GRED: Graph Representation Editing for Efficient Dataset Recommendation

Chenyang Zhao[1,2,†], Xiaolei Du[1,2,†], Haotian Chen[1,2], Qingqing Long[1,2],
Zhiyuan Ning[3], Meng Xiao[1,2], Hengshu Zhu[1,2], Yuanchun Zhou[1,2,*]
[1]Computer Network Information Center, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]Westlake University, Hangzhou, China
zhaocy@cnic.cn, duxiaolei25@mails.ucas.ac.cn, htchen@cnic.cn, qqlong@cnic.cn,
zyningcn@gmail.com, shaow@cnic.cn, hszhu@cnic.cn, zyc@cnic.cn

***Abstract*—Graph-based recommendation systems have gained widespread adoption in recent years. Yet most methods still fail to fully capture the dynamics of user behavior efficiently and often require updating all parameters in the user-node embedding table during finetuning, making it hard to adjust to frequent changes in user behaviors. The problem is amplified in sparse interaction graphs, such as those observed in scientific dataset platforms, where users interact with only three items on average, far fewer than those in commercial platforms, leading to marked drops in accuracy. To tackle these issues, Graph Representation EDitor (GRED) is introduced as a lightweight, inject-able module set that "edits" node embeddings to enhance task adaptation. GRED utilizes LLM-based embedding with Graph Neural Network (GNN) to extract complementary semantic and structural signals from history logs. The design unifies static item content and relational structure, yielding more dynamic, and more transferable representations. GRED is used to fine-tune the model for recommendation and conduct extensive evaluations under multiple configurations. The results show that GRED achieves over 10% improvement against state-of-the-art recommendation systems in hit ratio@20 while requiring significantly less fine-tuning time and computational cost.**

***Index Terms*—Graph Representation Editing, Graph Representation Learning, Dataset Recommendation, Recommendation System, Graph Neural Network.**

I. Introduction

Recommendation systems provide users with personalized and relevant content, which are widely used in commercial platforms and academic research. Efficient recommendation systems not only improve user satisfaction and retention, but also facilitate content discovery and reduce cognitive load. They can be viewed as an information filtering system, aiming to predict users' future preferences based on their history activities [1]. Extending this paradigm to scientific data, dataset recommendation aims to automatically suggest relevant datasets that match a researcher's objectives, queries, or prior usage. Unlike traditional item or content recommendation, it must account for the semantic, structural, and contextual relationships among datasets, research topics, and user intents. Traditional recommendation systems are commonly categorized into content-based, collaborative, and hybrid approaches [2].

We have implemented collaborative filtering (CF) and content-based (CBF) recommendation methods on our dataset-sharing platform ScienceDB [3]. Here we denote each dataset as an *item*. The platform currently hosts on the order of 10^7 items, which necessitates an implicit, behavior-driven recommendation mechanism for both the dataset side panel and the scroll-based feed interface. Consequently, the recommendation algorithm must be computationally efficient and capable of generating high-quality results under strict latency constraints. Fine-tuning for frequent user behavior changes is also required. In practice, user attention is typically confined to the **top 10-20 items**, which represents the effective content capacity of the interface. Therefore, our optimization objective focuses primarily on ranking performance within this range. Although large language models (LLMs) are effective for sorting, filtering, and re-ranking, applying them in real-time implicit recommendation settings is impractical, as the end-to-end pipeline latency does not scale with the increasing number of candidate items. Through an analysis of historical recommendation logs, we further identify several distinct patterns in user access behavior that introduce significant challenges for conventional recommendation systems:

(1) User-item interactions are sparse. Traditional CF approaches typically rely on a dense user-item interaction matrix and require frequent user activity to uncover meaningful relationships. In sparsely populated graphs like ours, niche or domain-specific datasets such as those targeting small, specialized scientific communities are hardly recommended, regardless of their value.

(2) ScienceDB hosts a highly diverse collection of datasets, spanning various topics, formats, and geographic regions. While static embedding similarity-based methods can be used for recommendations in this context, they fail to capture the dynamic nature of user behavior and evolving interests.

(3) Many items remain unvisited, while new items are constantly added. Traditional CF and contrastive learning approaches require global embedding updates for each user-item addition, leading to growing computational costs and

[†]These authors contributed equally as first authors.
[*]Corresponding author.
This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No.XDB1350102.

scalability issues [4].

(4) Viewing histories on ScienceDB can reveal sensitive research interests. In CF-based systems, sparsity increases the risk of privacy leakage [5], [6]. As illustrated in Figure 1, if a cold item was viewed by A, recommending another related item can inadvertently expose that A's history. These characteristics pose significant challenges for conventional recommendation systems on our platform.

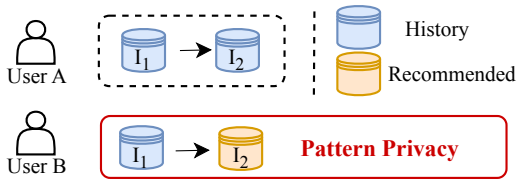


Fig. 1. An illustration of the **privacy leaks of CF based recommendation systems**. Specifically, user A have seen history items I_1 and I_2, if both items and users have no other interactions then if user B clicked I_1, in CF based system I_2 will have certain high priority in recommendation system without doubt, causing privacy leaks. This could leak user A's research interest unintentionally.

Recently, graph-based methods have gained prominence by modeling complex user-item relationships through Graph Neural Networks (GNNs) [7]–[9]. To address the limitations of CF-based recommendation systems and to explore methods in GNN fine-tuning in recommendation utilizing both content information and user behavior data, we developed a novel recommendation framework that integrates both entity-level semantic information and dynamic relational data into a unified embedding space using graph neural network. To reduce the redundant embedding updates commonly seen in CF-based systems, we leverage a sentence-transformer model based on large language models (LLMs) to generate embeddings for users and items. We construct a homogeneous graph by extracting user-item interactions and augment item connectivity with keyword nodes, which are derived from item description using LLM-based extraction. These keywords are then linked to the corresponding items to enrich graph connectivity and enhance sparse-related item recommendation. To preserve user activity signal and enhance the generalization ability of the pre-trained GNN encoder, we employ user-centric subgraph sampling. This strategy introduces controlled randomness and allows the GNN to learn robust user representations from local structures while protecting user from deterministic user-to-user inference. Finally, we incorporate Graph Representation Editing (GRED) as a lightweight and efficient fine-tuning technique. GRED enables the pre-trained GNN to adapt to evolving user behaviors without requiring full retraining, making it well-suited for dynamic recommendation scenarios. The main contributions of this work are as follows:

- We propose a GNN-based recommendation framework that integrates user/item properties and relational structures. By using sampled, keyword-enhanced subgraphs, it improves performance on sparse items and offers empirical mitigation against deterministic user-to-user inference, reducing reliance on full-graph training.
- We propose a highly efficient fine-tuning method, Graph Representation Editing (GRED), which can be seamlessly integrated into pre-trained GNNs to adapt them to downstream recommendation tasks. GRED enables localized model updates without retraining the full network.
- We verified the effectiveness of our method through extensive experiments and ablation studies. Compared with existing recommendation baselines, our approach achieves superior performance with a 10.2% relative improvement over the second-best model in Hit Ratio@20 while remaining parameter-efficient.

Due to content limitations, we are unable to include all experimental details in this paper. Additional information, including detailed experimental configurations and the LLM prompt templates used for keyword extraction, is provided in the accompanying code repository. The training and evaluation data used in our experiments contain sensitive information related to user behavior and therefore cannot be publicly released. Nevertheless, the complete source code and full experimental setup details are made available to facilitate reproducibility at: https://github.com/ZCY24-coder/gred_codebase_repo. Note that generative AIs were used to clean code and polish grammar expression of this paper.

II. Related Work

Recommendation systems have evolved from traditional collaborative filtering and content-based methods to graph-based architectures and large-model-driven approaches leveraging pre-training/fine-tuning paradigms. We review the works in related aspects as follows:

CF-based Recommendation [10] provides a survey on deep learning enhancements for CF systems, covering MLPs, CNNs, RNNs, autoencoders, RBMs, and GANs, and analyze their impacts on recommendation quality. [11] combine CF with social network signals to improve recommendation precision, demonstrating that integrating peer connections boosts relevance. These works showcase how dense neural architectures enhance traditional CF, yet most neglect higher-order structural dependencies in user interactions.

GNN-based Recommendation Graph Neural Networks (GNNs) are well-suited for modeling complex user behaviors, including multi-type interactions and higher-order relational patterns. LightGCN retains only the core mechanism of neighbor aggregation while removing self-connections, feature transformation, and nonlinear activation, thereby simplifying the model structure and improving recall and NDCG performance. MBH-GNN [12] introduces a multi-behavior, high-hop-aware design that dynamically weights diverse user-item interactions and captures long-range dependencies. In addition, [13] presents a GNN framework with layer-wise residuals and identity mappings to mitigate over-smoothing, thereby enhancing interpretability and accuracy in dynamic user scenarios. LightKG [14] introduces a lightweight GNN-based framework for knowledge graph-aware recommendation, addressing data sparsity and training inefficiency by using scalar relation encoding and linear aggregation with an efficient

contrastive layer. By adding noise to the embedding instead of graph augmentation, XSimGCL [15] enables the model to learn a more uniformly distributed representation, thereby mitigating popularity bias and significantly improving long-tail recommendation performance. BUIR [16] eliminates negative sampling by employing two mutually-learned encoders with random data augmentation, effectively alleviating data sparsity in one-class collaborative filtering.

Fine-Tuning and Prompting for Recommendation Fine-tuning LLMs to adapt to recommended tasks mainly includes the following two ways: (1) Pre-trained fine-tuning paradigms. [17] provides a comprehensive taxonomy analyzing how pre-trained language models are adapted to recommendation tasks. They discuss training strategies including fine-tuning and prompting, emphasizing transfer learning, sparsity mitigation, and efficiency improvements. (2) Prompt-Based and Parameter-Efficient Tuning [18], [19]. [18] survey prompt tuning approaches, categorizing them into direct prompt learning and transfer-based methods. They highlight innovations such as low-rank reparameterization and transfer strategies like SPoT [20] and ATTEMPT [21].

Scientific Dataset Recommendation Systems Easy scientific dataset searching can facilitate scientific discovery [22]. [23] proposes the first method that uses co-author networks to recommend datasets: starting from seed datasets, they navigate n-hop co-author graphs to identify candidate authors and associated datasets, enhanced further by graph embeddings plus metadata ranking.

While these advances have significantly improved personalization, alleviated data sparsity, and enhanced the modeling of complex user-item interactions, adapting these techniques to domain-specific scenarios, for example, scientific dataset recommendation remains challenging due to highly specialized user intents, sparse interactions, and heterogeneous metadata. These approaches emphasize fine-grained behavioral signals in GNNs, none simultaneously tackle fine-tuning GNN for dataset recommendation. In contrast, our work uniquely integrates fine-grained user behavior modeling with adaptive fine-tune strategies. By jointly leveraging multi-type user interactions and task-specific tuning, we enhance both personalization and generalization in dataset recommendation. Compared to graph fine-tuning methods GRED edits intermediate representations directly, enabling localized adaptation with minimal parameter overhead.

III. Method

In this section, we present the methodology behind our proposed framework. We begin by introducing the semantic pre-embedding process used to represent all entities in the dataset. Next, we describe the design of our graph-based recommendation framework, which is optimized for effective fine-tuning of GNNs. Finally, we explain how the proposed approach reduces task loss and improves recommendation performance.

A. Unified-Space-Based Semantic Embedding

To effectively compress high-dimensional sparse user-item interaction data into low-dimensional, dense representations, we apply semantic embedding techniques that map users and items into a shared latent space. This helps to distinguish entities within the GNN more effectively. To better capture the full scope of user intent and item semantics, we employ an LLM-based sentence encoder, the QWen3-8B-Sentence-Transformer [24] for initial embedding. This approach helps encode semantic similarity and contrast more effectively prior to graph-based processing.

Note that these embeddings are treated as **fixed** during training within our framework. Dataset descriptions and their corresponding keyword representations tend to be stable over time and do not require frequent re-embedding. Therefore, recomputing these semantic embeddings is generally unnecessary.

B. Dataset-User-Keyword Graph Construction

We collected user activity data from ScienceDB spanning from 2023 to 2024. To ensure data quality and meaningful behavioral patterns, we selected users who accessed more than 4 datasets. Users with only one or two dataset visits typically exhibited minimal engagement and rarely returned, making them less informative for modeling user intent. Our selected cohort includes both logged-in and temporary (non-login) users, with the latter introducing additional noise and reducing the overall consistency of interaction patterns.

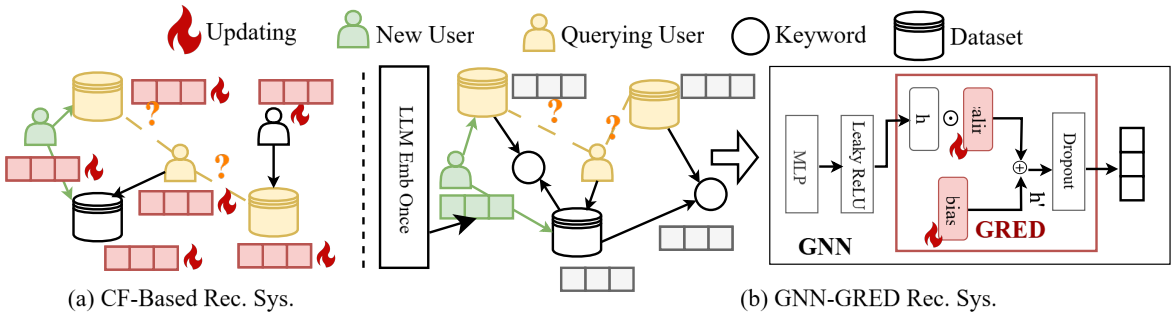
To enrich the graph with semantic context, we used QWen3-2507-8B [25] as the basic LLM to extract keywords from dataset descriptions. These extracted keywords were added to the graph as additional nodes, and linked to their corresponding datasets. This graph augmentation step was empirically shown to improve the final recommendation performance.

We identify datasets, users, and keywords as nodes in our constructed graph. The graph has two types of edges, from user to datasets and datasets to keywords. Reverse edges were added before training to ensure correct propagation.

C. Graph Representation Editor (GRED)

Inspired by the representation editing framework proposed by Wu et al. [26], which demonstrates that inserting lightweight linear mappings after Transformer blocks enables efficient fine-tuning of large language models, we extend this idea to graph neural networks. We assume that fine-tuning pre-trained graph encoder within same domain task can be approximated as a local, additive correction in the representation space. Under this assumption, the discrepancy between the pre-trained representation and the task-adapted representation can be modeled as a shifted linear approximation. As shown in Figure 2, this is a simple linear projector can be represented as follows:

$$h_i' = l_{scaling} \odot h_i + l_{bias}, \tag{1}$$



(a) CF-Based Rec. Sys. (b) GNN-GRED Rec. Sys.

Fig. 2. **Traditional CF based Rec Sys V.S. Graph Representation Editor.** The figure illustrates how a recommendation system is fine-tuned with the introduced relationships (blue). Query datasets are under recommendation comparison for the query user (yellow). Panel (b) shows the GNN-GRED-based recommendation framework, where GRED edits the GNN layer representations by injecting lightweight modules composed of linear projectors with bias terms. During fine-tuning, only a significantly smaller subset of parameters compared to those in panel (a) needs to be updated, rather than retraining the entire GNN or the full user-item embedding table.

where h_i represents the pre-edited output at i-th layer, h'_i represents the modified output at the i-th layer. While $l_{scaling}$ and l_{bias} are vectors with the same length as the embedding dimension. This transformation rescales each embedding dimension element-wise and adds a bias.

Assuming we have desired representation h_w at certain layer j, which has error e compared to current generated representation h_n, the following equation should apply:

$$h_w = h_n + e. \tag{2}$$

As for the current input x for this layer, we have a fixed continuous transformation function (Graph Encoder) $f(x) = h_n$, which should not be modified directly due to its complexity. It does not match our wanted output h_w. If we were applying representation editing, the transformed output h' can be represented as follows:

$$h' = l_{scaling} \odot h_n + l_{bias}, \tag{3}$$

Let the layer's current input be x, and the fixed encoder function be $f(x) = h_n$, which is continuous but complex and not subject to modification. Since $f(x)$ does not match the target output h_w, we apply a representation editing function, the residual δ can be obtained from the following expression:

$$\delta = h' - h_n = (l_{\text{scaling}} - \mathbf{1}) \odot h_n + l_{\text{bias}}. \tag{4}$$

If $h' \approx h_w$, then we have the approximation:

$$\delta \approx e. \tag{5}$$

Minimizing $\mathcal{L}(h')$ with respect to l_{scaling} and l_{bias} allows the GRED layer to learn a residual correction δ that improves task performance without modifying the base encoder $f(x)$. We want optimized parameters $l_{scaling}, l_{bias}$ to satisfy the following objective:

$$\min_{l_{\text{scaling}}, l_{bias}} \|e - \delta\|_2^2, \tag{6}$$

Given that parameters can be tuned, with any optimized δ^*, compared to original encoder $f(x)$ which would always have error e, we can claim that:

$$\|e - \delta^*\|_2^2 \leq \|e - 0\|_2^2, \tag{7}$$

For any given encoder, using representation editing can theoretically reduce the representation error under optimal parameterization, and leads to loss reduction during training.

Let η represent the learning rate during training, L as loss function. For each gradient descent step, we can perform the following operations:

$$\begin{aligned} l_{scaling} &\xleftarrow{update} l_{scaling} - \eta \frac{\partial \mathcal{L}}{\partial h'} \odot h_n, \\ l_{bias} &\xleftarrow{update} l_{bias} - \eta \frac{\partial \mathcal{L}}{\partial h'}. \end{aligned} \tag{8}$$

We insert GRED components between activation function and layer's dropout layer. Thus, GRED integrates seamlessly with each GNN layer to adjust the GNN's encoding to better align the GNN [27], [28] encoding with user-dataset-specific recommendation objectives. After the injection, assuming we are using GIN graph convolution, for i-th layer's input $h^{(i)}$, we will have the layer process it's value as the following expression:

$$h^{(i+1)} = f'_i(h^{(i)}) = l_{scaling} \odot GIN(h^{(i)}) + l_{bias}. \tag{9}$$

Here we use mean propagation for GIN [29] since a popular item could have exploded message aggregation if applied sum.

IV. Experiment

In this section, we present our experiment. We begin by describing the experimental settings, including datasets, evaluation metrics, and baseline models. Then we report the main results of our method compared to the baselines. Next, we conduct ablation studies to examine the contribution of each core component. Finally, we explore the task generalization capability of our GRED as a GNN fine-tune method in other domains.

A. Experimental Settings

Dataset ScienceDB is an open data platform established by the Chinese Academy of Sciences to provide multidisciplinary data services for scientific research. We collect user behavior data on the ScienceDB platform span from 2023-2024. As previously stated in I, the user behavior data was collected and sampled to build a user-dataset interaction graph. The graph containing sampled 10,000 users and 15,017 datasets with total amount of 76,511 keywords. The entire graph has 98,524 user-to-dataset-visit relationships, average density for user-dataset relation is 0.000656. For graph-fine-tune based methods, we used 45% user-interaction edges for pretraining, 5% for pre-training evaluation, 30% for fine-tuning training, 10% for fine-tune evaluation, with last 10% for testing. At the time of data collection, our platform employed CF-based methods and history-content similarity for recommendation.

1) Baseline Methods: We selected several representative recommendation methods spanning multiple modeling paradigms for comparison with our approach. These methods consist of the following: (1) Collaborative Filtering (CF)-based methods: *LightGCN* (denoted as L-GCN) removes feature transformation and non-linear operations from traditional GCNs, linearly propagating user-item embeddings over the interaction graph. The final embedding is obtained by aggregating representations from multiple layers, capturing high-order collaborative relations. (2) Contrastive Learning (CL)-based methods: *XSimGCL* replaces graph augmentations with a simple noise-based embedding perturbation, generating contrastive views that improve representation consistency and robustness. For CF/CL-based methods, we adopt a unified set of hyperparameters, including a learning rate of 1×10^{-3}, an embedding dimension of 64, and a maximum of 100 training epochs. These configurations follow the default or recommended settings reported in the original works, and are commonly used in standard recommendation benchmarks. (3) Graph Prompt based methods: *AdapterGNN* (denoted as ADGNN) injects small bottleneck-MLP with both after layer and bypass layer connection, by adjusting their parameters to fine-tune GNN network. *GPF* operates on the input graph's feature space, to align GNN representation with downstream tasks. For a fair comparison, GRED and the graph prompt-based baselines use the same graph construction and training pipeline. (4) Knowledge Graph based method: *LightKG* [14] represents a recent line of work that focuses on reducing model complexity while maintaining high recommendation accuracy. It employs a simplified GNN layer and an efficient contrastive module that directly minimizes node similarity in the original graph, significantly improving both performance and efficiency. (5) History Content Similarity: the method implemented on our site during data collection, which takes raw embedding of user's history item and calculate weighted mean embedding and retrieve items by content similarity. It's excluded from ranking due to potential data contamination.

We kept nearly original settings for those methods to get expected results. More detail regarding the baseline settings are described in the provided source code repository due to main content size limitation.

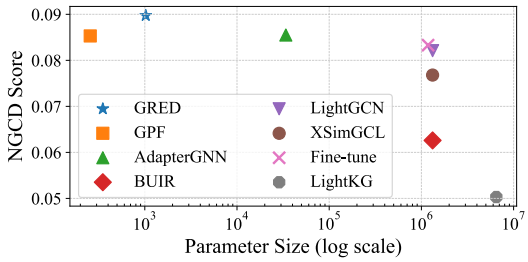


Fig. 3. NDCG@20 of different models by hot parameter size

2) GRED Settings: We use the Graph Isomorphism Network (GIN) [29] as our graph convolution method due to its proven expressiveness in capturing complex graph structures. For the initial input, we process static LLM embeddings using a linear transformation followed by a batch normalization layer, reducing their dimensionality for downstream graph processing. These compressed embeddings are then passed through multiple layers of GIN-based convolution to learn task-relevant representations. A dot-product predictor is used to assess whether a link between two nodes should exist.

The pretraining phase is formulated as a link prediction task. We randomly sample edges from the interaction graph and construct small graph patches centered around target links. Both positive (true) edges and negative (false) samples (ratio: $k = 1$) are used to guide the model in learning discriminative representations for user-dataset, dataset-keyword relationships. We perform link prediction with BPR loss as the pre-training task to leverage global structural information from the graph. A two-layer GIN convolution network with width $d = 256$ is used as the base GNN model, serving as the foundation for all fine-tuning methods. A dot product predictor is employed to compute similarity scores between node pairs. For GRED method, we apply both input edit and layer edit, using batch normalization for input normalization and layer normalization for output normalization. More settings regarding the detail of implementation is in the code repository.

3) Evaluation Metrics: Inspired by previous works ([30], [31]), we employ metric Hit Ratio@K (HR), Precision@K, Recall@K and NDCG@K to measure our model performance. We select $K = 1, 10, 20$ to get overall performance. We evaluate our model with a focus on metrics where $K \geq 10$.

B. Main Results

We highlighted the top and second results per metric in Table I. The parameters adjusted during model fine-tuning are listed below each model name. The result test variance are shown below each corresponding metric value. We report the evaluation metrics at $K = 1, 10, 20$, corresponding to single-shot recommendation, related-content recommendation page, and full-page recommendation scenarios, respectively. From the result we can conclude that:

TABLE I
PERFORMANCE METRICS OF DIFFERENT RECOMMENDATION MODELS (METRICS $\times 100$)

K	Metrics	BUIR	L-GCN	XSimGCL	HistSim	GPF	ADGNN	Fine-tune	LightKG	**GRED**
	# Param.	1.32M	1.32M	1.32M	(Ref-Only)	0.25K	33K	1.2M	6.5M	1K
1	HR	1.43 0.0011	1.97 0.0007	1.68 0.03	2.84 0.09	1.57 0.07	1.57 0.06	1.46 0.03	1.79 0.07	0.98 0.06
	Recall	1.63 0.0013	2.33 0.0009	2.07 0.01	3.89 0.09	1.98 0.06	1.94 0.07	1.92 0.04	1.19 0.05	1.20 0.07
	Precision	2.51 0.0020	3.46 0.0012	2.96 0.05	5.11 0.16	2.81 0.12	2.82 0.11	2.62 0.06	1.79 0.07	1.77 0.11
	NDCG	2.51 0.0020	3.46 0.0012	2.96 0.05	5.11 0.16	2.81 0.12	2.82 0.11	2.62 0.06	1.79 0.07	1.77 0.11
10	HR	7.92 0.0019	10.02 0.0018	9.73 0.11	9.84 0.06	10.23 0.10	10.21 0.12	9.78 0.18	10.09 0.14	11.25 0.14
	Recall	8.50 0.0008	11.19 0.0021	11.20 0.17	12.89 0.08	12.60 0.09	12.58 0.09	12.28 0.19	7.04 0.14	14.14 0.17
	Precision	1.39 0.0003	1.76 0.0003	1.71 0.02	1.77 0.01	1.84 0.02	1.83 0.02	1.76 0.03	1.14 0.02	2.02 0.01
	NDCG	5.05 0.0011	6.77 0.0010	6.58 0.07	8.57 0.10	6.97 0.06	6.93 0.09	6.79 0.06	4.12 0.06	7.18 0.08
20	HR	12.08 0.0018	15.08 0.0015	13.58 0.06	12.76 0.11	15.18 0.13	15.28 0.13	14.53 0.19	13.98 0.40	16.84 0.15
	Recall	12.68 0.0024	16.17 0.0019	14.99 0.01	16.37 0.13	18.15 0.19	18.33 0.11	17.77 0.16	10.13 0.35	20.52 0.18
	Precision	1.06 0.0002	1.32 0.0001	1.19 0.01	1.15 0.01	1.36 0.01	1.37 0.01	1.31 0.02	0.86 0.03	1.51 0.03
	NDCG	6.26 0.0013	8.21 0.0011	7.68 0.06	9.54 0.10	8.53 0.07	8.55 0.07	8.33 0.0583	5.03 0.12	8.97 0.09

(1) GRED Requires Fewer Trainable Parameters. Our experiment shows that GRED based recommendation used about 1K active parameters during training, only higher than GPF. Other CF, CL based methods require updating all embedding items to maintain performance. We mark the estimated number of updated parameters below each method name. Note that GRED parameters were *only affected by GNN width and GNN layers*, while LightGCL and other dynamic embedding methods would require *updating much more parameters* under larger sets of users and items. Therefore GRED would scale well to larger scenarios.

(2) GRED Demonstrates Strong Recommendation Performance According to our results, the proposed method consistently demonstrates competitive performance against various recommendation baselines. We observe that graph fine-tuning methods generally underperform at smaller K values. While at larger K settings, for instance, at $K = 10$, GRED achieves a relative improvement in hit ratio by 9.97% compared to GPF. At $K = 20$, the relative gain increases further, reaching 10.02% improvement compared to AdapterGNN. While methods such as LightGCN also exhibit strong performance, they come with significantly larger parameter sizes and higher computational costs. While it seems that content history similarity achieved great results when K is at small settings, it's very likely due to how the user's behavior data is collected based on current system, therefore we excluded it from ranking comparison. LightKG as a new method, can perform well under low K settings, while degrades with larger K. We attribute to dependency on fine graded knowledge graph structure which simple keywords link enhancement is unable to provide. We can observe significant performance drop in large K settings, suggesting that this method cannot effectively capture user preferences in larger batch of recommendation.

(3) Higher Standard Deviation in Graph Fine-tuning Methods. This mainly arises during the graph patch sample stage, when during the recommendation process, graph fine-tuning based methods usually need to sample a subgraph centered on selected users and target datasets to generate embeddings. It was done by the **neighbor sampling** process in the graph computation framework [32]. Only a subset of neighbors is included in the sampled patch of the calculation graph, therefore introducing more randomness. The stochasticity induced by graph sampling increases uncertainty in the model's output, which can be beneficial from a privacy perspective.

In other words, it becomes more difficult to infer another user's item preferences, offering a form of protection against user preference inference attacks. We provide an illustrative example below.

Privacy Leakage Illustrative Example: We observed the following case in our experiment:

- **User A (ID: 101345)** viewed items `3165` and `12056`.
- **User B (ID: 98650)** viewed only item `3165`.

with the *LightGCN* model, item `12056` was consistently ranked at position 1 for User B under different seeds, likely due to the shared interaction with item `3165`. In contrast, with GRED, item `12056` was ranked at position between 1,661 and 1,351 with standard variance of 137.60 for the same user on the same model under different seed. This stark difference highlights how the sampling-induced randomness in GRED disrupts direct user-to-user preference transfer, potentially enhancing user privacy. For this attempt of GRED recommendation, human examination of the first 20-items have similar semantic meaning showing that GRED could leverage item content information rather than user-preference

information only.

C. Ablation Studies

In this section, we analyze the individual components of our framework to evaluate their respective contributions to the overall recommendation performance. All other parameters and components are kept identical to those in the main experiments, unless explicitly stated otherwise.

1) Keyword-based Enhancement: In theory, better connectivity can improve recommendation performance due to a great number of items from our platform are cold therefore isolated. We employed a LLM to extract keyword nodes and link them to relevant items. These keyword nodes serve as semantic anchors that enrich the graph structure. However, due to the inherent uncertainty of LLM-generated content, this process may also introduce noise into the graph. To evaluate the impact of keyword augmentation, we compared two versions of GRED in our experiments: one trained on the enhanced graph with keyword nodes, and the other on the base graph without them. As shown in Figure 4, incorporating keyword nodes significantly improves the NDCG score across all evaluated K values. On average, the score increased by 0.0147, which is a substantial gain given the relatively low base value.

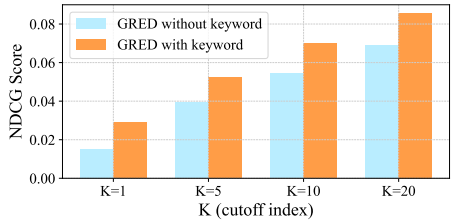


Fig. 4. Comparison of GRED with and without keyword nodes (NDCG@K).

2) Input Edit vs Layer Edit: GRED can be applied at both the input layer and the internal GNN layers. We aim to evaluate the relative contribution of each component to the overall recommendation performance. As shown in Figure 5, the *Layer Edit* component has a greater impact on Hit Ratio@20 and overall recommendation results compared to the *Input Edit*. This suggests that editing intermediate GNN representations leads to more meaningful updates for improving recommendation accuracy. Using layer edit only achieves optimal results.

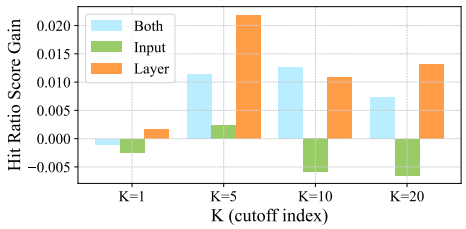


Fig. 5. Performance gain of Hit Ratio@K GRED in different edit location. Compared to fine-tune.

3) Normalization: The final prediction is based on non-normalized dot product to preserve semantic structure of GNN encoding. However, also introduced high variance across layers. After GRED is applied to the GNN layer i, a hidden dimension representation h_i may have excessively large activations for popular items. How to balance signal magnitudes within individual item embeddings and across a batch should be considered.

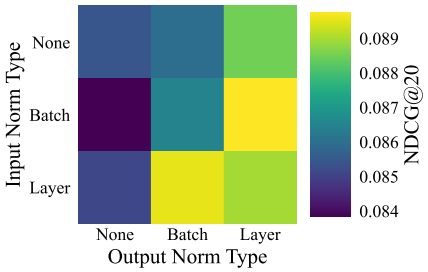


Fig. 6. GRED with different sets of normalization (Hit Ratio@20).

According to the result shown in Figure 6, layer normalization and batch normalization are preferred for GRED to achieve maximum recommendation performance. However, applying such normalization techniques tends to reduce performance in lower K metrics which can be seen in Figure 5.

4) Scaling and Bias effectiveness: To evaluate the effectiveness of scaling and bias within GRED, we designed controlled settings using input batch normalization and output layer normalization. Each component was enabled independently to assess its individual contribution to performance.

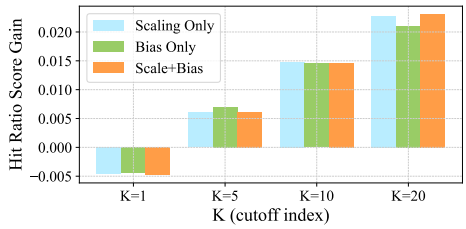


Fig. 7. Performance gain of Hit Ratio@K for GRED using scale or bias only. Compared to no-edit.

We adopt Hit Ratio as the primary evaluation metric in this study. While NDCG scores across the three settings exhibit only minor differences, this is likely due to the structural similarity between GRED and GPF under those conditions. In such configurations, stronger performance at lower-K values can significantly influence the overall NDCG. Figure 7 illustrates the performance gains achieved under both the scale and bias configurations. While using either scaling or bias alone yields comparable performance in low-K settings, their effectiveness diminishes at larger K, highlighting their complementary roles in enhancing recommendation quality.

D. Task Generalization Ability Evaluation

Besides the recommendation task built upon our own dataset, we also conduct additional experiments using public datasets from the biology and chemistry domains, as

introduced in Hu et al. [33], to verify GRED as general fine-tuning method. These experiments demonstrate that our method achieves competitive results in accuracy compared to GPF and AdapterGNN. However, node classification and graph classification tasks in the biology and chemistry domains are not directly related with the recommendation topic. A report of these experiments and their results is provided in code repository for reference.

V. Conclusion

In this work, we propose a recommendation framework under the pre-train-fine-tune paradigm, which integrates large language model (LLM) pre-embedded features with Graph Neural Networks (GNN) aggregation of user preferences over a hybrid graph structure. To enable effective fine-tuning, we introduce the **GRED**, a novel method for graph-based fine-tuning. We conduct extensive experiments on sampled data from ScienceDB. GRED achieves competitive performance while significantly reducing the number of tuning parameters compared to other approaches.

References

[1] M. Zhang, Z. Ren, Z. Wang, P. Ren, Z. Chen, P. Hu, and Y. Zhang, "Membership inference attacks against recommender systems," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 864–879.

[2] H. Ko, S. Lee, Y. Park, and A. Choi, "A survey of recommendation systems: recommendation models, techniques, and application fields," *Electronics*, vol. 11, no. 1, p. 141, 2022.

[3] Y. Zhou, P. Wang, C. Li, Z. Li, L. Jiang, Z. Zhang, and J. Liu, "The trusted system and international service capacity construction of science data bank (sciencedb)," in *China's e-Science Blue Book 2023*. Springer, 2024, pp. 427–445.

[4] R. Guan, H. Pang, F. Giunchiglia, Y. Liang, and X. Feng, "Cross-domain meta-learner for cold-start recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 8, pp. 7829–7843, 2022.

[5] C. Peng, F. Xia, M. Naseriparsa, and F. Osborne, "Knowledge graphs: Opportunities and challenges," *Artificial intelligence review*, vol. 56, no. 11, pp. 13 071–13 102, 2023.

[6] Z. Chen, Y. Wang, S. Zhang, H. Zhong, and L. Chen, "Differentially private user-based collaborative filtering recommendation based on k-means clustering," *Expert Systems with applications*, vol. 168, p. 114366, 2021.

[7] K. Yang and L. Toni, "Graph-based recommendation system," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2018, pp. 798–802.

[8] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: a survey," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37, 2022.

[9] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He *et al.*, "A survey of graph neural networks for recommender systems: Challenges, methods, and directions," *ACM Transactions on Recommender Systems*, vol. 1, no. 1, pp. 1–51, 2023.

[10] A. F. R. Abadi and J. Mohammadzadeh, "Leveraging deep learning techniques on collaborative filtering recommender systems." *Journal of Applied Research on Industrial Engineering*, vol. 10, no. 4, 2023.

[11] A. Fareed, S. Hassan, S. B. Belhaouari, and Z. Halim, "A collaborative filtering recommendation framework utilizing social networks," *Machine Learning with Applications*, vol. 14, p. 100495, 2023.

[12] G. Tan, "Nah-gnn: A graph-based framework for multi-behavior and high-hop interaction recommendation," *PloS one*, vol. 20, no. 4, p. e0321419, 2025.

[13] W. Liu, Z. Zhang, X. Li, J. Hu, Y. Luo, and J. Du, "Enhancing recommendation systems with gnns and addressing over-smoothing," in *2024 4th International Conference on Electronic Information Engineering and Computer Communication (EIECC)*. IEEE, 2024, pp. 1184–1189.

[14] Y. Li, D. Wang, Z. Sun, H. Zhang, and H. Guo, "Lightkg: Efficient knowledge-aware recommendations with simplified gnn architecture," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2025, pp. 1577–1588.

[15] J. Yu, X. Xia, T. Chen, L. Cui, N. Q. V. Hung, and H. Yin, "Xsimgcl: Towards extremely simple graph contrastive learning for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 2, pp. 913–926, 2023.

[16] D. Lee, S. Kang, H. Ju, C. Park, and H. Yu, "Bootstrapping user and item representations for one-class collaborative filtering," in *Proceedings of the 44th international ACM SIGIR conference on Research and Development in information retrieval*, 2021, pp. 317–326.

[17] P. Liu, L. Zhang, and J. A. Gulla, "Pre-train, prompt, and recommendation: A comprehensive survey of language modeling paradigm adaptations in recommender systems," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1553–1571, 2023.

[18] Z. Li, Y. Su, and N. Collier, "A survey on prompt tuning," *arXiv preprint arXiv:2507.06085*, 2025.

[19] Y. Yan, P. Zhang, Z. Fang, and Q. Long, "Inductive graph alignment prompt: bridging the gap between graph pre-training and inductive fine-tuning from spectral perspective," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 4328–4339.

[20] T. Vu, B. Lester, N. Constant, R. Al-Rfou', and D. Cer, "SPoT: Better frozen model adaptation through soft prompt transfer," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 5039–5059.

[21] A. Asai, M. Salehi, M. Peters, and H. Hajishirzi, "Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts," in *EMNLP*, 2022.

[22] Q. Long, H. Chen, C. Zhao, X. Du, X. Wang, P. Wang, C. Li, Y. Zhou, and H. Zhu, "Sciencedb ai: An llm-driven agentic recommender system for large-scale scientific data sharing services," *arXiv preprint arXiv:2601.01118*, 2026.

[23] X. Wang, F. van Harmelen, and Z. Huang, "Recommending scientific datasets using author networks in ensemble methods," *Data Science*, vol. 5, no. 2, pp. 167–193, 2022.

[24] Y. Zhang, M. Li, D. Long, X. Zhang, H. Lin, B. Yang, P. Xie, A. Yang, D. Liu, J. Lin *et al.*, "Qwen3 embedding: Advancing text embedding and reranking through foundation models," *arXiv preprint arXiv:2506.05176*, 2025.

[25] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[26] M. Wu, W. Liu, X. Wang, T. Li, C. Lv, Z. Ling, J. Zhu, C. Zhang, X. Zheng, and X. Huang, "Advancing parameter efficiency in fine-tuning via representation editing," *arXiv preprint arXiv:2402.15179*, 2024.

[27] Y. Sun, D. Zhu, Y. Wang, Y. Fu, and Z. Tian, "Gtc: Gnn-transformer co-contrastive learning for self-supervised heterogeneous graph representation," *Neural Networks*, vol. 181, p. 106645, 2025.

[28] Q. Long, Y. Jin, G. Song, Y. Li, and W. Lin, "Graph structural-topic neural network," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 1065–1073.

[29] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[30] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and 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*, 2020, pp. 639–648.

[31] J. Yu, H. Yin, X. Xia, T. Chen, J. Li, and Z. Huang, "Self-supervised learning for recommender systems: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 1, pp. 335–355, 2023.

[32] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai *et al.*, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

[33] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *International Conference on Learning Representations (ICLR)*, 2020.