. Nie, Gpr-opt a practical gaussian optimization criterion for implicit recommender systems, Information Processing & Management 61 (2024) 103525.