Eliminating Out-of-Domain Recommendations in LLM-based Recommender Systems: A Unified View

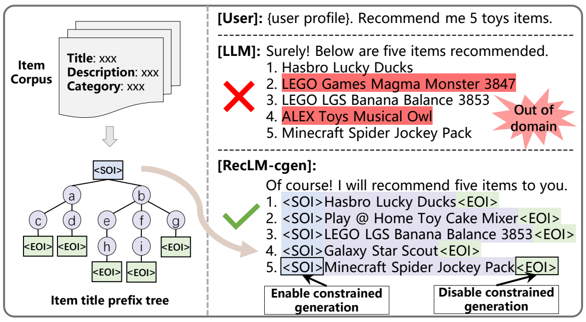
Anonymous ACL submission

Abstract

Recommender systems based on Large Language Models (LLMs) are often plagued by hallucinations of out-of-domain (OOD) items. To address this, we propose RecLM, a unified framework that bridges the gap between retrieval and generation by instantiating three grounding paradigms under a single architecture: embedding-based retrieval, constrained generation over rewritten item titles, and discrete item-tokenizer generation. Using the same backbone LLM and prompts, we systematically compare these three views on public benchmarks. RecLM strictly eradicates OOD recommendations (OOD@10 = 0) across all variants, and the constrained generation variants RecLM-cgen and RecLM-token achieve overall state-of-the-art accuracy compared to both strong ID-based and LLM-based baselines. Our unified view provides a systematic basis for comparing three distinct paradigms to reduce item hallucinations, offering a practical framework to facilitate the application of LLMs to recommendation tasks. Source code is at https://anonymous.4open.science/r/RecLM-cgen.

1 Introduction

Large language models (LLMs) are increasingly used to build conversational recommender systems, thanks to their strengths in language understanding, reasoning, and instruction following. Prior work either augments LLMs with prompt engineering or agentic retrieval (Yao et al., 2023; Gao et al., 2023; Huang et al., 2023), or fine-tunes them with domain knowledge (Lu et al., 2024; Zhang et al., 2024; Ji et al., 2024), bringing gains in recommendation quality but still suffering from out-of-domain (OOD) item recommendations (as illustrated in Figure 1) that can harm real-world systems.



[User]: {user profile}. Recommend me 5 toys items.

[LLM]: Surely! Below are five items recommended.
1. Hasbro Lucky Ducks
2. LEGO Games Magma Monster 3847
3. LEGO LGS Banana Balance 3853
4. ALEX Toys Musical Owl
5. Minecraft Spider Jockey Pack

[RecLM-cgen]:
Of course! I will recommend five items to you.
1. <SOI>Hasbro Lucky Ducks<EOI>
2. <SOI>Play @ Home Toy Cake Mixer<EOI>
3. <SOI>LEGO LGS Banana Balance 3853<EOI>
4. <SOI>Galaxy Star Scout<EOI>
5. <SOI>Minecraft Spider Jockey Pack<EOI>

Figure 1: An illustration of unconstrained versus constrained decoding.

Current attempts to mitigate OOD recommendations largely fall into three grounding paradigms. Retrieval-based methods map user and item information into an embedding space and retrieve in-domain items. Constrained generation methods restrict decoding to a catalog-dependent subspace, often via prefix trees over item titles. Item-tokenizer methods instead map each item to a compact sequence of discrete codes and generate over this learned token space. However, these paradigms are typically developed and evaluated in isolation, using different backbones and prompts, which makes it difficult to understand their relative strengths or to flexibly choose between them in practice.

In this work, we address the OOD issue and, more broadly, seek a unified view of these paradigms for LLM-based recommendation. We introduce <u>**Rec**</u>ommendation <u>**L**</u>anguage <u>**M**</u>odels (**RecLM**), a unified framework that bridges the gap between retrieval and generation by instantiating three complementary grounding strategies under a single architecture and training protocol. The key idea is to teach the LLM to first emit a special start-of-item token `<SOI>` to mark where recommendations should appear, and then plug in different grounding mechanisms after `<SOI>` while keeping the rest of the conversational behavior unchanged, as illustrated in Figure 1. Concretely, RecLM-ret uses the hidden state at `<SOI>` to retrieve an in-domain item from an embedding index, RecLM-cgen generates a rewritten item title under a prefix-tree constraint built over RL-optimized titles, and RecLM-token generates a short sequence of learned item tokens under a prefix tree and maps it back to a catalog item.

Our experiments on three public datasets demonstrate that all three RecLM variants successfully eliminate out-of-domain recommendations (OOD@10 = 0). Moreover, the constrained generation variants (RecLM-cgen and RecLM-token) achieve state-of-the-art accuracy, outperforming both strong ID-based and LLM-based baselines. By integrating retrieval, constrained generation, and item tokenization into a unified framework, RecLM enables a reliable comparison of these distinct paradigms and offers best practices for designing LLM-based recommendation systems in both academic and industrial settings.

To summarize, our main contributions are:

- We propose RECLM, a unified framework that trains an LLM to first emit a special start-of-item token (`<SOI>`) and then delegate recommendation to interchangeable grounding modules. This design unifies retrieval-based and generation-based recommendation paradigms within a single architecture and evaluation protocol.
- Within this framework, we introduce lightweight enhancement modules, including a reinforcement learning–based Title Rewriter and scope-mask training. These components compress verbose item metadata into concise, human-readable identifiers and improve the alignment between constrained generation and recommendation objectives under limited token budgets.
- Experimental results show that the three paradigms exhibit clear differences in ranking accuracy, OOD rate, efficiency, and conversational behavior across three public recommendation datasets. All RecLM variants strictly eliminate OOD recommendations (OOD@10 = 0), and the observed trade-offs reveal complementary strengths across retrieval, constrained textual generation, and item tokenization, offering practical guidance for method selection in different deployment scenarios.

2 Related Work

2.1 LLMs for Recommender Systems

LLMs have significantly influenced various NLP applications, including recommender systems. Their potential has been widely recognized in facilitating a new type of generative recommender systems (Wu et al., 2024; Lian et al., 2024; Lyu et al., 2024; Ji et al., 2024). Said (2025) provides a comprehensive review of the literature on using LLMs for generating recommendation explanations. Methods for selectively injecting domain-specific knowledge into prompts to enhance the recommendation capabilities of LLMs without fine-tuning are introduced by Yao et al. (2023) and Bacciu et al. (2024). Another line of research focuses on fine-tuning LLMs to inject domain knowledge, demonstrating significant improvements in recommendation performance (Zhang et al., 2024; Lu et al., 2024; Yang et al., 2023; Zhu et al., 2024). However, these approaches often face the challenge of OOD item generation, where LLMs may recommend items that are not present within the current domain, potentially leading to negative business impacts.

2.2 Addressing Out-of-domain Recommendations

The issue of OOD item generation is a critical challenge in LLM-based recommenders. Bao et al. (2025) proposes a generate-then-align method to ensure that recommended items are grounded within the domain item set. Gao et al. (2023) and Huang et al. (2023) leverage agentic frameworks where LLMs act as controllers and natural language interfaces for user interactions. When making recommendations, these frameworks call traditional recommender models to retrieve relevant items. Another promising direction is constrained generation. This paradigm restricts the LLM's decoding space to a subspace conditioned by the context, thereby avoiding OOD generation (Dong et al., 2024). Constrained generation methods maintain the traditional language generation process without necessitating significant modifications to the LLM. In addition to retrieval-based grounding and prompt-based constrained generation, recent work proposes an item-tokenizer paradigm for LLM-based recommenders, where each item is mapped to a compact sequence of discrete tokens and decoded under prefix-tree constraints (Tan et al., 2024; Wang et al., 2024; Zheng et al., 2024; Lin et al., 2025). These methods tightly couple catalog structure with generative modeling, but are typically studied in isolation from retrieval- and text-based grounding. From a unified perspective, our work brings together these three lines. It allows us to interpret existing approaches as points in a common design space and to empirically compare their behavior for OOD mitigation under the same backbone model and evaluation protocol.

3 Methodology

Our goal is to avoid recommending OOD items while preserving the conversational strengths of LLMs. To this end, we build three RecLM variants under two fundamental paradigms—in-domain retrieval and constrained catalog grounding—and implement them within a single lightweight framework that introduces minimal changes to the backbone model. The overall fraemwork is illustrated in Figure 2. The key mechanism is a pair of special item indicator tokens that tell the LLM when it is entering and leaving a recommendation segment, so that different grounding strategies can be plugged into the same decoding process.

3.1 Special Item Indicator Token

We equip the backbone LLM with two special tokens—`<SOI>` (start-of-item) and `<EOI>` (end-of-item)—to explicitly mark recommendation segments in its outputs. After fine-tuning on recommendation data, RecLM learns to produce sequences of the form `<SOI>` *item identifier* `<EOI>` at appropriate positions in a conversation. The emission of `<SOI>` signals that the model is entering an item segment where grounding constraints apply, and the appearance of `<EOI>` token marks its termination, after which the model resumes generating general text. As illustrated in Figure 2, what happens between `<SOI>` and `<EOI>` is then delegated to one of the three RecLM variants: retrieval over an item index (RecLM-ret), constrained decoding over rewritten titles (RecLM-cgen), or constrained decoding over discrete item tokens (RecLM-token).

3.2 RecLM-ret

RecLM-ret instantiates retrieval-style grounding within our unified framework. We build a domain-specific item index by encoding each item's title, description, and category with BGE-M3 (Chen et al., 2024), followed by a lightweight adapter that maps embeddings to the target space, yielding $\mathcal{E} = \{\mathbf{e}_i\}$. At inference time, when RecLM emits `<SOI>`, the corresponding hidden state $\mathbf{h}_{<SOI>}$ is projected into the item embedding space and the nearest item in $\mathcal{E}$ is retrieved; its title is then inserted into the output and closed by `<EOI>`. This design reuses standard embedding-based retrieval while cleanly fitting the shared `<SOI>`/`<EOI>` interface.

For training, sequence data <$I_{history}^{(1\ldots n)}$, $I_{rec}^{(1\ldots K)}$> are converted into instruction–response pairs <Instruction:X, Response:Y>, where X encodes the user history and Y contains the recommended items; we follow Lu et al. (2024) for data augmentation and provide prompt templates in Appendix A.5. RecLM-ret is optimized with a language-modeling loss over non-item tokens and an auxiliary retrieval loss that aligns each `<SOI>` hidden state with the embedding of its ground-truth item:

$$\mathcal{L}_{\text{lm}} = \sum_{\substack{j=1 \\ Y_j \notin \{\text{item}, \texttt{<EOI>}\}}}^{|Y|} -\log P_\theta(Y_j | Y_{<j}, X) \tag{1}$$

A retrieval loss further teaches the model to select the correct item in the embedding space. Let $\mathbf{h}_{<SOI>}^{(1\ldots K)}$ be the hidden states at `<SOI>` for the K recommended items in Y, and proj_ϕ a projection layer. We match the projected vectors to their ground-truth item embeddings in $\mathcal{E}$ using:

$$\mathcal{L}_{\text{ret}} = -\frac{1}{K} \sum_{j=1}^{K} \log(\sigma(\text{proj}_\phi(\mathbf{h}_{<SOI>}^{(j)}) \cdot \mathbf{e}_j)) \tag{2}$$

$$\mathcal{L}_{\text{RecLM-ret}} = \mathcal{L}_{\text{lm}} + \alpha_{\text{ret}} * \mathcal{L}_{\text{ret}} \tag{3}$$

where α_{ret} balances conversational modeling and retrieval alignment. This design keeps the conversational behavior of the backbone LLM largely intact while providing a simple, index-based grounding mechanism compatible with the `<SOI>`/`<EOI>` interface.

3.3 RecLM-cgen

RecLM-cgen instantiates text-based constrained grounding with rewritten titles. It first employs a Title Rewriter (TR) module to transform each item's verbose title and description into a new, concise, human-readable title. These rewritten titles are used both to construct user histories and to build a prefix tree over the catalog. During generation, once RecLM emits `<SOI>`, decoding is restricted to paths in the prefix tree until `<EOI>` appears, ensuring that all recommended items come from the catalog while still leveraging the LLM's language modeling capacity around the title.

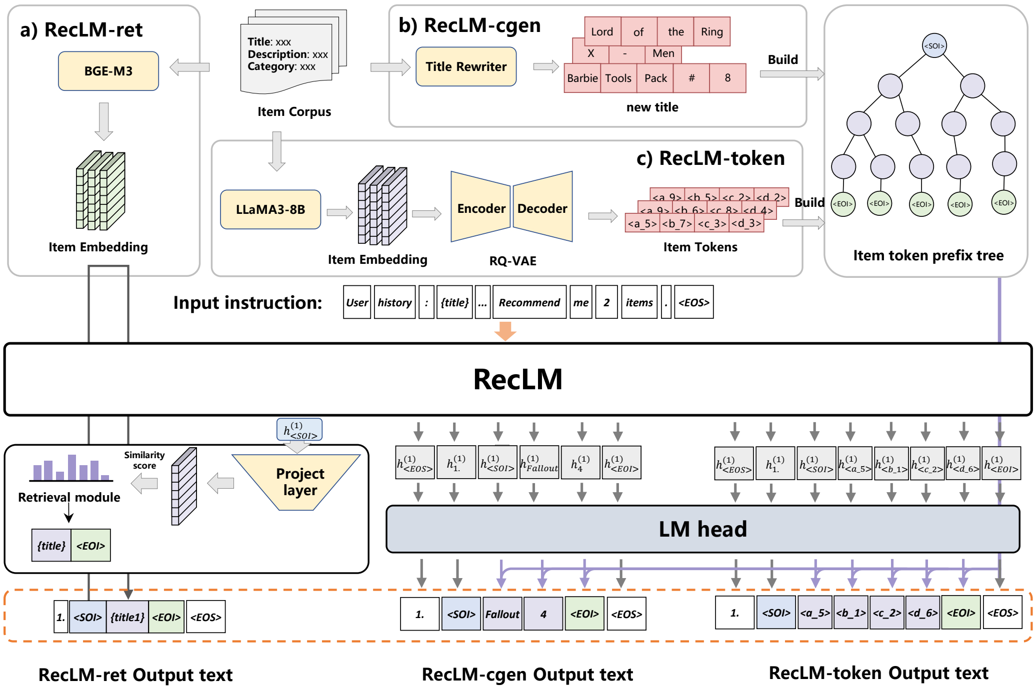


Figure 2: Overview of RecLM variants: embedding-based retrieval (RecLM-ret), constrained generation over rewritten item titles (RecLM-cgen), and discrete item-tokenizer generation (RecLM-token).

3.3.1 Title Rewriter

Item IDs are often lengthy or uninformative, whereas descriptions contain richer semantics but are too verbose under token budget constraints. The TR module addresses this by transforming raw item metadata into compact, human-readable titles that retain key semantic information and serve as the surface forms used in constrained generation.

Since title rewriting is inherently open-ended with no single ground-truth title, we train TR using the Group Relative Policy Optimization (GRPO) algorithm (Shao et al., 2024). We design five reward components that reflect our design goals: item-to-item similarity (I2I), user-to-item alignment (U2I), decoding complexity (DC), conciseness (CR), and discriminative power (DPR). These components are combined into a single reward (details in Appendix B.1):

$$\begin{aligned} R = \lambda_1 R_{\text{U2I}} + \lambda_2 R_{\text{I2I}} + \lambda_3 R_{\text{DC}} \\ + \lambda_4 R_{\text{CR}} + \lambda_5 R_{\text{DPR}}. \end{aligned} \tag{4}$$

Recommendation-oriented Reward. To enhance recommendation performance, we consider both item-to-item similarity and user-to-item alignment. From the **item-to-item** perspective, the generated short titles should preserve the neighborhood structure in the item space. We construct a contribution matrix from the original data as ground-truth item similarity, and define:

$$R_{\text{I2I}} = 0.5\big(1 + \text{Spearman}(\pi_{\text{orig}}, \pi_{\text{gen}})\big) \tag{5}$$

where π_{orig} represents the ground-truth ranking of similar items, derived from the contribution matrix, and π_{gen} denotes the ranking based on cosine similarities between the embedding of the rewritten item title and those of all other items.

From the perspective of **user-to-item**, the objective is to evaluate whether the generated titles better reflect user preferences and improve retrieval of the target item. User-to-Item Reward (R_{U2I}), applied to group-level tasks where TR rewrites a set of item names, measures how well the rewritten prompt preserves the target item's rank. Given the similarity ranking of the ground-truth item i^*, we define:

$$R_{\text{U2I}} = \exp\big(-(\text{rank}(i^*) - 1)/\tau\big), \; \tau = 2000 \tag{6}$$

Decoding Complexity. To keep titles easy for LLMs to process, we assess their decoding complexity using conditional perplexity and define:

$$R_{\text{DC}} = \exp(-\alpha_{\text{ppl}} \cdot \text{PPL}(y|X)) \tag{7}$$

where lower conditional perplexity corresponds to higher reward.

Conciseness. To encourage brevity, we introduce a length-based reward that favors shorter rewritten titles:

$$R_{\text{CR}} = \left(1 + (|y|/|x|)^2\right)^{-1} \tag{8}$$

where, $|x|$ and $|y|$ denote the number of tokens of the original and generated titles, respectively.

Discriminative Power. Finally, rewritten titles should be distinguishable from the titles of semantically similar items. We design a discrimination task where the generated title is used as a prompt to a language model, which is then asked to identify the correct original title from a set of four candidates: the true original title and the titles of the three most similar items. The reward function is:

$$R_{\text{DPR}} = \mathbb{I}[\text{correct}] \tag{9}$$

where, $\mathbb{I}[\text{correct}]$ is an indicator function that returns 1 if the model selects the correct title, and 0 otherwise.

3.3.2 Scope Mask Training

During constrained decoding, the next token is chosen from a prefix-tree–defined subset rather than the full vocabulary. To match this behavior at training time, we introduce a scope mask loss for RecLM-cgen: when computing the loss for item-title tokens, only tokens allowed by the prefix tree are included in the $softmax$ denominator:

$$\mathcal{L}_{\text{cgen}}^{\text{sm}} = \sum_{j=1}^{|Y|} -\log \frac{\exp(\text{logit}(Y_j|Y_{<j}, X, \theta))}{\sum_{t \in \text{NT}(Y_{<j})} \exp(\text{logit}(t|Y_{<j}, X, \theta))} \tag{10}$$

Here, $\text{NT}(Y_{<j})$ returns the valid next-token set given the current prefix. For general text (e.g., after `<EOI>`), it equals the full vocabulary; for item titles (between `<SOI>` and `<EOI>`), it is restricted to tokens available in the catalog prefix tree.

3.4 RecLM-token

RecLM-token casts recommendation as sequence generation over a finite vocabulary of discrete item tokens rather than natural-language titles: each catalog item is assigned a short code sequence (e.g., `<a_11><b_2><c_135><d_157>`), generated between `<SOI>` and `<EOI>` and deterministically mapped back to an item. To construct these codes, we encode item text with Llama3-8B-Instruct to obtain semantic embeddings, discretize them into short code sequences via an RQ-VAE codebook (Lee et al., 2022), and then fine-tune (plus RL) the LLM on tokenized recommendation sequences.

3.4.1 Item Tokenizer

Given the semantic embedding $\mathbf{e}_{sem} \in \mathbb{R}^{d_{\text{Llama3-8B}}}$ of each item, a multi-layer perceptron projects it into a latent vector suitable for quantization:

$$\mathbf{z}_e = \text{EncoderMLP}(\mathbf{e}_{sem}), \tag{11}$$

where $\mathbf{z}_e$ serves as the continuous representation to be discretized.

We then apply residual vector quantization (RQ) to turn $\mathbf{z}_e$ into a short sequence of discrete codes. Given quantization depth D, at each stage d, we select a code vector $\mathbf{z}_q^{(d)}$ from codebook $\mathcal{C}_d$ that best matches the current residual, obtaining the final quantized representation:

$$\hat{\mathbf{z}}_q = \sum_{d=1}^{D} \mathbf{z}_q^{(d)}. \tag{12}$$

The corresponding index tuple $t_i = (k_1, \ldots, k_D)$ serves as the item's tokenized identifier. Finally, a decoder network DecoderMLP reconstructs the semantic embedding $\hat{\mathbf{e}}_{sem}$ from $\hat{\mathbf{z}}_q$.

The tokenizer is trained end-to-end with a loss that balances reconstruction fidelity and quantization quality:

$$\mathcal{L}_{total} = \|\hat{\mathbf{e}}_{sem} - \mathbf{e}_{sem}\|_2^2 + \mathcal{L}_{quant}, \tag{13}$$

$$\mathcal{L}_{\text{quant}} = \sum_{d=1}^{D} \ell_{\text{VQ}}(\mathbf{r}_{d-1}, \mathbf{z}_q^{(d)}), \tag{14}$$

where $\ell_{\text{VQ}}(\mathbf{r}, \mathbf{z}) = \|\text{sg}[\mathbf{r}] - \mathbf{z}\|_2^2 + \beta\|\mathbf{r} - \text{sg}[\mathbf{z}]\|_2^2$. Here, $\mathbf{r}_{d-1}$ denotes the residual at depth d (with $\mathbf{r}_0 = \mathbf{z}_e$), and $\text{sg}[\cdot]$ is the stop-gradient operator.

The reconstruction loss encourages faithful recovery of the semantic representation. The quantization loss $\mathcal{L}_{quant}$ follows standard residual quantization formulations and includes a commitment term weighted by β; in all experiments, we set $\beta = 0.25$.

3.4.2 Reinforcement Learning for RecLM-token

After obtaining discrete item codebook, we first perform supervised fine-tuning to align the language model with the codebook. Unlike RecLM-cgen, the recommendation segments are expressed purely in terms of discrete identifiers (e.g., `<a_128>`), providing a fully symbolic interface between tokenizer and generator. We then apply GRPO to further refine generation behavior.

Reward Design. We optimize RecLM-token with a reward defined over ranked lists, decomposed into position-sensitive and inclusion-based components. When the target item appears in the generated list, we assign a reward that decreases with its rank position:

$$R_{\text{ord}} = \begin{cases} \frac{1}{\log_2(\text{rank}+1)}, & \text{if the target appears} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

where rank denotes the 1-based index of the target item.

To complement position-sensitive feedback, we add a reward that depends only on whether the target item appears in the prediction:

$$R_{\text{pre}} = \mathbb{I}(\text{target} \in \text{prediction}), \tag{16}$$

capturing coarse-grained relevance at the list level.

3.5 Multi-round Dialogue Training

To enable the LLM-based recommendation system to interact naturally with users beyond single-turn question-answering, we incorporate multi-round dialogue training. Without this, training solely on single-turn SFT samples `<Instruction:` X, `Response:` Y`>` biases the model toward a QA-style recommendation tool and weakens its conversational ability. We therefore augment about 10% of the data with multi-round conversation (MRC) samples, built by combining a randomly sampled ShareGPT dialogue[1] with a single-turn recommendation task. To diversify contexts, the recommendation turn appears before or after the dialogue with equal probability.

4 Experiments

4.1 Experiment Settings

We conduct experiments on three public sequential recommendation datasets: **Steam**[2], Amazon Movies & TV[3] (**Movies**), and Amazon Toys & Games[3] (**Toys**) (Ni et al., 2019). Users with fewer than 17 interactions are filtered out, and each interaction sequence is truncated to the 17 most recent items. After this, 10k users are randomly sampled for experiments. Following prior work (Kang and McAuley, 2018; Lu et al., 2024), we adopt a leave-one-out protocol: for each user, the last interaction is reserved for testing, the second-to-last for validation, and the remaining 15 for training; dataset statistics are summarized in Table 5.

Backbone and Fine-tuning. We use Llama3-8B-Instruct as the backbone for all RecLM variants.User histories are truncated to 10 interactions and embedded into instruction-style prompts; the maximum input and output lengths are both 512 tokens. All models are fine-tuned with LoRA on all linear layers using Adam (learning rate 1×10^{-4}, LoRA rank $r = 16$, scaling factor $\alpha = 8$, batch size 2), typically converging within 20 epochs. The token embeddings of `<SOI>` and `<EOI>` are initialized as the average embeddings of the phrases "start of an item" and "end of an item". All reported results are averaged over five runs, with significance assessed using paired tests ($p < 0.05$).

In the RecLM-token, we employ residual vector quantization with four codebooks of 256 entries each, yielding a four-token discrete representation per item. During RL training stage, we sample 16 candidate recommendation lists per prompt for relative reward normalization, generate with temperature 1.0 and maximum length 128, and optimize with learning rate 1×10^{-5}, batch size 32. The LoRA rank is 16. Detailed experimental settings can be found in Appendix A.3.

4.1.1 Metrics

We evaluate recommendation accuracy with Top-k Hit Ratio ($HR@k$) and Top-k Normalized Discounted Cumulative Gain ($NDCG@k$). To assess reliability, we additionally report $Repeat@k$, the fraction of duplicate items within the Top-k list, and $OOD@k$, the fraction of Top-k items that fall outside the domain catalog.

4.1.2 Baselines

We compare RecLM to 12 baselines grouped into four categories. (1) Traditional ID-based sequential recommenders include SASRec (Kang and McAuley, 2018) and GRU4Rec (Hidasi et al., 2016), which operate purely on interaction sequences. (2) Frozen LLM baselines include GPT-4o and Llama3-8B-Instruct, along with Llama3-cgen, a prompt-based constrained-generation variant of the latter. (3) Finetuned LLMs comprise BIGRec (Bao et al., 2025), CtrlRec (Lu et al., 2024), and PALR (Yang et al., 2023). Finally, (4) item-tokenizer LLMs include IDGenRec (Tan et al.,

---

[1] https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered

[2] https://www.kaggle.com/datasets/antonkozyriev/game-recommendations-on-steam

[3] https://mcauleylab.ucsd.edu/public_datasets/data/amazon_v2/categoryFiles

	Traditional Recommenders		LLMs (frozen)			LLMs (finetuned)			LLMs (item tokenizer)				LLMs (ours)		
Metrics	SASRec	GRU4Rec	GPT-4o	Llama3	Llama3-cgen	BIGRec	CtrlRec	PALR	IDGenRec	SETRec	LC-Rec	LETTER	RecLM-ret	RecLM-cgen	RecLM-token
Dataset: Steam															
HR@10 ↑	0.0694	0.0599	0.0383	0.0230	0.0261	0.0396	0.0756	0.0739	0.0682	0.0626	0.0725	0.0569	0.0600	**0.0868(+14.8%)**	0.0733
NDCG@10 ↑	0.0308	0.0281	0.0194	0.0120	0.0125	0.0244	0.0367	0.0408	0.0344	0.0303	0.0390	0.0287	0.0291	**0.0456(+11.8%)**	0.0388
HR@5 ↑	0.0428	0.0323	0.0234	0.0136	0.0147	0.0291	0.0507	0.0488	0.0416	0.0346	0.0473	0.0335	0.0359	**0.0579(+2.1%)**	0.0567
NDCG@5 ↑	0.0224	0.0193	0.0147	0.0090	0.0088	0.0201	0.0318	0.0305	0.0248	0.0213	0.0309	0.0212	0.0214	**0.0361(+9.1%)**	0.0331
repeat@10 ↓	–	–	1.07%	2.06%	**0.00%**	**0.00%**	1.08%	1.05%	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**
OOD@10 ↓	–	–	16.08%	15.26%	2.59%	**0.00%**	2.40%	2.46%	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**
Dataset: Movies															
HR@10 ↑	0.1510	0.0722	0.0046	0.0049	0.0246	0.0861	0.1347	0.1335	0.1243	0.0929	0.1607	0.1205	0.1145	0.1467	**0.1700(+5.8%)**
NDCG@10 ↑	0.1351	0.0556	0.0028	0.0025	0.0106	0.0760	0.1248	0.1244	0.1064	0.0823	0.1458	0.1097	0.1052	0.1311	**0.1622(+11.2%)**
HR@5 ↑	0.1422	0.0625	0.0027	0.0029	0.0123	0.0823	0.1304	0.1294	0.1107	0.0867	0.1532	0.1136	0.1038	0.1400	**0.1667(+8.8%)**
NDCG@5 ↑	0.1323	0.0525	0.0022	0.0019	0.0064	0.0747	0.1234	0.1230	0.1098	0.0803	0.1434	0.1002	0.0970	0.1290	**0.1611(+12.3%)**
repeat@10 ↓	–	–	0.89%	3.15%	**0.00%**	**0.00%**	9.02%	34.69%	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	1.00%
OOD@10 ↓	–	–	61.21%	52.52%	11.91%	**0.00%**	8.13%	14.85%	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**
Dataset: Toys															
HR@10 ↑	0.0589	0.0389	0.0031	0.0039	0.0354	0.0405	0.0473	0.0438	0.0498	0.0371	**0.0790**	0.0515	0.0596	0.0657	0.0714
NDCG@10 ↑	0.0484	0.0228	0.0013	0.0020	0.0153	0.0272	0.0378	0.0369	0.0402	0.0301	0.0584	0.0394	0.0437	0.0508	**0.0651(+11.5%)**
HR@5 ↑	0.0529	0.0276	0.0021	0.0019	0.0191	0.0311	0.0426	0.0407	0.0418	0.0327	0.0652	0.0420	0.0499	0.0543	**0.0686(+5.2%)**
NDCG@5 ↑	0.0464	0.0192	0.0010	0.0013	0.0104	0.0242	0.0363	0.0359	0.0376	0.0287	0.0540	0.0397	0.0405	0.0470	**0.0643(+19.1%)**
repeat@10 ↓	–	–	0.31%	2.10%	**0.00%**	**0.00%**	5.91%	29.50%	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	0.86%
OOD@10 ↓	–	–	89.57%	90.99%	4.16%	**0.00%**	7.80%	37.00%	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**

Table 1: Overall recommendation performance comparison on three datasets. Best results are in **bold**, second-best are underlined; traditional recommenders serve as ID-based reference baselines.

2024), SETRec (Lin et al., 2025), LC-Rec (Zheng et al., 2024), and LETTER (Wang et al., 2024), all built on Llama3-8B-Instruct. Additional implementation details for all baselines are given in Appendix A.2.

4.2 Overall Performance

Table 1 summarizes the overall comparison. On the key reliability metric, OOD@10, all three RecLM variants strictly avoid out-of-domain items across all benchmarks, matching the best mapping- and tokenizer-based baselines. This shows that, under a unified `<SOI>`-based grounding interface, both retrieval and constrained generation can enforce catalog fidelity rather than trading it off against recommendation quality.

In terms of accuracy, the constrained generation variants deliver the strongest performance overall: RecLM-cgen and RecLM-token consistently match or exceed the best non-ours LLM baselines. RecLM-cgen maintains $\text{repeat@10} = 0$, and RecLM-token also keeps repetition low, indicating that the unified framework can simultaneously control OOD, repetition, and ranking quality. RecLM-ret typically trails the constrained generation variants in ranking metrics but remains competitive with prior mapping-based methods.

Beyond quantitative metrics, a qualitative case study (Appendix D) shows that the Title Rewriter succinctly compresses verbose item metadata into clearer identifiers and that rewriting user-history titles can further improve recommendation quality.

4.3 Ablation Study

We choose RecLM-cgen for our ablation study as it incorporates the most newly proposed components. We analyze how RECLM-CGEN components contribute to accuracy and control. We define six variants: **v0** is a finetuned Llama3-8B-Instruct that can emit `<SOI>` but uses unconstrained decoding; **v1** adds constrained generation; **v2** adds the scope-mask loss; **v3** additionally uses multi-round dialogue data; **v4** incorporates a TR trained with three reward components; and **full** uses the five-reward TR, representing our complete configuration.

Dataset	Metrics	v0	v1	v2	v3	v4	full
Steam	HR@10 ↑	0.0731	0.0749	0.0746	0.0797	0.0829	**0.0868**
	NDCG@10 ↑	0.0396	0.0406	0.0410	0.0433	0.0447	**0.0456**
	HR@5 ↑	0.0495	0.0508	0.0502	0.0540	0.0571	**0.0579**
	NDCG@5 ↑	0.0320	0.0329	0.0332	0.0360	**0.0362**	0.0361
	Repeat@10 ↓	2.33%	0.00%	0.00%	0.00%	0.00%	0.00%
	OOD@10 ↓	1.75%	0.00%	0.00%	0.00%	0.00%	0.00%
Movies	HR@10 ↑	0.1331	0.1400	0.1443	0.1424	0.1433	**0.1467**
	NDCG@10 ↑	0.1240	0.1269	0.1318	0.1296	**0.1329**	0.1311
	HR@5 ↑	0.1297	0.1334	0.1396	0.1365	**0.1400**	**0.1400**
	NDCG@5 ↑	0.1229	0.1248	0.1303	0.1277	**0.1318**	0.1290
	Repeat@10 ↓	39.26%	0.00%	0.00%	0.00%	0.00%	0.00%
	OOD@10 ↓	17.48%	0.00%	0.00%	0.00%	0.00%	0.00%
Toys	HR@10 ↑	0.0400	0.0581	0.0605	0.0642	**0.0686**	0.0657
	NDCG@10 ↑	0.0346	0.0429	0.0442	0.0479	0.0481	**0.0508**
	HR@5 ↑	0.0380	0.0475	0.0496	0.0534	**0.0543**	**0.0543**
	NDCG@5 ↑	0.0340	0.0395	0.0407	0.0444	0.0435	**0.0470**
	Repeat@10 ↓	34.57%	0.00%	0.00%	0.00%	0.00%	0.00%
	OOD@10 ↓	35.85%	0.00%	0.00%	0.00%	0.00%	0.00%

Table 2: Ablation study on three datasets comparing the performance of six model variants.

Table 2 shows that each component of RecLM-cgen yields incremental gains. Introducing constrained generation (**v1**) consistently improves ranking metrics over **v0** while preserving low OOD and repetition rates. The scope mask (**v2**) further boosts accuracy by matching training to prefix-tree decoding, and multi-round dialogue training (**v3**) improves robustness when recommendation is interleaved with general conversation. Adding the TR module (**v4** and **full**) delivers the best or near-best accuracy, indicating that RL-optimized titles help the model use its constrained generation capacity more effectively. Together, these trends illus-

Model	Response R_1 HR@10 ↑	Response R_1 NDCG@10 ↑	Response R_1 $\mathrm{CSN}_{\mathrm{R}_1}^{\mathrm{n}=10}$ ↑	Response R_2 $\mathrm{ACC}_{\mathrm{gsm8k}}$ ↑	Response R_2 $\mathrm{CSN}_{\mathrm{R}_2}^{\mathrm{n}=0}$ ↑
Dataset: Steam					
Llama3-cgen	0.0258	0.0119	0.717	**0.676**	**1.000**
PALR	0.0629	0.0364	–	0.585	–
CtrlRec	0.0662	0.0349	–	0.022	–
RecLM-ret	0.0508	0.0257	0.998	0.669	0.987
RecLM-cgen	**0.0713**	**0.0410**	**1.000**	0.673	0.990
RecLM-token	0.0480	0.0221	**1.000**	0.660	0.998
Dataset: Movies					
Llama3-cgen	0.0296	0.0128	0.703	0.670	**1.000**
PALR	0.1425	0.1349	–	0.380	–
CtrlRec	0.1327	0.1233	–	0.456	–
RecLM-ret	0.1062	0.0944	0.998	0.653	0.987
RecLM-cgen	0.1509	0.1388	**1.000**	**0.703**	0.988
RecLM-token	**0.1524**	**0.1441**	**1.000**	0.652	**1.000**
Dataset: Toys					
Llama3-cgen	0.0403	0.0245	0.396	0.667	**1.000**
PALR	0.0462	0.0399	–	0.617	–
CtrlRec	0.0455	0.0404	–	0.591	–
RecLM-ret	0.0516	0.0396	0.998	0.640	**1.000**
RecLM-cgen	0.0584	0.0484	**1.000**	**0.718**	0.998
RecLM-token	**0.0644**	**0.0545**	**1.000**	0.638	0.998

Table 3: Results of the control symbol study in the multi-turn dialogue setting.

Dataset	Model	MMLU	GSM8K	CSQA	Humam-eval
-	Llama3	0.675	0.781	0.786	0.640
Steam	BIGRec	0.632	0.722	0.737	0.402
	PALR	**0.659**	0.745	**0.778**	0.512
	CtrlRec	0.646	0.697	0.764	0.567
	RecLM-ret	0.653	0.722	0.762	0.573
	RecLM-cgen	0.657	**0.777**	0.767	**0.591**
	RecLM-token	0.638	0.676	0.750	0.530
Movies	BIGRec	0.609	0.689	0.711	0.299
	PALR	0.651	0.747	0.747	0.555
	CtrlRec	0.649	0.729	0.756	0.549
	RecLM-ret	0.650	0.501	0.761	0.555
	RecLM-cgen	**0.658**	**0.772**	0.756	**0.579**
	RecLM-token	0.653	0.737	**0.762**	0.506
Toys	BIGRec	0.622	0.661	0.710	0.445
	PALR	0.645	0.728	0.737	0.561
	CtrlRec	0.623	0.721	0.728	0.561
	RecLM-ret	**0.653**	0.340	0.754	0.561
	RecLM-cgen	**0.653**	**0.767**	0.747	**0.598**
	RecLM-token	0.640	0.709	**0.756**	0.512

Table 4: General-task performance. Gray row: untuned Llama3-8B-Instruct.

trate how the unified RecLM design can be tuned along several axes—decoding constraints, training alignment, and title rewriting—to strengthen both reliability and recommendation quality.

We further probe robustness under domain shift by training and evaluating RecLM-cgen across mismatched domains (e.g., training on one catalog and testing on another). Detailed cross-domain results are reported in the Appendix C.3.

4.4 Control Symbol Study

To assess the reliability of control symbol generation, we use a multi-turn dialogue setting to ensure that the final model can interact naturally in daily conversations and provide effective recommendations. We construct a three-turn dialogue that interleaves GSM8K-style math questions with a recommendation request. The first and third turns are math reasoning tasks, while the second asks for 10 item recommendations. We measure $CSN_{R_*}^{n=k}$, the proportion of responses R_* that generate exactly k `<SOI>` symbols (10 for the recommendation turn, 0 for non-recommendation turns).

As shown in Table 3, both RecLM-cgen and RecLM-token achieve near-perfect CSN scores, correctly generating 10 `<SOI>` symbols in recommendation turns and almost never emitting them in reasoning turns. Importantly, this high level of control is achieved without sacrificing recommendation accuracy, as evidenced by their competitive HR@10 and NDCG@10 scores across all datasets. This indicates the robustness of the control-symbol interface in our unified framework.

4.5 General Tasks Evaluation

Finally, we examine how aligning models to recommendation and grounding affects their broader abilities. We evaluate MMLU (5-shot), GSM8K (8-shot), CommonsenseQA (7-shot), and HumanEval (0-shot), covering comprehension, mathematics, commonsense reasoning, and code generation. As shown in Table 4, while there is a general slight decline compared to the untuned Llama3-8B-Instruct due to the alignment tax, our `RecLM` variants demonstrate strong resilience against catastrophic forgetting. RecLM-cgen consistently achieves superior performance among tuned models; it largely tracks Llama3's capabilities in mathematical reasoning and significantly outperforms other baselines in code generation, indicating that our grounding-oriented fine-tuning effectively preserves general reasoning and language understanding.

5 Conclusion

Advancing the application of LLMs to recommendation systems, this paper introduced *RecLM*, a unified framework that eliminates OOD recommendations through control tokens (`<SOI>`/`<EOI>`) and interchangeable modules: retrieval-based (*RecLM-ret*), constrained generation (*RecLM-cgen*), and item tokenization (*RecLM-token*). All variants achieve OOD@10 = 0. The unified view enables fair comparison of OOD-avoidance paradigms, driving scientific findings on their trade-offs. *RecLM* variants attain state-of-the-art recommendation accuracy and serves as a practical tool for real-world model training and deployment.

Limitations

While RecLM-cgen and RecLM-Token demonstrate significant improvements in recommendation accuracy and successfully address the out-of-domain item generation problem, several limitations warrant further discussion and investigation.

5.1 Inference Latency and Scalability

The inference latency of LLM-based generative recommendations presents challenges for large-scale, real-time services that require millisecond-level response times. While our framework employs prefix-tree constrained decoding to reduce search space, the autoregressive nature of token-by-token generation remains computationally intensive. Future work should implement and benchmark specific optimization techniques: *model distillation* to create smaller, specialized variants; *speculative decoding* to accelerate generation; *quantization-aware training* for efficient deployment; and *hybrid architectures* that combine retrieval efficiency with generative refinement. These optimizations could make RecLM variants practical for industrial-scale applications.

5.2 Evaluation Beyond Accuracy

Our evaluation focused primarily on accuracy metrics (NDCG, Hit Rate), but comprehensive recommender system assessment requires examining diversity, fairness, and long-term user satisfaction. Future work should implement *multi-objective optimization* during training, incorporating diversity constraints and fairness regularizers. For evaluation, we recommend adopting established metrics like *intra-list diversity*, *coverage*, and *equity measures* across demographic segments. Additionally, *longitudinal user studies* and *online A/B testing* frameworks are needed to assess real-world impact beyond offline metrics. These enhancements would provide a more holistic assessment of RecLM's practical utility.

References

Andrea Bacciu, Enrico Palumbo, Andreas Damianou, Nicola Tonellotto, and Fabrizio Silvestri. 2024. Generating query recommendations via llms. *arXiv preprint arXiv:2405.19749*.

Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yancheng Luo, Chong Chen, Fuli Feng, and Qi Tian. 2025. A bi-step grounding paradigm for large language models in recommendation systems. *ACM Trans. Recomm. Syst.* Just Accepted.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.

Yixin Dong, Charlie F Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. 2024. Xgrammar: Flexible and efficient structured generation engine for large language models. *arXiv preprint arXiv:2411.15100*.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2023. Recommender ai agent: Integrating large language models for interactive recommendations. *arXiv preprint arXiv:2308.16505*.

Jianchao Ji, Zelong Li, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Juntao Tan, and Yongfeng Zhang. 2024. Genrec: Large language model for generative recommendation. In *Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part III*, page 494–502, Berlin, Heidelberg. Springer-Verlag.

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

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. 2022. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11523–11532.

Jianxun Lian, Yuxuan Lei, Xu Huang, Jing Yao, Wei Xu, and Xing Xie. 2024. Recai: Leveraging large language models for next-generation recommender systems. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 1031–1034.

Xinyu Lin, Haihan Shi, Wenjie Wang, Fuli Feng, Qifan Wang, See-Kiong Ng, and Tat-Seng Chua. 2025.

Order-agnostic identifier for large language model-based generative recommendation. In *Proceedings of the 48th international ACM SIGIR conference on research and development in information retrieval*, pages 1923–1933.

Wensheng Lu, Jianxun Lian, Wei Zhang, Guanghua Li, Mingyang Zhou, Hao Liao, and Xing Xie. 2024. Aligning large language models for controllable recommendations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 8159–8172. Association for Computational Linguistics.

Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Chris Leung, Jiajie Tang, and Jiebo Luo. 2024. LLM-rec: Personalized recommendation via prompting large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 583–612, Mexico City, Mexico. Association for Computational Linguistics.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197.

Alan Said. 2025. On explaining recommendations with large language models: a review. *Frontiers in Big Data*, 7:1505284.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. 2024. Idgenrec: Llm-recsys alignment with textual id learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 355–364, New York, NY, USA. Association for Computing Machinery.

Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 2400–2409, New York, NY, USA. Association for Computing Machinery.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, and 1 others. 2024. A survey on large language models for recommendation. *World Wide Web*, 27(5):60.

Fan Yang, Zheng Chen, Ziyan Jiang, Eunah Cho, Xiaojiang Huang, and Yanbin Lu. 2023. Palr: Personalization aware llms for recommendation. *arXiv preprint arXiv:2305.07622*.

Jing Yao, Wei Xu, Jianxun Lian, Xiting Wang, Xiaoyuan Yi, and Xing Xie. 2023. Knowledge plugins: Enhancing large language models for domain-specific recommendations. *arXiv preprint arXiv:2311.10779*.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024. Recommendation as instruction following: A large language model empowered recommendation approach. *ACM Trans. Inf. Syst.* Just Accepted.

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

Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. 2024. Collaborative large language model for recommender systems. In *Proceedings of the ACM on Web Conference 2024*, pages 3162–3172.

A Appendix

A.1 Data Augmentation and Experiment Setup

During the training process of RecLM-ret, RecLM-cgen(Excluding the training of the TR) and RecLM-token, we employ an online data augmentation strategy. For each user's historical interaction records, denoted as $I_{history}^{(1...n)}$, we randomly sample a continuous segment $I_{history}^{(a...b)}$ where $1 \leq a < b \leq 10 < n$, to serve as the augmented training data. To construct the corresponding training labels for $I_{history}^{(a...b)}$, denoted as $I_{rec}^{(1...k)}$, where k is a random integer between 1 and 10, we follow the method described in (Lu et al., 2024). Specifically, $I_{rec}^{(1)}$ corresponds to the next interaction item $I_{history}^{(b+1)}$, while $I_{rec}^{(2...k)}$ are provided by the teacher model SASRec based on $I_{history}^{(a...b)}$.

Before the start of each epoch, data augmentation sampling is performed for every user. As a result, the training data for each epoch corresponds to the total number of users in the dataset, with online augmentation ensuring greater diversity in the training samples. During the testing phase, no data augmentation is applied. Instead, the number of test samples remains fixed at $10,000$.

A.2 Baseline Details

LLMs (frozen)

- **GPT-4o**: The *gpt-4o-2024-05-13* version accessed via Azure OpenAI.
- **Llama3**: The *Llama3-8b-instruct* model, which also serves as the base model for our tuning.
- **Llama3-cgen**: A prompt-based variant where Llama3 is instructed to output a special symbol `<SOI>` before mentioning an item, triggering our constrained generation decoding.

LLMs (finetuned)

- **BIGRec** (Bao et al., 2025): Fine-tunes the LLM to generate item-related text, which is then mapped to the item corpus using an embedding model (BGE-M3).
- **CtrlRec** (Lu et al., 2024): Focuses on controllability using two-stage training (supervised fine-tuning and reinforcement learning). It employs SASRec as a teacher model for data augmentation during SFT.
- **PALR** (Yang et al., 2023): Relies on SFT to learn recommendation tasks. We explicitly enable the SASRec-based data augmentation for PALR in our experiments to match the performance gains demonstrated in (Lu et al., 2024).

CtrlRec and PALR both use SASRec-based data augmentation for fair comparison, which can be found in Appendix A.1.

LLMs (item tokenizer)

- **IDGenRec** (Tan et al., 2024): Optimizes an item generator to dynamically adjust textual item IDs and utilizes a prefix tree for decoding.
- **SETRec** (Lin et al., 2025): Introduces an order-agnostic set identifier paradigm. It integrates collaborative filtering and semantic information while employing a query-guided mechanism to enable simultaneous generation for enhanced efficiency.
- **LC-Rec** (Zheng et al., 2024): Addresses the semantic gap by utilizing a vector quantization-based item indexing mechanism with uniform semantic mapping. It employs multi-faceted alignment tuning tasks to integrate collaborative and language semantics.
- **LETTER** (Wang et al., 2024): Learns a hierarchical item tokenizer through codebooks and decodes items using prefix trees.

Dataset	#Users	#Items	#Inters	#Sparsity
Steam	10,000	11,726	170,000	99.85%
Movies	10,000	34,452	170,000	99.95%
Toys	10,000	49,985	170,000	99.96%

Table 5: General statistics of the three datasets used in our experiments.

A.3 Additional Training Details

Backbone and Fine-tuning. Unless otherwise stated, all RecLM variants are initialized from Llama3-8B-Instruct. User behavior sequences are truncated to at most 10 interactions and injected into instruction-style prompts as user profiles, with the maximum input and output lengths both set to 512 tokens. We fine-tune all linear layers of the backbone using LoRA (PEFT) with the Adam optimizer, a learning rate of 1×10^{-4}, LoRA rank $r = 16$, scaling factor $\alpha = 8$, and a batch size of 2. Training typically converges within 20 epochs. The embeddings of the control symbols `<SOI>` and `<EOI>` are initialized as the average of the token embeddings for the phrases “start of an item” and “end of an item”, respectively. All quantitative results are averaged over five independent runs, and we report significance using paired tests with $p < 0.05$.

Item Tokenization Setup. The item tokenizer is trained on all items that appear in the interaction logs. For each item, we combine its collaborative embedding with its semantic embedding and feed them as inputs, while discrete item codes are learned in an unsupervised manner. We use residual vector quantization with four codebooks, each containing 256 entries, so that every item is represented by a tuple of four codes. The tokenizer is optimized by minimizing a reconstruction loss together with a commitment-style quantization loss to encourage stable code assignments; we set the weighting coefficient $\lambda_6 = 1.0$ and the commitment parameter $\beta = 0.25$ in all experiments. To assess tokenization quality, we monitor both reconstruction error and collision rate, defined as the proportion of distinct items that are mapped to identical code tuples.

Reinforcement Learning Setup. Reinforcement learning data are constructed on-the-fly from raw user interaction sequences. Given a sequence $I_{hist} = [i_1, i_2, \ldots, i_N]$, we randomly sample a split point t and use the prefix $I_{hist,<t}$ as the observed history, while the item i_t is treated as the ground-truth target for reward computation and is never exposed to the model during generation. The history is formatted as an instruction-style prompt asking the model to produce a top-K recommendation list. During RL training, we sample 16 candidate recommendation lists per prompt to enable relative reward normalization and stable optimization. Unless otherwise specified, we use a sampling temperature of 1.0, a maximum generation length of 128 tokens, a learning rate of 1×10^{-5}, and a batch size of 32, with a KL regularization coefficient of 0.04. For LoRA during RL, the rank and scaling factor are set to 16 and 32, respectively. We fix the recommendation list length to $K = 10$ and the maximum history length in RL to 20, matching the main-sequence setting described in the experiment section.

A.4 Projection Layer of RecLM-ret

In RecLM-ret, to align the hidden representation $\mathbf{h}^{(i)}_{<SOI>}$ of the base model with the vector space of the pre-generated item embeddings $\mathcal{E}$, we introduce a projection layer. Its formulation is shown in Equation 17.

$$\text{proj}_{\phi}(\mathbf{h}^{(j)}_{<SOI>}) = GELU(\mathbf{h}^{(j)}_{<SOI>} \cdot \mathbf{W}_1) \cdot \mathbf{W}_2 \tag{17}$$

Here, $\mathbf{W}_1 \in \mathbb{R}^{d \times \frac{d}{2}}$ and $\mathbf{W}_2 \in \mathbb{R}^{\frac{d}{2} \times c}$ constitute the trainable parameters ϕ of the project layer. d is the dimension of base model. c is the dimension of item embeddings $\mathcal{E}$.

A.5 Prompt Settings in RecLM-ret and RecLM-cgen

We provide the prompts in Listing 1 which are used to convert user behaviors <$I^{(1\ldots n)}_{history}$, $I^{(1\ldots k)}_{rec}$> into Supervised Fine-Tuning data samples <Instruction:X, Response:Y>. Io increase the data diversity, we use four prompt templates.

A.6 Prefix Tree Structure and Constrained Generation

We construct a prefix tree based on the item titles within the given recommendation domain. This prefix tree is represented as $Node = n_1, ..., n_a$ and $Children = C_1, ..., C_a$, where a is the number of nodes in the prefix tree, and C_i is a set indicating the child nodes of node n_i. To avoid recommending the same item multiple times within the same response, we record the number of leaf nodes under the subtree of each node in this prefix tree as $L = l_1, ..., l_a$ (where l_i is the number of leaf nodes in the subtree corresponding to node n_i, indicating the maximum number of times node n_i can be accessed within a single response).

At a certain generation step during the inference phase, the input token sequence is $X = [t_1, ..., t_i]$, and the token sequence of the generated response is $Y = [t_{i+1}, ...t_{i+j}]$. We look for the most recent control token in sequence Y. If the most recent control token is `<SOI>` (Start of Item), then we activate the constrained decoding strategy. If the most recent control token is `<EOI>` (End of Item) or no control token has been generated yet, the constrained decoding strategy is not activated.

When the constrained decoding strategy is activated (the most recent control token t_{i+k} is `<SOI>`, where $1 \leq k \leq j$), we first need to count the access times of the recommended items in the generated response at their corresponding nodes in the prefix tree, denoted as $V = v_1, ..., v_a$, where $v_i \leq l_i$. Next, we locate the corresponding node n_b in the prefix tree based on the sequence $[t_{i+k}, ..., t_{i+j}]$ and obtain the set of candidate next tokens C_b. To avoid generating duplicate items, we exclude nodes in C_b whose access times have reached the maximum access times, resulting in $C_b^{'}$ as the final candidate set for token t_{i+j+1}. We set the logit values of tokens outside $C_b^{'}$ to negative infinity to prevent them from being generated.

A key feature of RecLM-cgen is its simplicity in inference, as demonstrated in Figure 3.

A.7 Discussions on RecLM-cgen vs. RecLM-ret

In this section, we provide some theoretical perspective on why RecLM-cgen tends to achieve higher recommendation accuracy than RecLM-ret.

The main difference in the paradigms of RecLM-cgen and RecLM-ret is **Single-Stage Generation** vs. **Two-Stage Retrieval**. RecLM-ret relies on a

```
class FastPrefixConstrainedLogitsProcessor(LogitsProcessor):
    def __init__(
        self,
        item_title_set: list[str],
        start_control_symbol: str,
        end_control_symbol: str,
        tokenizer
    ):
    ... ...

logits_processor = FastPrefixConstrainedLogitsProcessor(
    item_title_set,           # all in-domain item titles
    start_control_symbol,     # start control symbol token
    end_control_symbol,       # end control symbol token
    tokenizer,                # modeltokenizer
)
```

Figure 3: Example implementation of RecLM-cgen during inference. Only minimal code modifications are required to integrate the constrained generation mechanism

two-step process:

1. Generate a special `<SOI>` token.
2. Perform a similarity-based lookup in an external embedding index to select the item.

This split can degrade accuracy in two ways. First, any mismatch between the model's hidden-state embedding and the item corpus embeddings may select a suboptimal item. Second, because the retrieval is effectively an external "hard choice," it does not benefit from token-by-token language modeling feedback, i.e., once `<SOI>` is emitted, the model's subsequent text has no bearing on which item is retrieved.

RecLM-cgen, by contrast, never leaves its native autoregressive process. Once `<SOI>` is produced, the model continues to generate tokens for the item title, except it restricts that token distribution to valid item titles stored in a prefix tree. In other words, each token that forms the recommended item is chosen within the model's next-token probabilities. On the one hand, there is no embedding mismatch. The model's hidden state directly translates into item-token predictions, rather than relying on an external embedding query. On the other hand, it is using a unified generative signal. Every generated token refines the item selection process. The model's full contextual understanding, such as user preferences, conversation history, etc., affects which item tokens appear.

Mathematically, we can view RecLM-ret as factorizing the recommendation process into:

$$P(\text{item}) \approx \mathrm{NN}\big(\phi(\mathbf{h}_{\texttt{<SOI>}}), \mathbf{E}\big) \qquad (18)$$

where ϕ is a projection of the model's hidden state, and $\mathbf{E}$ is the precomputed item embedding base. Small errors in $\phi(\mathbf{h}_{\text{SOI}})$ can lead to suboptimal recommendations.

Conversely, RecLM-cgen effectively implements:

$$P(\text{item} \mid \text{context}) = \prod_i P_\theta(w_i \mid w_{<i}, \text{context}), \qquad (19)$$

with the prefix-tree constraint filtering out invalid tokens. This direct language modeling over the item strings harnesses the entire generative capacity of the LLM, typically converging to higher recommendation accuracy during training. On the one hand, the model is trained to maximize the probability of each correct token in an item title, directly linking language modeling loss to better item predictions; on the other hand, there is no discontinuity between item selection and item-text generation, each token reflects the same internal distribution that learned the user's context.

B Title Rewriter Training Details

B.1 The Design of the Reward

Since title rewriting lacks a single optimal answer, we train the TR using the GRPO algorithm with reinforcement learning. We design five reward functions to guide generation quality, focusing on recommendation effectiveness, LLM compatibility, appropriate length, and semantic distinctiveness from yet closeness to the original.

B.1.1 Recommendation-Oriented Rewards

The first category of rewards is designed to enhance the performance of the recommendation system. We consider two perspectives in this context: item-to-item similarity and user-to-item alignment. Since the generated titles serve as item identifiers, it is important that they should help in efficient item discovery and user profiling.

Item-to-Item Reward For the item-to-item perspective, the generated short titles should improve the identification of similar items. This is motivated by the observation that recommendation systems typically recommend items based on similarity. To quantify item similarity, we construct a contribution matrix $C \in \mathbb{R}^{N \times N}$, where each entry C_{ij} represents the number of users who have interacted with both item i and item j, and N is the number of items. To mitigate the influence of popularity bias, we normalize the similarity scores as follows:

$$S_{ij} = \frac{C_{ij}}{|C_{i\cdot}| \cdot |C_{\cdot j}|} \quad (20)$$

Here, $|C_{i\cdot}|$ and $|C_{\cdot j}|$ denote the number of users who interacted with items i and j, respectively.

During training, the TR samples multiple candidate titles for each item. These rewritten titles are combined with the item descriptions and embedded using the BGE-M3 model. We then compute the cosine similarity between the resulting embedding and the embeddings of other original items, and compare the similarity-based ranking to the normalized similarity scores derived from the contribution matrix. Specifically, we select the top-10 items most similar to the current item based on the original contribution scores. After rewriting, we assess how the similarity-based ranking of these items changes by computing the Spearman's rank correlation coefficient between the original and updated rankings.

User-to-Item Reward From the perspective of user-item interaction, the objective is to evaluate whether the generated item titles better reflect user preferences. Each user's interaction history is represented by K items. During the training of the TR, the TR is required to generate new titles for all K items in a single sampling step. Based on these rewritten titles, we use RecLM-ret to obtain the user's final embedding by computing the user representation from the updated item history via a projection layer. We then assess the alignment between this new user embedding and the embedding of a target item (i.e., the item to be recommended). Specifically, we calculate cosine similarities between the user embedding and all item embeddings, determine the ranking position of the target item among these, and convert this ranking into an appropriate reward value.

B.1.2 Decoding Complexity

To ensure that the generated titles are easy to interpret and process by language models, we assess their decoding complexity using Perplexity (PPL). A lower PPL indicates that the title is more natural and easier for a language model to decode. During training, we obtain each user's interaction history and require the TR to rewrite the target item's title. The user's interaction history and the rewritten target item are then concatenated into a single input sequence. We compute the log-likelihood of the target item portion within this sequence and use it to calculate the PPL of the newly generated item title.

B.1.3 Conciseness Reward

In addition to semantic quality, conciseness is another desirable property of generated titles. We define a length-based reward to encourage the generation of more concise titles. This reward is computed by taking the ratio of the length of the newly generated title to that of the original title, and then converting this ratio into a reward value, where lower ratios correspond to higher rewards. This design aims to minimize the length of the rewritten title, thereby promoting conciseness.

B.1.4 Discriminative Power Reward

Finally, the generated title should be distinguishable from the titles of semantically similar items. To evaluate this property, we design a discrimination task where the generated title is used as a prompt to a language model, which is then asked to identify the correct original title from a set of four candidates: the true original title and the titles of the three most similar items. These similar titles are selected based on cosine similarity between item embeddings. A higher identification accuracy indicates that the generated title is more distinctive and recognizable. The prompt template at this stage is as follows:

B.2 Parameter Settings for Training

The TR is trained using the GRPO with five reward functions. The base model is Llama3-8B-Instruct, fine-tuned with LoRA (rank 16, $\alpha = 32$). vLLM is used for efficient generation, with 4 completions sampled per input and a temperature of 0.6. Training is conducted with bfloat16 precision, a learning rate of 5×10^{-6}, for 1 epoch, using a per-device batch size of 4 and gradient accumulation steps of 2. Optimizer and model states are offloaded to reduce memory usage. Training was performed on 2 NVIDIA H100 GPUs and took approximately 6 hours to complete.

The optimal weights for the reward functions were determined empirically for each dataset. consistently, we set the weights for Conciseness (CR), Discriminative Power (DPR), and Item-to-Item Similarity (I2I) to 1 across all datasets. The weights for Decoding Complexity (DC) and User-to-Item Alignment (U2I) were tuned specifically: for the Steam dataset, we utilized $w_{DC} = 2.5$ and $w_{U2I} = 3$. For both the Movies and Toys datasets, we adopted $w_{DC} = 2$ and increased the alignment weight to $w_{U2I} = 4$.

During training, four reward functions—Item-to-Item Similarity, Decoding Complexity, Conciseness, and Discriminative Power—are applied to the STR data, while the User-to-Item Alignment reward is excluded. In contrast, the GTR data training exclusively employs the User-to-Item Alignment reward, as it is more consistent with the group-level modeling objective.

B.3 Model Selection during Training

The TR module is built upon the Llama3-8B-Instruct model as its base architecture. For semantic embedding of rewritten titles and item descriptions, we employ the BGE-M3 model. To ensure consistency across different reward components, the decoding complexity and discriminative power evaluation stages also rely on Llama3-8B-Instruct.

B.4 Prompt Settings in TR

During the GRPO training phase of TR, we design two types of tasks based on the requirements of the title rewriting objective. The first task requires the TR to rewrite a single title, referred to as single-title rewriting(STR). The second task involves rewriting a group of titles in a single pass and producing them in order, referred to as group-title rewriting(GTR). The prompt settings for both tasks are shown in the Figure 4.

C Additional Experiments

C.1 Prefix Tree Construction and Complexity

The prefix tree used in RecLM-cgen (and RecLM-token) is built once offline using a trie over all (rewritten) item titles. The construction cost scales approximately linearly with the total number of title tokens, and during inference the decoding cost depends on the tree depth (i.e., title length) rather than the catalog size, since only the children of the current trie node are considered when masking logits. As summarized in Table 6, scaling the catalog from 10k to 1M items leads to only a marginal increase in end-to-end inference time, confirming that constrained generation remains efficient even for very large item corpora.

C.2 Cold-start Study

In this section, we conducted experiments to study the cold start capability of the model. For each dataset, we include 10k new users who have not been included in the training set, and limit their history randomly within the length range 4 to 10. The target items for all these 10k users are not included in the training set, which simulate the cold-start item scenario. The results are shown in Table 7.

Prompt Template: Single Title Rewriting (STR)

Item's Information:
Title: `{item_title}`
Description: `{item_description}`
Category: `{item_tags}`

Instructions:
1. Rewrite the title to be concise, clear, and descriptive.
2. Ideally shorter than the original, capturing key features.
3. Ensure the new title flows naturally.
4. **Output ONLY the rewritten title. No explanations.**

Prompt Template: Group Title Rewriting (GTR)

Input List: `{item_list}`

Instructions:
- Rewrite each title in the list to be concise and descriptive.
- Keep it short, capturing key features.
- **Output strictly in the following format:**
  - 1. [new title]
  - 2. [new title]
  - ...
- **Do not include extra text.**

Figure 4: Prompt templates used for the STR and GTR tasks. Variable placeholders are denoted in brackets.

Catalog	Build (s)	Inference (10k users)
10k	0.08	8 m 37 s
100k	2.71	8 m 44 s
1M	29.78	8 m 45 s

Table 6: Scalability of prefix tree construction and constrained generation on catalogs of different sizes.

C.3 Cross-domain Recommendation

We additionally evaluate the cross-domain behavior of RecLM-cgen. Table 8 reports two zero-shot settings: training on Toys and testing on Movies, and training on Movies and testing on Toys. For clarity, we group RecLM-cgen variants using subscripts, where Our denotes the base configuration (v0 in section 4.3), and subscripts cg, sm, and mr indicate the inclusion of constrained generation, scope mask, and multi-round conversation training, respectively. Across both cross-domain scenarios,

Dataset	Model	HR@10	NDCG@10
Steam	Llama3-8b-cgen	0.0039	0.0021
	RecLM-cgen	0.0086	0.0057
Movies	Llama3-8b-cgen	0.0162	0.0089
	RecLM-cgen	0.0422	0.0291
Toys	Llama3-8b-cgen	0.0202	0.0100
	RecLM-cgen	0.0313	0.0183

Table 7: Cold-start performance comparison on the three datasets.

D_{train}	D_{test}	Model	HR@10 ↑	NDCG@10 ↑
Toys	Movies	Llama3-cgen	0.0246	0.0106
		$\mathrm{Our}_{\mathrm{cg}}$	0.0743	0.0651
		$\mathrm{Our}_{\mathrm{cg+sm}}$	0.0953(+28.26%)	0.0817(+25.50%)
		$\mathrm{Our}_{\mathrm{cg+mr}}$	0.0745	0.0648
		$\mathrm{Our}_{\mathrm{cg+sm+mr}}$	**0.1170 (+57.05%)**	**0.1029 (+58.80%)**
Movies	Toys	Llama3-cgen	0.0354	0.0153
		$\mathrm{Our}_{\mathrm{cg}}$	0.0503	0.0384
		$\mathrm{Our}_{\mathrm{cg+sm}}$	**0.0572 (+13.72%)**	**0.0447 (+16.41%)**
		$\mathrm{Our}_{\mathrm{cg+mr}}$	0.0481	0.0366
		$\mathrm{Our}_{\mathrm{cg+sm+mr}}$	0.0527(+9.56%)	0.0414(+13.11%)

Table 8: Cross-domain recommendation results. Our proposed models progressively integrate semantic memory (sm) and multi-round reasoning (mr) on top of base conversational generation (cg).

incorporating the scope mask component (sm) consistently yields better performance than its counterparts without sm, indicating that aligning the training objective with the constrained decoding space improves robustness under domain shift.

C.4 Inference Speed on RecLM-cgen

Dataset	cg	Token_{in}	Token_{out}	$\mathrm{Speed}_{.avg}$ (token/s)	Search Time_{in} (ms/token)	Search Time_{out} (ms/token)
Steam	w/	7726	7552	35.0385	1.0725	0.3234
	w/o	7872	7552	36.6996	-	-
Movies	w/	12970	7552	34.5347	1.4535	0.3221
	w/o	11900	7552	36.3846	-	-
Toys	w/	20838	7552	34.0883	1.9922	0.3237
	w/o	19910	7552	36.9466	-	-

Table 9: Computation cost analysis of constrained generation (cg) during inference across three datasets. Results are compared between models with (w/) and without (w/o) constrained generation.

To illustrate that the constrained generation does not cause significant latency on the LLM inference, we conduct an inference throughput experiment. We select 128 test samples from the test set of three datasets, generating 10 item recommendations per test sample. The model is deployed using the Hugging Face Transformers library[4] on a single A100 GPU (40GB), with an inference batch size set to 1. We used 5 test samples for warm-up and ignored the time it took to generate the first token. We then aggregate the number of inner prefix tree tokens ($Token_{in}$) and outer prefix tree tokens ($Token_{out}$), calculating the average search time for both token types in the settings. Here search time corresponds to the operation to determine the valid space in next token decoding. Table 9 shows the average results of 5 repeated experiments, numbers are aggregated from the response text of the 128 test samples, we report both settings with and without constrained generation.

For the Steam dataset, with constrained generation enabled, a total of 7,726 inner tokens and 7,552 outer tokens are generated. The average generation speed is 35.0385 tokens/second. The average search time for inner tokens is 1.0725 ms/token, while for outer tokens, it is 0.3234 ms/token. As the length of item titles increases from the Steam dataset (6.0359 tokens/item) to the Movies dataset (10.1328 tokens/item) and further to the Toys dataset (16.2797 tokens/item), the search time for outer tokens remains stable, whereas the search time for inner tokens gradually increases.

C.5 In Context Learning Study

We conducted experiments comparing the performance of contextual learning with RecLM-cgen. In Table 10, Llama3-8b-cgen refers to the unfinetuned LLM with constrained generation enabled. We observed that performance actually drops when using a 5-shot setting. This finding aligns with existing literature(Yao et al., 2023). The reason is that LLMs can become severely biased towards the provided few-shot examples. Unless these examples are dynamically retrieved by a recommender model tailored to the current data sample, improvements from few-shot strategies are unlikely. Therefore, we recommend fine-tuning the LLM to some extent for better recommendation performance.

C.6 Sensitivity Analysis of Reward Weights

This section provides a sensitivity analysis of the reward weight configuration used in the TR module. We vary the five reward weights in Eq. (4) while keeping all other training settings fixed, and report performance on the Steam dataset.

Table 11 shows that the model performance remains relatively stable across different weight combinations. Although certain configurations yield marginally higher values on specific metrics, no single weight dominates the performance, and rea-

---

[4] https://github.com/huggingface/transformers

Dataset	Model	HR@10	NDCG@10
Steam	Llama3-8b-cgen (0-shot)	0.0261	0.0125
	Llama3-8b-cgen (5-shot)	0.0211	0.0112
	RecLM-cgen (Ours)	**0.0797**	**0.0433**
Movies	Llama3-8b-cgen (0-shot)	0.0246	0.0106
	Llama3-8b-cgen (5-shot)	0.0121	0.0059
	RecLM-cgen (Ours)	**0.1424**	**0.1296**
Toys	Llama3-8b-cgen (0-shot)	0.0354	0.0153
	Llama3-8b-cgen (5-shot)	0.0303	0.0137
	RecLM-cgen (Ours)	**0.0642**	**0.0479**

Table 10: In-context learning (ICL) comparison across three datasets.

Component Weights					**Evaluation Metrics**			
DC	CR	DPR	I2I	U2I	H@10	N@10	H@5	N@5
1	1	1	1	1	0.0820	0.0412	0.0560	0.0328
1	1	1	1	2	0.0900	0.0490	0.0630	0.0404
1	1	1	1	3	0.0780	0.0390	0.0530	0.0310
2	1	1	1	3	**0.0930**	0.0485	**0.0640**	0.0384
3	1	1	1	3	0.0890	0.0472	0.0580	0.0373
1	1	2	2	3	0.0920	**0.0496**	**0.0640**	**0.0406**
1	2	1	1	3	0.0860	0.0418	0.0500	0.0317
1	3	1	1	3	0.0820	0.0436	0.0570	0.0354

Table 11: Sensitivity analysis showing the impact of different weight configurations. H@K denotes HR@K and N@K denotes NDCG@K.

sonable variations do not lead to significant degradation. This indicates that the TR module does not rely on finely tuned hyperparameters and is robust to moderate changes in reward weighting.

Based on this analysis, we use a balanced default configuration for all reported experiments.

D Case Study

D.1 Title Rewriting for Enhanced Recommendation

As shown in Table 12, our case study illustrates that the TR module effectively combines item titles and descriptions to retain key information while removing redundancy. For example, it rewrites the complex game title "Ukrainian Ball in Search of Gas" as the concise "Gas Quest," adds contextual details to movies like "55 Days at Peking (1900 Boxer Rebellion)" for clearer disambiguation, and simplifies toy product names by dropping irrelevant phrases like "Colors May Vary" while preserving brand identity. These enhancements result in more concise, readable, and informative identifiers, thereby improving LLM comprehension.

Field	Content
Case 1: Steam Games	
Original Title	Ukrainian Ball in Search of Gas
Description	... our hero touched it with a gorilka and turned into a ball! ... You play as a man turned into a ball and **you need to steal all the gas from the forest at all costs** ...
Rewritten	**Gas Quest**
Case 2: Movies	
Original Title	55 Days at Peking VHS
Description	Diplomats, soldiers and other representatives ... fend off the siege of the International Compound in Peking **during the 1900 Boxer Rebellion** ...
Rewritten	**55 Days at Peking (1900 Boxer Rebellion)**
Case 3: Toys	
Original Title	SwimWays Toypedo Revolution - Colors May Vary
Description	**The SwimWays Toypedo Revolution dive toy** rockets through the pool up to 30 feet with amazing hydrodynamic action! ...
Rewritten	**SwimWays Toypedo Dive Toy**

Table 12: Case study of the TR module.

D.2 Additional Case Study on Cross-domain Generalization

Table 13 presents an additional qualitative case study to further examine the cross-domain and cold-start behavior of the proposed framework. Unlike standard recommendation settings, cross-domain scenarios involve items that are entirely absent from the training interactions, making historical collaborative signals unavailable.

In this example, a RecLM-cgen model trained exclusively on the Steam dataset is prompted with an item from the Movies domain. Despite the domain shift and the lack of domain-specific interaction data, the model generates a coherent and semantically accurate description of the unseen item. This observation suggests that the model relies primarily on semantic cues derived from item titles rather than memorizing domain-specific interaction patterns.

D.3 Impact of Title Rewriting on Recommendation Outcomes

The case study in Table 14 illustrates how TR improves recommendation accuracy in different domains. In the movie domain, rewritten titles enhance semantic clarity, enabling the system to correctly rank relevant items—such as "X-Men: Apocalypse VHS"—within the top-10 recommendations,

Field	**Content**
Item Title	*Take the Money and Run* VHS
Original Desc.	Woody Allen's feature-film debut, *Take the Money and Run*, is a mockumentary combining sight gags, sketch-like scenes, and stand-up humor. Allen plays Virgil Starkwell, a music-loving nebbish who turns to a life of crime at an early age . . .
Generated Desc.	*Take the Money and Run* is a 1969 American comedy film directed by Woody Allen. The film stars Allen and Janet Margolin and is known for its fast-paced humor and mockumentary style. It is considered one of Allen's early works and showcases his distinctive comedic voice. The film is available in VHS format through various retailers.

Table 13: Cross-domain qualitative case study demonstrating the model's generation capabilities.

where previously it was missed. Similarly, in the toy domain, verbose and inconsistent original titles hindered accurate recommendations, but after rewriting, the system successfully identified and ranked items aligned with user preferences near the top. These examples demonstrate the practical impact of the TR module in enhancing recommendation relevance.

E Formal Definition of Out-of-domain (OOD) Recommendations

For each dataset, let $\mathcal{I}$ denote the finite domain catalog of in-domain items. Given a user context x (including interaction history and dialogue turns), a recommender f_θ produces a list of candidate recommendations $R(x) = (\hat{i}_1(x), \ldots, \hat{i}_T(x))$ in some output space (e.g., natural-language titles, item identifiers, or discrete codes). An individual prediction $\hat{i}_t(x)$ is *out-of-domain* if it cannot be mapped to any catalog item, i.e., $\hat{i}_t(x) \notin \mathcal{I}$. The out-of-domain recommendation problem is to design models and grounding mechanisms such that, for all user contexts x and all recommendation positions t, every generated item lies within the catalog, i.e., $\hat{i}_t(x) \in \mathcal{I}$.

Case	Stage	User History	Target Item	Top-10 Recommendations	Hit?
Case 1	Before	• The Hobbit: The Battle of the Five Armies [DVD] [2015] • Tinker Bell and the Legend of the Neverbeast • 55 Days at Peking VHS • The Incredibles (Mandarin Chinese Edition) • The Legend of Longwood • Batman vs. Robin • The Last Witch Hunter Digital • Batman: Bad Blood • Justice League vs Teen Titans (DVD) • Batman v Superman: Dawn of Justice	X-Men-Apocalypse - The Cure/Come The Apocalypse VHS	• The Martian • X-Men VHS • The Young Riders: The Series - Season 1-3 • The Last Witch Hunter Digital • The Martian: Extended Edition 4K Ultra-HD • The Revenant • The Hunger Games: Catching Fire 2013 • The Hobbit: The Desolation of Smaug • The Hobbit: The Battle of the Five Armies [DVD] [2015] • The Hobbit: An Unexpected Journey	✗
	After	• The Hobbit: Battle of the Five Armies [DVD] [2015] • Tinker Bell and the Neverbeast • 55 Days at Peking (1900 Boxer Rebellion) • The Incredibles (Mandarin Chinese Edition) • The Legend of Longwood: A Magical Quest • Batman vs. The Court of Owls • The Last Witch Hunter • Batman: Dark Knight Down • Justice League vs Teen Titans: A Heroic Showdown • Batman V Superman: Dawn of Justice (2016)	X-Men: Apocalypse VHS (2017)	• The Hobbit: An Unexpected Journey (2012) • Captain America: Civil War (1) • The Martian Stranded • Guardians of the Galaxy (2014) [Region Free] • **X-Men: Apocalypse VHS** • Jurassic World • Avengers: Ultron Rising • Ant-man: The Tiny Hero • The Revenant (Multi-Region PAL/NTSC DVD) • The Big Short: A Tale of Financial Forecasts	✓
Case 2	Before	• Barbie Fashion Complete Look 2-Pack, Pop Concert Set • Barbie Complete Look Fashion Pack #5 • Barbie Style Doll, Jean Jacket and Black/White Skirt • Barbie Style Nikki Doll • Barbie Fashion Complete Look 2-Pack, Movie Set • Barbie Style Nikki Doll • Barbie Fifth Harmony Lauren Doll • Barbie Complete Look Fashion Pack #3 • Barbie Style Summer Doll with Pink Paisley Dress and Jacket • Barbie Complete Look Fashion Pack #4	Barbie Complete Look Fashion Pack #8	• Barbie Style Doll, Black and Silver Jacket • Barbie Style Teresa Doll • Barbie Style Midge Doll • Barbie Style Raquelle Doll, Leopard Print Jacket • Barbie Style Nikki Doll • Barbie Style Doll, Jean Jacket and Black/White Skirt • Barbie Style Glam Doll with Pink Retro Print Dress • Barbie Style Barbie Doll • Barbie Style Summer Doll with Pink Paisley Dress and Jacket • Barbie Style Nikki Doll	✗
	After	• Barbie Fashion 2-Pack: Pop Concert Outfits • Barbie Fashion Pack #5 • Barbie Fashion Doll with Jean Jacket and Skirt • Barbie Fashion Doll (1) • Barbie Fashion 2-Pack: Movie Date Outfits • Barbie Style Nikki Doll: Fashionista Friend • Barbie Fifth Harmony Lauren Doll • Barbie Fashion Pack: 6 Dresses, 2 Coats, Shoes & Purse • Barbie Fashion Doll with Pink Dress and Jacket • Barbie Fashion Pack #4: 6 Dresses, 2 Coats, Shoes, and Purse	Barbie Fashion Pack #8	• Barbie Fashion Doll with Dark Hair • **Barbie Fashion Pack #8** • Barbie Fashion Pack #6: 6 Dresses, 2 Coats, Shoes & Purse • Barbie Fashion Doll with Leather Jacket and Accessories • Barbie Fashion Doll with Black and Silver Jacket • Barbie Fashion Doll with Jean Jacket and Skirt • Barbie Fashion Pack #3: Dresses, Shoes, and Handbags • Barbie Fashion Doll with Pink Dress and Jacket • Barbie Fashion Doll with Accessories • Barbie Fashion Pack #5	✓

Table 14: Comparison of recommendation results. (✓) Success, (✗) Failure.

```
System: You are an expert recommender engine as well as a helpful, respectful and honest assistant.

Instruction 1: You need to generate a recommendation list considering user's preference from
    historical interactions.The historical interactions are provided as follows: {history}. You
    need to generate a recommendation list with {item_count} different items. Each item should be
    enclosed by <SOI> and <EOI>. <SOI> should be generated before item title, and <EOI> should be
    generated after item title.
Output: {item_list}

Instruction 2: You need to select a recommendation list considering user's preference from
    historical interactions.The historical interactions are provided as follows: {history}. The
    candidate items are: {candidate_titles}. You need to select a recommendation list with
    {item_count} different items from candidate items. Each item should be enclosed by <SOI> and
    <EOI>. <SOI> should be generated before item title, and <EOI> should be generated after item
    title.
Output: {item_list}

Instruction 3: Your task is generating a recommendation list according user's preference from
    historical interactions.The historical interactions are provided as follows: {history}. Please
    generate a recommendation list with {item_count} different items.
Output: {item_list}

Instruction 4: Your task is selecting a recommendation list according user's preference from
    historical interactions.The historical interactions are provided as follows: {history}. The
    candidate items are: {candidate_titles}. Please select a recommendation list with {item_count}
    different items from candidate items.
Output: {item_list}
```

Listing 1: Prompts for training