# UniRank: Unified List-wise Reranking via Confidence-Ordered Denoising

**Pengyue Jia**[1], **HailanYang**[2], **Shuchang Liu**[2], **Xiaobei Wang**[2], **Wanyu Wang**[1], **Xiang Li**[2], **Yongqi Liu**[2], **Kaiqiao Zhan**[2], **Kun Gai**[2], **Xiangyu Zhao**[1*]
[1]City University of Hong Kong, [2]Kuaishou Technology
jia.pengyue@my.cityu.edu.hk,xianzhao@cityu.edu.hk

## Abstract

List-wise reranking arranges a request-specific pool of candidate items into an ordered slate that maximizes user satisfaction. Existing generative rerankers fall into two paradigms: Autoregressive (AR) rerankers construct the slate left to right and capture inter-item dependencies in the exposure list, but they suffer from error propagation because early mistakes affect subsequent slots. Non-autoregressive (NAR) rerankers predict all slots in parallel and avoid error propagation, but they weaken inter-item interaction modeling under a slot independence assumption. This raises a central question: is there a unified architecture that combines the strengths of both paradigms and delivers stronger reranking performance? We answer this question with **UniRank**, *a unified list-wise reranking framework whose inference time variants recover AR and NAR rerankers as special cases*. UniRank integrates bidirectional slate modeling into an iterative denoising process and fills the most confident slot at each step. To instantiate this framework for reranking, we introduce the Task Grounded Diffusion Interface (TGD), which performs denoising in the (item, slot) space and restricts prediction to the request-specific candidate pool. TGD aggregates each item's semantic tokens into a single item embedding and scores each slot directly against the candidate pool. Experiments on Amazon Books, MovieLens-1M, and an industrial short video dataset show that UniRank consistently outperforms state-of-the-art baselines. Online A/B tests on a real-world industrial platform further validate its effectiveness, yielding significant improvements of +0.159% in average user app-time and +1.016% in share rate.

## 1 Introduction

Recommender systems on modern online content-sharing platforms typically follow a multi-stage cascade [1, 2] of retrieval [3, 4], ranking [5–8], and reranking [9]. As the final stage of this cascade, reranking [10–14] takes a request-specific local candidate set produced by earlier stages and arranges the items into an ordered slate for user exposure. Unlike retrieval and ranking, which often evaluate items independently, reranking must model *inter-item patterns*, including substitution, complementarity, and diversity [15]. For example, placing two near-identical items next to each other can reduce slate diversity, while placing complementary items together may improve user satisfaction. The central challenge in reranking is, therefore, to generate an ordered slate that captures these patterns while respecting the candidate pool returned by earlier stages.

Recent work frames reranking as a sequence generation problem $P(\mathbf{S}|u, \mathbf{C})$, where the model constructs an exposure slate $\mathbf{S}$ from the user context $u$ and a request-specific candidate pool $\mathbf{C}$. The majority follow two paradigms: *Autoregressive (AR) rerankers* [16–18] factorize the slate in a fixed left-to-right order and condition each item based on previously generated items. This factorization

*Corresponding author.

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

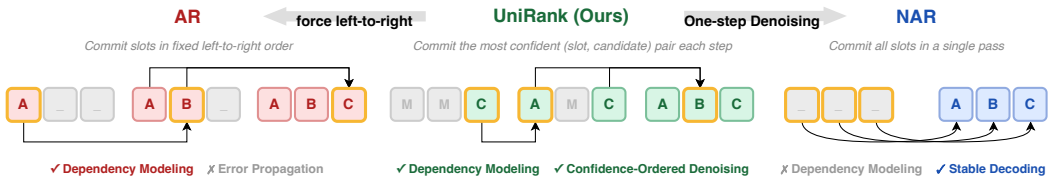


Figure 1: Comparison of AR, NAR, and UniRank reranking paradigms. AR reranking (left) decodes the slate in a fixed order, while NAR reranking (right) predicts all positions in parallel. UniRank (middle) uses confidence-ordered iterative denoising with bidirectional slate context; forcing the denoising order to follow slate positions recovers AR decoding, while using a single parallel denoising step recovers NAR decoding.

captures dependencies among exposed items, but the causal attention mask prevents each slot from using information from later slots, and an early mistake can propagate irreversibly to subsequent slots [19]. *Non-autoregressive (NAR) rerankers* [12, 20, 21] predict all slots in parallel and avoid this error propagation, but they usually rely on a slot-wise independence assumption that weakens dependency modeling within the exposure slate. A recent diffusion-based reranker [22] introduces iterative slate refinement, but it operates after an initial complete list has already been given. Thus, its performance largely depends on the initial slate and is likely to fall into local optima, where the inter-item dependency modeling introduces unstable refinements (**Appendix A**). As illustrated in Figure 1, existing generative rerankers therefore expose a tradeoff between **exposure dependency modeling** and **generation stability**.

This raises the central question of this paper: *can we design a unified architecture that combines the strengths of both paradigms, modeling bidirectional inter-item interactions across a slate of exposure while mitigating the generation instability caused by fixed-order error propagation?*

We answer this question with UniRank, a unified list-wise reranking framework that constructs slates via confidence-ordered iterative denoising. In other words, UniRank follows a slate-diffusion procedure in the item-slot space. Starting from an all-mask slate, UniRank uses bidirectional slate modeling to score candidate assignments for each unfilled position, commits the most confident position-item assignment, and repeats this process until the slate is complete. This design lets every prediction use the current slate context while avoiding a fixed left-to-right commitment order. Under different inference schedules, the same framework recovers both paradigms: 1) Forcing the denoising order to follow left-to-right gives an AR reranker, while 2) collapsing the process into all-slot parallel generation steps gives an NAR reranker. Thus, UniRank provides a unified inference view that connects AR decoding, NAR decoding, and confidence-ordered iterative decoding.

To instantiate UniRank for reranking, the diffusion process must also match the item-level modeling and candidate-constrained nature of the task. Mechanically, we introduce the Task Grounded Diffusion (TGD) interface, which contains two modules: 1) *Semantic Fusion Layer (SFL)* aggregates each item's semantic representations into a single item embedding, so each denoising position corresponds to one item rather than a sequence of tokens. 2) *Latent Pool Selection (LPS)* replaces vocabulary-level decoding with dot product scoring against the local candidate pool, so each prediction is made directly over legal candidates for the current request. Extensive experiments on Amazon Books, MovieLens-1M, and an industrial short video dataset show relative improvements of up to 5.4% in Precision, 4.4% in NDCG, and 6.4% in MAP over the strongest baseline. Online A/B tests on a real-world industrial platform further confirm real-world effectiveness, with +0.159% in user average app-time and +1.016% in share-rate. In summary, our main contributions are as follows:

- We present a unified reranking architecture, UniRank, that combines bidirectional slate modeling with confidence-ordered iterative denoising, and recovers AR and NAR rerankers as special cases.
- We introduce the Task Grounded Diffusion (TGD) Interface, consisting of the Semantic Fusion Layer (SFL) and the Latent Pool Selection (LPS), which instantiates diffusion reranking at the item level and within the request-specific candidate pool.
- We conduct experiments on three datasets, and the results show that UniRank outperforms state-of-the-art reranking baselines, with ablations validating the contribution of the proposed components.

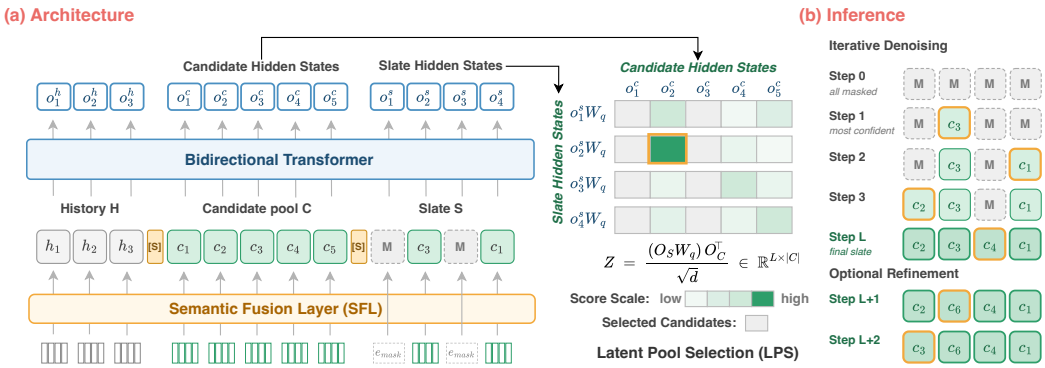


Figure 2: Overview of UniRank.

## 2 Methodology

### 2.1 Problem Setup

A reranking request consists of a user input $u$ and a candidate pool $\mathbf{C} = \{c_1, \ldots, c_{|\mathbf{C}|}\}$ provided by an upstream ranker/retriever. The reranker solves a conditional policy $p_\theta(\mathbf{S} \mid u, \mathbf{C})$ that produces an ordered slate $\mathbf{S} = [s_1, \ldots, s_L] \in \mathbf{C}^L$ of $L$ distinct items. Under sequential modeling, the user input $u$ is typically converted into a sequential representation $\mathbf{H}$ [23, 24], which may include but is not limited to user history and other profile features. Training seeks parameters $\theta$ that maximize the expected list-wise utility of the generated slate [25]:

$$\theta^\star \;=\; \arg\max_\theta \; \mathbb{E}_{(u,\mathbf{C})\sim\mathcal{D}} \, \mathbb{E}_{\mathbf{S}\sim p_\theta(\cdot\mid u,\mathbf{C})} \big[\, U(\mathbf{S}) \,\big], \tag{1}$$

where $\mathcal{D}$ is the distribution of requests and $U$ is a slate-level metric (e.g., NDCG@L or sum of clicks) that scores the full ordered slate $\mathbf{S}$.

### 2.2 UniRank: Unified Confidence-Ordered Denoising

In UniRank, we instantiate $p_\theta$ with a masked discrete diffusion model whose denoising backbone attends jointly to the history, the candidate pool, and a partially filled slate, modeling reranking as masked discrete diffusion over slate positions (Figure 2).

Formally, let $\mathbf{x}_0 = (\mathbf{x}_0^{(1)}, \ldots, \mathbf{x}_0^{(L)})$ denote the ground truth slate, where each $\mathbf{x}_0^{(j)}$ is the item representation assigned to slot $j$. Given a mask ratio $\alpha \in (0, 1]$, we draw a subset $\mathcal{M} \subseteq \{1, \ldots, L\}$ of size $\lceil \alpha L \rceil$ uniformly at random and form the corrupted slate $\mathbf{x}_\alpha$ by

$$\mathbf{x}_\alpha^{(j)} \;=\; \begin{cases} \mathbf{e}_{\text{mask}}, & j \in \mathcal{M}, \\ \mathbf{x}_0^{(j)}, & j \notin \mathcal{M}, \end{cases} \tag{2}$$

where each slot is either replaced by the learnable mask embedding ($\mathbf{e}_{\text{mask}} \in \mathbb{R}^d$) or kept as its original item representation. Masking is applied only to the output slate $\mathbf{S}$, while the user history $\mathbf{H}$ and candidate pool $\mathbf{C}$ remain clean throughout.

The reverse model $f_\theta$ is a shared $\mathbf{x}_0$ predictor parameterized by a stack of bidirectional Transformer blocks, implementing the distribution $p_\theta(\,\cdot \mid \mathbf{x}_\alpha, \mathbf{H}, \mathbf{C})$ that predicts the item generation probability (i.e., item confidence) at every masked slot (details in Section 2.3). We introduce a separator embedding $\mathbf{e}_{\text{sep}}$ to delimit the history, candidate, and slate in the input sequence $[\,\mathbf{H} \mid \mathbf{e}_{\text{sep}} \mid \mathbf{C} \mid \mathbf{e}_{\text{sep}} \mid \mathbf{x}_\alpha\,]$ under full self-attention. Since the random masking of Eq.(2) produces $\mathbf{x}_\alpha$ for all mask ratios $\alpha$, UniRank can learn a single shared denoiser across all denoising steps. This design enables full bidirectional attention over the slate at every prediction step, while deferring the commit order to inference time (e.g., committing the slot-item pair with the maximum confidence at each step).

### 2.3 Task Grounded Diffusion Interface

To align with the item-level and candidate-constrained nature of the reranking task, we introduce the Task Grounded Diffusion (TGD) interface, comprising the Semantic Fusion Layer (SFL) and the Latent Pool Selection (LPS) module.

#### 2.3.1 Semantic Fusion Layer

Following recent generative recommendation work [26–28], we represent each item by a semantic ID (SID) of $M$ discrete tokens $(\text{sid}_{i,1}, \ldots, \text{sid}_{i,M})$ drawn from $M$ parallel codebooks, sharing a token embedding table $\mathbf{E} \in \mathbb{R}^{V \times d}$ across all codebook positions. For each item $i$, SFL aggregates its $M$ SID token embeddings via concatenation and projection into a single item representation $\mathbf{e}_i$, so that each denoising position corresponds to exactly one item:

$$\mathbf{e}_i \;=\; \mathbf{W}_{\text{agg}} \left[ \mathbf{E}[\text{sid}_{i,1}] \,\|\, \cdots \,\|\, \mathbf{E}[\text{sid}_{i,M}] \right] + \mathbf{b}_{\text{agg}}, \qquad \mathbf{W}_{\text{agg}} \in \mathbb{R}^{d \times Md},\ \mathbf{b}_{\text{agg}} \in \mathbb{R}^{d}. \tag{3}$$

At each denoising step, the model input is $[\,\mathbf{H} \mid \mathbf{e}_{\text{sep}} \mid \mathbf{C} \mid \mathbf{e}_{\text{sep}} \mid \mathbf{x}_\alpha\,]$, where $\mathbf{x}_\alpha$ is the partially masked slate defined in Eq.(2).

#### 2.3.2 Latent Pool Selection

Latent Pool Selection replaces vocabulary-level decoding with direct selection from the request-specific candidate pool. Let $\mathbf{O}_S \in \mathbb{R}^{L \times d}$ collect the slate slot hidden states returned by $f_\theta$, and $\mathbf{O}_C \in \mathbb{R}^{|\mathbf{C}| \times d}$ collect the candidate hidden states from the same forward pass. We compute the slot-to-candidate score matrix as

$$\mathbf{Z} \;=\; \frac{(\mathbf{O}_S \mathbf{W}_q)\, \mathbf{O}_C^\top}{\sqrt{d}} \;\in\; \mathbb{R}^{L \times |\mathbf{C}|}, \tag{4}$$

where $\mathbf{W}_q \in \mathbb{R}^{d \times d}$ is a learnable query projection. The distribution for slot $j$'s item generation is $p_\theta(\cdot \mid \mathbf{x}_\alpha, \mathbf{H}, \mathbf{C}) = \text{softmax}(\mathbf{Z}_{j,:})$, whose support is exactly the current candidate pool. This design makes candidate constraints a part of the model architecture rather than a post hoc correction process.

### 2.4 Optimization

We train the model with a denoising surrogate that predicts target items at masked slots, optimizing a label-reweighted cross-entropy over masked positions. At each training step, we draw a mask count $n \sim \text{Uniform}\{1, \ldots, L\}$ and mask $n$ slate slots uniformly at random following Eq.(2); the remaining $L - n$ slots keep their ground-truth item embeddings. The induced mask ratio $\alpha = n/L$ is therefore uniformly distributed across training steps. For each masked slot $j$, let $\ell_j \in \{1, \ldots, |\mathbf{C}|\}$ be the index of the ground-truth item $\mathbf{x}_0^{(j)}$ in the candidate pool $\mathbf{C}$. The per-example loss is

$$\mathcal{L} \;=\; \frac{1}{L} \sum_{j \in \mathcal{M}} \frac{r_j}{\alpha}\, \text{CE}\Big( \text{softmax}(\mathbf{Z}_{j,:}),\ \ell_j \Big), \tag{5}$$

where $\mathcal{M}$ is the set of masked slots and $\mathbf{Z}_{j,:}$ is the corresponding row of Eq.(4). The factor $1/\alpha$ keeps the expected per-slot gradient contribution invariant to the sampled mask ratio. The factor $r_j$ provides label-aware reweighting for each item in the slate, allowing the same framework to incorporate richer feedback signals [29, 30] without changing the denoising architecture:

- When $r_j \equiv 1$, the loss reduces to the standard denoising objective.
- When the slate utility $U(\mathbf{S})$ is not slot-decomposable, setting $r_j \equiv U(\mathbf{S})/L$ recovers a utility-rewarded policy gradient.
- When $U$ is linear, i.e., $U = \sum_j r_j$, Eq.(5) simultaneously optimizes the average slate utility ($\frac{U}{\alpha L} \log p_\theta(\ell_j \mid \cdot)$) and an item-wise advantage term ($\frac{A_j}{\alpha} \log p_\theta(\ell_j \mid \cdot)$, where $A_j = r_j - U/L$), ensuring better alignment of item-level preferences with the context-aware model $p_\theta$.

### 2.5 Inference

At inference time, UniRank starts from a fully masked slate and iteratively commits items to slots via the shared denoiser, as illustrated in Figure 2(b). The inference behavior is primarily determined by the *commit policy*: how many ⟨slot, candidate⟩ pairs to commit per step and whether the slot order is constrained across steps.

**Default UniRank policy.** At step $k$, we obtain the score matrix $\mathbf{Z}^{(k)}$ from Eq.(4), mask out the filled slots and selected candidates, and select the highest-confidence slot-candidate pair:

$$(j_k^\star,\, i_k^\star) \;=\; \arg\max_{(j,\,i)} \; \mathbf{Z}^{(k)}_{j,\,i}, \tag{6}$$

writing $c_{i_k^\star}$ into slot $j_k^\star$, for $k = 1, \ldots, L$. This confidence-ordered strategy makes easy decisions first and leaves uncertain slots for later rounds, when more slate context is available. After the $L$ denoising steps, an optional $T \geq 0$ refinement steps follow, where each step remasks the lowest-confidence item and reapplies the same denoiser (see **Appendix A**).

**Relation to existing paradigms.** If the commit policy fills one pre-specified slot per step in left-to-right order, the framework recovers AR-style serial decoding. If it fills all slots in a single denoising pass, it recovers NAR-style parallel decoding. In all cases, UniRank uses the same bidirectional denoiser and candidate-constrained interface (only the commit policy changes). More details are given in **Appendix B**.

## 3 Experiments

**Datasets and Evaluation Metrics.** We conduct experiments on two public benchmarks and one industrial dataset: Amazon Books [31], MovieLens-1M [32], and Industry. Industry is a proprietary log collected from a real-world short-video platform, containing anonymized user interaction sequences. Dataset statistics are reported in **Appendix C**. For each dataset, 80% of the data is used for training and 20% for testing. Each data sample consists of a history of $|H| = 50$ most recent interactions, a candidate pool of $|C| = 50$ items, and the reranker output lists of length $L = 6$ along with binary labels representing user feedback. We report four list-wise metrics computed on the top-$L$ positions of the predicted list: Precision@$L$, NDCG@$L$, MAP@$L$, and F1@$L$. All metrics are higher-is-better and are averaged over requests in the test set.

**Baselines.** We consider nine reranking baselines spanning two families that dominate the recent literature. The *Generator-only (G-only)* family trains a single generator to produce the output list end-to-end: DNN [1], DLCM [10], PRM [12], GoalRank [18], and GloRank [33]. The *Generator-Evaluator (G-E)* family decouples list generation from list scoring so that a separately trained evaluator can rescore candidate lists produced by the generator: PIER [34], NAR4Rec [20], G-$n$ [14] with $n \in \{3, 20, 100\}$, and DCDR [22]. Full descriptions for baselines are provided in **Appendix D**.

**Implementation Details.** For Semantic ID construction, item text descriptions are encoded by Qwen3-4B into dense embeddings, reduced by PCA to `128` dimensions, and then quantized by a multi-head VQ-VAE [35] with `M=4` disjoint codebooks of `256` entries each. The backbone is a stack of `4` bidirectional Transformer blocks with a hidden width of `d=256`, `4` attention heads, and a dropout rate of `0.1`. We optimize the objective in Equation 5 with AdamW [36] at a peak learning rate of $3 \times 10^{-3}$, weight decay of `0.05`, batch size of `64`, and `bf16` mixed precision. At each training step, for each example, we draw a mask count $n \sim \mathrm{Uniform}\{1, \ldots, L\}$ and mask $n$ slate slots uniformly at random. We set the label-aware weight $r_j = 1$ for items with positive feedback (click or long watch, depending on the dataset) and $r_j = 0$ for the remaining slots. In inference, we run the default UniRank confidence-ordered denoising policy for $L$ steps, committing one item per step.

### 3.1 Main Results

Table 1 reports the overall comparison between UniRank and the reranking baselines on three datasets. We summarize three observations below:

- **UniRank achieves the best overall performance.** It ranks first on all four metrics for three datasets. The gains are especially clear on Amazon Books, where UniRank improves over the strongest baseline GloRank by 5.43% in Precision, 4.35% in NDCG, 6.44% in MAP, and

Table 1: Overall performance comparison on Amazon Books, MovieLens-1M, and Industry. All metrics are reported as percentages with $L = 6$. The **best** and second-best results are highlighted. The last two rows mark our method and its relative improvement over the strongest baseline.

| | | Amazon Books | | | | MovieLens-1M | | | | Industry | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Family | Method | P@$L$ | NDCG@$L$ | MAP@$L$ | F1@$L$ | P@$L$ | NDCG@$L$ | MAP@$L$ | F1@$L$ | P@$L$ | NDCG@$L$ | MAP@$L$ | F1@$L$ |
| G-only | DNN | 60.28 | 69.61 | 58.58 | 62.45 | 56.86 | 70.30 | 59.28 | 62.16 | 32.80 | 51.26 | 41.02 | 38.97 |
| | DLCM | 66.80 | 75.88 | 65.39 | 69.28 | 62.31 | 73.87 | 63.82 | 67.96 | 47.86 | 73.03 | 63.16 | 56.89 |
| | PRM | 67.86 | 76.88 | 66.44 | 70.42 | 60.09 | 72.85 | 62.21 | 65.51 | 47.89 | 67.09 | 54.59 | 56.88 |
| | GoalRank | 80.35 | 84.88 | 77.91 | 83.44 | 73.56 | 83.43 | 76.16 | 80.15 | 60.16 | 88.08 | 82.06 | 71.48 |
| | GloRank | 83.75 | 90.08 | 85.10 | 86.97 | 75.79 | 87.56 | 81.19 | 82.57 | 62.56 | 90.15 | 84.84 | 74.32 |
| G-E | PIER | 71.14 | 80.22 | 71.62 | 73.74 | 62.74 | 75.99 | 65.98 | 68.74 | 56.69 | 80.15 | 70.52 | 67.52 |
| | NAR4Rec | 70.08 | 79.46 | 70.69 | 72.66 | 62.81 | 75.01 | 65.42 | 68.31 | 51.93 | 65.16 | 53.13 | 61.56 |
| | G-3 | 68.76 | 76.36 | 65.82 | 71.33 | 55.51 | 67.39 | 55.52 | 55.51 | 48.53 | 67.38 | 54.89 | 57.67 |
| | G-20 | 72.99 | 78.68 | 68.66 | 75.72 | 58.66 | 69.86 | 58.60 | 64.18 | 51.58 | 69.29 | 57.26 | 61.08 |
| | G-100 | 77.21 | 82.15 | 73.78 | 80.09 | 60.64 | 70.97 | 59.93 | 66.29 | 53.21 | 70.67 | 58.80 | 63.06 |
| | DCDR | 80.56 | 86.73 | 80.33 | 83.66 | 73.04 | 84.61 | 77.17 | 79.60 | 50.02 | 62.48 | 49.26 | 59.44 |
| | **UniRank(Ours)** | **88.30** | **94.00** | **90.58** | **91.70** | **76.68** | **88.65** | **82.64** | **83.59** | **63.54** | **91.34** | **86.53** | **75.49** |
| | *Rel. Improv.* | +5.43% | +4.35% | +6.44% | +5.44% | +1.44% | +1.24% | +1.79% | +1.24% | +1.57% | +1.32% | +1.99% | +1.57% |

5.44% in F1. On MovieLens-1M and Industry, UniRank also remains consistently above the best baseline method on every metric. This demonstrates the superiority of UniRank and verifies the effectiveness of the unified framework.

- **Both AR and NAR reranking paradigms have intrinsic bottlenecks.** Existing AR rerankers such as GoalRank and GloRank can model inter-item dependencies within the exposure slate, achieving superior performance within baselines. Yet, they are constrained by unidirectional causal attention and suffer from error propagation during decoding, resulting in suboptimal performance compared to UniRank. Existing NAR rerankers (DNN, DLCM, PRM, NAR4Rec) ignore the exposure dependency modeling and thus generally perform worse than AR rerankers in practice. DCDR uses diffusion-based refinement and achieves inter-item relation modeling, but its performance is also suboptimal compared to UniRank due to its inherent dependency on the initial slate quality.
- **Larger evaluator search alone does not remove the gap to UniRank.** Empirically, NAR rerankers perform better in the G-E paradigm (PIER and NAR4Rec) than in the G-only paradigm (DNN, DLCM, and PRM). This shows that the limitation caused by ignoring the exposure dependency modeling could be partially alleviated by exploring multiple slates as candidates, then using a slate-wise evaluator to select the best one. Within the G-$n$ variants, enlarging the slate candidate pool from G-3 to G-20 and G-100 steadily improves NDCG on Amazon Books from 76.36 to 78.68 and 82.15. However, even the strongest variant G-100 remains well below UniRank at 94.00. Similar gaps also appear on MovieLens-1M and Industry. This result suggests that better evaluator side search cannot efficiently compensate for weaker slate generation.

It is worth noting that we also provide simulator-based experimental results [37] in **Appendix E** from a complementary perspective.

### 3.2 Ablation Study

We conduct an ablation study to examine the contribution of each component in UniRank. The variants fall into two groups:

- **Input Components.** (1) w/o SID, replacing SID with item embeddings; (2) w/o Hist. Padding, removing history padding.
- **Model Architecture.** (1) w/o SFL, removing the Semantic Fusion Layer and feeding raw token embeddings into the Transformer; (2) w/o Iter. Decoding, collapsing the iterative denoising process into a single parallel step; and (3) w/o LPS, replacing Latent Pool Selection with direct SID token generation over the entire vocabulary.

Table 2 reports the results, and we can draw the following conclusions:

**Input Components.** (1) *Semantic ID provides marginal but consistent gains.* Replacing SID with raw item IDs (w/o SID) leads to consistent drops across all metrics. Semantic IDs encode structural item features through quantization, providing richer signals than opaque item indices. (2) *History*

Table 2: Ablation study on Amazon Books and MovieLens-1M. All metrics are reported as percentages with $L = 6$. The full model is shown at the top for reference.

| | Amazon Books | | | | MovieLens-1M | | | |
|---|---|---|---|---|---|---|---|---|
| Method | P@$L$ | NDCG@$L$ | MAP@$L$ | F1@$L$ | P@$L$ | NDCG@$L$ | MAP@$L$ | F1@$L$ |
| UniRank (full) | **88.30** | **94.00** | **90.58** | **91.70** | **76.68** | **88.65** | **82.64** | **83.59** |
| *Input Components* | | | | | | | | |
| w/o SID | 88.27 | 93.84 | 90.38 | 91.65 | 76.33 | 88.17 | 82.08 | 83.19 |
| w/o Hist. Padding | 88.25 | 93.78 | 90.32 | 91.64 | 76.35 | 88.41 | 82.38 | 83.23 |
| *Model Architecture* | | | | | | | | |
| w/o SFL | 81.13 | 88.31 | 82.24 | 84.25 | 75.21 | 86.69 | 79.94 | 81.96 |
| w/o Iter. Decoding | 88.20 | 93.89 | 90.42 | 91.59 | 76.22 | 88.29 | 82.16 | 83.08 |
| w/o LPS | 87.24 | 92.93 | 89.07 | 90.61 | 75.12 | 87.28 | 80.83 | 81.89 |

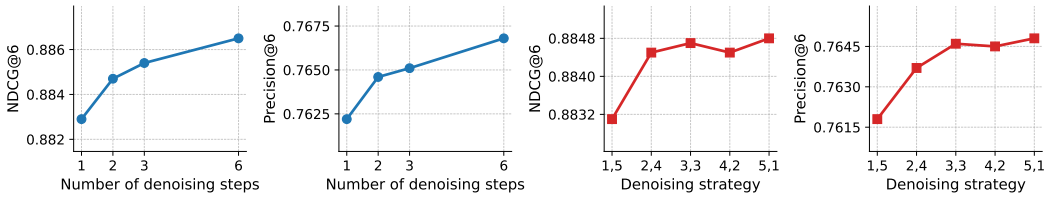


Figure 3: Effect of Denoising Parameters.

*padding stabilizes the input context.* Removing history padding (w/o Hist. Padding) leads to small but consistent drops, particularly in MAP. History padding ensures that the Transformer attends over a fixed-length user interaction context, reducing variance introduced by requests with shorter histories.

**Model Architecture.** (1) *The Semantic Fusion Layer (SFL) is essential for aligning item representations with the denoising process.* Removing SFL (w/o SFL) causes the largest drops across all settings, with Precision on Amazon Books falling from 88.30 to 81.13 and MAP dropping from 90.58 to 82.24. This breaks the alignment between denoising positions and items, making it substantially harder for the model to perform item-level scoring during the diffusion process. As a consequence, we suggest that practitioners should favor item-level modeling rather than token SID-level modeling when using diffusion in the reranking task. (2) *Iterative decoding provides consistent improvements over single-step generation.* A single parallel generation step (w/o Iter. Decoding) corresponds to the NAR extreme of UniRank, which performs slightly worse than the iterative decoding strategy due to the downgraded modeling of exposure dependency in the slate. Fortunately, the performance drop is not severe; for systems operating under strict latency constraints, UniRank w/o Iterative Decoding remains a highly competitive choice, as it avoids the computational overhead of the iterative generation process. (3) *Latent Pool Selection (LPS) is important for candidate-constrained generation.* Replacing LPS with direct SID token generation (w/o LPS) leads to notable drops, for example from 88.30 to 87.24 in Precision and from 90.58 to 89.07 in MAP on Amazon Books. Without LPS, the model must produce valid SIDs through a large shared vocabulary rather than selecting directly from the local candidate pool, which introduces errors from out-of-pool generation and weakens item comparison within a request.

### 3.3 Denoising Parameter Analysis

As shown in Figure 3, we investigate how the number of denoising steps and the denoising strategy affect the performance. Full results across all four metrics are provided in **Appendix F**.

**Number of denoising steps.** We vary the number of denoising steps among $1, 2, 3, 6$, where a step count of 1 means the model generates the entire list in a single inference pass, and a step count of 6 means each step generates exactly one item. For each step count $K$, we use an equal allocation that commits $L/K$ items per step. Both NDCG and Precision improve consistently as the step count increases. This behavior resembles test-time scaling in large language models: each additional inference step allows the model to condition on items already placed in prior steps, providing stronger contextual signals when resolving positions with lower prediction confidence.

**Denoising strategy.** We fix the number of denoising steps at 2 and vary the allocation of items across the two steps. The label $a, b$ denotes generating $a$ items in the first step and $b$ items in the second step. Performance generally improves as more items are assigned to the first step. This is consistent with the confidence-based ordering underlying our denoising process: the first step resolves the positions with the highest confidence, where the model predictions are already reliable and additional inference budget yields limited gain. The remaining positions, denoised in the second step, have lower confidence and benefit more from the contextual information established in the first step.

### 3.4 Scaling Analysis

We examine how model size affects recommendation performance to identify the effective operating range of UniRank. We evaluate five model configurations on both Books and MovieLens-1M: XS (0.29M), S (0.88M), M (5.08M, default), L (11.01M), and XL (27.59M) parameters. The complete results are provided in **Appendix G**. As shown in Figure 4, performance improves consistently from XS to M on both datasets, then declines as model size increases further. This rise-then-fall pattern suggests that the training data volume imposes a practical capacity ceiling: models beyond the M configuration have sufficient parameters to overfit the available training signal, which leads to degraded generalization. The 5.08M default configuration achieves the best trade-off between model expressiveness and data efficiency.

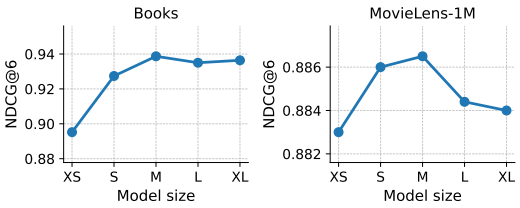

Figure 4: Scaling analysis with five configurations.

### 3.5 Online Experiments

To evaluate UniRank in a real-world industrial environment, we conduct an online A/B test on a short-video platform with billion-scale daily user interactions. The production system follows a multi-stage recommendation pipeline, where retrieval and ranking first produce a candidate pool, and the reranking stage selects the final exposed slate. In this deployment, the reranking stage follows the same 50-to-6 setting as our offline evaluation. The resulting permutation space contains nearly $1.5 \times 10^{10}$ possible slates, making exhaustive enumeration impractical.

**Experimental Setup.** The control system is the deployed reranking system. Its reranker follows a G-$n$-style [14], where multiple generators produce candidate slates and an evaluator selects the final slate. The generator module is mainly based on a GoalRank-style model [18]. In the treatment system, we keep the online pipeline unchanged and replace the generator module in the reranking stage with UniRank. This design isolates the effect of UniRank under the same retrieval, ranking, and evaluator components.

Table 3: Online A/B test results.

| Category | Metric | Rel. Improv. |
|---|---|---|
| Platform | Overall app-time | +0.146%* |
| | User avg. app-time | +0.159%* |
| | Overall Realshow | +0.273%* |
| | Long term DAU | +0.085% |
| Interaction | Like-rate | +0.588%* |
| | Follow-rate | +0.842%* |
| | Comment-rate | +0.285% |
| | Share-rate | +1.016%* |
| | Hate UV (↓ better) | -2.038% |
| | Effective interests | +0.082% |

**Traffic Split and Test Period.** We randomly allocate 10% of online traffic to the control group and another 10% to the treatment group, ensuring comparable user distributions across both buckets. The experiment runs for 7 consecutive days. This duration allows us to measure both statistical significance and cross-day stability under normal traffic changes.

**Evaluation Metrics.** We report both platform-level metrics and user interaction metrics. The platform metrics include overall app-time, user average app-time, overall Realshow, and long-term DAU, reflecting the system's impact on broad usage patterns and platform health. The interaction metrics include like-rate, follow-rate, comment-rate, share-rate, hate UV, and effective interests, capturing fine-grained user feedback on recommended content. Together, these metrics assess whether the reranker improves user engagement while maintaining stable long-term platform behavior.

**Main Results.** Table 3 reports the relative improvements of UniRank over the control system. Results marked with an asterisk (*) are statistically significant ($p<0.05$). The results show that replacing the GoalRank-style generator with UniRank improves the online reranking system in practice. This confirms that the proposed UniRank framework can be deployed in a production multi-stage recommender system and can provide gains in a real-world environment.

# 4 Related Work

## 4.1 Reranking in Recommender Systems

Reranking is the final stage of a modern multi-stage recommender pipeline, and it determines the list that is ultimately shown to the user [13]. Existing neural rerankers can be broadly grouped into two paradigms: Generator-only and Generator-Evaluator. Generator-only methods use a single model to produce the slate directly [10, 16, 12, 18]. DLCM [10] refines pointwise scores with a recurrent model that captures full list context; Seq2Slate [16] formulates slate construction as autoregressive decoding with a pointer network; PRM [12] replaces recurrent encoding with self-attention so that each candidate can interact with all other candidates in one pass; and GoalRank [18] revisits this paradigm with a large generator trained by group relative optimization. Generator-Evaluator methods [34, 20, 14] separate list proposal from list scoring. PIER [34] combines a SimHash-based permutation generator with a context-aware evaluator, NAR4Rec [20] generates slot assignments in parallel with unlikelihood training and contrastive decoding, and G-$n$ [14] uses multiple complementary generators to enlarge the candidate list space. Both paradigms still rely on a fixed decoding order, which is either autoregressive and vulnerable to error propagation or fully parallel and therefore limited in modeling dependencies among target positions. UniRank instead formulates reranking as masked discrete diffusion over the slate and commits slot to candidate assignments in order of their confidence, which unifies list generation and refinement within a unified framework and yields bidirectional slate attention together with an adaptive any-order decoding schedule. The closest prior work to UniRank is DCDR [22], which is equivalent to the refinement stage of UniRank applied to an externally given initial list. As a result, its output is anchored to that initial list and its quality is tightly bounded by the quality of the starting point.

## 4.2 Generative Methods in Recommender Systems

Generative recommendation reformulates recommendation as a sequence generation problem over learned item identifiers, where the model predicts the next item token by token or through iterative denoising instead of assigning a pointwise matching score [38]. Progress in this direction can be organized along three axes. (1) Tokenization. Recent work replaces atomic item identifiers with content-informed representations. P5 [39] converts user item interactions, item metadata, and related signals into natural language sequences within a unified text-to-text framework, while TIGER [26] learns hierarchical semantic identifiers from pretrained item embeddings through residual vector quantization. (2) Architecture. Encoder-decoder models such as M6-Rec [40] use a unified pre-trained language model to support retrieval, ranking, and explanation, whereas decoder-only models emphasize industrial scaling, with HSTU [41] proposing a high-throughput sequential transduction architecture for long and dynamic user histories. More recently, LLaDA-Rec [35] replaces left-to-right decoding with a masked discrete diffusion backbone [27] that generates the semantic identifier of the next item. (3) Optimization. Beyond next token prediction, TallRec [42] shows that instruction tuning with small amounts of recommendation data can improve few-shot recommendation and cross-domain transfer. In contrast to these methods, which generate the next item from a global item vocabulary conditioned on user history, UniRank targets the reranking setting, where the output must be selected from a small request-specific candidate pool.

# 5 Conclusion

We propose UniRank, a unified reranking framework that formulates slate generation as confidence-ordered denoising, bridging the trade-off between dependency modeling and generation stability in existing AR and NAR rerankers. We further introduce the Task Grounded Diffusion Interface, which aligns diffusion-based generation with the reranking task. Experiments on three offline datasets and an online test show that UniRank consistently outperforms strong reranking baselines.

## Acknowledgements

This research was partially supported by National Natural Science Foundation of China (No.62502404), Hong Kong Research Grants Council (Research Impact Fund No.R1015-23, Collaborative Research Fund No.C1043-24GF, RGC Research Fellow Scheme No.RFS2627-1S03, General Research Fund No.11218325, No.11212926), Institute of Digital Medicine of City University of Hong Kong (No.9229503), and Kuaishou (CCF-Kuaishou Large Model Explorer Fund No.2025008, Kuaishou University Cooperation Project).

## References

[1] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In Shilad Sen, Werner Geyer, Jill Freyne, and Pablo Castells, editors, *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 191–198. ACM, 2016.

[2] Qi Liu, Kai Zheng, Rui Huang, Wuchao Li, Kuo Cai, Yuan Chai, Yanan Niu, Yiqun Hui, Bing Han, Na Mou, Hongning Wang, Wentian Bao, Yunen Yu, Guorui Zhou, Han Li, Yang Song, Defu Lian, and Kun Gai. Recflow: An industrial full flow recommendation dataset. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[3] Pengyue Jia, Yiding Liu, Xiangyu Zhao, Xiaopeng Li, Changying Hao, Shuaiqiang Wang, and Dawei Yin. MILL: mutual verification with large language models for zero-shot query expansion. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 2498–2518. Association for Computational Linguistics, 2024.

[4] Pengyue Jia, Yiding Liu, Xiaopeng Li, Xiangyu Zhao, Yuhao Wang, Yantong Du, Xiao Han, Xuetao Wei, Shuaiqiang Wang, and Dawei Yin. G3: an effective and adaptive framework for worldwide geolocalization using large multi-modality models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[5] Pengyue Jia, Derong Xu, Xiaopeng Li, Zhaocheng Du, Xiangyang Li, Yichao Wang, Yuhao Wang, Qidong Liu, Maolin Wang, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. Bridging relevance and reasoning: Rationale distillation in retrieval-augmented generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, volume ACL 2025 of *Findings of ACL*, pages 4242–4256. Association for Computational Linguistics, 2025.

[6] Pengyue Jia, Seongheon Park, Song Gao, Xiangyu Zhao, and Sharon Li. Georanker: Distance-aware ranking for worldwide image geolocalization. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

[7] Zichuan Fu, Xiangyang Li, Chuhan Wu, Yichao Wang, Kuicai Dong, Xiangyu Zhao, Mengchen Zhao, Huifeng Guo, and Ruiming Tang. A unified framework for multi-domain CTR prediction via large language models. *ACM Trans. Inf. Syst.*, 43(5):117:1–117:33, 2025.

[8] Yuhao Wang, Ha Tsz Lam, Yi Wong, Ziru Liu, Xiangyu Zhao, Yichao Wang, Bo Chen, Huifeng Guo, and Ruiming Tang. Multi-task deep recommender systems: A survey. *CoRR*, abs/2302.03525, 2023.

[9] Weiwen Liu, Yunjia Xi, Jiarui Qin, Fei Sun, Bo Chen, Weinan Zhang, Rui Zhang, and Ruiming Tang. Neural re-ranking in multi-stage recommender systems: A review. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 5512–5520. ijcai.org, 2022.

[10] Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. Learning a deep listwise context model for ranking refinement. In Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz, editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 135–144. ACM, 2018.

[11] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan, editors, *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 95–103. ACM, 2018.

[12] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, and Dan Pei. Personalized re-ranking for recommendation. In Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk, editors, *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, pages 3–11. ACM, 2019.

[13] Weiwen Liu, Yunjia Xi, Jiarui Qin, Fei Sun, Bo Chen, Weinan Zhang, Rui Zhang, and Ruiming Tang. Neural re-ranking in multi-stage recommender systems: A review. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 5512–5520. ijcai.org, 2022.

[14] Hailan Yang, Zhenyu Qi, Shuchang Liu, Xiaoyu Yang, Xiaobei Wang, Xiang Li, Lantao Hu, Han Li, and Kun Gai. Comprehensive list generation for multi-generator reranking. In Nicola Ferro, Maria Maistro, Gabriella Pasi, Omar Alonso, Andrew Trotman, and Suzan Verberne, editors, *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2025, Padua, Italy, July 13-18, 2025*, pages 2298–2308. ACM, 2025.

[15] Derong Xu, Yi Wen, Pengyue Jia, Yingyi Zhang, Wenlin Zhang, Yichao Wang, Huifeng Guo, Ruiming Tang, Xiangyu Zhao, Enhong Chen, and Tong Xu. From single to multi-granularity: Toward long-term memory association and selection of conversational agents. In Carl Vondrick, Bharath Hariharan, Colin Raffel, Lerrel Pinto, Diyi Yang, and Aleksandra Faust, editors, *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. proceedings.iclr.cc, 2026.

[16] Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. Seq2slate: Re-ranking and slate optimization with rnns. *arXiv preprint arXiv:1810.02019*, 2018.

[17] Shuchang Liu, Qingpeng Cai, Zhankui He, Bowen Sun, Julian J. McAuley, Dong Zheng, Peng Jiang, and Kun Gai. Generative flow network for listwise recommendation. In Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye, editors, *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 1524–1534. ACM, 2023.

[18] Kaike Zhang, Xiaobei Wang, Shuchang Liu, Hailan Yang, Xiang Li, Lantao Hu, Han Li, Qi Cao, Fei Sun, and Kun Gai. Goalrank: Group-relative optimization for a large ranking model. In Carl Vondrick, Bharath Hariharan, Colin Raffel, Lerrel Pinto, Diyi Yang, and Aleksandra Faust, editors, *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. proceedings.iclr.cc, 2026.

[19] Weiqin Yang, Bohao Wang, Zhenxiang Xu, Jiawei Chen, Shengjia Zhang, Jingbang Chen, Canghong Jin, and Can Wang. BEAR: towards beam-search-aware optimization for recommendation with large language models. In *The 49st International ACM SIGIR Conference on*

*Research & Development in Information Retrieval, SIGIR 2026, Melbourne, VIC, Australia, July 20-24, 2026*. ACM, 2026.

[20] Yuxin Ren, Qiya Yang, Yichun Wu, Wei Xu, Yalong Wang, and Zhiqiang Zhang. Non-autoregressive generative models for reranking recommendation. In Ricardo Baeza-Yates and Francesco Bonchi, editors, *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 5625–5634. ACM, 2024.

[21] Wenyu Mao, Shuchang Liu, Hailan Yang, Xiaobei Wang, Xiaoyu Yang, Xu Gao, Xiang Li, Lantao Hu, Han Li, Kun Gai, An Zhang, and Xiang Wang. Denoising neural reranker for recommender systems. In Carl Vondrick, Bharath Hariharan, Colin Raffel, Lerrel Pinto, Diyi Yang, and Aleksandra Faust, editors, *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. proceedings.iclr.cc, 2026.

[22] Xiao Lin, Xiaokai Chen, Chenyang Wang, Hantao Shu, Linfeng Song, Biao Li, and Peng Jiang. Discrete conditional diffusion for reranking in recommendation. In Tat-Seng Chua, Chong-Wah Ngo, Roy Ka-Wei Lee, Ravi Kumar, and Hady W. Lauw, editors, *Companion Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, Singapore, May 13-17, 2024*, pages 161–169. ACM, 2024.

[23] Chengxi Li, Yejing Wang, Qidong Liu, Xiangyu Zhao, Wanyu Wang, Yiqi Wang, Lixin Zou, Wenqi Fan, and Qing Li. Strec: Sparse transformer for sequential recommendations. In Jie Zhang, Li Chen, Shlomo Berkovsky, Min Zhang, Tommaso Di Noia, Justin Basilico, Luiz Pizzato, and Yang Song, editors, *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18-22, 2023*, pages 101–111. ACM, 2023.

[24] Ziwei Liu, Qidong Liu, Yejing Wang, Wanyu Wang, Pengyue Jia, Maolin Wang, Zitao Liu, Yi Chang, and Xiangyu Zhao. SIGMA: selective gated mamba for sequential recommendation. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 12264–12272. AAAI Press, 2025.

[25] Shuchang Liu, Qingpeng Cai, Bowen Sun, Yuhao Wang, Ji Jiang, Dong Zheng, Peng Jiang, Kun Gai, Xiangyu Zhao, and Yongfeng Zhang. Exploration and regularization of the latent action space in recommendation. In Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben, editors, *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, pages 833–844. ACM, 2023.

[26] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Mahesh Sathiamoorthy. Recommender systems with generative retrieval. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[27] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *CoRR*, abs/2502.09992, 2025.

[28] Yupeng Hou, Jiacheng Li, Ashley Shin, Jinsung Jeon, Abhishek Santhanam, Wei Shao, Kaveh Hassani, Ning Yao, and Julian J. McAuley. Generating long semantic ids in parallel for recommendation. In Luiza Antonie, Jian Pei, Xiaohui Yu, Flavio Chierichetti, Hady W. Lauw, Yizhou Sun, and Srinivasan Parthasarathy, editors, *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.2, KDD 2025, Toronto ON, Canada, August 3-7, 2025*, pages 956–966. ACM, 2025.

[29] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1040–1048. ACM, 2018.

[30] Chi Zhang, Rui Chen, Xiangyu Zhao, Qilong Han, and Li Li. Denoising and prompt-tuning for multi-behavior recommendation. In Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben, editors, *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, pages 1355–1363. ACM, 2023.

[31] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In Ricardo Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto, editors, *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 43–52. ACM, 2015.

[32] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[33] Pengyue Jia, Xiaobei Wang, Yingyi Zhang, Shuchang Liu, Yupeng Hou, Hailan Yang, Xu Gao, Xiaopeng Li, Yejing Wang, Julian McAuley, Xiang Li, Lantao Hu, Yongqi Liu, Kaiqiao Zhan, Han Li, Kun Gai, and Xiangyu Zhao. From local indices to global identifiers: Generative reranking for recommender systems via global action space, 2026.

[34] Xiaowen Shi, Fan Yang, Ze Wang, Xiaoxu Wu, Muzhi Guan, Guogang Liao, Yongkang Wang, Xingxing Wang, and Dong Wang. PIER: permutation-level interest-based end-to-end re-ranking framework in e-commerce. In Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye, editors, *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 4823–4831. ACM, 2023.

[35] Teng Shi, Chenglei Shen, Weijie Yu, Shen Nie, Chongxuan Li, Xiao Zhang, Ming He, Jianping Fan, and Jun Xu. Llada-rec: Discrete diffusion for parallel semantic ID generation in generative recommendation. In Hung-Yi Lee, Sunipa Dev, Junyi Jessy Li, and Desmond Elliott, editors, *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, EMNLP 2026, Budapest, Hungary, October 24-29, 2026*, pages 14152–14165. Association for Computational Linguistics, 2026.

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[37] Kesen Zhao, Shuchang Liu, Qingpeng Cai, Xiangyu Zhao, Ziru Liu, Dong Zheng, Peng Jiang, and Kun Gai. Kuaisim: A comprehensive simulator for recommender systems. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[38] Xiaopeng Li, Bo Chen, Junda She, Shiteng Cao, You Wang, Qinlin Jia, Haiying He, Zheli Zhou, Zhao Liu, Ji Liu, et al. A survey of generative recommendation from a tri-decoupled perspective: Tokenization, architecture, and optimization. 2025.

[39] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In Jennifer Golbeck, F. Maxwell Harper, Vanessa Murdock, Michael D. Ekstrand, Bracha Shapira, Justin Basilico, Keld T. Lundgaard, and Even Oldridge, editors, *RecSys '22: Sixteenth ACM Conference on Recommender Systems, Seattle, WA, USA, September 18 - 23, 2022*, pages 299–315. ACM, 2022.

[40] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. M6-rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084*, 2022.

[41] 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.

[42] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In Jie Zhang, Li Chen, Shlomo Berkovsky, Min Zhang, Tommaso Di Noia, Justin Basilico, Luiz Pizzato, and Yang Song, editors, *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18-22, 2023*, pages 1007–1014. ACM, 2023.

[43] Xiangyu Zhao, Long Xia, Lixin Zou, Hui Liu, Dawei Yin, and Jiliang Tang. Usersim: User simulation via supervised generativeadversarial network. In Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia, editors, *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 3582–3589. ACM / IW3C2, 2021.

[44] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 6111–6120. Association for Computational Linguistics, 2019.

[45] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 11305–11315. IEEE, 2022.

# Technical Appendix

## Table of Contents


**A Optional Refinement Stage** 16

**B Relation to Existing Paradigms** 17

**C Dataset Statistics** 17

**D Baseline Descriptions** 17

**E Simulator-based Evaluation** 18

**F Additional Analysis on Denoising Parameters** 18

**G Additional Scaling Results** 19

**H Limitations** 19

**I Relation to Masked Iterative Decoding** 20

**J Hypothesis-Driven Analysis of Decoding** 20

**K Effect of Slate Length on Iterative Decoding** 21

**L Inference Latency** 21

Table 4: Effect of optional refinement on MovieLens-1M. The insertion-only row is the default UniRank inference policy. Refinement reuses the same denoising model at inference time. Higher values are better for all metrics.

| Criterion | Steps | Precision@6 | NDCG@6 | MAP@6 | F1@6 |
|---|---|---|---|---|---|
| Insertion only | – | 0.7668 | 0.8865 | 0.8264 | 0.8359 |
| *prob* | 1 | 0.7668 | 0.8865 | 0.8267 | 0.8359 |
| *prob* | 2 | **0.7671** | 0.8868 | 0.8267 | **0.8363** |
| *prob* | 3 | **0.7671** | 0.8865 | 0.8267 | 0.8363 |
| *prob* | 4 | 0.7669 | 0.8867 | 0.8266 | 0.8360 |
| *prob* | 5 | 0.7665 | 0.8866 | 0.8266 | 0.8356 |
| *prob* | 6 | 0.7667 | 0.8860 | 0.8261 | 0.8358 |
| *delta* | 1 | 0.7666 | 0.8864 | 0.8265 | 0.8356 |
| *delta* | 2 | 0.7666 | 0.8863 | 0.8261 | 0.8357 |
| *delta* | 3 | 0.7666 | 0.8858 | 0.8257 | 0.8356 |
| *delta* | 4 | 0.7667 | 0.8862 | 0.8260 | 0.8358 |
| *delta* | 5 | 0.7670 | 0.8868 | 0.8268 | 0.8362 |
| *delta* | 6 | 0.7667 | **0.8869** | **0.8272** | 0.8358 |

# A Optional Refinement Stage

**Refinement procedure.** We describe the optional refinement stage used after the insertion process in UniRank. After inference finishes, the model has produced a complete slate $S = [s_1, \ldots, s_L]$, where each $s_j$ is one candidate item from $C$. Refinement reuses the same denoising model and the same Latent Pool Selection module, without additional training. At each refinement step, we reconsider one filled slot by a leave-one-out denoising query. For slot $j$, we replace its current item representation with $e_{\text{mask}}$ and keep all other filled slots unchanged, which gives a corrupted slate $S_{\setminus j}$. We then run the denoiser on $[\, H \mid e_{\text{sep}} \mid C \mid e_{\text{sep}} \mid S_{\setminus j} \,]$ and obtain a candidate score vector $Z_{j,:}$ for the masked slot, following Equation 4. The valid replacement set contains unused candidates in $C$ and the current item $s_j$, so the refined slate still contains distinct items, and the model can also keep the original choice.

We use two criteria to decide which slot to reconsider. Let

$$p_{j,i} = \mathrm{softmax}(Z_{j,:})_i \tag{7}$$

be the probability assigned to candidate $c_i$ when slot $j$ is masked, and let $i_j$ be the candidate index of the current item $s_j$. The *prob* criterion treats $p_{j,i_j}$ as the confidence of the current item and selects the slot with the lowest confidence. The *delta* criterion computes the replacement margin

$$\delta_j = \max_{i \in \mathcal{A}_j} p_{j,i} - p_{j,i_j}, \tag{8}$$

where $\mathcal{A}_j$ is the valid replacement set for slot $j$, and selects the slot with the largest margin. After selecting a slot, refinement writes the highest-scoring valid candidate into that slot. This procedure is repeated for a fixed number of refinement steps.

**Analysis.** Table 4 reports the refinement results on MovieLens-1M. Refinement gives small changes over the insertion-only result. With the *prob* criterion, two refinement steps give the best Precision@6 and F1@6, improving Precision@6 from 0.7668 to 0.7671 and F1@6 from 0.8359 to 0.8363. With the *delta* criterion, six refinement steps give the best NDCG@6 and MAP@6, improving NDCG@6 from 0.8865 to 0.8869 and MAP@6 from 0.8264 to 0.8272.

The results indicate that the insertion process already produces stable slates, while refinement can make inference-time corrections by rechecking filled slots under bidirectional slate context. The gains are not uniform across metrics or step counts. More refinement steps do not always improve the result, which suggests that repeatedly replacing filled items may also disturb correct earlier choices. We therefore treat refinement as an optional inference-time extension of UniRank, rather than a required part of the main method.

| Model | | #step | #token per step | Commit policy | Training objective |
|---|---|---|---|---|---|
| AR | Seq2Slate [16] | $L$ | 1 | $P(s_k \mid s_{<k}, u, C)$ | Step-wise sequence CE |
| | GoalRank [18] | $L$ | 1 | $P(s_k \mid s_{<k}, u, C)$ | Group-relative CE/KL |
| | GloRank [33] | $L \times M$ | 1 | $P(\text{sid}_k \mid \text{sid}_{<k}, u, C)$ | NTP CE + GRPO |
| NAR | NAR4Rec [20] | 1 | $L$ | Parallel generation | Unlikelihood + contrastive losses |
| | PRM [12] | 1 | $L$ | top-$L$ ranking | Click CE/NLL |
| | DNR [21] | 1 | $L$ | top-$L$ ranking | $\max P(Y \mid \tilde{Y}, u, C, S)$ |
| UniRank(Default) | | $L+T$ | 1 | $\arg\max_{(j,i)} Z_{j,i}$ | $\frac{r_j}{\alpha}\text{CE}(P(i_j \mid x_\alpha, u, C), y_j)$ |
| UniRank(NAR) | | 1 | $L$ | | |

Table 5: Model variations under the UniRank framework.

Table 6: Dataset statistics.

| Dataset | Users | Items | Interactions | Lists |
|---|---|---|---|---|
| MovieLens-1M | 6,020 | 3,043 | 995,154 | 161,646 |
| Amazon Books | 35,732 | 38,121 | 1,960,674 | 311,386 |
| Industry | 200,775 | 17,014 | 5,173,698 | 589,751 |

## B Relation to Existing Paradigms

**Explaining existing works under UniRank terminologies:** We provide detailed model settings of existing methods and UniRank in Table 5.

## C Dataset Statistics

As noted in the experimental setup, the UniRank experiments use three datasets: MovieLens-1M, Amazon Books, and Industry Dataset. Table 6 reports the number of users, items, interactions, and lists for each dataset.

## D Baseline Descriptions

In this section, we detail the descriptions of each baseline in our experiments.

**Generator-only baselines.** These methods directly produce the output list or assign final item scores with a single model.

- **DNN** [1] is adapted from the deep ranking model in the YouTube recommendation system. It scores each candidate item independently with a neural network and ranks items by the predicted scores, without explicit list-level dependency modeling.
- **DLCM** [10] is a listwise context model originally proposed for ranking refinement in information retrieval. It sequentially encodes the top-ranked items with a recurrent neural network and uses the learned local ranking context to adjust item scores.
- **PRM** [12] is a personalized reranking model based on a Transformer encoder. It models the mutual influence among items in the input list and incorporates personalized vectors to capture user-specific preferences before producing reranking scores.
- **GoalRank** [18] is a generator-only one-stage ranking framework for training a large ranker with group-relative optimization. It constructs a reference policy from reward-model scores within candidate-list groups and trains the generator to align with this group-relative policy.
- **GloRank** [33] reformulates reranking from selecting local candidate indices to generating global item identifiers represented by Semantic IDs. It uses constrained decoding over the candidate set to produce valid non-duplicated output lists.

Table 7: Simulator-based evaluation on the reranking task. We report the average reward, where higher is better. The **best** and second-best results are highlighted.

| Method | Reward (avg) ↑ | Method | Reward (avg) ↑ |
|---|---|---|---|
| DNN | 1.6146 | G-3 | 1.7997 |
| DLCM | 1.5920 | G-20 | 1.8878 |
| PRM | 1.5796 | G-100 | 1.8914 |
| NAR4Rec | 1.5723 | DCDR | 2.8200 |
| PIER | 1.7837 | GloRank | 2.9730 |
| GoalRank | 1.8851 | **Ours** | **3.0551** |

**Generator-Evaluator baselines.** These methods decouple candidate-list generation from list scoring, where an evaluator selects the final list from generated candidates.

- **PIER** [34] is a Generator-Evaluator reranking framework for e-commerce recommendation. It first selects promising candidate permutations according to user permutation-level interests and then evaluates these permutations with an omnidirectional context-aware prediction module.
- **NAR4Rec** [20] is a non-autoregressive generative model for reranking recommendation. It generates all target positions in parallel, which reduces the sequential decoding cost of autoregressive generators while retaining a Generator-Evaluator reranking structure.
- **G-$n$** [14] is a multi-generator reranking framework, where $n$ generators produce candidate lists that are then scored by an evaluator. We evaluate its variants with $n \in \{3, 20, 100\}$ to study the effect of expanding the generated candidate-list set.
- **DCDR** [22] applies discrete conditional diffusion to reranking. It defines a discrete forward process over item sequences and learns a conditional reverse process to refine reranked lists under expected user-feedback conditions.

## E Simulator-based Evaluation

To evaluate UniRank from a complementary perspective, we assess the list-level utility of the reranked output using a user interaction simulator, which provides a direct measure of practical recommendation effectiveness. Following previous work [43, 37], we train a list-wise simulator on MovieLens-1M and use it to score the output list of each method on the test set. UniRank and each baseline method are pre-trained on the same training set. The average reward across all test instances is reported as the evaluation metric. As shown in Table 7, UniRank achieves the highest average reward of 3.055, outperforming the strongest baseline GloRank and DCDR. The remaining baselines, including GoalRank, G-100, and PIER, cluster in the range of 1.57 to 1.89. The result confirms that UniRank generates lists with higher overall user utility, which is consistent with its discrete denoising process that reasons over the full list at each step.

## F Additional Analysis on Denoising Parameters

Figure 5 reports the complete hyperparameter analysis for the denoising process. The upper row varies the number of denoising steps from 1 to 6. All four metrics improve as the number of steps increases, showing that iterative denoising helps refine the output list. The gain from 1 to 2 steps is clear, while later steps bring smaller additional improvements. The lower row studies the item allocation strategy under two denoising steps. The label $a, b$ means that $a$ items are generated in the first step and $b$ items are generated in the second step. The strategy $1, 5$ gives the weakest results across all metrics. Moving more items to the first step generally improves performance, and $5, 1$ achieves the best NDCG@6, Precision@6, and F1@6, while matching the best MAP. This trend supports the confidence-ordered denoising design, where high-confidence positions are fixed first, and the remaining lower-confidence positions are generated with the context established by the first step.

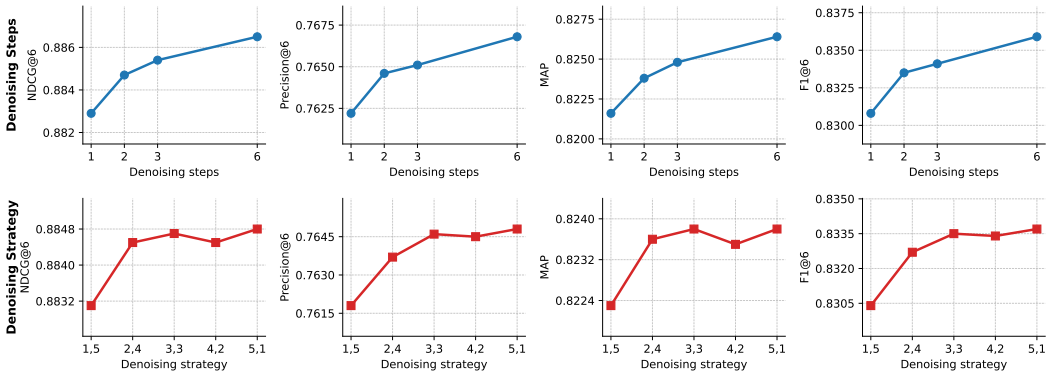


Figure 5: Effect of denoising parameters on all evaluation metrics. The upper row varies the number of denoising steps, and the lower row varies the item allocation strategy under two denoising steps.

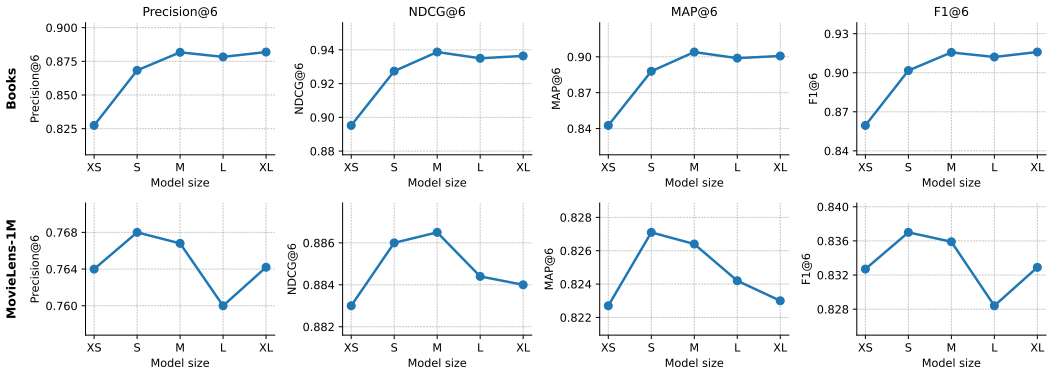


Figure 6: Complete scaling results on Books and MovieLens-1M.

## G Additional Scaling Results

Figure 6 reports the complete scaling results on Books and MovieLens-1M across Precision@6, NDCG@6, MAP@6, and F1@6. We evaluate five model configurations: XS with 0.29M parameters, S with 0.88M parameters, M with 5.08M parameters, L with 11.01M parameters, and XL with 27.59M parameters. On Books, performance improves substantially from XS to M across all metrics. Larger models do not bring consistent gains: L is lower than M on all four metrics, while XL only slightly improves Precision@6 and F1@6 but remains below M on NDCG@6 and MAP@6. On MovieLens-1M, the best results are concentrated around S and M, and both L and XL perform worse than these medium-size configurations. These results support the conclusion that increasing model capacity is useful up to a moderate scale, while larger models can overfit the available training signal and reduce generalization.

## H Limitations

UniRank is designed for the reranking stage and therefore selects items from a request-specific candidate pool. This design matches the standard multi-stage recommendation pipeline, but it also means that the final slate quality is bounded by the upstream retrieval and ranking stages. A natural extension is to jointly optimize candidate generation and diffusion-based reranking, or to use UniRank as feedback for improving the upstream candidate pool. Our experiments fix the candidate pool size at $|C| = 50$, and evaluating UniRank with larger candidate pools is left for future work.

Table 8: Hypothesis-driven analysis. All metrics are reported in percentage, and Avg is the average over the four metrics. The **best** and second-best results are highlighted.

| Variant | Tests | P@$L$ | NDCG@$L$ | MAP@$L$ | F1@$L$ | Avg |
|---|---|---|---|---|---|---|
| UniRank (bidirectional, confidence-ordered) | – | **76.68** | **88.65** | **82.64** | **83.59** | **82.89** |
| Left-to-right decoding | H1 | 76.51 | 88.23 | 82.25 | 83.40 | 82.59 |
| Random-order decoding | H1 | 76.57 | 87.70 | 81.23 | 83.48 | 82.24 |
| One-step NAR decoding | H3 | 76.22 | 88.29 | 82.16 | 83.08 | 82.43 |
| Causal slate attention + AR training/decoding | H2 | 74.58 | 86.61 | 79.97 | 81.29 | 80.61 |

# I Relation to Masked Iterative Decoding

UniRank belongs to the family of masked iterative decoding methods. Mask-Predict [44] first predicts all target words of a translation in parallel, and then repeatedly masks and regenerates the words that the model is least confident about. MaskGIT [45] trains a bidirectional Transformer to predict randomly masked image tokens, and at each decoding iteration it predicts all tokens in parallel but keeps only the most confident ones. LLaDA [27] applies masked diffusion to large language modeling, and LLaDA-Rec [35] uses it to generate the semantic identifier of the next item in generative recommendation. These methods share one decoding skeleton, in which a bidirectional denoiser starts from a fully masked output and commits confident predictions over several steps. The contribution of UniRank does not lie in this skeleton itself, but in how reranking is formulated under it and in the components that make it work for this task. We discuss three differences below.

**Reranking as candidate-constrained assignment.** The methods above generate tokens from a fixed vocabulary, whereas the output space of reranking is a request-specific candidate pool that changes with every request. UniRank therefore treats reranking as assigning candidates to slots rather than generating tokens. SFL aggregates the semantic ID tokens of each item into one embedding, so that each denoising position corresponds to one item. LPS scores each slot against the candidate pool, which serves as a dynamic vocabulary, and removes committed candidates from the following steps. Neither mechanism has a counterpart in the token-level methods above.

**Commit policy as a unifying variable.** In UniRank, the commit policy is the variable that connects existing generative rerankers. A left-to-right schedule that commits one slot per step recovers AR decoding, a single parallel step recovers NAR decoding, and confidence-ordered decoding lies between the two. As a result, AR-style, NAR-style, and confidence-ordered decoding can be compared on the same trained model by changing only the commit policy.

**Commit allocation in reranking.** MaskGIT uses a concave mask schedule that commits few tokens in early iterations and many in later ones, following the intuition that image generation has a less-to-more information flow. Our denoising strategy shows the opposite trend for reranking, where performance generally improves as more items are committed in the first step. This difference has a structural cause. Image generation starts from an almost empty context, so early predictions are unreliable. In reranking, only the slate is masked, while the user history $\mathbf{H}$ and the candidate pool $\mathbf{C}$ are fully visible from the first step. The model is therefore strongly conditioned from the start and can commit high-confidence slots right away.

# J Hypothesis-Driven Analysis of Decoding

To connect the central question of this paper with direct evidence, we decompose it into three hypotheses. H1: confidence-ordered denoising outperforms fixed left-to-right decoding by committing reliable slots first, which mitigates error propagation. H2: bidirectional slate context improves inter-item dependency modeling over causal (unidirectional) modeling. H3: iterative denoising improves over one-step parallel prediction.

We test the three hypotheses on MovieLens-1M with $L = 6$. For H1 and H3, we use the same trained checkpoint without retraining and vary only the inference-time commit policy, comparing confidence-ordered (default), left-to-right, random-order, and one-step parallel decoding. For H2, we retrain a variant that replaces bidirectional slate attention with a causal mask and uses AR training and decoding, while keeping all other components identical.

Table 9: Iterative decoding versus one-step NAR decoding under different slate lengths. The better result under each slate length is shown in **bold**.

| $L$ | Decoding | P@$L$ | NDCG@$L$ | MAP@$L$ | F1@$L$ | Avg Gap |
|---|---|---|---|---|---|---|
| 6 | Iterative ($L$ steps) | **76.68** | **88.65** | **82.64** | **83.59** | +0.46 |
| | One-step NAR | 76.22 | 88.29 | 82.16 | 83.08 | |
| 20 | Iterative ($L$ steps) | **77.02** | **88.44** | **79.17** | **83.89** | +1.11 |
| | One-step NAR | 75.68 | 87.64 | 78.39 | 82.35 | |

Table 10: Inference latency under different commit policies.

| Model | Setting | Forward passes | ms/request (bs=1) | ms/request (bs=64) | Params (M) |
|---|---|---|---|---|---|
| UniRank | 1 step (NAR) | 1 | 2.64 | 0.043 | 5.08 |
| | 2 steps | 2 | 4.33 | 0.069 | 5.08 |
| | 3 steps | 3 | 5.73 | 0.095 | 5.08 |
| | 6 steps | 6 | 10.18 | 0.173 | 5.08 |
| GloRank | – | 24 | 129.32 | 27.41 | 4.59 |

Table 8 shows that all three hypotheses are supported. For H1, confidence-ordered decoding outperforms both left-to-right and random-order decoding under the same step budget, which shows that the adaptive order itself drives the gain. Random-order decoding performs even worse than one-step decoding, since committing uncertain slots early propagates their errors into the context of subsequent steps. For H2, replacing bidirectional slate attention with causal attention causes the largest degradation, which confirms that bidirectional inter-item dependency modeling is the foundation of the architecture. For H3, iterative decoding under a sensible order (confidence-ordered or left-to-right) outperforms one-step parallel prediction. In terms of effect size, the bidirectional architecture (H2) contributes the most, followed by the commit order (H1) and the number of iterations (H3). This is consistent with our view that the TGD architecture provides the foundation, while confidence-ordered decoding adds complementary gains at inference time.

## K Effect of Slate Length on Iterative Decoding

To examine whether the benefit of iterative denoising grows with task difficulty, we compare the original setting ($L = 6$) with a longer slate setting ($L = 20$) on MovieLens-1M. The model is retrained for each setting, since the input sequence length changes with the slate length. With the same trained model, we compare confidence-ordered iterative decoding ($L$ steps) against one-step NAR decoding.

As shown in Table 9, the advantage of iterative decoding grows with the slate length on all four metrics, and the average gap increases from +0.46 at $L = 6$ to +1.11 at $L = 20$. This trend has a structural explanation. One-step NAR decoding predicts all slots in a single pass under a slot-wise independence assumption, and the cost of this assumption compounds as the slate grows. A longer slate contains more inter-item dependencies to coordinate and more low-confidence positions, which benefit from being resolved after their context is established.

## L Inference Latency

We report the wall-clock inference latency of UniRank under different numbers of commit steps and compare it with the AR baseline GloRank. All models are measured on the same GPU with bf16 precision, and the latency is averaged over the test set at batch sizes 1 and 64. For 2 and 3 steps, UniRank uses an equal allocation that commits 3 and 2 items per step, respectively.

Table 10 leads to three observations. First, the latency of UniRank scales linearly with the number of commit steps, since each step is a single forward pass. The commit policy therefore serves as an accuracy-latency knob on one trained model. Second, even the most expensive 6-step mode is 12.7× faster than GloRank at batch size 1 (10.18 vs. 129.32 ms) and 158× faster at batch size 64. The gap

arises because AR generation requires sequential per-slot decoding with beam search (24 forward passes with beam width 20, each over SID token vocabularies), whereas UniRank performs at most $L$ beam-free passes with candidate-constrained scoring. Third, the accuracy-latency tradeoff is favorable across the whole range. The 1-step NAR mode runs at 2.64 ms, which fits latency-sensitive scenarios, and moving from 1 to 6 steps improves accuracy at a cost that remains an order of magnitude below the AR baseline.

## 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 accurately reflect the paper's contributions and scope, see the contributions summary and introduction section.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Please refer to the limitation section in appendix

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

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:

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

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: Please refer to the implementation details section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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: [No]

Justification: We will release the code after it passes the company's internal review.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

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: please refer to the implementation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

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: please refer to the experiments section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

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: please refer to the implementation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

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: This work does not involve data collection that would raise ethical concerns and is submitted anonymously in compliance with NeurIPS guidelines.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: please refer to the introduction section.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

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:

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

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 cites all datasets and models used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

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: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.