LightKG: Efficient Knowledge-Aware Recommendations with Simplified GNN Architecture

Yanhui Li
Zhejiang University
Hangzhou, China
YanHuiLi@zju.edu.cn

Dongxia Wang*
Zhejiang University
Hangzhou, China
dxwang@zju.edu.cn

Zhu Sun
Singapore University of Technology and Design
Singapore
sunzhuntu@gmail.com

Haonan Zhang
Zhejiang University
Hangzhou, China
haonanzhang@zju.edu.cn

Huizhong Guo
Zhejiang University
Hangzhou, China
huiz_g@zju.edu.cn

Abstract

Recently, Graph Neural Networks (GNNs) have become the dominant approach for Knowledge Graph-aware Recommender Systems (KGRSs) due to their proven effectiveness. Building upon GNN-based KGRSs, Self-Supervised Learning (SSL) has been incorporated to address the sparity issue, leading to longer training time. However, through extensive experiments, we reveal that: (1)compared to other KGRSs, the existing GNN-based KGRSs fail to keep their superior performance under sparse interactions even with SSL. (2) More complex models tend to perform worse in sparse interaction scenarios and complex mechanisms, like attention mechanism, can be detrimental as they often increase learning difficulty. Inspired by these findings, we propose LightKG, a simple yet powerful GNN-based KGRS to address sparsity issues. LightKG includes a simplified GNN layer that encodes directed relations as scalar pairs rather than dense embeddings and employs a linear aggregation framework, greatly reducing the complexity of GNNs. Additionally, LightKG incorporates an efficient contrastive layer to implement SSL. It directly minimizes the node similarity in original graph, avoiding the time-consuming subgraph generation and comparison required in previous SSL methods. Experiments on four benchmark datasets show that LightKG outperforms 12 competitive KGRSs in both sparse and dense scenarios while significantly reducing training time. Specifically, it surpasses the best baselines by an average of 5.8% in recommendation accuracy and saves 84.3% of training time compared to KGRSs with SSL. Our code is available at https://github.com/1371149/LightKG.

*Corresponding author.

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737026

CCS Concepts

• **Information systems** → **Recommender systems**; • **Knowledge Graph** → **Graph Neural Networks**.

Keywords

Recommender Systems, Knowledge Graph, Sparse Scenarios, Graph Neural Network

ACM Reference Format:
Yanhui Li, Dongxia Wang, Zhu Sun, Haonan Zhang, and Huizhong Guo. 2025. 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 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3737026

1 Introduction

Recommender Systems (RSs) aim to capture user preferences from their behavior to filter irrelevant information in domains such as movies, news, and e-commerce [9]. Due to the Matthew effect [3], interactions in real-world scenarios often concentrate on a few popular items or active users, resulting in "sparse scenarios" where the majority of users and items have limited interactions. Such scenarios pose significant challenges to RSs in providing accurate and personalized recommendations, where poor performance could lead to user churn and significant economic losses [18].

To address this, Knowledge Graphs (KGs) have been incorporated into RSs to provide rich auxiliary information, forming Knowledge Graph-aware Recommender Systems (KGRSs) [31]. Based on the model framework, KGRSs can be categorized into three types, among which Graph Neural Network (GNN) emerges as the dominant solution [10], primarily due to their proven effectiveness in encoding hierarchical relational patterns in KGs. Recent advancements leverage Self-Supervised Learning (SSL) [36, 43] to extract more supervisory signals, enhancing recommendation performance in sparse scenarios while leading to longer training time (see Tab. 1).

However, we evaluate the recommendation accuracy of 12 KGRSs under varying levels of interaction sparsity and observe that *while GNN-based KGRSs always achieve the best performance compared to other KGRSs in dense scenarios, they generally fail to do so in sparse scenario even with SSL*. Then, we analyze the complexity of each GNN-based KGRS and find that *more complex models tend to*

Table 1: The Recall@10 and training time per epoch for GNN-based KGRSs. More recently-proposed models tend to have longer training time, indicating increased model complexity.

	Model	Amazon-book		ML-1M		Book-Crossing		Last.FM	
		Recall	time	Recall	time	Recall	time	Recall	time
Supervised method	KGCN(2019)	0.1550	0.76s	0.1594	2.13s	0.0467	0.72s	0.2149	0.71s
	KGNN-LS(2019)	0.1508	1.52s	0.1592	5.92s	0.0330	1.50s	0.2117	1.33s
	KGAT(2019)	0.1925	1.78s	0.1830	8.73s	0.0499	1.81s	0.2583	0.80s
	KGIN(2021)	0.2090	8.21s	0.1969	19.26s	0.0801	10.21s	0.2727	1.48s
Self-Supervised method	MCCLK(2022)	0.2025	8.48s	0.1853	161.48s	0.0607	59.06s	0.2699	5.01s
	KGRec(2023)	0.2035	4.47s	0.1960	99.69s	0.1033	21.15s	0.2560	3.10s
	DiffKG(2024)	0.2039	16.87s	0.1846	115.46s	0.0581	60.92s	0.2520	2.58s
	CL-SDKG(2024)	0.2036	52.52s	0.1861	281.2s	0.0924	9.39s	0.2409	1.92S
	LightKG(Ours)	0.2120	3.70s	0.2029	19.07s	0.1154	1.58s	0.2929	0.92s

perform worse in sparse scenarios. To further explore the influence of model complexity, we then design experiments to investigate whether and how simplifying GNN-based KGRSs, e.g., removing the attention mechanism employed by six representative KGRSs (e.g., KGAT [27]), would affect their recommendation accuracy. Surprisingly, the results show that *the removal mostly influences little, while often even increases the accuracy slightly*. These observations suggest that complex mechanisms in GNN-based KGRSs may be unsuitable for sparse scenarios, potentially leading to overfitting or poor generalization. This insight inspires us to look for a simpler yet effective GNN-based KGRS to address the sparsity issue.

To this end, we propose LightKG, a simple yet powerful GNN-based KGRS, to address the sparsity issue. It includes a *simplified GNN layer* to derive node embeddings and an *efficient contrastive layer* to implement SSL from the perspective of alleviating the over-smoothing issue. In the GNN layer, unlike other KGRSs that encode relations as vectors or transformation matrices, LightKG encodes them simply as scalar pairs. Then, LightKG employs these scalars to a linear aggregation framework, which significantly reduces the model's complexity. In the contrastive layer, LightKG avoids the generation and comparison of subgraphs, which is time-consuming required in previous SSL methods. Instead, LightKG directly minimizes the similarity between nodes on the original graph, which significantly reduces the training time for SSL.

Equipped with a simpler scalar-based relation encoding strategy, a linear information aggregation framework, and efficient contrastive layers, we examine how LightKG performs compared to the existing KGRSs. We apply LightKG to four benchmark datasets which vary greatly in sizes and sparsity levels. The experimental results show that LightKG outperforms 12 State-Of-The-Art (SOTA) KGRSs of diverse types on these benchmark datasets, not only with sparse but also with dense interactions. Meanwhile, LightKG shows high training efficiency compared to other KGRSs with SSL. To summarize, our contributions are highlighted as follows:

- For the first time, we empirically reveal that simpler GNN-based KGRSs can perform better with sparse interactions, whereas complex mechanisms like attention in GNNs have minimal impact or may even degrade the recommendation accuracy.
- We propose LightKG, a simple yet powerful GNN-based KGRS which incorporates a much simpler relation encoding strategy, a linear information aggregation framework and efficient contrastive layers. These designs allow LightKG to significantly reduce training time while enhancing recommendation accuracy, particularly in sparse scenarios.
- We evaluate LightKG on four benchmark datasets against 12 SOTA KGRSs. The results demonstrate its superiority in both recommendation accuracy and training efficiency. Particularly, it surpasses the best baseline by margins ranging from 1.4% to 11.7% on accuracy, and requires only 16.8% to 82.7% of the training time compared to SOTA GNN-based KGRSs with SSL.

2 Preliminary

In this section, we will briefly introduce the related concepts, including how a GNN-based KGRS generally works and the Collaborative Knowledge Graph (CKG) [27] it usually employs.

Usually a GNN-based KGRS is trained on a CKG, which consists of a user-item interaction graph $\mathcal{G}_{UI}$ and a KG $\mathcal{G}_{KG}$. For example, as shown in Fig. 2, *Tom buys a book written by Jane*, where *Tom* is a user, the *book* is an item, the author *Jane* is an entity, and the relation r between *Jane* and the *book* is 'book-author'. Specifically, let $\mathcal{U}, \mathcal{I}, \mathcal{V}$ denote the set of users, items and entities respectively. $\mathcal{G}_{UI} = \{(u, Interact, i)|u \in \mathcal{U}, i \in \mathcal{I}\}$, where *Interact* is a relation, meaning an observed interaction between user u and item i. $\mathcal{G}_{KG} = \{(h, r, t)|h, t \in (\mathcal{I} \cup \mathcal{V}), r \in \mathcal{R}'\}$, where $\mathcal{R}'$ denotes the set of relations in KG ($Interact \notin \mathcal{R}'$). The user-item graph can be integrated with a KG as a CKG: $\mathcal{G} = \{(h, r, t)|h, t \in \mathcal{K}, r \in \mathcal{R}\}$, where $\mathcal{K} = (\mathcal{U} \cup \mathcal{I} \cup \mathcal{V})$, $\mathcal{R} = \mathcal{R}' \cup \{Interact\}$.

To train a model, each node and relation will firstly be mapped to the feature space. Let $\boldsymbol{e}_k^{(0)}$ denote the embedding of node k, and $\boldsymbol{e}_r$ denote the embedding of relation r. Let $\mathcal{N}_k$ denote the set of neighbors of node k. The node embeddings are randomly initialized. Then with the involvement of the relations, the model updates the node embeddings by aggregating information from its neighboring nodes, performing it iterately for L times as shown in Equ. 1. Finally, a more accurate node embedding is obtained for RSs, given by,

$$\boldsymbol{e}_k^{(l)} = \sum_{(k,r,t)\in\mathcal{N}_k} \boldsymbol{Agg}(\boldsymbol{e}_t^{(l-1)}, \boldsymbol{e}_r), l \in 1, 2, ..., L, \tag{1}$$

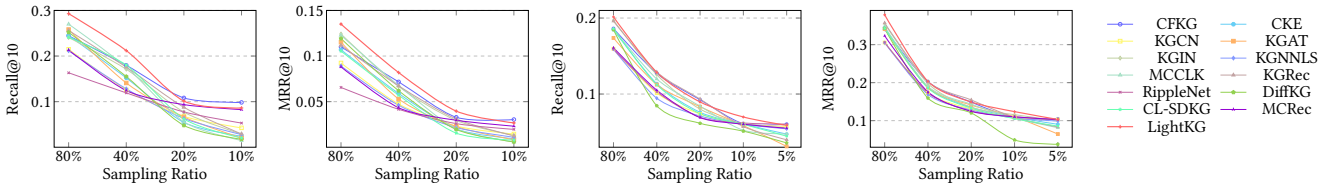
where $\boldsymbol{Agg}(\cdot)$ is a function used for embedding aggregation, which varies in different models. Embedding $\boldsymbol{e}_r$ can be encoded as vectors or matrices depending on the specific GNN-based KGRS. Together with the embeddings obtained by the aggregation in the L layers, we obtain $L+1$ embeddings for node k: $\{\boldsymbol{e}_k^{(0)}, \boldsymbol{e}_k^{(1)}, \boldsymbol{e}_k^{(2)}, ..., \boldsymbol{e}_k^{(L)}\}$. The embedding of node k for model prediction, i.e., $\boldsymbol{e}_k^*$ can be obtained using a combination function,

$$\boldsymbol{e}_k^* = \boldsymbol{Comb}(\boldsymbol{e}_k^{(0)}, \boldsymbol{e}_k^{(1)}, \boldsymbol{e}_k^{(2)}, ..., \boldsymbol{e}_k^{(L)}). \tag{2}$$

Finally, the embeddings $\boldsymbol{e}_u^*, \boldsymbol{e}_i^*$ of each user-item pair (u, i) are used to get the prediction score $\hat{y}_{ui} = \boldsymbol{e}_u^{*\top} \boldsymbol{e}_i^*$. Based on these prediction scores, all candidate items are ranked in descending order. The top-ranked items are then selected and recommended to the users.

3 Motivation

In this section, we present a series of exploratory experiments, the observations of which motivate our proposal for LightKG. We focus on scenarios where user-item interaction is sparse. First, we evaluate 12 KGRSs at different sparsity levels and observed that *while GNN-based KGRSs always achieve the best performance compared to other KGRSs in dense scenarios, they fail to do so in sparse scenario*. Then, we analyze the complexity of each GNN model and *find that higher-complexity models tend to perform poorly in sparse scenarios*. To further explore the connection between complexity and performance under sparse scenarios, we simplify several SOTA KGRSs by removing their attention mechanisms. Such removal yields a



(a) Last.FM (b) Last.FM (c) ML-1M (d) ML-1M

Figure 1: Performance of KGRSs across varying sparsity levels. Similar trends can be noted on other datasets in our study. Due to space limitations, we only present the results on Last.FM and ML-1M, with the remaining results in the appendix.

Table 2: The best Recall@10 achieved by each group of KGRSs at different sparsity levels on the Last.FM. Bold values highlight the top performance for each sparsity level. The "Improve" shows the relative improvement of LightKG over the best GNN-based KGRSs.

Model	80%	40%	20%	10%
Path_{max}	0.2132	0.1243	0.0935	0.0825
Embedding_{max}	0.2453	0.1797	**0.1082**	**0.0983**
GNN_{max}	**0.2727**	**0.1802**	0.0876	0.0425
LightKG	0.2929	0.2120	0.1012	0.0861
Improve	8.52%	17.66%	15.49%	102.40%

surprising result: *the attention mechanisms in these models seem not only ineffective but also potentially detrimental.*

3.1 Impact of Interaction Sparsity

We evaluate 12 recent and representative KGRSs on Last.FM and ML-1M datasets (detailed statistics are provided in Tab. 5), using Recall@10 and MRR@10 as evaluation metrics. To simulate varying degrees of sparse interactions, we build datasets with varying sparsity levels by adjusting the sampling ratio of the full datasets. Sampling ratios are set according to the original density levels of each dataset, with 10% and 5% representing sparse scenarios, and 80%, in comparison, serves as relatively dense scenario. Hyperparameters are tuned for all models across varying sparsity levels, including the number of GNN layers, learning rates, negative sampling ratio, and other key parameters.

Fig. 1 presents the experimental results, with the x-axis representing sampling ratio (e.g., 40% means 40% of the full dataset is sampled for training). To systematically evaluate the performance of KGRSs in sparse scenarios, we categorized them into three groups based on their frameworks [9]: path-based (MCRec, RippleNet), embedding-based (CFKG, CKE), and GNN-based (KGCN, KGNNLS, KGAT, KGIN, MCCLK, KGRec, DiffKG, CL-SDKG). Tab. 2 summarizes the highest Recall achieved by each group at different sparsity levels on the Last.FM dataset. For example, "Path_{max} in 40%" represents the highest Recall achieved by path-based KGRSs (RippleNet, MCRec) at a 40% sampling ratio.

The results reveal several key observations. **First**, as expected, the accuracy of all KGRSs declines with increasing data sparsity as shown in Fig. 1 and Tab. 2. **Second**, while GNN-based KGRSs demonstrate superior performance in dense scenarios (sampling ratios of 80% and 40%), their performance degrades more significantly in sparse scenarios (sampling ratios of 20% and 10%) even with SSL, as shown in Tab. 2. Notably, attempts to mitigate this degradation through expanded GNN receptive fields prove ineffective. In fact, introducing additional GNN layers often brings in noise, which outweighs the potential benefits of additional information [42]. **Lastly**, Fig. 1 reveals that simpler models like KGCN and KGNN-LS frequently outperform more complex counterparts such as KGAT and MCCLK in sparse scenarios. This indicates a notable correlation between model complexity and accuracy in sparse scenarios, which will be further explored in the following subsections.

Table 3: Complexity analysis and Recall@10 on Last.FM (sampling Ratio: 10%) and ML-1M (sampling Ratio: 5%). Correlation refers to the Pearson correlation coefficient between complexity ranking and Recall.

Model	Time Complexity (ranked from low to high)	Last.FM	ML-1M
KGCN(2019)	$O(d(\lvert\mathcal{G}_{KG}\rvert + \lvert\mathcal{R}\rvert \times \lvert\mathcal{U}\rvert + \lvert\mathcal{I}\rvert))$	0.0425	0.0575
KGNNLS(2019)	$O(d(\lvert\mathcal{G}_{KG}\rvert + \lvert\mathcal{V}\rvert + \lvert\mathcal{U}\rvert \times \lvert\mathcal{R}\rvert))$	0.0378	0.0542
KGIN(2021)	$O(d\lvert\mathcal{G}_{KG}\rvert + d^2\lvert\mathcal{G}_{UI}\rvert)$	0.0289	0.0557
KGRec(2023)	$O(d^2\lvert\mathcal{G}_{KG}\rvert + d\lvert\mathcal{G}_{UI}\rvert)$	0.0291	0.0398
DiffKG(2024)	$O(d^2\lvert\mathcal{G}_{KG}\rvert + d\lvert\mathcal{G}_{UI}\rvert)$	0.0177	0.0357
CL-SDKG(2024)	$O(d^2(\lvert\mathcal{G}_{KG}\rvert + \lvert\mathcal{U}\rvert) + d\lvert\mathcal{G}_{UI}\rvert)$	0.0204	0.0445
KGAT(2019)	$O(d^2(\lvert\mathcal{G}_{KG}\rvert + \lvert\mathcal{G}_{UI}\rvert + \lvert\mathcal{U}\rvert + \lvert\mathcal{I}\rvert + \lvert\mathcal{V}\rvert))$	0.0202	0.0311
MCCLK(2022)	$O(d^3\lvert\mathcal{G}_{KG}\rvert + d\lvert\mathcal{G}_{UI}\rvert)$	0.0149	0.0475
Correlation		-0.9374	-0.6682

3.2 Impact of Model Complexity

Due to the significant differences in frameworks among various types of KGRSs, comparing their complexities is challenging. Therefore, we focus on GNN-based KGRSs, which is the most widely adopted framework [10]. We analyze the time complexity of a single GNN layer, as it serves as a fundamental building block for most GNN-based KGRSs. Let d denote the embedding size, while $|\mathcal{G}_{UI}|$ and $|\mathcal{G}_{KG}|$ represent the number of triplets in the user-item interaction graph and KG, respectively.

Tab. 3 ranks KGRSs by their time complexity and reports their Recall@10 on Last.FM (sampling ratio: 10%) and ML-1M (sampling ratio: 5%). The correlation shows the Pearson correlation coefficient between complexity ranking and Recall@10, with negative values indicating an inverse relationship. The results reveal the following observations. **(1) Impact of Complexity in Sparse Scenarios**: Models with higher complexity generally perform worse in sparse scenarios. This is supported by the strongly negative Pearson correlation between model complexity and Recall in sparse scenarios (e.g., -0.9374 on Last.FM and -0.6682 on ML-1M). **(2) Effectiveness of SSL**: SSL helps mitigate the issue of insufficient training data in sparse scenarios by extracting more supervisory signals. This can be supported by the fact that although KGRec is more complex than KGIN, its SSL training mechanism leads to superior performance on Last.FM compared to KGIN.

Table 4: The Recall@10 after removing the attention mechanism on Last.FM. "Average" and "Average$_{a-}$" represent the average Recall of the six SOTA KGRSs with and without attention mechanisms, respectively, while "Improve" indicates the latter's improvement over the former.

Model	80%	40%	20%	10%
KGAT	0.2583	0.1423	**0.0677**	0.0202
KGAT$_{a-}$	**0.2690**	**0.1466**	0.0667	**0.0228**
KGIN	**0.2727**	0.1711	0.0770	0.0289
KGIN$_{a-}$	0.2718	**0.1718**	**0.0796**	**0.0307**
KGRec	0.2560	**0.1758**	**0.0876**	0.0291
KGRec$_{a-}$	**0.2573**	0.1740	0.0872	0.0291
MCCLK	**0.2699**	0.1758	0.0555	0.0149
MCCLK$_{a-}$	0.2698	**0.1792**	**0.0557**	**0.0152**
DiffKG	0.2520	0.1551	0.0479	0.0177
DiffKG$_{a-}$	**0.2564**	**0.1599**	**0.0488**	**0.0214**
CL-SDKG	0.2409	**0.1591**	0.0431	0.0203
CL-SDKG$_{a-}$	**0.2424**	0.1571	**0.0450**	**0.0204**
Average	0.2583	0.1632	0.0631	0.0219
Average$_{a-}$	0.2611	0.1648	0.0638	0.0233
Improve	**+1.09%**	**+0.97%**	**+1.10%**	**+6.05%**

3.3 Impact of Removing Attention Mechanisms

Tab. 3 shows that complex models may sometimes underperform simpler models in sparse scenarios. To further explore this issue, a natural approach is to examine whether and how making models simpler, e.g., removing their attention mechanisms, would influence recommendation accuracy.[1] Specifically, We implement it by fixing attention weights to 1 in six SOTA KGRSs and conduct the same experiment as in Section 3.1. Notably, this removal is rather simple and can lead to substantial disruption of the model's structure. For example, the original structure of KGAT is as follows:

$$a(h,r,t) = (\boldsymbol{W}_r \boldsymbol{e}_t)^\top \tanh\Big((\boldsymbol{W}_r \boldsymbol{e}_h + \boldsymbol{e}_r)\Big), \tag{3}$$

$$\pi(h,r,t) = \frac{\exp(a(h,r,t))}{\sum_{(h,r',t')\in\mathcal{N}_h}\exp(a(h,r',t'))}, \tag{4}$$

$$\boldsymbol{e}_h^{(l)} = \sum\nolimits_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t)\boldsymbol{e}_t^{(l-1)}, \tag{5}$$

where $\boldsymbol{W}_r \in \mathbb{R}^{d\times d}$ is the transformation matrix of relation r. The removal of attention impairs KGAT's ability to account for the influence of different relations, simplifying its GNN structure as:

$$\boldsymbol{e}_h^{(l)} = \sum\nolimits_{(h,r,t)\in\mathcal{N}_h} \boldsymbol{e}_t^{(l-1)}. \tag{6}$$

The results are presented in Tab. 4. The subscript $_{a-}$ (e.g., KGAT$_{a-}$) denotes the altered model (of KGAT) with the attention mechanism removed. Due to space limitation, we only present the Recall@10 results for Last.FM, as similar conclusions are observed across other datasets in appendix. Surprisingly, despite the simplicity of the removal method, we observed that: **(1)** *in most cases, removing the attention mechanism slightly improves recommendation accuracy*; and **(2)** *as the data sparsity level increases, the improvement becomes more pronounced.* These observations strongly support simpler models are more suited for sparse scenarios.

Intuitively, complex models often require substantial training data to effectively learn their parameters and achieve optimal performance. In sparse scenarios, they often underfit or generalize poorly. Therefore, simpler models are better suited to address the challenges posed by sparse scenarios.

[1]A unified ablation study is challenging due to the structural and functional diversity of these models, and our focus is solely on verifying the influence of simpler models.

4 LightKG

Our previous experimental explorations inspire us to design a simpler yet effective KGRS, LightKG. Fig. 2 displays its workflow. First, it employs a simplified GNN layer, with *scalar-based relation encoding and linear aggregation framework*, for aggregating neighbor information. Second, to shorten the training time for SSL, it employs an efficient contrastive layer that directly minimizes the similarity between nodes of the same type, eliminating the need for generating and comparing subgraphs—a common yet computationally expensive approach adopted by previous SSL methods. Finally, it derives matching scores for the recommendation task.

4.1 Simplified GNN Layer

We propose a simplified GNN layer, which serves to derive node representations in the CKGs. The existing models such as KGRec, MCCLK, and KGAT encode relations as vectors or matrices, which are then incorporated into mechanisms such as attention and multi-graph learning. To simplify the process of GNNs, we encode relations of the same type as scalar pairs, significantly reducing complexity while retaining essential information. For example, consider the triplet (*book1*, book-author, *Jane*). LightKG will encode the relation "book-author" as a pair of learned scalars to denote the importance of authors to books ("*write*") and books to authors ("*written by*"), respectively. These scalars, similar to embeddings, are learned through backpropagation after initialization. Although this may seem simplistic in capturing the semantic information in the CKGs, our subsequent experiments prove its effectiveness (refer to Tab. 7).

Inspired by the success of LightGCN [7], we propose a simple and linear GNN framework tailored for CKGs by leveraging scalar-based relations and excluding non-linear activation function. Specifically, for a node k, we use Equ. 7 to update its embedding $\boldsymbol{e}_k^{(l)}$ in the l-th GNN layer,

$$\boldsymbol{e}_k^{(l)} = \sum\nolimits_{(k,r,t)\in\mathcal{N}_k} \frac{\alpha_{r_{tk}}}{\sqrt{|\mathcal{N}_k|}\sqrt{|\mathcal{N}_t|}} \boldsymbol{e}_t^{(l-1)}, \tag{7}$$

where $\alpha_{r_{tk}}$ denotes scalar-based relation from node t to k. Note that $\alpha_{r_{tk}} \neq \alpha_{r_{kt}}$. For each node of user (u) and item (i) in CKG, their final embeddings, which are used for model prediction, are generated by combining the outputs from each GNN layer:

$$\boldsymbol{e}_u^* = \sum\nolimits_{l=0}^{L} \frac{1}{L+1}\boldsymbol{e}_u^{(l)}, \boldsymbol{e}_i^* = \sum\nolimits_{l=0}^{L} \frac{1}{L+1}\boldsymbol{e}_i^{(l)}. \tag{8}$$

By employing a simple and linear neighbor-information aggregation framework with scalar-based relation encoding, LightKG greatly simplifies the existing GNN-KGRSs, thereby reducing the learning difficulty especially in sparse scenarios.

4.2 Efficient Contrastive Layer

Contrastive learning, as a form of SSL, can alleviate the issue of insufficient supervised training data in sparse scenarios [36]. The recent advances in contrastive learning show that SSL is effective as it enhances the distinction between node embeddings [43]. Such distinction can help alleviate the over-smoothing issue [37], a phenomenon where node embeddings in GNNs become indistinguishable due to excessive aggregation of neighborhood information.

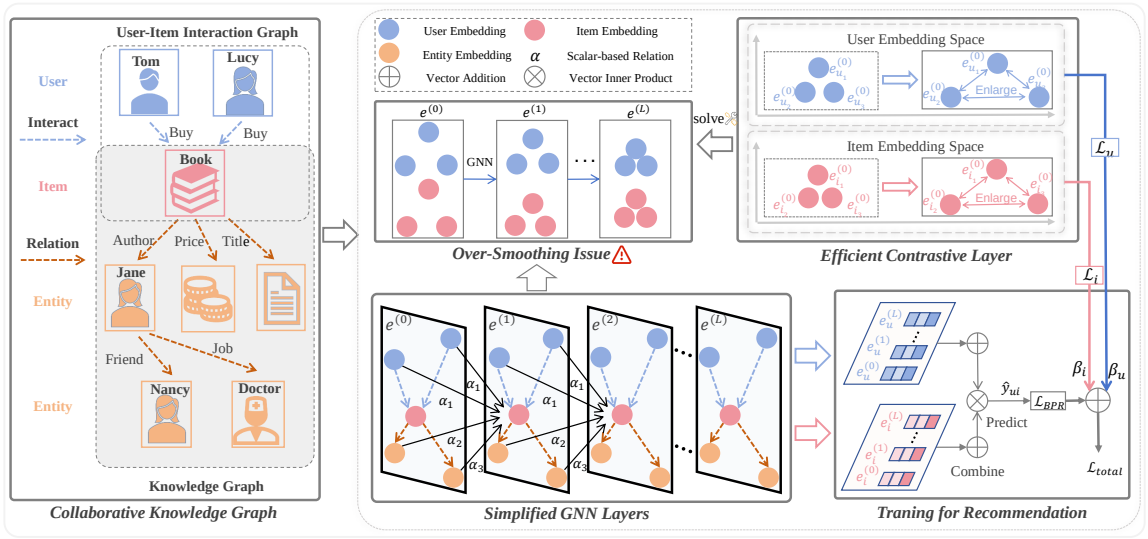


Figure 2: Illustration of our proposed LightKG.

However, existing contrastive learning methods, such as those used in MCCLK and KGRec, are time-consuming since they need to generate subgraphs and make comparisons between them.

To address it, we aim to simplify the contrastive learning with a specific focus on the over-smoothing issue. Inspired by [26], we propose to gain more distinctive node embeddings by directly minimizing similarities between embeddings of different nodes. Empirically, we find this works best when applied separately to nodes of the same type, such as between users or between items, as follows,

$$\min \mathcal{L}_u = \sum_{u_1,u_2 \in \mathcal{U}} \exp(\boldsymbol{e}_{u_1}^{(0)\top} \boldsymbol{e}_{u_2}^{(0)}), \tag{9}$$

$$\min \mathcal{L}_i = \sum_{i_1,i_2 \in \mathcal{I}} \exp(\boldsymbol{e}_{i_1}^{(0)\top} \boldsymbol{e}_{i_2}^{(0)}), \tag{10}$$

where $\boldsymbol{e}_{u_1}^{(0)}, \boldsymbol{e}_{u_2}^{(0)}, \boldsymbol{e}_{i_1}^{(0)}$ and $\boldsymbol{e}_{i_2}^{(0)}$ are normalized user and item embeddings at layer 0. We empirically find that adding the contrastive layer at layer 0 yields the best performance, as it enables the model to learn discriminative features from the outset, making deeper layer representations naturally distinct.

However, Equ. 9 and Equ. 10 ignore two key factors. First, they overlook the inherent similarity that exists between nodes. For two users who have highly overlapping interaction records, their embeddings should not be excessively differentiated. Second, over-smoothing impacts nodes unevenly, as those with more neighbors are more susceptible to losing their unique characteristics in GNNs. Thus, it is crucial to consider the number of neighbors. By incorporating these factors, we refine Equ. 9 and Equ. 10 as follows:

$$\min \mathcal{L}_u = \sum_{u_1,u_2 \in \mathcal{U}} w_{u_1,u_2} \exp((1 - s_{u_1,u_2}) \boldsymbol{e}_{u_1}^{(0)\top} \boldsymbol{e}_{u_2}^{(0)}), \tag{11}$$

$$\min \mathcal{L}_i = \sum_{i_1,i_2 \in \mathcal{I}} w_{i_1,i_2} \exp((1 - s_{i_1,i_2}) \boldsymbol{e}_{i_1}^{(0)\top} \boldsymbol{e}_{i_2}^{(0)}), \tag{12}$$

where $s_{i,j} = \frac{\mathcal{N}_i \cap \mathcal{N}_j}{\sqrt{|\mathcal{N}_i||\mathcal{N}_j|}}$ denotes the similarity, and $w_{i,j} = 1 - \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_j|}}$ reflects the impact of neighbor counts. By optimizing Equ. 11 and Equ. 12, we effectively and efficiently address the over-smoothing issue. Unlike prior works, we avoids the computationally expensive task of generating and comparing subgraphs. Consequently, our method not only enhances recommendation accuracy (see Fig. 3) but also greatly shortens the training time (see Tab. 1).

4.3 Model Prediction and Loss Function

The embeddings obtained from the GNN layer can be used to derive matching scores between users and items using $\hat{y}_{ui} = \boldsymbol{e}_u^{*\top} \boldsymbol{e}_i^*$. For the recommendation task, we choose the BPR loss [16]. It optimizes the personal ranking of items for each user by maximizing the probability that he/she prefers a positively interacted item $\hat{y}_{ui}$ over a randomly chosen non-interacted one $\hat{y}_{uj}$:

$$\mathcal{L}_{\text{BPR}} = \sum_{i \in \mathcal{N}_u, \notin j \mathcal{N}_u} - \ln \sigma \left(\hat{y}_{ui} - \hat{y}_{uj} \right), \tag{13}$$

where σ is the sigmoid function. Together with our objectives for the contrastive layer, we obtain the following objective function:

$$\mathcal{L}_{total} = \mathcal{L}_{\text{BPR}} + \beta_u \mathcal{L}_u + \beta_i \mathcal{L}_i + \lambda \|\Theta\|_2^2, \tag{14}$$

where Θ represents the model parameter set; β_u and β_i are two hyperparameters that determine the respective strengths of $\mathcal{L}_u$ and $\mathcal{L}_i$; and λ is the regularization coefficient.

4.4 Model Analysis

In this section, we analyze the complexity of LightKG, showing that its time complexity is lower than that of all existing GNN-based KGRSs. Therefore, LightKG is a simpler yet effective model, capable of achieving robust learning even in sparse scenarios. We further demonstrate that LightKG can be seen as an extension of LightGCN, which means that LightKG inherits LightGCN's efficient capability to extract information from interaction data. Finally, we discuss the encoding of relations as scalar pairs, which can be interpreted as node labels, enabling the model to distinguish different node types during representation learning.

4.4.1 Complexity Analysis. By encoding relations as scalar pairs and incorporating a linear aggregation framework, LightKG achieves a time complexity of $O(d(|\mathcal{G}_{UI}| + |\mathcal{G}_{KG}|))$, which is significantly lower than that of existing GNN-based KGRSs. This reduced complexity not only improves computational efficiency but also effectively addresses learning challenges in sparse scenarios. The simplicity of LightKG's architecture allows it to learn effectively from limited training data, making it particularly well suited for

sparse scenarios, where traditional GNN-based KGRSs often struggle due to their complexity and high data demands. Moreover, in dense scenarios, LightKG retains the strengths of GNN, leveraging rich interaction data to deliver stronger performance. Thus, LightKG is adaptable and effective for both sparse (see Tab. 2) and dense (see Tab. 6) recommendation scenarios.

4.4.2 Relationship with LightGCN. For a better comparison, here we consider only the interaction graphs, as LightGCN cannot be applied to KG directly. We use $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ to denote the interaction matrix, where $\boldsymbol{A}_{(u,i)} = 1$ means an observed interaction between user u and item i. Based on this, the matrix form of GNN in LightGCN can be denoted as follows:

$$\boldsymbol{E}^{(l)} = \boldsymbol{D}^{-\frac{1}{2}} \begin{pmatrix} 0 & \boldsymbol{A} \\ \boldsymbol{A}^T & 0 \end{pmatrix} \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{E}^{(l-1)}, \tag{15}$$

where $\boldsymbol{D} \in \mathbb{R}^{(|\mathcal{U}|+|\mathcal{I}|)\times(|\mathcal{U}|+|\mathcal{I}|)}$ is a diagonal matrix. For LightKG, we use a pair of scalars α_u and α_i to denote the relation encoding between users and items. Then, the matrix form of GNN in LightKG can be denoted as follows:

$$\boldsymbol{E}^{(l)} = \boldsymbol{D}^{-\frac{1}{2}} \begin{pmatrix} 0 & \alpha_{iu}\boldsymbol{A} \\ \alpha_{ui}\boldsymbol{A}^T & 0 \end{pmatrix} \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{E}^{(l-1)}, \tag{16}$$

$$\begin{aligned} \boldsymbol{E}^{(l)} = {} & \boldsymbol{D}^{-\frac{1}{2}} \begin{pmatrix} 0 & \boldsymbol{A} \\ \boldsymbol{A}^T & 0 \end{pmatrix} \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{E}^{(l-1)} \\ & + \boldsymbol{D}^{-\frac{1}{2}} \begin{pmatrix} 0 & (\alpha_{iu}-1)\boldsymbol{A} \\ (\alpha_{ui}-1)\boldsymbol{A}^T & 0 \end{pmatrix} \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{E}^{(l-1)}. \end{aligned} \tag{17}$$

Since α_{iu} and α_{ui} are learnable, LightKG becomes equivalent to LightGCN when $\alpha_{iu} = \alpha_{ui} = 1$. Therefore, LightKG can be viewed as an extension of LightGCN, inheriting its ability to efficiently extract information from interaction graph. This is particularly important, as many items, especially new ones, may not link to KGs. Our subsequent experiments (Tab. 7) reveal that many KGRSs fail to outperform LightGCN when KGs are removed. In contrast, LightKG achieves superior accuracy both with and without KGs.

4.4.3 Scalar-based Relations as Node Labels. In LightKG, the relations are encoded as scalar pairs such as α_{iu} and α_{ui}, which can be interpreted as implicit labels for different node types. To clarify this concept clearly, we focus on interaction graphs for simplicity and without loss of generality. Considering an extreme scenario where all node embeddings ($\boldsymbol{e}$) are identical, and each node has the same number of neighbors ($|\mathcal{N}|$). In traditional models like LightGCN and KGAT, this uniformity leads to identical aggregation coefficients. As a consequence, all nodes converge to identical embeddings after GNN propagation, thereby losing the distinction between different node types such as users and items. In contrast, LightKG leverages scalar weights to differentiate node types. Taking one layer of the GNN as an example, the updated embeddings of user (u) and item (i) are computed as follows:

$$\boldsymbol{e}_u^{(l)} = \sum_{i\in\mathcal{N}_u} \frac{\alpha_{iu}}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \boldsymbol{e}_i^{(l-1)} = \frac{\alpha_{iu}}{|\mathcal{N}|}\boldsymbol{e}, \tag{18}$$

$$\boldsymbol{e}_i^{(l)} = \sum_{u\in\mathcal{N}_i} \frac{\alpha_{ui}}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \boldsymbol{e}_u^{(l-1)} = \frac{\alpha_{ui}}{|\mathcal{N}|}\boldsymbol{e}. \tag{19}$$

The distinction between α_{iu} and α_{ui} ensures that the user and item embeddings diverge after the propagation of GNN. These scalars act as implicit labels. For example, α_{iu} is the label of user nodes, as it only appears in the expression of $\boldsymbol{e}_u$.

Our subsequent experiments (Fig. 4) reveal that the optimal values of β_u and β_i (refer to Equ. 14) may differ significantly, suggesting that the embeddings of users and items follow distinct distributions. As proven in [2], labeling different node types enable the model to effectively account for the unique characteristics of each node type, thereby improving its capacity to capture patterns specific to users and items during the representation learning process.

Table 5: Statistics of the datasets.

Statistics	User-Item Graph			Knowledge Graph		
	users	items	interactions	entities	relations	triplets
Amazon-book (AMZ-B)	20,347	4,230	234,323	12,230	21	46,522
MovieLens-1M (ML-1M)	6,039	3,499	573,637	77,799	51	378,151
Book-Crossing (BX)	11,018	9,059	24,644	77,904	27	151,500
Last.FM	1,873	3,847	21,173	9,367	62	15,518

5 Experiments

We conduct extensive experiments to demonstrate the effectiveness and efficency of our proposed LightKG. We aim to answer the following four research questions:

- RQ1: How does LightKG perform compared to the SOTA KGRSs, in terms of both the recommendation accuracy and the training efficiency in dense scenarios?
- RQ2: How does LightKG perform in sparse scenarios compared to the SOTA KGRSs?
- RQ3: With its simplified relation encodings and aggregation framework, does LightKG effectively utilize KGs in recommendation compared to the SOTA KGRSs?
- RQ4: How does the contrastive layer contribute to the overall performance of our proposed LightKG?

5.1 Experimental Settings

5.1.1 Datasets and Evaluation Protocol. We select four benchmark datasets: Last.FM[2], Amazon-Book[3] (AMZ-B), Book-Crossing[4] (BX), and MovieLens-1M[5] (ML-1M). Following SOTA methods [13], we preprocess ML-1M and AMZ-B by retaining interactions with ratings of at least 4. For AMZ-B, we further apply a 20-core setting, ensuring that both users and items have a minimum of 20 interactions. For Last.FM and BX, we use the versions released in [22, 25] without additional modifications. Tab. 5 presents the statistical details of the datasets. For evaluation, we use the full-rank approach to generate top-10 recommendations. To evaluate the recommendation accuracy, we adopt widely used metrics, Recall@10 and MRR@10, following [39].

5.1.2 Baselines. We compare our proposed LightKG with 13 SOTA baselines, including KG-free RS (LightGCN [7]), embedding-based KGRSs (CFKG [40], CKE [38]), path-based KGRSs (RippleNet [22], MCRec [8]), supervised GNN-based KGRSs (KGCN [25], KGNN-LS [23], KGAT [27], KGIN [29]) and self-supervised GNN-based KGRSs (MCCLK [43], KGRec [35], DiffKG [9], CL-SDKG [17]).

5.1.3 Implementation Details. All experiments are conducted on Ubuntu 18.04 with an Intel(R) Xeon(R) Gold 6226R CPU running at 2.90GHz, 64GB of memory, and 8 NVIDIA GeForce GTX 3090 GPU. To reduce randomness, all experiments are repeated five times. To ensure the rigor of the experiments, we implement LightKG and

[2]https://grouplens.org/datasets/hetrec-2011/
[3]https://jmcauley.ucsd.edu/data/amazon/
[4]http://www2.informatik.uni-freiburg.de/ cziegler/BX/
[5]https://grouplens.org/datasets/movielens/1m/

Table 6: The results of Recall@10 and MRR@10 of all methods on the four benchmark datasets. * denotes statistically significant different by the paired t-test with $p-\text{value} < 0.01$.

Model	AMZ-B		ML-1M		BX		Last.FM	
	Recall	MRR	Recall	MRR	Recall	MRR	Recall	MRR
LightGCN	0.1980	0.1052	0.1855	0.3453	0.0468	0.0198	0.2695	0.1204
CFKG	0.1968	0.0987	0.1862	0.3405	0.0802	0.0384	0.2444	0.1100
CKE	0.1979	0.1037	0.1848	0.3457	0.0313	0.0152	0.2453	0.1069
RippleNet	0.1561	0.0838	0.1590	0.3062	0.0445	0.0194	0.1633	0.0656
MCRec	0.1524	0.0791	0.1610	0.3233	0.0512	0.0241	0.2132	0.0941
KGCN	0.1550	0.0738	0.1594	0.3456	0.0867	0.0433	0.2149	0.0924
KGNN-LS	0.1508	0.0750	0.1592	0.3051	0.0731	0.0371	0.2117	0.0891
KGAT	0.1925	0.0997	0.1830	0.3412	0.0499	0.0233	0.2583	0.1152
KGIN	0.2090	0.1099	0.1969	0.3551	0.0801	0.0399	0.2727	0.1242
MCCLK	0.2025	0.1065	0.1853	0.3474	0.0607	0.0359	0.2699	0.1228
KGRec	0.2035	0.1094	0.1960	0.3570	0.1033	**0.0540**	0.2560	0.1118
DiffKG	0.2039	0.1116	0.1846	0.3428	0.0581	0.0318	0.2520	0.1192
CL-SDKG	0.2036	0.1134	0.1861	0.3428	0.0924	0.0532	0.2409	0.1054
LightKG	**0.2120***	**0.1173***	**0.2029***	**0.3785***	**0.1154***	0.0515*	**0.2929***	**0.1350***

all the baselines in RecBole [41], which is a unified, comprehensive and efficient recommendation library. We set the "stopping-step" parameter in Recbole to 20, which means that the model will stop training if no improvement is observed on the validation set for 20 consecutive epochs. The datasets are split into training, validation, and testing sets using a ratio of 8:1:1. For a fair comparison, the embedding size of all the models is fixed to 64 and the Adam optimizer is used for optimization with a fixed batch size of 2048. We use Bayesian Optimization [19] for hyperparameter tuning and each model is optimized 30 trails per dataset. All parameters including scalar-based relations are initialized by Xavier uniform. Specifically, we search learning rate in $\{0.01, 0.005, 0.001, 0.0005, 0.0001\}$, the GNN layers in $\{1, 2, 3, 4\}$ for all GNN-based KGRSs and the number of negative samples for training in $\{1, 2, 5, 10\}$. For β_u and β_i (refer to Equ. 14), we tune them in $\{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}\}$.

5.2 Experimental Results and Analysis

5.2.1 Result of RQ1. We evaluate both the recommendation accuracy and efficiency of LightKG in dense scenario (sampling ratio: 80%). The results are presented in Tab. 6, where the bolded numbers represent the best results and the underlined numbers indicate the second-best. We can draw the following conclusions.

First, while LightKG is specifically designed for sparse scenarios, it also demonstrates outstanding performance in dense scenarios. LightKG performs exceptionally well across all benchmark datasets, achieving the highest *Recall/MRR* on Last.FM, AMZ-B and ML-1M, with improvements ranging from 1.42% to 10.5%. For BX, it obtains the highest *Recall* while ranking third in *MRR*, closely following KGRec and CL-SDKG by a narrow margin. These results highlight the strong generalizability of our simplified GNN framework.

Second, some KGRSs, like KGAT, perform worse than LightGCN which does not utilize KGs, highlighting their limitations in effectively using KGs or uncovering valuable insights from interaction graphs. Similar phenomenons are observed in [36]. Note that LightGCN outperforms KGAT on AMZ-B, Last.FM and Ml-1M. It aligns with the slight improvement we observed after removing the attention mechanism from KGAT (as shown in Tab. 4), as KGAT becomes more similar to LightGCN once its attention mechanism is removed.

Third, no single baseline model consistently outperforms the others across the four datasets. Self-supervised methods (e.g., MCCLK and KGRec) do not always surpass supervised methods (e.g., KGIN). This may stem from the limitations of random graph augmentation or overly simplistic, handcrafted cross-view pairing, which may fail to capture meaningful KG information.

Table 7: Recall@10 on KG exploitation experiment. Bold values indicate the best Recall and improvement among all models and the underlined values indicate the second-best.

	Last.FM			BX		
	KG	w/o KG	Improve	KG	w/o KG	Improve
LightGCN	–	0.2695	–	–	0.0468	–
CFKG	0.2444	0.2389	2.29%	0.0802	0.0579	38.51%
KGAT	0.2583	0.2640	-2.17%	0.0499	0.0267	86.89%
KGIN	0.2727	0.2588	5.37%	0.0801	0.0345	132.12%
MCCLK	0.2699	0.2608	3.49%	0.0607	0.0306	98.30%
KGRec	0.2560	0.2571	-0.43%	0.1033	0.0487	112.07%
DiffKG	0.2520	0.2517	0.12%	0.0581	0.0210	**176.67%**
SDKG	0.2327	0.2323	0.17%	0.0924	0.0462	99.87%
LightKG	**0.2929**	**0.2725**	**7.49%**	**0.1154**	**0.0654**	76.48%

Lastly, we compare the training efficiency, i.e., time consumed per training epoch, between LightKG and other GNN-based KGRSs, as shown in Tab. 1. The results indicate that *LightKG achieves a substantially shorter training time than other self- supervised models, highlighting the efficiency of our designed contrastive layer.* Although models like KGCN, KGNN-LS and KGAT achieve better training efficiency than LightKG due to the absence of SSL methods, they fail to obtain accurate recommendation.

5.2.2 Results of RQ2. We now investigate the performance of LightKG across different sparsity levels. Following the experimental setup in Section 4.1, we employ random sampling to generate datasets of varying sizes, representing scenarios with different sparsity levels. The specific sampling ratios used are {80%, 40%, 20%, 10%} on Last.FM and {80%, 40%, 20%, 10%, 5%} on ML-1M, with each value denoting the proportion of interaction records allocated to the training set. The results are presented in Tab. 2 and Fig. 1.

***First**, when interaction records are sparse, LightKG outperforms all other GNN-based KGRSs.* As sparsity level increases, LightKG's advantage over other GNN-based KGRSs becomes more evident, ranging from 8.52% to 102.4% on Last.FM. This supports our hypothesis that overly complex structures for capturing full KG semantics are not always beneficial and can hinder learning in sparse scenarios. ***Second**, despite outperforming other GNN-based KGRSs, LightKG falls behind CFKG, which is an embedding-based KGRS.* We attribute this to the lower complexity of CFKG, which is only $O(d \times batch\_size)$, making it even simpler than LightKG. Compared to CFKG, LightKG exhibits a clear advantage (with an improvement of 20.1% on average) in dense scenarios and falls slightly behind (with a drop of 1.39% on average) in sparse scenarios. Therefore, considering performance across both sparse and dense scenarios, LightKG shows optimal overall effectiveness.

5.2.3 Results of RQ3. To check whether LightKG can capture the semantic information in KG with scalar-based relations, we remove the KG (denoted as w/o KG) and compare the performance before and after the removal, following the setting of [39]. We select several recently released or well-performing SOTA models for comparison.

The results are illustrated in Tab. 7. Notably, the "Improve" here ignores marginal effects of performance improvements. For example, a 50% increase from 60 to 90 is much harder than from 10 to

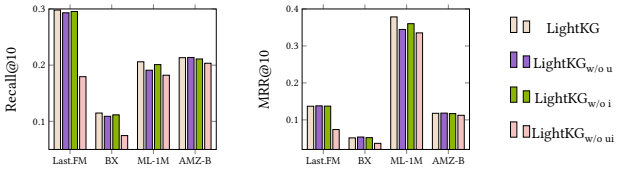


Figure 3: The impacts of contrastive layer.

15. ***(1)*** *With the integration of the KG, LightKG shows a significant improvement in Recall, indicating its effective utilization of the KG to improve recommendation accuracy.* This demonstrates that, despite the seemingly simplistic approach of encoding relations as scalar pairs, it effectively captures the semantic information of the KG while reducing model complexity. ***(2)*** *Many models perform worse than LightGCN when completely deprived of KG, indicating their limited ability to extract meaningful information solely from interaction graphs.* As demonstrated previously, LightKG, as an extension of LightGCN, can effectively extract meaningful information from interaction graphs. Therefore, LightKG maintains high accuracy even with a reduced or absent KG, which demonstrates its robustness and adaptability in various scenarios. ***(3)*** *In certain scenarios, the removal of KGs leads to improved accuracy for some models.* For example, removing KG from KGAT and KGRec improves accuracy on Last.FM. This suggests that some overly complex framework may backfire, failing to effectively leverage KGs. The same phenomenons have also been observed in [39].

5.2.4 Results of RQ4. We aim to evaluate the effectiveness of contrastive layers by conducting ablation experiments with three variants of LightKG.

- **LightKG**$_{w/o\ u}$: This variant eliminates the contrastive layer for users by setting $\beta_u = 0$.
- **LightKG**$_{w/o\ i}$: This variant eliminates the contrastive layer for items by setting $\beta_i = 0$.
- **LightKG**$_{w/o\ ui}$: This variant eliminates contrastive layers for both users and items by setting $\beta_u = \beta_i = 0$.

The results in Fig. 3 lead to the following observations. **(1)** *Comparing LightKG and LightKG$_{w/o\ ui}$, we observe a notable drop in performance for LightKG$_{w/o\ ui}$ after the removal of the contrastive layers.* This highlights the critical role and effectiveness of contrastive layers, emphasizing their importance in enhancing the overall performance of LightKG. **(2)** *A single contrastive layer (either user-side or item-side) can achieve performance comparable to the complete framework.* For example, LightKG$_{w/o\ u}$, LightKG$_{w/o\ i}$ and LightKG achieve similar performance on Last.FM and BX. This phenomenon occurs because the CKG connects users and items through a unified graph structure. When the contrastive layer is applied to one side (e.g., users or items), the impact naturally spreads to the other side.

To further explore the impact of contrastive layer parameters, we fixed the parameter β_u and β_i separately on Last.FM, BX, ML-1M and AMZ-B, while varying the range of the other parameter. The results are presented in Fig. 4. It can be observed that: **(1)** Both β_u and β_i have an optimal value, and deviating from these values—either higher or lower—leads to a degradation of model performance. **(2)** The optimal values of β_u and β_i can differ significantly. For example, on the ML-1M, the optimal β_u is 10^{-5}, while the optimal β_i is 10^{-7}.

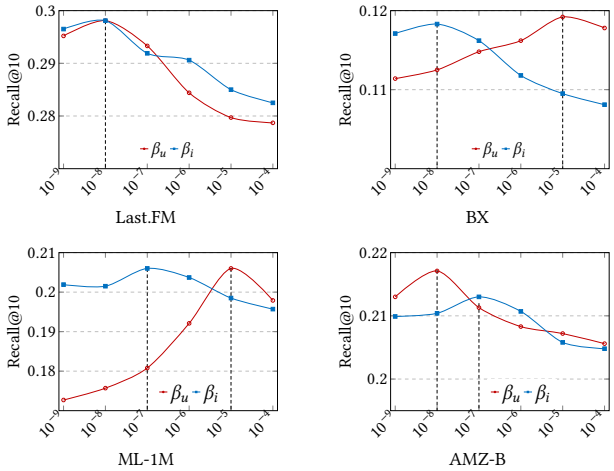


Figure 4: The impacts of β_u and β_i.

This indicates that the distributions of different node types may vary considerably, further validating the rationale behind LightKG's node labeling, as demonstrated earlier.

5.3 Further Exploration of LightKG

5.3.1 Semantic Modeling Capabilities. Since LightKG represents relations as scalar values, its ability to capture semantic information may be limited. To evaluate this issue, we calculate the variance of aggregation coefficients during GNN propagation - a higher variance suggests better discrimination between different relations. For LightKG, these coefficients are computed via Eq. 7, whereas other models derive them from attention weights. As shown in Tab. 8, LightKG consistently achieves the highest variance across all datasets, demonstrating its superior capability to capture global semantic patterns despite its simple scalar-based approach.

Table 8: The variance of aggregation coefficients during GNN.

	Last.FM	ML-1M	BX	AMZ-B
KGAT	0.0670	0.0492	0.2248	0.0368
KGRec	0.1273	0.0672	0.3874	0.0514
CL-SDKG	0.1147	0.0694	0.3991	0.0446
KGIN	0.1204	0.0727	0.3433	0.0371
LightKG	**0.5500**	**0.0967**	**0.4728**	**0.0932**

While LightKG's scalar-based relation encoding offers computational efficiency, it exhibits limited capability to discern fine-grained relational nuances in complex scenarios. For instance, in the ML-1M dataset, the node "Titanic" is linked to 51 other movies via the "film-actor-film" relation. LightKG assigns nearly identical aggregation weights (variance = 0.0009) to these edges, whereas KGAT (variance = 0.0174) demonstrates stronger discriminative power. This limitation is supported by LightKG's suboptimal performance on the MRR metric in the BX dataset. Since MRR relies on the precise ranking position of positive samples, it demands fine-grained discrimination among items with subtle semantic differences—a task inherently more challenging than achieving high Recall.

In summary, LightKG's simple design excels at capturing global semantic information, but this comes at the cost of reduced precision in distinguishing subtle relational details.

Table 9: The scalar values of top-5 relation from each dataset. For clarity and conciseness, the names of relations have been approximately substituted.

	Last.FM	ML-1M	BX	AMZ-B
1	U-I (5.3)	I-U (12.2)	U-I (3.2)	U-I (9.9)
2	I-U (5.2)	U-I (7.6)	I-U (2.7)	I-U (9.4)
3	artist (4.4)	type (6.4)	type (1.2)	char (8.8)
4	born (3.5)	outfit (6.4)	written (0.9)	written (8.8)
5	act (2.6)	act (5.9)	series (0.3)	write (7.7)

5.3.2 The impact of scalar-based relation. In LightKG, scalar values quantitatively measure the relative importance of different relationship types for recommendation tasks. To validate this design, we identify the top-5 most influential relations per dataset based on their scalar weights. The results are shown in Tab. 9, where the values in parentheses represent the learned relation scalars.

It can be observed that user-item and item-user interactions ("U-I/I-U") achieve the highest scalar values across all datasets, indicating that direct user-item interactions provide the most influential recommendation signals. These results demonstrate that direct user-item interactions consistently outweigh other relational signals in recommender systems. This empirical evidence aligns with and strengthens the arguments presented in [39], which critically examines the actual effectiveness of knowledge graph augmentation in recommender systems.

6 Related Work

6.1 Knowledge-aware Recommender Systems

Existing KGRSs can be roughly categorized into three types [9].

Embedding-based KGRSs [5, 11, 24, 38, 40] use the distances and directions in the embedding space to represent relationships between nodes. For example, CKE [38] uses TransR to enhance recommendation accuracy by capturing structured semantic relationships between items in the KGs; CFKG [40] adopts TransE to model user-item interactions, integrating KG to enhance recommendation accuracy and explainability; and KG4Vis [11] adapts TransE to model KG relations specifically for automated visualization recommender systems.

Path-based KGRSs [1, 4, 8, 15] use methods like random walks to explore item connections in KGs. For example, RippleNet [22] uncovers latent user-item associations through "ripple propagation". MCRec [8] significantly enhances both recommendation accuracy and diversity by effectively leveraging meta-path contexts, while PGPR [1] employs reinforcement learning to optimize the quality of reasoning paths.

GNN-based KGRSs are based on the information aggregation mechanism of GNNs and can be divided into **supervised methods** [23, 25, 29] and **self-supervised methods** [9, 17, 34–36, 43]. Regarding supervised methods, KGCN [25] and KGNN-LS [23] utilize GCN to aggregate neighborhood information of items in KGs. Then, KGAT [27] integrates user-item interactions and KGs, using graph attention to improve recommendations. Inspired by the success of self-supervised learning, contrastive learning, as a form of self-supervised learning, has been increasingly integrated into GNN-based methods. For example, MCCLK [43] aligns knowledge and interaction graphs via cross-view contrastive learning. KGCL [36] reduces data noise through graph contrastive learning [9, 17].

6.2 Sparsity Issue in Recommender Systems

Sparse scenarios arise from limited user-item interaction data, leading to challenges in providing accurate recommendations [18]. Failure to address it will lead to user attrition and significant economic lossess. To tackle this issue, various data augmentation approaches have been proposed. Auxiliary information, such as KGs [43] and social networks [33], has been integrated into RSs to enrich the data available for users or items with sparse interactions. Togashi, et al. [21] employ an enhanced negative sampling method to mitigate the popularity bias in sparse scenarios. Moreover, incorporating self-supervised learning techniques into RSs has emerged as a promising trend, which addresses the data sparsity issue by extracting additional supervisory signals from raw data [35, 36, 43]. However, none of these methods have explored what kind of framework is suitable for sparse scenarios.

6.3 Simplifying GNN for Different Tasks

Several studies [6, 12] have shown that not all components of GNNs are universally beneficial. For example, SGC [32], a linear simplification of GCNs, achieves superior computational efficiency and parameter economy while maintaining competitive accuracy. Recent work by Luo et al. [14] establishes that traditional GNNs outperform all subsequent architectural modifications on standard node classification benchmarks. In RSs, LightGCN [7], derived from NGCF [28] through systematic ablation studies, validates that removing nonlinearities and feature transformations can enhance recommendation performance. These findings collectively underscore a critical design principle: streamlined GNN architectures—when carefully tailored to specific application requirements—consistently deliver enhanced efficiency without compromising effectiveness.

7 Conclusion and Future Work

In this work, we empirically reveal the limited or even detrimental effect of complex mechanisms, such as attention mechanism in the existing GNN-based KGRSs. Motivated by this finding, we propose a simple yet powerful GNN-based KGRS, LightKG, which features a simplified GNN structure with scalar-based relation encoding and linear neighbor-information aggregation mechanism, as well as an efficient contrastive layer to address the over-smoothing issue. Extensive experimental results demonstrate LightKG's exceptional performance, achieving superior recommendation accuracy while significantly improving training efficiency. Our approach offers valuable insights for the design of lightweight and effective GNN-based KGRS. For future work, we will further investigate the role of each module of LightKG to increase its interpretability. Also, considering that the emergence of large language models [20, 30] is generating new types of auxiliary data and also new types of RSs, we plan to adapt LightKG accordingly.

8 Acknowledgment

This research is supported by the State Key Laboratory of Industrial Control Technology, China (Grant No. ICT2024C01), and partially supported by the Fundamental Research Funds for the Central Universities (2025ZFJH02) and the Ministry of Education, Singapore, under its MOE AcRF Tier 1 SUTD Kickstarter Initiative (SKI 2021_06_12).

References

[1] Giacomo Balloccu, Ludovico Boratto, Gianni Fenu, and Mirko Marras. 2023. Reinforcement recommendation reasoning through knowledge graphs for explanation path quality. *Knowledge-Based Systems* 260 (2023), 110098.

[2] Yuanchen Bei, Weizhi Chen, Hao Chen, Sheng Zhou, Carl Yang, Jiapei Fan, Longtao Huang, and Jiajun Bu. 2024. Correlation-Aware Graph Convolutional Networks for Multi-Label Node Classification. *arXiv preprint arXiv:2411.17350* (2024).

[3] Xiaocong Chen, Siyu Wang, Julian McAuley, Dietmar Jannach, and Lina Yao. 2024. On the opportunities and challenges of offline reinforcement learning for recommender systems. *ACM Transactions on Information Systems* 42, 6 (2024), 1–26.

[4] Zhendong Chu, Hongning Wang, Yun Xiao, Bo Long, and Lingfei Wu. 2023. Meta policy learning for cold-start conversational recommendation. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining*. 222–230.

[5] Saman Forouzandeh, Kamal Berahmand, Razieh Sheikhpour, and Yuefeng Li. 2023. A new method for recommendation based on embedding spectral clustering in heterogeneous networks (RESCHet). *Expert Systems with Applications* 231 (2023), 120699.

[6] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).

[7] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. 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*. 639–648.

[8] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1531–1540.

[9] Yangqin Jiang, Yuhao Yang, Lianghao Xia, and Chao Huang. 2024. Diffkg: Knowledge graph diffusion model for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 313–321.

[10] Dongze Li, Hanbing Qu, and Jiaqiang Wang. 2023. A survey on knowledge graph-based recommender systems. In *2023 China Automation Congress (CAC)*. IEEE, 2925–2930.

[11] Haotian Li, Yong Wang, Songheng Zhang, Yangqiu Song, and Huamin Qu. 2021. KG4Vis: A knowledge graph-based approach for visualization recommendation. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2021), 195–205.

[12] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan. 2019. Label efficient semi-supervised learning via graph filtering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9582–9591.

[13] Xinhang Li, Zhaopeng Qiu, Xiangyu Zhao, Zihao Wang, Yong Zhang, Chunxiao Xing, and Xian Wu. 2022. Gromov-wasserstein guided representation learning for cross-domain recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1199–1208.

[14] Yuankai Luo, Lei Shi, and Xiao-Ming Wu. 2024. Classic GNNs are Strong Baselines: Reassessing GNNs for Node Classification. *Advances in Neural Information Processing Systems* (2024).

[15] Sung-Jun Park, Dong-Kyu Chae, Hong-Kyun Bae, Sumin Park, and Sang-Wook Kim. 2022. Reinforcement learning over sentiment-augmented knowledge graphs towards accurate and explainable recommendation. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*. 784–793.

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *Uncertainty in Artificial Intelligence* (2012).

[17] Lei Shi, Jiapeng Yang, Pengtao Lv, Lu Yuan, Feifei Kou, Jia Luo, and Mingying Xu. 2024. Self-derived Knowledge Graph Contrastive Learning for Recommendation. In *Proceedings of the 32nd ACM International Conference on Multimedia*. 7571–7580.

[18] Wentao Shi, Xiangnan He, Yang Zhang, Chongming Gao, Xinyue Li, Jizhi Zhang, Qifan Wang, and Fuli Feng. 2024. Enhancing Long-Term Recommendation with Bi-level Learnable Large Language Model Planning. *arXiv preprint arXiv:2403.00843* (2024).

[19] Zhu Sun, Hui Fang, Jie Yang, Xinghua Qu, Hongyang Liu, Di Yu, Yew-Soon Ong, and Jie Zhang. 2022. Daisyrec 2.0: Benchmarking recommendation for rigorous evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 7 (2022), 8206–8226.

[20] Zhu Sun, Hongyang Liu, Xinghua Qu, Kaidong Feng, Yan Wang, and Yew Soon Ong. 2024. Large language models for intent-driven session recommendations. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 324–334.

[21] Riku Togashi, Mayu Otani, and Shin'ichi Satoh. 2021. Alleviating cold-start problems in recommendation through pseudo-labelling over knowledge graph. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 931–939.

[22] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 417–426.

[23] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 968–977.

[24] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. In *The World Wide Web Conference*. 2000–2010.

[25] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *The World Wide Web Conference*. 3307–3313.

[26] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*. PMLR, 9929–9939.

[27] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 950–958.

[28] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Vonference on Research and Development in Information Retrieval*. 165–174.

[29] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the Web Conference 2021*. 878–887.

[30] Ziyan Wang, Yingpeng Du, Zhu Sun, Haoyan Chua, Kaidong Feng, Wenya Wang, and Jie Zhang. 2025. Re2llm: Reflective reinforcement large language model for session-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 12827–12835.

[31] Minwei Wen, Hongyan Mei, Wei Wang, Xiaorong Xue, and Xing Zhang. 2024. Multi-task recommendation based on dynamic knowledge graph. *Applied Intelligence* (2024), 1–19.

[32] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International Conference on Machine Learning*. PMLR, 6861–6871.

[33] Le Wu, Junwei Li, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2020. Diffnet++: A neural influence and interest diffusion network for social recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 10 (2020), 4753–4766.

[34] Zhizhong Wu. 2024. An efficient recommendation model based on knowledge graph attention-assisted network (kgatax). *arXiv preprint arXiv:2409.15315* (2024).

[35] Yuhao Yang, Chao Huang, Lianghao Xia, and Chunzhen Huang. 2023. Knowledge graph self-supervised rationalization for recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3046–3056.

[36] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge graph contrastive learning for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1434–1443.

[37] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems* 33 (2020), 5812–5823.

[38] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 353–362.

[39] Haonan Zhang, Dongxia Wang, Zhu Sun, Yanhui Li, Youcheng Sun, Huizhi Liang, and Wenhai Wang. 2024. Does Knowledge Graph Really Matter for Recommender Systems? *ACM Transactions on Information Systems* (2024).

[40] Y Zhang, Q Ai, X Chen, and P Wang. 2018. Learning over knowledge-base embeddings for recommendation. *arXiv preprint arXiv:1803.06540* (2018).

[41] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, et al. 2022. RecBole 2.0: towards a more up-to-date recommendation library. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4722–4726.

[42] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.

[43] Ding Zou, Wei Wei, Xian-Ling Mao, Ziyang Wang, Minghui Qiu, Feida Zhu, and Xin Cao. 2022. Multi-level cross-view contrastive learning for knowledge-aware recommender system. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1358–1368.

A APPENDIX

A.1 The Complete Results of Fig. 1

In Section 3.1, we examine the performance of 12 SOTA KGRSs across different sparsity levels on Last.FM, ML-1M and AMZ-B.

Table 10: The Recall@10 of different models across different sparsity scenarios on Last.FM.

Model	80%	40%	20%	10%
CFKG	0.2444	0.1797	0.1082	0.0983
CKE	0.2453	0.1522	0.0544	0.0243
RippleNet	0.1633	0.1192	0.0774	0.0530
MCRec	0.2132	0.1243	0.0935	0.0825
KGCN	0.2149	0.1259	0.0677	0.0425
KGNNLS	0.2117	0.1281	0.0631	0.0378
KGAT	0.2583	0.1410	0.0664	0.0202
KGIN	0.2727	0.1711	0.0770	0.0289
MCCLK	0.2699	0.1759	0.0555	0.0149
KGRec	0.2560	0.1758	0.0876	0.0291
Diffkg	0.2520	0.1551	0.0479	0.0177
CL-SDKG	0.2409	0.1802	0.0621	0.0204
path_{max}	0.2132	0.1243	0.0935	0.0825
embedding_{max}	0.2453	0.1797	**0.1082**	**0.0983**
GNN_{max}	**0.2727**	**0.1802**	0.0876	0.0425
LightKG	0.2929	0.21202	0.1012	0.0861
Improve	7.41%	17.66%	15.49%	102.40%

Table 11: The MRR@10 of different models across different sparsity scenarios on Last.FM.

Model	80%	40%	20%	10%
CFKG	0.1100	0.0717	0.0329	0.0304
CKE	0.1069	0.0590	0.0200	0.0094
RippleNet	0.0656	0.0416	0.0259	0.0196
MCRec	0.0941	0.0433	0.0296	0.0231
KGCN	0.0924	0.0474	0.0232	0.0145
KGNNLS	0.0891	0.0460	0.0216	0.0110
KGAT	0.1152	0.0529	0.0227	0.0082
KGIN	0.1242	0.0667	0.0294	0.0130
MCCLK	0.1228	0.0676	0.0204	0.0058
KGRec	0.1118	0.0673	0.0310	0.0125
Diffkg	0.1192	0.0620	0.0195	0.0059
CL-SDKG	0.1054	0.0571	0.0155	0.0079
path-based_{max}	0.0941	0.0433	0.0296	0.0231
embedding_{max}	0.1100	**0.0717**	**0.0329**	**0.0304**
GNN_{max}	**0.1242**	0.0676	0.0310	0.0145
LightKG	0.1350	0.0819	0.0396	0.0267
Improve	8.61%	21.15%	27.74%	84.14%

Table 12: The Recall@10 of different models across different sparsity scenarios on ML-1M.

Model	80%	40%	20%	10%	5%
CFKG	0.1862	0.1272	0.0925	0.0615	0.0603
CKE	0.1848	0.1136	0.0760	0.0604	0.0473
RippleNet	0.1590	0.1031	0.0694	0.0619	0.0575
MCRec	0.1610	0.1051	0.0690	0.0603	0.0550
KGCN	0.1594	0.1014	0.0738	0.0604	0.0575
KGNNLS	0.1592	0.0945	0.0711	0.0601	0.0542
KGAT	0.1830	0.1124	0.0833	0.0585	0.0311
KGIN	0.1969	0.1254	0.0802	0.0612	0.0557
MCCLK	0.1853	0.1198	0.0841	0.0523	0.0475
KGRec	0.1960	0.1290	0.0934	0.0572	0.0399
DiffKG	0.1846	0.0846	0.0615	0.0512	0.0357
CL-SDKG	0.1861	0.1122	0.0742	0.0602	0.0445
path_{max}	0.1610	0.1051	0.0694	**0.0619**	0.0575
embedding_{max}	0.1862	0.1272	0.0925	0.0615	**0.0603**
GNN_{max}	**0.1969**	**0.1290**	**0.0934**	0.0612	0.0575
LightKG	0.2015	0.1284	0.0996	0.0699	0.0587
Improve	2.33%	-0.47%	6.65%	14.22%	2.16%

Table 13: The MRR@10 of different models across different sparsity scenarios on ML-1M.

Model	80%	40%	20%	10%	5%
CFKG	0.3405	0.2025	0.1487	0.1077	0.1047
CKE	0.3457	0.1878	0.1354	0.1092	0.0904
RippleNet	0.3062	0.1729	0.1250	0.1103	0.1021
MCRec	0.3233	0.1757	0.1245	0.1099	0.1032
KGCN	0.3456	0.1734	0.1307	0.1084	0.1029
KGNNLS	0.3051	0.1688	0.1279	0.107	0.0989
KGAT	0.3412	0.1870	0.1408	0.1101	0.0647
KGIN	0.3551	0.1976	0.1349	0.1157	0.1004
MCCLK	0.3474	0.1900	0.1431	0.1063	0.0815
KGRec	0.3570	0.2029	0.1544	0.1086	0.0834
DiffKG	0.3428	0.1594	0.1202	0.0492	0.0378
CL-SDKG	0.3428	0.1858	0.1329	0.1052	0.0857
path_{max}	0.3233	0.1757	0.125	0.1103	0.1032
embedding_{max}	0.3457	0.2025	0.1487	0.1077	**0.1047**
GNN_{max}	**0.3570**	**0.2029**	**0.1544**	**0.1157**	0.1029
LightKG	0.3785	0.2032	0.1491	0.1237	0.1032
Improve	6.02%	0.14%	-3.43%	5.10%	0.09%

Table 14: The Recall@10 of different models across different sparsity scenarios on AMZ-B.

Model	80%	40%	20%	10%
CFKG	0.1968	0.1357	0.1098	0.0840
CKE	0.1979	0.1177	0.0631	0.0237
RippleNet	0.1561	0.1006	0.0753	0.0446
MCRec	0.1524	0.0983	0.0723	0.0461
KGCN	0.1550	0.1058	0.0757	0.0551
KGNNLS	0.1508	0.0948	0.0728	0.0525
KGAT	0.1925	0.1356	0.0679	0.0351
KGIN	0.2090	0.1455	0.0911	0.0765
MCCLK	0.2025	0.1468	0.1005	0.0671
KGRec	0.2035	0.1448	0.0986	0.0771
DiffKG	0.2039	0.1355	0.0843	0.0439
CL-SDKG	0.2036	0.1367	0.0947	0.0685
path_{max}	0.1561	0.1006	0.0753	0.0461
embedding_{max}	0.1979	0.1357	**0.1098**	**0.0840**
GNN_{max}	**0.2090**	**0.1468**	0.1005	0.0771
LightKG	0.2120	0.1582	0.1148	0.0931
Improve	1.44%	7.77%	14.23%	20.75%

Table 15: The MRR@10 of different models across different sparsity scenarios on AMZ-B.

Model	80%	40%	20%	10%
CFKG	0.0987	0.0632	0.0515	0.0422
CKE	0.1037	0.0575	0.0266	0.0146
RippleNet	0.0838	0.0484	0.0422	0.0257
MCRec	0.0791	0.0472	0.0415	0.0274
KGCN	0.0738	0.0501	0.0418	0.0309
KGNNLS	0.0750	0.0482	0.0396	0.0247
KGAT	0.0997	0.0652	0.0283	0.0169
KGIN	0.1099	0.0722	0.0452	0.0407
MCCLK	0.1065	0.0715	0.5050	0.0389
KGRec	0.1094	0.0707	0.0483	0.0415
DiffKG	0.1116	0.0622	0.0457	0.0286
CL-SDKG	0.1134	0.0635	0.0473	0.0404
path_{max}	0.0838	0.0484	0.0422	0.0274
embedding_{max}	0.1037	0.0632	**0.0515**	**0.042**2
GNN_{max}	**0.1134**	**0.0722**	0.0505	0.0415
LightKG	0.1173	0.0762	0.0577	0.0472
Improve	3.44%	5.54%	14.26%	13.73%

Due to the sparsity of the BX dataset, it is difficult to further reduce the sampling rate, making it infeasible for us to conduct experiments on it.

A.2 The Complete Results of Tab. 3

We show the complete results of Tab. 3. The sampling ratio is set 10% for Last.FM and AMZ-B, 5% for ML-1M. Since the BX dataset is already extremely sparse, 80% sampling ratio is sufficient. **The Pearson correlation coefficients on four datasets are all negative**.

Table 16: The results of complexity analysis.

Model	Time Complexity (ranked from low to high)	Last.FM	ML-1M	Amz-B	BX
KGCN	$O(d(\lvert\mathcal{G}_{KG}\rvert + \lvert\mathcal{R}\rvert \times \lvert\mathcal{U}\rvert + \lvert\mathcal{I}\rvert))$	0.0425	0.0575	0.0551	0.0867
KGNNLS	$O(d(\lvert\mathcal{G}_{KG}\rvert + \lvert\mathcal{V}\rvert + \lvert\mathcal{U}\rvert \times \lvert\mathcal{R}\rvert))$	0.0378	0.0542	0.0525	0.0731
KGIN	$O(d\lvert\mathcal{G}_{KG}\rvert + d^2\lvert\mathcal{G}_{UI}\rvert)$	0.0289	0.0557	0.0765	0.0801
KGRec	$O(d^2\lvert\mathcal{G}_{KG}\rvert + d\lvert\mathcal{G}_{UI}\rvert)$	0.0291	0.0398	0.0771	0.1033
DiffKG	$O(d^2\lvert\mathcal{G}_{KG}\rvert + d\lvert\mathcal{G}_{UI}\rvert)$	0.0177	0.0357	0.0439	0.0581
CL-SDKG	$O(d^2(\lvert\mathcal{G}_{KG}\rvert + \lvert\mathcal{U}\rvert) + d\lvert\mathcal{G}_{UI}\rvert)$	0.0204	0.0445	0.0685	0.0924
KGAT	$O(d^2(\lvert\mathcal{G}_{KG}\rvert + \lvert\mathcal{G}_{UI}\rvert + \lvert\mathcal{U}\rvert + \lvert\mathcal{I}\rvert + \lvert\mathcal{V}\rvert))$	0.0202	0.0311	0.0351	0.0499
MCCLK	$O(d^3\lvert\mathcal{G}_{KG}\rvert + d\lvert\mathcal{G}_{UI}\rvert)$	0.0149	0.0475	0.0671	0.0607
Correlation		-0.9374	-0.6682	-0.1145	-0.4836

A.3 The Complete Results of Tab. 4

Last, we present the rest of experimental results with the attention mechanism removed. Noticing that removing the attention mechanism leads to a slight decline in performance on the AMZ-B, we attribute this to the method we used to remove the attention mechanism being too crude, which damaged the model's structure. This slight decline does not affect our conclusion: the attention mechanism is useless even harmful in GNN-based KGRSs.

Table 17: The MRR@10 after removing the attention mechanism on Last.FM.

model	80%	40%	20%	10%
KGAT	0.1152	**0.0529**	0.0227	0.0082
KGAT$_{a-}$	**0.1172**	0.0475	**0.0264**	**0.0096**
KGIN	**0.1243**	**0.0667**	0.0294	0.0130
KGIN$_{a-}$	0.1241	0.0654	**0.0301**	**0.0138**
MCCLK	**0.1228**	**0.0676**	0.0204	0.0058
MCCLK$_{a-}$	0.1215	0.0668	**0.0209**	**0.0059**
KGRec	0.1117	**0.0673**	0.0310	0.0125
KGRec$_{a-}$	**0.1140**	0.0669	**0.0311**	0.0125
DiffKG	**0.1192**	0.0620	0.0195	0.0059
DiffKG$_{a-}$	0.1177	**0.0627**	0.0195	**0.0067**
CL-SDKG	0.1054	**0.0571**	0.0155	0.0079
CL-SDKG$_{a-}$	**0.1059**	0.0562	**0.0157**	0.0079
Average	0.1164	0.0623	0.0231	0.0089
Average$_{a-}$	0.1167	0.0609	0.0240	0.0094
Improve	0.26%	-2.17%	3.75%	5.82%

Table 18: The Recall@10 after removing the attention mechanism on ML-1M.

Model	80%	40%	20%	10%	5%
KGAT	0.1830	0.1124	**0.0833**	0.0585	0.0311
KGAT$_{a-}$	**0.1837**	**0.1152**	0.0795	**0.0611**	**0.0317**
KGIN	0.1969	**0.1254**	**0.0802**	**0.0612**	**0.0557**
KGIN$_{a-}$	**0.1971**	0.1250	0.0799	0.0607	0.0554
MCCLK	0.1853	**0.1198**	0.0841	**0.0723**	0.0475
MCCLK$_{a-}$	**0.1860**	0.1193	0.0841	0.0693	**0.0478**
KGRec	0.1960	0.1290	0.0934	0.0572	0.0399
KGRec$_{a-}$	**0.1966**	**0.1293**	**0.0942**	**0.0600**	0.0399
DiffKG	0.1846	0.0846	**0.0615**	0.0512	0.0357
DiffKG$_{a-}$	**0.1892**	**0.0877**	0.0611	**0.0557**	**0.0391**
CL-SDKG	0.1861	0.1122	0.0742	0.0602	0.0445
CL-SDKG$_{a-}$	**0.1881**	**0.1134**	**0.0795**	**0.0611**	**0.0473**
Average	0.1887	0.1139	0.0795	0.0601	0.0424
Average$_{a-}$	0.1901	0.1150	0.0797	0.0613	0.0435
Improve	0.76%	0.95%	0.33%	2.02%	2.70%

Table 19: The MRR@10 after removing the attention mechanism on ML-1M.

model	80%	40%	20%	10%	5%
KGAT	0.3412	0.187	**0.1408**	0.1101	**0.0647**
KGAT$_{a-}$	**0.3413**	**0.1885**	0.1349	0.1101	0.0638
KGIN	0.3551	**0.1976**	0.1349	**0.1157**	**0.1004**
KGIN$_{a-}$	**0.3596**	0.1950	**0.1362**	0.1152	0.0996
MCCLK	0.3474	**0.1900**	0.1431	0.1063	0.0815
MCCLK$_{a-}$	**0.3475**	0.1898	**0.1434**	**0.1081**	**0.0823**
KGRec	**0.3570**	0.2029	0.1544	0.1086	0.0834
KGRec$_{a-}$	0.3567	**0.2070**	**0.1565**	**0.1124**	0.0834
DiffKG	0.3428	0.1594	0.1202	0.0492	0.0378
DiffKG$_{a-}$	**0.3441**	**0.1616**	**0.1215**	**0.0509**	**0.0385**
CL-SDKG	0.3428	0.1858	0.1329	0.1052	0.0857
CL-SDKG$_{a-}$	**0.3444**	**0.1872**	**0.1338**	**0.1067**	**0.0877**
Average	0.3477	0.1871	0.1377	0.0992	0.0756
Average$_{a-}$	0.3489	0.1882	0.1377	0.1006	0.0759
Improve	0.35%	0.57%	0.00%	1.39%	0.40%

Table 20: The Recall@10 after removing the attention mechanism on AMZ-B.

Model	80%	40%	20%	10%
KGAT	**0.1925**	**0.1356**	**0.0679**	**0.0351**
KGAT$_{a-}$	0.1872	0.1342	0.0666	0.0316
KGIN	**0.2090**	0.1455	**0.0911**	**0.0765**
KGIN$_{a-}$	0.2081	**0.1466**	0.0901	0.0738
MCCLK	**0.2025**	0.1468	0.1005	0.0671
MCCLK$_{a-}$	0.2016	**0.1477**	**0.1042**	0.0671
KGRec	0.2035	0.1448	**0.0986**	0.0771
KGRec$_{a-}$	**0.2041**	**0.1476**	0.0976	**0.0773**
DiffKG	0.2039	0.1355	0.0843	0.0439
DiffKG$_{a-}$	**0.2098**	**0.1394**	**0.0875**	**0.0477**
CL-SDKG	0.2036	0.1367	0.0947	0.0685
CL-SDKG$_{a-}$	**0.2050**	**0.1379**	**0.0964**	**0.0697**
Average	0.2025	0.1408	0.0895	0.0614
Average$_{a-}$	0.2026	0.1422	0.0904	0.0612
Improve	0.07%	1.01%	0.99%	-0.28%

Table 21: The MRR@10 after removing the attention mechanism on AMZ-B.

model	80%	40%	20%	10%
KGAT	**0.0997**	**0.0652**	**0.0283**	**0.0169**
KGAT$_{a-}$	0.0956	0.0644	0.0275	0.0143
KGIN	**0.1099**	0.0722	0.0452	**0.0407**
KGIN$_{a-}$	0.1097	**0.0731**	0.0452	0.0391
MCCLK	0.1065	**0.0715**	0.0505	0.0389
MCCLK$_{a-}$	**0.1067**	0.0711	**0.0508**	0.0389
KGRec	**0.1094**	**0.0707**	**0.0483**	**0.0415**
KGRec$_{a-}$	0.1083	0.0699	0.0477	0.0406
DiffKG	0.1116	0.0622	0.0457	0.0286
DiffKG$_{a-}$	**0.1159**	**0.0627**	**0.0462**	**0.0291**
CL-SDKG	0.1134	**0.0635**	0.0473	0.0404
CL-SDKG$_{a-}$	**0.1138**	0.0631	**0.0479**	**0.0408**
Average	0.1084	0.0676	0.0442	0.0345
Average$_{a-}$	0.1083	0.0674	0.0442	0.0338
Improve	-0.08%	-0.25%	0.00%	-2.03%