R-GRec: Relation-Guided Generative Recommendation via Collaborative Graph Supervision

Xiangfu Meng[1,†] **Jiapeng Yang**[2,†] **Lei Shi**[3,*] **Xue Wang**[4] **Yuefeng Zhan**[2] **Chen Gu**[2]
Hao Sun[2] **Weiwei Deng**[2] **Tao Yao**[1,*] **Feng Sun**[2] **Qi Zhang**[2]

[1]Shanghai Jiao Tong University, Shanghai, China
[2]Microsoft
[3]State Key Laboratory of Media Convergence and Communication,
Communication University of China, China
[4]Independent Researcher

Abstract

Generative recommendation has recently emerged as a promising paradigm for large-scale recommendation by reformulating next-item prediction as discrete sequence generation. However, current SID-based models rely mainly on history-to-next-item supervision, leaving item–item collaborative topology only indirectly captured through sparse user sequences. To address this gap, we propose R-GRec, a graph-derived structural supervision framework for SID-based generative recommendation. From user interactions, R-GRec builds an item-item graph and derives three complementary auxiliary objectives-Neighbor Prediction, Topological Contrast, and Link Prediction — to make collaborative topology explicit during generative training. Extensive experiments on public Amazon datasets demonstrate that R-GRec achieves state-of-the-art performance over representative traditional, generative, and LLM-based recommenders. Further ablation and analytical studies verify the contribution of each graph-derived objective and show that collaborative topology acts as a complementary supervision signal to Semantic ID generation, improving generative recommendation without adding inference-time graph computation.

1 Introduction

In recent years, generative recommendation has emerged as an important paradigm beyond conventional ranking-based recommendation. Instead of scoring items from a predefined candidate set, generative recommenders retrieve items by generating textual descriptions, semantic identifiers, or discrete token sequences. Among them, semantic ID (SID)-based methods represent each item as a compact sequence of discrete codes, enabling item retrieval through identifier generation within a unified generative framework. However, most SID-based generative recommenders are still trained primarily with user-history-to-next-item supervision. This objective teaches the model to generate the next interacted item, but provides limited explicit guidance on how items should be organized according to collaborative relations.

From the perspective of collaborative filtering, recommendation is not only about predicting the next item, but also about modeling relational structures among users and items. Classical collaborative filtering and graph-based recommendation methods have long exploited item–item similarities, local neighborhoods, and higher-order connectivity, yet these signals are not naturally encoded in standard

*Corresponding authors: Lei Shi (leiky_shi@cuc.edu.cn) and Tao Yao (taoyao@sjtu.edu.cn).
[†]Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

SID-based generative training. This mismatch raises a key question: can collaborative structures be reformulated as generative-compatible supervision signals? To answer this question, we propose **R-GRec**, a relation-guided generative recommendation framework that constructs an item–item graph from user interaction sequences and converts graph relations into three auxiliary SID-generation tasks: **neighbor prediction** for local item neighborhoods, **topological contrast** for relative structural proximity, and **link prediction** for higher-order co-neighborhood closure. These tasks provide SID-native auxiliary supervision without replacing the downstream recommendation objective.

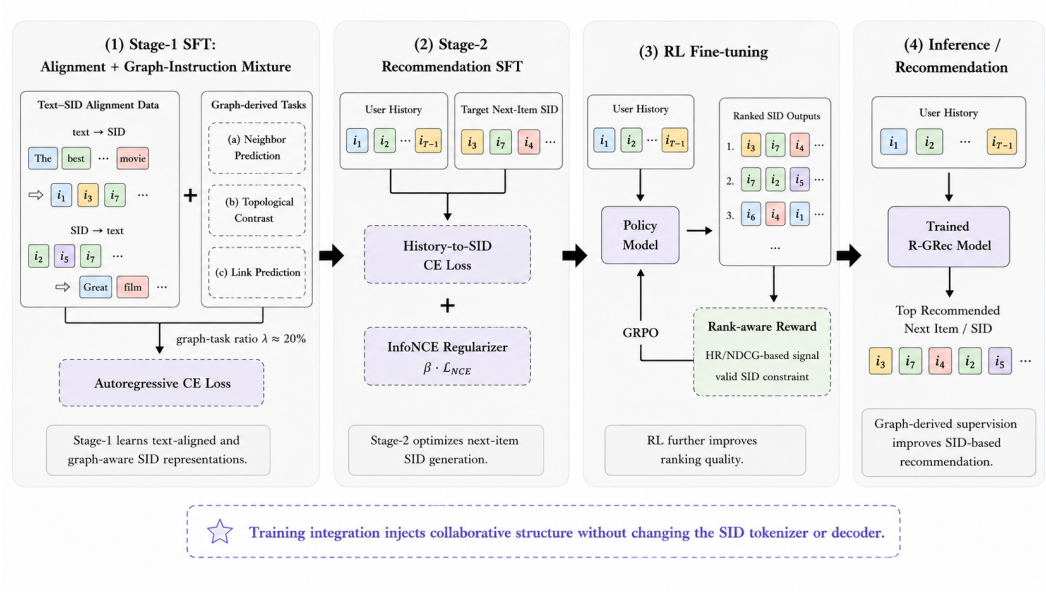


Figure 1: Training pipeline of R-GRec. Graph-derived structural tasks are injected in Stage-1 SFT.

Unlike conventional graph-based recommendation methods, R-GRec does not adopt graph neural networks, graph propagation, or graph embeddings as the recommendation backbone. Instead, it reformulates collaborative structures into supervision forms compatible with generative recommendation, allowing graph-derived relational knowledge to shape SID generation while keeping the SID tokenizer, decoder, and inference procedure unchanged. In this way, R-GRec provides a lightweight and architecture-preserving way to incorporate collaborative structure into SID-based generative recommendation.

Our experiments on two Amazon recommendation benchmarks show that R-GRec improves downstream generative recommendation performance over strong traditional, generative, and LLM-based baselines. Further control experiments and ablations show that the gains cannot be explained solely by additional auxiliary training data, and that different graph-derived tasks and supervision ratios provide complementary effects.

The main contributions of this work are as follows:

- We identify a structural supervision gap in SID-based generative recommendation: standard history-to-next-item learning provides limited explicit constraints for preserving collaborative item relations in the SID-generating space.
- We propose **R-GRec**, a relation-guided framework that converts item–item neighborhoods, relative topological proximity, and higher-order co-neighborhood closure into SID-native auxiliary generation tasks.
- We integrate graph-derived structural supervision into the SID-based generative recommendation pipeline without changing the SID tokenizer, decoder, or inference procedure.
- Through control experiments and task-level ablations, we show that R-GRec improves downstream recommendation performance and that its graph-derived tasks provide complementary supervision beyond merely adding auxiliary training data.

2 Related Work

2.1 Collaborative structure in recommendation

Collaborative structure has long been a central signal in recommender systems. Classical item-based collaborative filtering shows that item neighborhoods provide stable and scalable evidence for recommendation [15, 22]. Sequential and co-occurrence-aware methods further exploit item transition and co-occurrence structures, as exemplified by FPMC [21] and CoFactor [13]. Graph-based recommenders make such structures explicit by propagating collaborative signals over user–item or item–item graphs. PinSage scales graph convolution and random-walk neighborhood aggregation to web-scale recommendation [30], LightGCN highlights simple neighborhood aggregation as a key mechanism for graph collaborative filtering [6], and self-supervised graph methods such as SGL and NCL further leverage graph augmentations, high-order neighbors, and neighborhood-enriched contrastive learning [14, 27]. These works demonstrate the value of item neighborhoods, relative proximity, and higher-order connectivity. However, they typically use collaborative structure through representation learning, message passing, contrastive objectives, or candidate scoring, rather than converting item–item topology into generative supervision over semantic ID outputs.

2.2 Semantic IDs, tokenization, and collaborative fusion

Generative recommendation reformulates item retrieval as sequence generation, making item identifiers a key interface between recommendation targets and generative models. Earlier work such as RecForest [5] shows that representation learning and index construction can be jointly optimized for efficient retrieval. TIGER introduces semantic IDs for generative recommendation by assigning each item a discrete identifier and training a sequence-to-sequence model to generate the target SID from user history [20]. Subsequent studies improve this paradigm through item ID design [10], textual ID learning [23], learnable item tokenization [25], high-order collaborative tokenization [18], end-to-end tokenizer–recommender learning [16], and contextual action tokenization [9]. These methods mainly study how items, users, or actions should be discretized so that semantic or collaborative information can be encoded into the token space.

Recent generative recommenders further inject collaborative information through alignment, architecture design, or semantic-collaborative code learning. LC-Rec aligns language semantics and collaborative semantics through learned item indices and alignment tuning [32]. EAGER uses a two-stream architecture to model behavior tokens and semantic tokens with auxiliary objectives [26]. UNGER and DAS explore unified semantic-collaborative codes, progressive semantic-collaborative knowledge fusion, and dual-aligned semantic IDs for industrial recommendation [28, 29]. These works are closely related to our motivation, but they incorporate collaborative information mainly by changing identifiers, learning tokenizers, adding alignment losses, or modifying model architectures. In contrast, R-GRec keeps the SID tokenizer and decoder unchanged, and uses item–item topology to construct auxiliary generation tasks whose targets are expressed in the same SID output space as downstream recommendation.

2.3 Decoding, scaling, and post-training

Another line of work improves generative recommendation through scaling, decoding, and post-training. HSTU studies large-scale generative recommenders as sequential transducers [31]. BIGRec investigates LLM grounding for recommendation [2], OneRec unifies retrieval and ranking with iterative preference alignment [4], and MiniOneRec provides an open-source SID-based pipeline covering SID construction, supervised fine-tuning, constrained decoding, and recommendation-oriented reinforcement learning [12]. Decoding-oriented methods show that inference strategies also affect recommendation quality: D3 addresses amplification bias and homogeneity through debiasing-diversifying decoding [1], RPG improves long SID generation with parallel prediction and graph-guided decoding [8], and S-DPO studies preference optimization for LLM-based recommendation [3]. These methods are complementary to R-GRec. Unlike graph-guided decoding methods that use graph information to constrain or guide inference-time generation, R-GRec uses the item graph only to construct training-time SID-generation tasks and preserves the standard graph-free decoding interface at inference.

2.4 Position of this work

R-GRec studies a complementary way of injecting collaborative structure into SID-based generative recommendation. Instead of learning new identifiers, modifying tokenizers or architectures, changing decoding strategies, or applying post-training preference optimization, it converts item–item topology into training-time SID-generation tasks. Since both auxiliary and recommendation targets are valid SID sequences, collaborative structure supervises the same auto regressive output space used for downstream recommendation while preserving graph-free inference.

3 Method

3.1 Overview

We build on the standard SID-based generative recommendation paradigm, where each item v is represented by a semantic ID (SID) $\mathbf{s}(v) = [s_1(v), \ldots, s_L(v)]$. Given a user history $\mathbf{x}_u = [v_{u,1}, \ldots, v_{u,t_u-1}]$, the model autoregressively generates the SID of the next item y_u:

$$p_\theta(s(y_u) \mid x_u) = \prod_{\ell=1}^{L} p_\theta(s_\ell(y_u) \mid x_u, s_{<\ell}(y_u)) \,. \tag{1}$$

We keep the original SID-based recommendation model unchanged and add a set of auxiliary SID-generation objectives derived from an item–item graph constructed from the training interactions. Our framework, R-GRec, uses the collaborative item–item relations in this graph to construct three types of SID-level structural supervision: neighbor prediction, topological contrast, and link prediction. All auxiliary targets are expressed as SID sequences, matching the output format of next-item recommendation. The graph is used only to create supervision targets, not as a graph encoder or message-passing backbone. This allows the generative recommender to learn structural regularities of the item graph while remaining fully compatible with the original SID-based training pipeline.

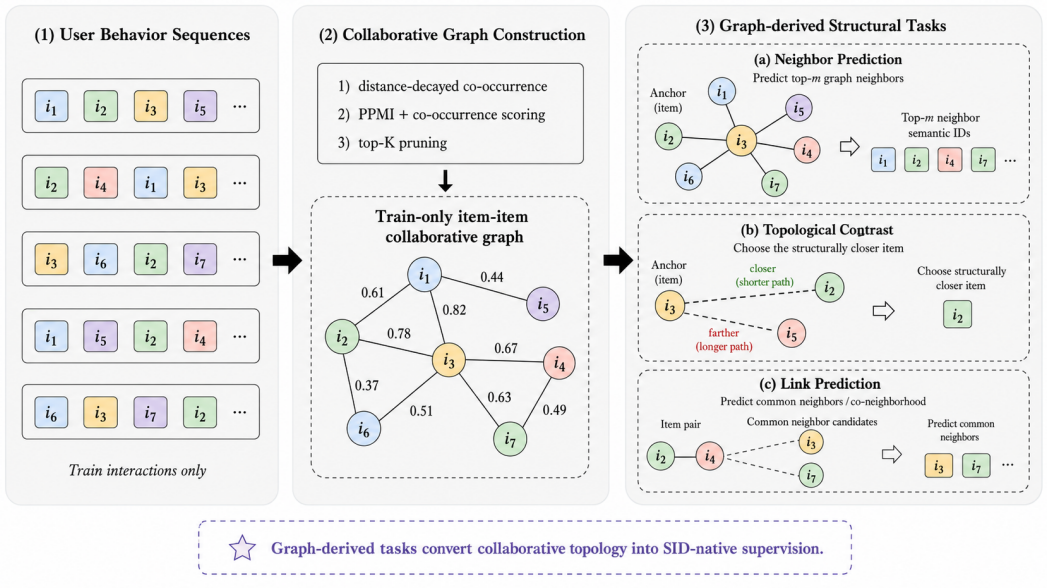


Figure 2: Overview of graph construction and graph-derived structural task creation in R-GRec. Starting from user interaction sequences, we build a train-only item–item collaborative graph and derive three SID-native structural tasks: Neighbor Prediction, Topological Contrast, and Link Prediction.

3.2 Item–Item Graph Construction

We construct a directed weighted item–item graph from training interactions only. For each training example $(\mathbf{x}_u, y_u)$, we form the complete training sequence

$$\tilde{\mathbf{x}}_u = [v_{u,1}, v_{u,2}, \ldots, v_{u,t_u-1}, y_u]. \tag{2}$$

For every ordered pair of positions $i < j$ in $\tilde{\mathbf{x}}_u$, we add a distance-decayed co-occurrence weight:

$$w(v_{u,i} \to v_{u,j}) \mathrel{+}= \exp\big(-\alpha((j-i)-1)\big), \tag{3}$$

where α controls how quickly long-range co-occurrences are down-weighted. This assigns larger weights to nearby item pairs, reflecting that short-range co-occurrences usually provide stronger collaborative evidence.

After aggregating all training sequences, we prune weak edges and retain the top outgoing neighbors for each item according to a relevance score $r(a, b)$ derived from the aggregated co-occurrence statistics. The retained scores are normalized into a transition distribution:

$$P(b \mid a) = \frac{r(a,b)}{\sum_{c \in \mathcal{N}(a)} r(a,c)}, \tag{4}$$

where $\mathcal{N}(a)$ is the retained neighbor set of item a. The graph is used only to construct auxiliary supervision targets. It is not used as a graph encoder, message-passing backbone, or inference-time retrieval module.

3.3 Graph-Derived Structural Tasks

Based on this graph, R-GRec converts three complementary item–item relations into SID-level generation tasks: local neighborhoods, relative structural proximity, and higher-order co-neighborhood closure. All tasks use SIDs in the input prompt and predict SID sequences as targets, so they share the same autoregressive training format as next-item generation.

Neighbor Prediction. Given an anchor item a, the model generates the SIDs of its top-m outgoing neighbors under $P(\cdot \mid a)$:

$$\mathbf{t}_{\text{nei}}(a) = [\mathbf{s}(n_1), \ldots, \mathbf{s}(n_m)], \quad n_i \in \text{Top-}m(\mathcal{N}(a)). \tag{5}$$

This task exposes local co-occurrence neighborhoods.

Topological Contrast. Given an anchor item a and two candidates (b^+, b^-), where b^+ is a high-relevance neighbor and b^- is sampled from structurally farther items, the model generates the SID of b^+. This task teaches relative structural proximity rather than only edge existence.

Link Prediction. Given a retained item pair (a, b), the model generates SIDs of their top common neighbors:

$$\mathcal{C}(a,b) = \mathcal{N}(a) \cap \mathcal{N}(b), \qquad \mathbf{t}_{\text{link}}(a,b) = [\mathbf{s}(c_1), \ldots, \mathbf{s}(c_m)]. \tag{6}$$

This link-prediction-style task captures higher-order co-neighborhood and triadic-closure structure.

3.4 Integration into the MiniOneRec Training Pipeline

R-GRec is implemented as a training-time augmentation to the MiniOneRec pipeline, rather than a replacement of its SID tokenizer, decoder, or downstream recommendation objective. The key design is to inject graph-derived collaborative supervision into the early SID-alignment stage, while keeping the later recommendation training and inference interface unchanged.

In Stage-1 SFT, we train the model on a mixture of the original text–SID alignment corpus and our graph-instruction corpus:

$$\mathcal{D}_{\text{SFT1}}(\lambda) = (1-\lambda)\mathcal{D}_{\text{align}} + \lambda \mathcal{D}_{\text{graph}}, \tag{7}$$

where $\mathcal{D}_{\text{align}}$ contains the original text-to-SID and SID-to-text alignment examples, and $\mathcal{D}_{\text{graph}}$ consists of the three graph-derived structural tasks defined above. The mixing ratio λ controls the

empirical fraction of graph-instruction examples in Stage-1, rather than serving as a separate loss coefficient. All examples are optimized with the same autoregressive cross-entropy objective in the SID output space.

After Stage-1, we follow the original MiniOneRec workflow for recommendation-oriented training. Stage-2 SFT optimizes history-to-next-SID generation on user interaction sequences, optionally with the same sampled SID-level contrastive regularization used in the base pipeline. We further apply GRPO with a rank-aware NDCG reward for recommendation alignment. Importantly, the item–item graph is used only to construct Stage-1 supervision targets; R-GRec does not introduce a graph encoder, graph propagation module, or any inference-time graph computation. Full training objectives and implementation details are provided in Appendix A.

Overall, graph-derived supervision is confined to the Stage-1 graph-instruction mixture; Stage-2 SFT and RL continue to optimize recommendation-oriented SID generation. This separation allows us to study whether explicit collaborative structure improves a standard SID-based generative recommender without changing its tokenizer, decoder, or graph-free inference interface.

4 Experiments

We organize the experiments around the following research questions.

RQ1: *Does R-GRec achieve state-of-the-art SID-based generative recommendation performance on the primary HR@10 and NDCG@10 metrics?*

RQ2: *Do the gains come from meaningful collaborative structure rather than merely adding graph-style instruction data or task templates?*

RQ3: *Which graph-derived structural tasks contribute most to the gains, and are their effects complementary?*

RQ1 is answered by the overall comparison with traditional sequential, generative, and LLM-based baselines. RQ2 is answered by control experiments that replace or disrupt the real collaborative graph while preserving the graph-instruction format. RQ3 is investigated through single-task and leave-one-out task ablations.

4.1 Experimental Settings

Datasets. Following MiniOneRec [12], we evaluate on two Amazon product recommendation benchmarks: *Industrial* and *Office* [17]. We keep the same preprocessing protocol, semantic ID construction, and train/validation/test split as the MiniOneRec baseline to ensure a fair comparison.

Evaluation Metrics. We report Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) at cutoffs $K \in \{3, 5, 10\}$. We use HR@10 and NDCG@10 as the primary metrics for performance comparisons, while reporting HR@3/5 and NDCG@3/5 for completeness. For R-GRec on *Office*, HR@10 and NDCG@10 are averaged over five runs with random seeds (42, 43, 504, 1119, and 2026), keeping the data split, training configuration, and evaluation protocol fixed.

Baselines. We compare against three groups of methods:

- **Traditional sequential recommenders:** GRU4Rec [7], Caser [24], and SASRec [11].
- **Generative recommenders:** HSTU [31], TIGER [20], and LC-Rec [32].
- **LLM-based recommenders:** BIGRec [2], D^3 [1], S-DPO [3], and MiniOneRec [12].

Among them, MiniOneRec is the most relevant baseline, since our method is built on top of its SID-based generative recommendation pipeline.

4.2 RQ1: Overall Performance

Table 1 reports the overall comparison on *Industrial* and *Office*. On the primary HR@10 and NDCG@10 metrics, R-GRec achieves the best performance among the evaluated baselines on both benchmarks, outperforming traditional sequential recommenders, existing generative baselines,

Table 1: Overall performance comparison on Amazon *Industrial* and *Office* datasets. Best results are bolded and second-best results are underlined.

Dataset	Category	Method	HR@3	NDCG@3	HR@5	NDCG@5	HR@10	NDCG@10
Industrial	**Traditional**	GRU4Rec	0.0638	0.0542	0.0774	0.0598	0.0999	0.0669
		Caser	0.0618	0.0514	0.0717	0.0555	0.0942	0.0628
		SASRec	0.0790	0.0700	0.0909	0.0748	0.1088	0.0806
	Generative	HSTU	0.0927	0.0885	0.1037	0.0918	0.1163	0.0958
		TIGER	0.0852	0.0742	0.1010	0.0807	0.1321	0.0908
		LC-Rec	0.0915	0.0805	0.1057	0.0862	0.1332	0.0952
	LLM-based	BIGRec	0.0931	0.0841	0.1092	0.0907	0.1370	0.0997
		D^3	0.1024	0.0991	0.1213	0.0989	0.1500	0.1082
		S-DPO	0.1032	0.0906	0.1238	0.0991	0.1524	0.1082
		MiniOneRec	<u>0.1143</u>	<u>0.1011</u>	<u>0.1321</u>	<u>0.1084</u>	<u>0.1586</u>	<u>0.1167</u>
	Ours	R-GRec	**0.1147**	**0.1012**	**0.1344**	**0.1090**	**0.1662**	**0.1192**
Office	**Traditional**	GRU4Rec	0.0629	0.0528	0.0789	0.0595	0.1019	0.0669
		Caser	0.0748	0.0615	0.0865	0.0664	0.1093	0.0737
		SASRec	0.0861	0.0769	0.0949	0.0805	0.1120	0.0858
	Generative	HSTU	0.1134	0.1031	0.1252	0.1079	0.1400	0.1126
		TIGER	0.0986	0.0852	0.1163	0.0960	0.1408	0.1002
		LC-Rec	0.0921	0.0807	0.1048	0.0859	0.1237	0.0920
	LLM-based	BIGRec	0.1069	0.0961	0.1204	0.1017	0.1434	0.1091
		D^3	0.1204	0.1055	0.1406	0.1139	<u>0.1634</u>	0.1213
		S-DPO	0.1169	0.1033	0.1356	0.1110	0.1587	<u>0.1255</u>
		MiniOneRec	<u>0.1217</u>	<u>0.1088</u>	<u>0.1420</u>	<u>0.1172</u>	<u>0.1634</u>	0.1242
	Ours	R-GRec	**0.1302**	**0.1157**	**0.1474**	**0.1228**	**0.1745**	**0.1316**

and strong LLM-based methods. This supports the state-of-the-art performance claim under the MiniOneRec evaluation protocol.

Compared with MiniOneRec, R-GRec improves HR@10/NDCG@10 from 0.1586/0.1167 to 0.1662/0.1192 on *Industrial*, and from 0.1634/0.1242 to 0.1745/0.1316 on *Office*. The advantage is most pronounced on the primary top-10 metrics.

Answer to RQ1. Yes. R-GRec achieves the best HR@10 and NDCG@10 among the evaluated baselines on both datasets.

4.3 RQ2: Do the Gains Come from Meaningful Collaborative Structure?

RQ1 shows that R-GRec achieves strong overall performance, but it does not by itself explain where the gains come from. A possible alternative explanation is that the improvement comes from adding graph-style instruction data, exposing the model to additional SID targets, or using task templates, rather than from meaningful collaborative topology. To isolate this factor, we compare R-GRec with MiniOneRec and two graph-style control variants.

- **MiniOneRec**: the original SID-based generative recommendation baseline without graph-instruction supervision;
- **Random-Graph**: graph-instruction examples generated from a randomly rewired graph with destinations sampled by original weighted in-degree, preserving the graph-task format while disrupting observed item–item relations;
- **Shuffled-Target**: graph-instruction examples whose target SIDs are shuffled, preserving task templates and target exposure while breaking the correspondence between graph contexts and targets;
- **R-GRec**: graph-instruction examples generated from the train-only collaborative item–item graph.

Table 2: Control experiments isolating the effect of real collaborative graph structure. Raw HR@10 and NDCG@10 values are reported, with relative changes over MiniOneRec shown in parentheses.

Method	Industrial		Office	
	HR@10	NDCG@10	HR@10	NDCG@10
MiniOneRec	0.1586	0.1167	0.1634	0.1242
Random-Graph	0.1577 (-0.6%)	0.1146 (-1.8%)	0.1717 (+5.1%)	0.1293 (+4.1%)
Shuffled-Target	0.1610 (+1.5%)	0.1117 (-4.3%)	0.1671 (+2.3%)	0.1257 (+1.2%)
R-GRec	**0.1662 (+4.8%)**	**0.1192 (+2.1%)**	**0.1745 (+6.8%)**	**0.1316 (+6.0%)**

Table 2 reports mean HR@10 and NDCG@10 over five random seeds, with relative changes over MiniOneRec shown in parentheses. R-GRec outperforms all controls on both metrics across both datasets, improving HR@10/NDCG@10 over MiniOneRec by 4.8% and 2.1% on *Industrial* and 6.8% and 6.0% on *Office*. While Random-Graph and Shuffled-Target yield mixed gains across datasets and metrics, R-GRec improves consistently, supporting the value of meaningful collaborative topology beyond graph-style task templates and additional target exposure.

Answer to RQ2. Yes. R-GRec outperforms both Random-Graph and Shuffled-Target on HR@10 and NDCG@10 across both datasets. This consistent advantage across all four dataset–metric pairs demonstrates the additional value of preserving meaningful collaborative relations: graph-style task templates and target exposure alone cannot match the gains achieved with real collaborative topology.

4.4 RQ3: Which Graph-Derived Tasks Matter?

To understand which structural signals are responsible for the gains, we conduct task-level ablations from two perspectives. First, we evaluate single-task variants, where Stage-1 SFT uses only one graph-instruction task. Second, we conduct leave-one-out ablations, where one task is removed from the full graph-task mixture. These two analyses allow us to distinguish the standalone effect of each task from its marginal contribution when combined with other structural signals.

Table 3: Task-level ablation results on HR@10 and NDCG@10. For all variants, parentheses show relative changes over MiniOneRec.

Variant	Industrial		Office	
	HR@10	NDCG@10	HR@10	NDCG@10
Single-task variants: raw value (relative change over MiniOneRec)				
MiniOneRec	0.1586	0.1167	0.1634	0.1242
Neighbor only	0.1624 (+2.4%)	0.1129 (-3.3%)	0.1710 (+4.7%)	0.1305 (+5.1%)
Contrast only	0.1562 (-1.5%)	0.1093 (-6.3%)	0.1667 (+2.0%)	0.1263 (+1.7%)
Link only	0.1661 (+4.7%)	0.1186 (+1.6%)	0.1697 (+3.9%)	0.1286 (+3.5%)
Full R-GRec	**0.1662 (+4.8%)**	**0.1192 (+2.1%)**	**0.1745 (+6.8%)**	**0.1316 (+6.0%)**
Leave-one-out variants: raw value (relative change over MiniOneRec)				
w/o Link	0.1639 (+3.3%)	0.1172 (+0.4%)	0.1652 (+1.1%)	0.1245 (+0.2%)
w/o Contrast	0.1619 (+2.1%)	0.1156 (-0.9%)	0.1687 (+3.2%)	0.1278 (+2.9%)
w/o Neighbor	0.1613 (+1.7%)	0.1166 (-0.1%)	0.1597 (-2.3%)	0.1229 (-1.0%)

Single-task effects. The single-task results in Table 3 evaluate each graph-derived task as the only source of structural supervision under the same overall graph-task budget. Therefore, a single-task variant allocates all graph-instruction examples to one task type, whereas the full model distributes the same budget across multiple structural signals. These results should be interpreted as the standalone effectiveness of a task when it receives the full graph-supervision budget, rather than as its marginal contribution inside the full mixture.

Among the single-task variants, Link Prediction performs best on *Industrial*, while Neighbor Prediction performs best on *Office*. The full R-GRec model outperforms every single-task variant on both HR@10 and NDCG@10 across both datasets. These results suggest that combining complementary

structural signals is more effective than allocating the entire graph-supervision budget to any single task.

Leave-one-out task contribution. To measure marginal contribution within the full graph-task mixture, we further compare the full model with two-task variants.

$$\Delta_{\text{Link}} = M_{\text{Full}} - M_{\text{Neighbor+Contrast}}, \tag{8}$$

$$\Delta_{\text{Contrast}} = M_{\text{Full}} - M_{\text{Neighbor+Link}}, \tag{9}$$

$$\Delta_{\text{Neighbor}} = M_{\text{Full}} - M_{\text{Contrast+Link}}. \tag{10}$$

Unlike the single-task setting, which reallocates the entire graph budget to one task type, leave-one-out ablation evaluates what is lost when one structural signal is removed from the mixture. The leave-one-out results show positive marginal contributions for all three tasks on the primary HR@10/NDCG@10 metrics. This indicates that the tasks are partially complementary: Neighbor Prediction captures local co-occurrence structure, Topological Contrast provides relative proximity supervision, and Link Prediction models co-neighborhood relations. Additional diagnostics in Appendix F show that graph supervision changes the pre-RL SID generation distribution and improves the structural consistency of recommendations on *Office*.

Answer to RQ3. Single-task variants reveal the effectiveness of individual structural signals when the full graph-supervision budget is concentrated on one task type, with Neighbor Prediction particularly effective on *Office*. The full R-GRec model nevertheless outperforms all single-task variants on both HR@10 and NDCG@10 across both datasets. Leave-one-out ablations further show that removing any task reduces performance, supporting the positive marginal contribution of each structural signal. Together, these results indicate that local neighborhood, relative proximity, and co-neighborhood supervision provide complementary benefits, and that combining all three yields stronger performance than concentrating the supervision budget on any single task.

5 Conclusion

In this work, we studied how to explicitly introduce collaborative structure into SID-based generative recommendation beyond standard history-to-next-item supervision. We proposed R-GRec, a graph-derived structural supervision framework built on top of the MiniOneRec training pipeline. R-GRec constructs a train-only item–item collaborative graph and converts its structural relations into three SID-native auxiliary tasks: Neighbor Prediction, Topological Contrast, and Link Prediction. These tasks inject local neighborhood, relative proximity, and co-neighborhood signals into the SID generation space without changing the tokenizer, generator, or decoding mechanism.

Experiments on Amazon *Industrial* and *Office* show that R-GRec achieves the best HR@10 and NDCG@10 among the evaluated traditional, generative, and LLM-based baselines under the MiniOneRec evaluation protocol. Control experiments suggest that the gains cannot be explained solely by additional graph-style instruction data or task templates, although random graph-style supervision can provide regularization benefits in some settings. Task-level ablations further show that the three graph-derived tasks provide partially complementary structural signals. Overall, our results indicate that explicit collaborative structure remains a valuable source of supervision for modern SID-based generative recommendation.

6 Limitations

This work has several limitations. First, our experiments are conducted on two Amazon product recommendation benchmarks under the MiniOneRec evaluation protocol, so the conclusions are most directly supported within the SID-based generative recommendation setting studied here. Second, R-GRec relies on item–item co-occurrence graphs derived from historical training interactions; its effectiveness may depend on the density, quality, and temporal stability of these collaborative signals. Third, although the graph is not used during inference, graph construction and graph-task generation add training-time preprocessing cost. Finally, because evaluation follows SID-level exact matching, SID collisions may introduce ambiguity between SID-level success and item-level retrieval when multiple items share the same semantic ID.

Acknowledgments and Disclosures

This work is supported by the Key Laboratory of Public Opinion Governance and Computational Communication (No. YQKFYB202501), and the Fundamental Research Funds for the Central Universities (No. CUC26TD13). The authors declare no competing interests.

References

[1] Keqin Bao, Jizhi Zhang, Yang Zhang, Xinyue Huo, Chong Chen, and Fuli Feng. Decoding matters: Addressing amplification bias and homogeneity issue for llm-based recommendation. *arXiv preprint arXiv:2406.14900*, 2024.

[2] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yanchen Luo, Chong Chen, Fuli Feng, and Qi Tian. A bi-step grounding paradigm for large language models in recommendation systems. *ACM Transactions on Recommender Systems*, 3(4):1–27, 2025.

[3] Yuxin Chen, Junfei Tan, An Zhang, Zhengyi Yang, Leheng Sheng, Enzhi Zhang, Xiang Wang, and Tat-Seng Chua. On softmax direct preference optimization for recommendation. *Advances in Neural Information Processing Systems*, 37:27463–27489, 2024.

[4] Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965*, 2025.

[5] Chao Feng, Wuchao Li, Defu Lian, Zheng Liu, and Enhong Chen. Recommender forest for efficient retrieval. *Advances in Neural Information Processing Systems*, 35:38912–38924, 2022.

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

[7] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[8] Yupeng Hou, Jiacheng Li, Ashley Shin, Jinsung Jeon, Abhishek Santhanam, Wei Shao, Kaveh Hassani, Ning Yao, and Julian McAuley. Generating long semantic ids in parallel for recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 956–966, 2025.

[9] Yupeng Hou, Jianmo Ni, Zhankui He, Noveen Sachdeva, Wang-Cheng Kang, Ed H Chi, Julian McAuley, and Derek Zhiyuan Cheng. Actionpiece: Contextually tokenizing action sequences for generative recommendation. *arXiv preprint arXiv:2502.13581*, 2025.

[10] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. How to index item ids for recommendation foundation models. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 195–204, 2023.

[11] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[12] Xiaoyu Kong, Leheng Sheng, Junfei Tan, Yuxin Chen, Jiancan Wu, An Zhang, Xiang Wang, and Xiangnan He. Minionerec: An open-source framework for scaling generative recommendation. *arXiv preprint arXiv:2510.24431*, 2025.

[13] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM conference on recommender systems*, pages 59–66, 2016.

[14] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *Proceedings of the ACM web conference 2022*, pages 2320–2329, 2022.

[15] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[16] Enze Liu, Bowen Zheng, Cheng Ling, Lantao Hu, Han Li, and Wayne Xin Zhao. Generative recommender with end-to-end learnable item tokenization. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 729–739, 2025.

[17] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1018. URL https://aclanthology.org/D19-1018/.

[18] Haohao Qu, Wenqi Fan, Zihuai Zhao, and Qing Li. Tokenrec: Learning to tokenize id for llm-based generative recommendations. *IEEE Transactions on Knowledge and Data Engineering*, 37(10):6216–6231, 2025. doi: 10.1109/TKDE.2025.3599265.

[19] Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

[20] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

[21] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820, 2010.

[22] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[23] Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. Idgenrec: Llm-recsys alignment with textual id learning. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 355–364, 2024.

[24] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573, 2018.

[25] Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2400–2409, 2024.

[26] Ye Wang, Jiahao Xun, Minjie Hong, Jieming Zhu, Tao Jin, Wang Lin, Haoyuan Li, Linjun Li, Yan Xia, Zhou Zhao, et al. Eager: Two-stream generative recommender with behavior-semantic collaboration. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3245–3254, 2024.

[27] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 726–735, 2021.

[28] Longtao Xiao, Haozhao Wang, Cheng Wang, Linfei Ji, Yifan Wang, Jieming Zhu, Zhenhua Dong, Rui Zhang, and Ruixuan Li. Unger: Generative recommendation with a unified code via semantic and collaborative integration. *ACM Transactions on Information Systems*, 44(2):1–31, 2025.

[29] Wencai Ye, Mingjie Sun, Shaoyun Shi, Peng Wang, Wenjin Wu, and Peng Jiang. Das: Dual-aligned semantic ids empowered industrial recommender system. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 6217–6224, 2025.

[30] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[31] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *arXiv preprint arXiv:2402.17152*, 2024.

[32] Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 1435–1448. IEEE, 2024.

A Training Objectives and Pipeline Details

A.1 Training Pipeline built on MiniOneRec

Our method is implemented as a pipeline-level augmentation to MiniOneRec rather than a replacement of its SID tokenizer, decoding procedure, or downstream recommendation objective. The key design is to inject explicit collaborative structure into the early SID-alignment stage through graph-instruction data, while keeping the later history-to-next-item recommendation training compatible with the original MiniOneRec workflow.

Stage-1 SFT with graph-task mixture. In the first SFT stage, we train the model on a mixture of text–SID alignment tasks and graph-instruction tasks. Let $\mathcal{D}_{\text{align}}$ denote the alignment corpus, which contains text-to-SID and SID-to-text examples, and let $\mathcal{D}_{\text{graph}}$ denote the graph-instruction corpus generated from the train-only item interaction graph. The graph-instruction corpus consists of the structural tasks defined above, including neighbor prediction, topological contrast, and co-neighborhood prediction. All graph tasks are serialized into the same instruction-to-SID format as the original SFT data.

The purpose of this mixture is twofold. The text–SID alignment examples ground the discrete SID tokens in item semantics, while the graph-instruction examples inject collaborative topology into the SID generation space. Thus, Stage-1 introduces graph-derived structural supervision without turning the model into a pure graph-reconstruction system.

We define an example-level graph-task mixing ratio $\lambda \in [0, 1]$. Stage-1 SFT can be viewed as optimizing the following empirical mixture objective:

$$\mathcal{L}_{\text{SFT1-CE}}(\theta; \lambda) = (1 - \lambda)\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{align}}} \left[\ell_{\text{CE}} \left(f_\theta(x), y\right)\right] + \lambda\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{graph}}} \left[\ell_{\text{CE}} \left(f_\theta(x), y\right)\right] \quad (11)$$

where ℓ_{CE} denotes the per-example token-averaged autoregressive cross-entropy loss, and f_θ denotes the language model.

In implementation, λ is realized by controlling the number of graph-instruction examples mixed into the Stage-1 SFT corpus:

$$\lambda = \frac{|\mathcal{D}_{\text{graph}}|}{|\mathcal{D}_{\text{align}}| + |\mathcal{D}_{\text{graph}}|}. \quad (12)$$

Thus, λ controls the empirical example-level sampling frequency and coverage of graph-derived supervision, rather than serving as an additional scalar multiplier on the loss. No separate graph-specific loss coefficient is introduced; alignment examples and graph-instruction examples are optimized with the same autoregressive CE objective. In our default setting, we use $\lambda \approx 0.2$, corresponding to $|\mathcal{D}_{\text{align}}| = 7316$ and $|\mathcal{D}_{\text{graph}}| = 1830$.

Stage-2 recommendation SFT with contrastive regularization. After Stage-1 establishes a text-aligned and graph-aware SID space, Stage-2 fine-tunes the model on recommendation-oriented SFT examples. Given a user history $\mathbf{x}_u$ and the target item y_u, the base recommendation SFT objective is

$$\mathcal{L}_{\text{SFT2-CE}} = -\mathbb{E}_{(\mathbf{x}_u, y_u)\sim\mathcal{D}_{\text{rec}}} \log p_\theta \left(\mathbf{s}(y_u) \mid \mathbf{x}_u\right), \quad (13)$$

where $\mathbf{s}(y_u)$ denotes the three-token Semantic ID of the target item. This stage follows the original MiniOneRec history-to-next-SID training format.

When enabled, we add a sampled InfoNCE regularizer in Stage-2. Instead of constructing separate text-side and SID-side item encoders, we use the model's contextual hidden state during target SID generation as the anchor. For a training example i, let $\mathbf{s}_i = (s_{i,1}, s_{i,2}, s_{i,3})$ denote the target SID sequence. Let $\mathbf{h}_i$ be the final-layer hidden state at the target SID position, and let

$$\mathbf{e}(\mathbf{s}_i) = \text{norm}\left(\frac{1}{3}\sum_{m=1}^{3} E(s_{i,m})\right) \quad (14)$$

be the normalized mean embedding of the three SID tokens, where $E(\cdot)$ denotes the model input embedding table.

For each valid example in a mini-batch, we sample K negative SID sequences $\{\tilde{\mathbf{s}}_{i,k}\}_{k=1}^{K}$ from a frequency-smoothed distribution

$$q(\mathbf{s}) \propto \mathrm{freq}(\mathbf{s})^{\alpha}. \tag{15}$$

The sampled InfoNCE loss is

$$\mathcal{L}_{\mathrm{NCE}} = -\frac{1}{|\mathcal{B}'|} \sum_{i \in \mathcal{B}'} \log \frac{\exp\left(\mathrm{sim}(\mathbf{h}_i, \mathbf{e}(\mathbf{s}_i))/\tau\right)}{\exp\left(\mathrm{sim}(\mathbf{h}_i, \mathbf{e}(\mathbf{s}_i))/\tau\right) + \sum_{k=1}^{K} \exp\left(\mathrm{sim}(\mathbf{h}_i, \mathbf{e}(\tilde{\mathbf{s}}_{i,k}))/\tau\right)}, \tag{16}$$

where $\mathcal{B}'$ denotes the valid examples in the mini-batch that contain target SID tokens, $\mathrm{sim}(\cdot,\cdot)$ denotes cosine similarity, and τ is the temperature.

The Stage-2 objective is then

$$\mathcal{L}_{\mathrm{SFT2}} = \mathcal{L}_{\mathrm{SFT2\text{-}CE}} + \beta \mathcal{L}_{\mathrm{NCE}}, \tag{17}$$

where β corresponds to the implementation hyperparameter `infonce_weight` and controls the strength of the auxiliary sampled SID-embedding contrastive regularizer.

We distinguish β from the graph-task mixing ratio λ: λ controls the example-level composition of the Stage-1 SFT corpus, whereas β controls the auxiliary InfoNCE regularization strength in Stage-2.

RL fine-tuning. After SFT, we apply GRPO to optimize generated SID sequences under a rank-aware reward.We use a rule-based NDCG reward, following the original MiniOneRec design. Although we tested several more complex reward variants, such as graph-aware and candidate-calibrated rewards, they did not provide consistent gains; therefore, we use the simple NDCG reward in the final pipeline.

For a user u, the model generates a ranked list of item SIDs. If the ground-truth item y_u appears at rank r_u, the reward is

$$R_u = \frac{1}{\log_2(r_u + 1)}; \quad R_u = 0 \text{ if } y_u \text{ is not retrieved.} \tag{18}$$

For each prompt, GRPO samples G responses and normalizes their rewards within the group:

$$A_{u,g} = \frac{R_{u,g} - \mu_u}{\sigma_u + \epsilon}. \tag{19}$$

The policy is optimized with the clipped GRPO objective:

$$\mathcal{L}_{\mathrm{GRPO}} = -\mathbb{E}_{u,g}\left[\min\left(\rho_{u,g} A_{u,g}, \mathrm{clip}(\rho_{u,g}, 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}) A_{u,g}\right)\right] + \beta_{\mathrm{KL}} D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}}), \tag{20}$$

where

$$\rho_{u,g} = \frac{\pi_\theta(\hat{\mathbf{s}}_{u,g} \mid \mathbf{x}_u)}{\pi_{\theta_{\mathrm{old}}}(\hat{\mathbf{s}}_{u,g} \mid \mathbf{x}_u)}. \tag{21}$$

A.2 Dataset, Evaluation Protocol, and SID Construction Details

We follow the MiniOneRec preprocessing and evaluation protocol. The experiments use the Amazon Review 2018 5-core subsets for `Industrial_and_Scientific` and `Office_Products`. We keep interactions within the timestamp window from October 2016 to November 2018, apply the standard 5-core filtering and metadata title filtering, and sort each user's interactions chronologically. From each user sequence, we construct next-item prediction instances by using previous interactions as context and the next interaction as the target. The history length is capped at 10 items. For evaluation prompts, the last item in the historical sequence is treated as the current item, while the preceding items form the chronological behavior sequence.

The train/validation/test split is not a per-user leave-one-out split. Instead, after constructing all next-item prediction instances, we sort instances by target timestamp and use a chronological 8:1:1 split. Table 4 reports the resulting dataset statistics. The number of interactions corresponds to the filtered user-item interactions before next-item instance construction; the train/validation/test counts correspond to the generated next-item prediction instances.

Table 4: Dataset statistics after MiniOneRec preprocessing. Sparsity is computed as $1 - |\mathcal{R}|/(|\mathcal{U}||\mathcal{I}|)$, where $|\mathcal{R}|$ is the number of filtered interactions.

Dataset	#Users	#Items	#Interactions	Avg. Seq. Len.	#Train/#Val/#Test	Sparsity
Industrial	7,694	3,686	53,018	6.89	36,259 / 4,532 / 4,533	99.813%
Office	8,328	3,459	56,984	6.84	38,924 / 4,866 / 4,866	99.802%

Table 5: SID statistics. "Affected test targets" is the percentage of test instances whose target item belongs to a SID shared by multiple items.

Dataset	SID Len.	Atomic SID Vocab	Unique SIDs	Collision Groups	Collision Rate	Affected Test Targets
Industrial	3	560	3,670	15	0.434%	6.993%
Office	3	600	3,444	15	0.434%	1.726%

Evaluation protocol. We evaluate recommendation quality with the official MiniOneRec SID-generation protocol. For each test instance, the model generates a ranked list of semantic IDs (SIDs), and HR@K and NDCG@K are computed by exact matching between the generated SID string and the target SID string. Thus, the reported metrics are SID-level exact-match metrics. In the main experiments, we use full-catalog constrained beam decoding rather than sampled-negative evaluation: all valid catalog SIDs from the corresponding `info` file are inserted into a prefix trie, and decoding is constrained so that every generated sequence is a valid SID. We use beam size 50, return 50 sequences, maximum generation length 256 tokens, length penalty 0, and deterministic beam search without sampling. This is the same protocol used for MiniOneRec-style HR@K/NDCG@K comparison.

SID construction and collisions. Each item is represented by a length-3 semantic ID of the form `<a_x><b_y><c_z>`. These SID tokens are added to the tokenizer vocabulary before training. Table 5 summarizes the SID statistics. The atomic SID vocabulary size counts unique `a`, `b`, and `c` tokens that appear in the dataset. A small number of items share the same SID. When this happens, the MiniOneRec protocol cannot distinguish those items at decoding time because they correspond to the same token sequence; a prediction is counted as correct if the generated SID exactly matches the target item's SID. We therefore report the collision rate explicitly. The collision rate is computed as $(\#\text{items} - \#\text{unique SIDs})/\#\text{items}$.

For completeness, the atomic SID vocabularies contain 48/256/256 `a`/`b`/`c` tokens for Industrial and 88/256/256 tokens for Office. All catalog items in both datasets are encoded as exactly three SID tokens. During evaluation, the valid SID trie is built from all catalog SIDs in the corresponding `info` file, which ensures that generated recommendations are valid catalog identifiers.

B Full Graph Construction Details

This section gives the complete construction procedure for the item-item interaction graph used to derive graph-supervised alignment tasks. The graph is constructed separately for each dataset and uses only the training split. Validation and test instances, including their target items, are never used to construct edges, item frequencies, PMI statistics, transition probabilities, or graph-derived task samples.

Algorithm. Let a training instance contain a chronological sequence $s = (i_1, \ldots, i_n)$ formed by concatenating the history items and the next-item target from the training CSV. For each ordered pair (i_p, i_q) with $p < q$, we add a directed edge from i_p to i_q with a distance-decayed weight. This corresponds to `BuildItemGraph(train_sequences, alpha, topK, min_count, pmi_correction)` in pseudocode. The complete procedure is:

1. Initialize item frequencies f_i, an item-to-SID map, and a directed weighted adjacency matrix W.
2. For every training sequence $s = (i_1, \ldots, i_n)$, add every observed item to the node set and update f_i from both history and target occurrences.

Table 6: Statistics of the train-only item interaction graphs. Degree denotes directed out-degree after all graph filtering steps.

Dataset	#Nodes	#Edges	Avg. Degree	Median Degree	Density	Isolated Items
Industrial	3,647	39,624	10.86	8	0.298%	1.810%
Office	3,445	37,924	11.01	8	0.320%	0.697%

3. For every ordered pair $p < q$, update

$$W_{i_p,i_q} \leftarrow W_{i_p,i_q} + \exp\{-\alpha(q-p-1)\}, \tag{22}$$

where α is the distance decay coefficient. Adjacent items therefore receive weight 1, and longer-range co-occurrences are exponentially down-weighted.

4. Prune edges with $W_{ij} < \tau$, where τ is the minimum co-occurrence weight threshold.

5. Compute PPMI scores on the remaining directed edges. Let $T = \sum_{i,j} W_{ij}$ and let $m_i = \sum_j W_{ij} + \sum_j W_{ji}$. We define

$$\mathrm{PMI}(i,j) = \log \frac{W_{ij}/T}{(m_i/2T)(m_j/2T)}, \quad \mathrm{PPMI}(i,j) = \max(\mathrm{PMI}(i,j), 0). \tag{23}$$

In implementation, a numerical constant 10^{-10} is added inside the logarithm argument for stability.

6. Min-max normalize the raw co-occurrence weights and PPMI weights separately, then combine them as

$$\widetilde{W}_{ij} = \lambda\, \widehat{\mathrm{PPMI}}(i,j) + (1-\lambda)\, \widehat{W}_{ij}. \tag{24}$$

7. Apply frequency correction

$$\widetilde{W}_{ij} \leftarrow \widetilde{W}_{ij} \cdot \min(f_i, f_j)^{\beta}. \tag{25}$$

8. Retain the largest outgoing edges per source. In the final configuration we use adaptive top-K:

$$K_i = \min\{K_{\max}, \lfloor K_{\mathrm{scale}} \log(1+f_i) \rfloor + K_{\min}\}. \tag{26}$$

9. Normalize outgoing edges to obtain the transition distribution

$$P(j \mid i) = \frac{\widetilde{W}_{ij}}{\sum_k \widetilde{W}_{ik}}. \tag{27}$$

Graph statistics. Table 6 reports the final train-only graphs after pruning, mixed PPMI/co-occurrence scoring, frequency correction, adaptive top-K, and transition normalization. Because the graph is directed, average and median degree are reported as directed out-degree over all graph nodes, and isolated items are nodes with zero outgoing edges after pruning and top-K truncation. Density is computed as $|E|/(|V|(|V|-1))$.

Graph hyperparameters. Table 7 lists the graph construction hyperparameters used in the main experiments. The same construction parameters are used before generating the real graph tasks and before applying graph-control transformations.

Leakage prevention. The graph is built only from the training CSV of each dataset. The training CSV contains only next-item instances whose target timestamp belongs to the chronological training segment described in Appendix A.2. The validation and test CSVs are not read by the graph-construction script. Consequently, validation/test targets do not affect edge weights, item frequencies, PPMI values, top-K neighbor lists, transition probabilities, or graph-derived training tasks. The item-to-SID map used for graph construction is also collected from the training CSV rows only.

Table 7: Graph construction hyperparameters.

Component	Hyperparameter	Value
Distance decay	α	0.7
Edge pruning	Minimum edge weight τ	1.0
Raw graph	Directed edges	Yes
PMI correction	Use PPMI	Yes
PMI correction	Numerical smoothing	10^{-10} inside log argument
Mixed scoring	λ in PPMI/co-occurrence mixture	0.7
Frequency correction	β in $\min(f_i, f_j)^\beta$	0.3
Neighbor truncation	Fixed top-K fallback	20
Adaptive top-K	Enabled	Yes
Adaptive top-K	$K_{\min}, K_{\max}, K_{\text{scale}}$	5, 30, 5.0
Edge normalization	Transition distribution	L1-normalize outgoing edges
Task node sampling	Frequency exponent γ	0.5
Neighbor task target size	Top neighbors m	3
Contrastive negatives	2-hop / 3-hop / random ratio	0.4 / 0.4 / 0.2

Graph variants. We use three graph-task variants in control experiments.

Real-Graph is the graph produced by the train-only construction algorithm above. All graph-derived tasks use neighbors, paths, and common-neighbor sets from this real train graph.

Random-Graph first builds the same train-only real graph and then rewires edges before task generation. For each source node, the transformation preserves the source out-degree and the multiset of outgoing edge weights, but resamples destination nodes. By default, destinations are sampled from active graph nodes according to their original weighted in-degree, and original edges are disallowed when enough alternative destinations exist. This makes the control degree- and popularity-preserving while breaking the original collaborative topology. After rewiring, outgoing edges are renormalized and the same task generator is applied.

Shuffled-Target keeps the real graph task inputs unchanged but randomly reassigns outputs within each task type. The shuffle is performed separately for neighbor prediction, topological contrast, and link prediction, using a derangement-like procedure so that an output is not paired with its original input whenever possible. This preserves the prompt distribution, target length distribution, and number of training examples, while breaking the input-output correspondence.

C Graph-Derived Task Generation and Prompt Templates

After constructing the train-only graph, we generate graph-derived language modeling tasks in JSONL format. Each row has fields `task`, `input`, and `output`. During SFT, each graph task is wrapped with the fixed instruction "Answer the question about the item interaction graph." The model is trained only on the output tokens; the instruction and user input are masked in the language-modeling labels. SID targets are serialized as raw SID strings and terminated by a newline and EOS token. When a target contains multiple items, their SIDs are concatenated directly, e.g., `<a_1><b_2><c_3><a_4><b_5><c_6>`. For all graph tasks, samples are skipped if the required item-to-SID mapping is missing or if the resulting target set is empty.

Neighbor Prediction. For an anchor item i, let $N_m(i)$ be the top-m outgoing neighbors ranked by $P(j \mid i)$. The task asks the model to recover these high-probability neighbors from the anchor SID. Anchors are sampled from active graph nodes. With frequency-aware sampling enabled, anchor sampling is proportional to f_i^γ, where $\gamma = 0.5$; otherwise anchors are sampled uniformly. We use $m = 3$ in the main experiments. If the anchor has no valid SID or no valid neighbor SID, the sample is skipped.

Prompt template: `In the item interaction graph, the most frequently co-occurring items with <anchor_sid> are:`

Target template: `<neighbor_sid_1><neighbor_sid_2><neighbor_sid_3>`

Example input: `In the item interaction graph, the most frequently co-occurring items with <a_12><b_7><c_31> are:`

Example target: <a_3><b_18><c_9><a_44><b_2><c_11><a_8><b_91><c_5>

Topological Contrast. For an anchor item i, the positive candidate is sampled from its top-3 direct neighbors under $P(j \mid i)$. The negative candidate is sampled from one of three sources: 2-hop non-neighbors, 3-hop non-neighbors, or random far nodes. The intended mixture is 40% 2-hop, 40% 3-hop, and 20% random. A 2-hop candidate is accepted only if it is not a direct neighbor and its common-neighbor ratio with the anchor is below 0.5, where the ratio is $|N(i) \cap N(k)|/|N(i)|$. If the requested negative source is unavailable, the generator falls back to another available 3-hop or far-node candidate. Candidate order is randomized with probability 0.5, so the positive can appear as either Candidate 1 or Candidate 2. Popularity is controlled through frequency-weighted anchor sampling rather than exact popularity matching of the negative candidate.

Prompt template:

```
Which item is more closely related to <anchor_sid> in the
interaction graph?
Candidate 1: <candidate_sid_1>
Candidate 2: <candidate_sid_2>
Answer:
```

Target template: <positive_sid>

Example target: <a_3><b_18><c_9>

Link Prediction. For link prediction, the generator samples an observed edge pair (i, j) and asks for items connected to both endpoints. The common-neighbor set is

$$C(i, j) = (N(i) \cap N(j)) \setminus \{i, j\}. \tag{28}$$

Pairs with fewer than one common neighbor are skipped. Candidate common neighbors are ranked by

$$\text{score}(c \mid i, j) = \widetilde{W}_{ic} + \widetilde{W}_{jc}, \tag{29}$$

and the target serializes the top three common-neighbor SIDs. In the real graph generator, edge pairs are deduplicated by retaining pairs with `src < dst`; the order of the two endpoint SIDs in the prompt is randomized with probability 0.5.

Prompt template: `In the item interaction graph, <sid_a> and <sid_b> are frequently co-viewed. Which items are connected to both of them?`

Target template: <common_sid_1><common_sid_2><common_sid_3>

Example input: `In the item interaction graph, <a_12><b_7><c_31> and <a_9><b_14><c_80> are frequently co-viewed. Which items are connected to both of them?`

Example target: <a_3><b_18><c_9><a_44><b_2><c_11>

Task mixture. In the main Industrial configuration, the Stage-1 graph-task set contains 1,830 graph-derived samples: 549 neighbor-prediction tasks, 549 topological-contrast tasks, and 732 link-prediction tasks. This corresponds to a 3:3:4 task ratio and approximately 20% graph tasks relative to the 7,316 item alignment samples. For Office, the generated graph-task set contains 1,730 samples: 990 neighbor-prediction tasks, 494 topological-contrast tasks, and 246 link-prediction tasks. Random-walk graph tasks are disabled in the main configuration.

D Full Training and Hyperparameter Details

The complete training pipeline consists of Stage-1 alignment SFT, Stage-2 recommendation SFT with an auxiliary InfoNCE loss, and GRPO fine-tuning. Table 8 summarizes Stage 1. Stage 1 extends the tokenizer with all atomic SID tokens and trains only the newly added SID embedding rows; the base language-model parameters and the original tokenizer embeddings are frozen by a gradient mask. This stage mixes item-level text/SID alignment tasks with the graph-derived tasks from Appendix C.

Table 8: Stage-1 SFT hyperparameters.

Category	Hyperparameter	Value
Backbone	Base model	Qwen2.5-0.5B
Tokenizer	Added tokens	Atomic SID tokens `<a_x>`, `<b_y>`, `<c_z>`
Trainable parameters	Stage-1 freezing	New SID embedding rows only
Data	Alignment samples $\lvert D_{\text{align}} \rvert$	7,316
Data	Graph samples $\lvert D_{\text{graph}} \rvert$	1,830
Data	Graph task distribution	549 / 549 / 732
Data	Graph task ratio	Approximately 20% of Stage-1 samples
Optimization	Learning rate	1×10^{-4}
Optimization	Epochs	10
Optimization	Global batch size	2,048
Optimization	Micro batch size	256
Optimization	Optimizer	AdamW
Optimization	Warmup steps	20
Optimization	Scheduler	Linear warmup/decay (Transformers default)
Sequence	Maximum length	512 tokens
Precision	Numeric format	bfloat16
Checkpointing	Save strategy	Per epoch, keep final checkpoint

Compute budget. All main training runs were conducted on NVIDIA B200 GPUs with approximately 180GB memory per GPU. In the Industrial main pipeline, Stage-1 embedding-only alignment is lightweight and takes about 1–2 minutes, Stage-2 recommendation SFT takes about 36 minutes on 8 B200 GPUs, and GRPO fine-tuning takes about 3.7 hours on 4 B200 GPUs. The end-to-end training wall-clock time is therefore approximately 4.3 hours, excluding graph preprocessing, checkpoint evaluation, and final decoding-based test evaluation. This corresponds to roughly 20 B200 GPU-hours of training compute for one full Industrial run.

Table 9 summarizes Stage 2. Stage 2 starts from the Stage-1 checkpoint, unfreezes the full model, and trains recommendation tasks. The default Stage-2 supervised data contains both history-SID-to-SID and history-title-to-SID prediction. The auxiliary InfoNCE loss is computed over the hidden state at the final SID-token position and the average embedding of the three target SID tokens. Negatives are sampled from training target SIDs with probability proportional to item frequency raised to the exponent 0.75. The objective is

$$\mathcal{L}_{\text{Stage2}} = \mathcal{L}_{\text{CE}} + \eta \, \mathcal{L}_{\text{InfoNCE}}, \tag{30}$$

where $\eta = 0.01$.

Table 10 summarizes the GRPO fine-tuning configuration. The policy is initialized from the best Stage-2 checkpoint selected by the checkpoint-evaluation step. The final reward type is `ranking`, which combines exact-match reward and a rule-based NDCG-position reward under constrained SID generation. We use synchronized reference-model updates and a KL penalty controlled by $\beta_{\text{KL}} = 5 \times 10^{-3}$.

Reward variants. The main RL stage uses the rule-based `ranking` reward because it was the most stable configuration in our experiments. We also implemented graph-aware reward variants, including direct graph-proximity rewards, graph-negative penalties, graph-aware ranking losses, and calibrated graph rewards normalized by popularity buckets. Candidate-list rewards based on HR@K and NDCG@K were also implemented for candidate-generation and reranking experiments. These reward-shaping variants were useful controls, but they did not provide consistent gains over the final exact-match plus NDCG-position reward, so they are not used in the main reported model.

E Graph-Supervision Ratio Sensitivity

We sweep the proportion of graph-derived instruction examples used in Stage-1 SFT while keeping the graph construction procedure, Stage-2 recommendation SFT, InfoNCE configuration, and GRPO

Table 9: Stage-2 SFT and InfoNCE hyperparameters.

Category	Hyperparameter	Value
Initialization	Base checkpoint	Stage-1 final checkpoint
Trainable parameters	Stage-2 freezing	Full model unfrozen
Data	Recommendation tasks	history-SID-to-SID; history-title-to-SID
Optimization	Learning rate	1×10^{-4}
Optimization	Epochs	20
Optimization	Global batch size	2,048
Optimization	Micro batch size	64
Optimization	Optimizer	AdamW
Optimization	Warmup steps	20
Optimization	Scheduler	Linear warmup/decay (Transformers default)
Sequence	Maximum length	512 tokens
Precision	Numeric format	bfloat16
Selection	Validation criterion	Best validation loss
InfoNCE	Loss weight η	0.01
InfoNCE	Temperature τ	0.15
InfoNCE	Number of negatives	64
InfoNCE	Negative sampling	$p(i) \propto f_i^{0.75}$ over training target SIDs

Table 10: GRPO fine-tuning hyperparameters.

Category	Hyperparameter	Value
Initialization	Policy model	Best Stage-2 checkpoint
Reference model	Synchronization	Enabled
Reward	Reward type	`ranking`
Reward	Components	Exact SID match + NDCG-position shaping
Optimization	Learning rate	5×10^{-6}
Optimization	Epochs	2
Optimization	Train/eval batch size	1,280 / 1,280
Optimization	Gradient accumulation	2
Optimization	Optimizer	paged AdamW 32-bit
Optimization	Scheduler	Cosine
Optimization	Warmup ratio	0.03
Optimization	Max gradient norm	0.3
GRPO	Rollouts per prompt	32
GRPO	KL coefficient β_{KL}	5×10^{-3}
GRPO	Clipping epsilon	Not used; no explicit PPO-style clipping
Generation	Max completion length	32 tokens
Generation	Beam search during rollout	Enabled
Generation	Temperature	1.0
Generation	Ground-truth injection	Disabled
Algorithm variants	DAPO / GSPO	Disabled
Precision	Numeric format	bfloat16

configuration fixed. Let

$$\lambda = \frac{|D_{\mathrm{graph}}|}{|D_{\mathrm{align}}| + |D_{\mathrm{graph}}|}, \tag{31}$$

where D_{align} denotes item text/SID alignment examples and D_{graph} denotes graph-derived instruction examples. The default setting in the main pipeline uses $|D_{\mathrm{align}}| = 7{,}316$ and $|D_{\mathrm{graph}}| = 1{,}830$, giving $\lambda \approx 0.2$. The $\lambda = 0$ condition removes graph-derived Stage-1 instruction examples but keeps the remaining Stage-2 and GRPO training recipe unchanged.

Table 11 shows that the default $\lambda \approx 0.2$ setting is the best point among the tested ratios after full SFT+GRPO training. Very small graph mixtures provide little improvement over the no-graph setting, while larger mixtures do not yield further gains and can slightly reduce ranking quality. This supports

Table 11: Sensitivity to the Stage-1 graph-supervision ratio on Industrial after the full SFT+GRPO pipeline. Metrics are computed with the MiniOneRec constrained SID-generation protocol.

Graph Ratio λ	Industrial HR@10	Industrial NDCG@10
0.0	0.1573	0.1122
0.1	0.1549	0.1117
0.2	**0.1662**	**0.1192**
0.3	0.1610	0.1147
0.4	0.1606	0.1142
0.5	0.1582	0.1142

using graph-derived instructions as a moderate Stage-1 auxiliary signal rather than allowing graph tasks to dominate item alignment.

This sensitivity analysis is limited to Industrial; Office uses the same default graph-supervision ratio.

F Mechanism Analysis

We compare the pre-RL SFT checkpoints of MiniOneRec and R-GRec on *Office* under the same test protocol to examine how graph supervision affects SID generation before reinforcement learning. We measure valid-next-SID token entropy to characterize the concentration of the conditional generation distribution. We also measure TCR@10, defined as the fraction of top-10 recommended items connected to at least one item in the user's history in the observed training graph.

Table 12: Mechanism diagnostics on *Office* using pre-RL SFT checkpoints.

Diagnostic	MiniOneRec	R-GRec
Valid-next-SID token entropy	1.0829	1.9875
TCR@10	0.2666	0.3131

Table 12 shows that R-GRec produces a less concentrated distribution over valid next SID tokens and recommendations that are more consistent with the observed collaborative graph. The TCR@10 improvement is 0.0465, with a paired bootstrap 95% confidence interval of [0.0409, 0.0522]. After adjusting for recommended-item interaction frequency, the estimated improvement remains positive at 0.0076 (95% CI: [0.0012, 0.0139]). These results support the role of graph supervision in aligning SID generation with collaborative neighborhoods before RL-based ranking optimization.

G Broader Impacts

R-GRec studies training-time structural supervision for SID-based generative recommendation. A potential positive impact is that it can improve recommendation quality while keeping inference graph-free, which may reduce deployment complexity compared with methods that require inference-time graph computation. At the same time, recommendation systems can have negative societal impacts, including popularity-bias amplification, reduced exposure diversity, preference bubbles, and unfair exposure across items, sellers, or content providers. Since R-GRec learns from historical item–item co-occurrence structure, it may inherit or amplify biases already present in user interaction logs. Although our experiments use public benchmark datasets and aggregate item-level graphs, practical deployment should carefully evaluate privacy, fairness, exposure diversity, and temporal distribution shift.

H Assets, Licenses, and Terms of Use

We use the Amazon Review 2018 datasets[17] for offline recommendation evaluation, specifically the Industrial_and_Scientific and Office_Products subsets after MiniOneRec-compatible preprocessing. The dataset is available at `https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/index.html`. The dataset authors state that they do not own the data and are not in a position to

provide a license, and request that the data be used for research purposes. We use the data only for offline benchmark evaluation.

Our implementation builds on the MiniOneRec pipeline (`https://github.com/AkaliKong/MiniOneRec`)and follows its preprocessing, SID construction, training, and evaluation protocol. MiniOneRec is released under the Apache-2.0 license. The language-model backbone used in our experiments is Qwen2.5-0.5B, [19] (`https://huggingface.co/Qwen/Qwen2.5-0.5B`), which is also released under the Apache-2.0 license.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the paper's main contribution as a training-time structural supervision framework for SID-based generative recommendation. The claimed contributions are limited to constructing a train-only item–item graph, deriving graph-based auxiliary SID-generation tasks, and integrating them into the MiniOneRec pipeline without changing the tokenizer, decoder, or inference procedure. The empirical claims are evaluated under the MiniOneRec protocol on two Amazon benchmarks, with HR@10 and NDCG@10 used as the primary comparison metrics. The paper also includes control and ablation experiments to qualify the effect of graph-derived supervision and the contribution of individual tasks.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The paper includes a dedicated Limitations section before the references. It discusses the empirical scope of the evaluation, noting that experiments are conducted on two Amazon benchmarks under the MiniOneRec SID-based generative recommendation protocol. It also describes the dependence of R-GRec on the density, quality, and temporal stability of train-only item–item co-occurrence graphs, the additional training-time preprocessing cost from graph construction and graph-task generation, and the ambiguity introduced by SID collisions under SID-level exact-match evaluation.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A].

   Justification: The paper does not present formal theoretical results, theorems, or proofs. The equations in the paper define the SID-generation objective, graph construction procedure, auxiliary graph-derived tasks, and training losses used by the method.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: The paper provides the information needed to reproduce the main experimental results. Appendix A describes the MiniOneRec-based training pipeline, dataset preprocessing, chronological split, SID construction, SID-level evaluation protocol, constrained beam decoding setup, and SID collision statistics. Appendix B gives the full train-only item–item graph construction procedure and graph hyperparameters. Appendix C specifies graph-derived task generation, prompt templates, task filtering rules, and task mixture sizes. Appendix D reports the Stage-1 SFT, Stage-2 SFT, InfoNCE, and GRPO hyperparameters used in the main experiments.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The supplementary material includes anonymized code and instructions for reproducing the main experimental results.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The paper specifies the experimental setting in the main text and provides full details in the appendices. The main text describes the evaluated Amazon benchmarks, metrics, baselines, and MiniOneRec evaluation protocol. Appendix A details preprocessing, chronological train/validation/test splits, SID construction, constrained beam decoding, and SID-level exact-match evaluation. Appendix B reports the train-only graph construction algorithm and graph hyperparameters. Appendix C describes task generation and prompt templates. Appendix D lists the Stage-1 SFT, Stage-2 SFT, InfoNCE, and GRPO training hyperparameters, including optimizer choices, learning rates, batch sizes, epochs, precision, and decoding settings.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: The Office HR@10 and NDCG@10 results for R-GRec are averaged over five training runs with different random seeds under a fixed data split, training configuration, and evaluation protocol. The Office mechanism analysis additionally reports a paired bootstrap 95% confidence interval for the TCR@10 improvement between the pre-RL SFT checkpoints of R-GRec and MiniOneRec, together with a 95% confidence interval for the popularity-adjusted difference. Both intervals exclude zero, supporting a positive difference in structural consistency. These confidence intervals concern the mechanism diagnostics and do not establish statistical significance for all HR/NDCG comparisons.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The paper includes a compute budget paragraph reporting the GPU type, GPU memory, number of GPUs, and wall-clock time for the main MiniOneRec-based training pipeline. It provides per-stage execution time for Stage-1 alignment, Stage-2 recommendation SFT, and GRPO fine-tuning, as well as an estimated GPU-hour cost for one full Industrial run.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The research uses public recommendation benchmarks and does not involve human-subject experiments, private user data collection, deception, or deployment affecting real users. The submission and supplementary materials are prepared to preserve reviewer anonymity, and the work is intended for offline evaluation of recommendation algorithms under standard benchmark protocols.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: The appendix includes a Broader Impacts section. It discusses potential positive impacts, including improved recommendation quality and graph-free inference, as well as possible negative impacts such as popularity-bias amplification, reduced exposure diversity, preference bubbles, unfair exposure across items or providers, inherited biases from historical interaction logs, and privacy and fairness considerations for deployment.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release a high-risk general-purpose model, image generator, scraped unsafe dataset, or other artifact with substantial misuse or dual-use risk. The experiments are conducted on public recommendation benchmarks in an offline evaluation setting, and the proposed method is a training-time supervision framework for SID-based recommendation rather than a broadly deployable generative model.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: The paper and appendix credit the existing assets used in the work, including the Amazon Review 2018 datasets, the MiniOneRec pipeline, and the Qwen2.5-0.5B backbone. The appendix states the applicable licenses or terms of use: MiniOneRec and Qwen2.5-0.5B are released under Apache-2.0, while the Amazon Review 2018 dataset source notes that the data owners do not provide a formal license and request research-purpose use. The experiments use these assets only for offline benchmark evaluation and follow the corresponding citation and usage requirements.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes].

    Justification: The supplementary material includes anonymized code and documentation for the new R-GRec components, including train-only item–item graph construction, graph-derived task generation, graph-instruction prompt templates, training scripts, evaluation scripts, and configuration files. The documentation describes how to reproduce the main experiments from public Amazon Review 2018 data and MiniOneRec-compatible preprocessing. No new human-subject dataset is collected, and no additional consent is required beyond the terms of the public benchmark data used for offline evaluation.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects. All experiments are conducted offline using public recommendation benchmark data.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing or research with human subjects. The experiments are offline evaluations on public recommendation benchmark data, so IRB approval or equivalent human-subject review is not applicable.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes].

Justification: The paper describes the use of a pretrained language model as the backbone of the SID-based generative recommender. In particular, Appendix D specifies Qwen2.5-0.5B as the base model and reports how it is adapted through Stage-1 SID alignment, Stage-2 recommendation SFT, and GRPO fine-tuning. The LLM is used as the generative recommendation backbone, while graph-derived supervision is introduced through training data and objectives rather than through an additional proprietary or undisclosed LLM component.