PRISMIC: Reconstructing User Preference via Intent Decomposition and Consolidation

Donghee Han **Jiwon Jeong** **Hwanjun Song**† **Mun Yong Yi**†
KAIST, Republic of Korea
{handonghee, zzioni, songhwanjun, munyi}@kaist.ac.kr
†Corresponding authors

Abstract

Large language models (LLMs) have recently improved sequential recommendation, yet the task remains a retrieval problem without explicit user queries: the system must infer the next item from user histories where diverse intents and preferences are intertwined. Existing methods typically compress this heterogeneous evidence into a single embedding or query, while LLM-based recommenders can produce overly general queries that are semantically plausible but weakly aligned with retrieval. We propose **PRISMIC** (**P**reference **R**econstruction via **I**ntent **S**ynthesis and **M**ulti-signal **I**nference **C**onsolidation), which reformulates sequential recommendation as multi-signal intent consolidation. PRISMIC first generates multiple candidate queries from different historical signals, then trains an LLM-based consolidator with GRPO using an NDCG-based retrieval reward to merge relevant signals into a single query. The resulting consolidated query is not directly used at inference time; instead, it serves as a semantic supervision target for training a user encoder, enabling LLM-free inference. Across four real-world datasets, PRISMIC consistently outperforms strong baselines. Ablations further show that the gains come not from GRPO alone, but from consolidating multiple intent-signals and distilling the consolidated intent into an encoder.

1 Introduction

Sequential recommendation aims to infer a user's next item from an interaction history. A long line of work has addressed this problem with attention-based or sequence modeling architectures, which learn to directly extract useful signals from user histories [1, 2]. More recent methods extend this paradigm by using large language models (LLMs) as semantic encoders or generative query builders [3–6]. Despite their differences, these approaches largely share the same underlying assumption: a heterogeneous user history can be directly compressed into a single representation or query. We refer to this failure mode as *Premature Intent Collapse*: heterogeneous intent signals are compressed into a single semantic representation before the model has any mechanism to separate, compare, or suppress competing cues by their retrieval utility. The issue is not merely that the generated query is short or generic, but that the compression happens before the model has identified which historical signals are predictive of the next interaction. This failure becomes more visible when user histories contain intertwined short-term intents, long-term preferences, exploratory behaviors, and incidental interactions, where the model must identify the decisive next-intent signal only through the final compressed output.

In CF-based sequential recommendation, prior work has explored *multi-interest* modeling, where multiple intent vectors are extracted from a single history through attention or capsule routing [7–9]. However, this line of work has not yet been carried over to LLM-based recommendation: existing LLM recommenders still compress the entire history into a single semantic query or representation,

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

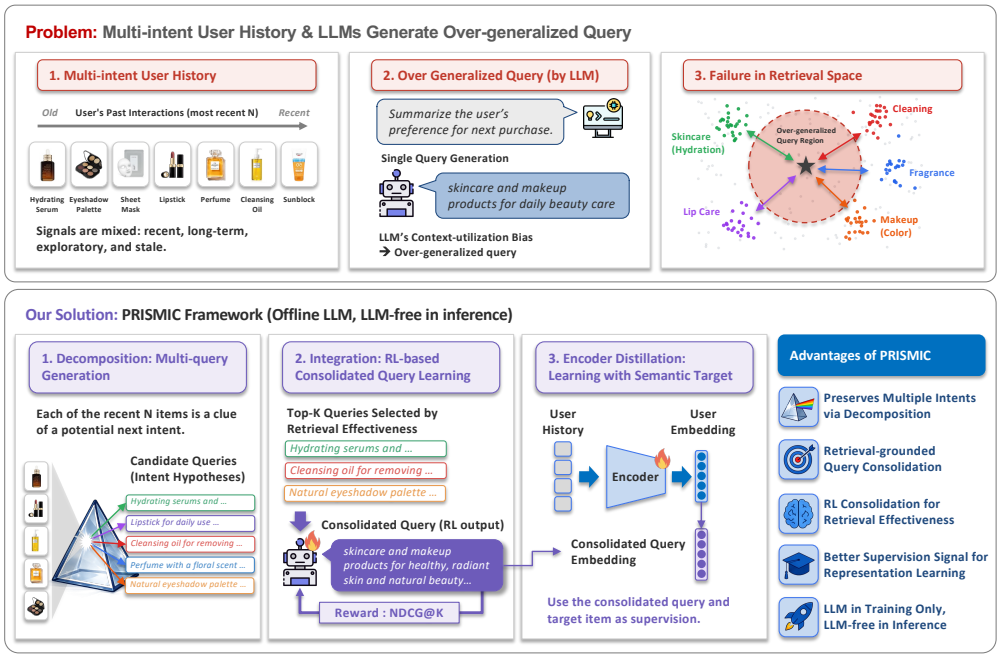


Figure 1: Motivation and overview of PRISMIC. To prevent LLMs from over-generalizing heterogeneous user histories, PRISMIC decomposes the history into multiple candidate intents, selects high-utility queries by training-only retrieval effectiveness, learns a retrieval-optimized consolidated query with GRPO, and distills the resulting semantic target into a user encoder.

without explicitly separating heterogeneous intent signals before consolidation. This limitation becomes more pronounced in LLM-based recommendation. Although LLMs can interpret rich item semantics and reason over user histories, they are also susceptible to context-utilization bias: given a long and heterogeneous history, an LLM tends to incorporate weakly related interactions into a fluent but overly broad query [10–12]. As a result, the generated query may appear semantically coherent while failing to preserve the fine-grained intent needed for accurate next-item retrieval. Thus, the central challenge is not merely to generate a better textual summary of the user history, but to avoid premature consolidation of multiple intent signals before the model has identified which ones are retrieval-effective.

Motivated by this view, we propose **PRISMIC**, **P**reference **R**econstruction via **I**ntent **S**ynthesis and **M**ulti-signal **I**nference **C**onsolidation. To avoid Premature Intent Collapse, PRISMIC adopts a *decomposition-and-consolidation* strategy. As illustrated in Figure 1, PRISMIC first decomposes the user history into multiple candidate queries, where each query is conditioned on a different historical item and represents a possible next-intent direction. These candidates are then evaluated by retrieval effectiveness, allowing PRISMIC to retain only the signals that are more predictive for the next item. This step prevents the model from aggregating all historical evidence indiscriminately and provides a more focused set of intent candidates for downstream learning.

PRISMIC then trains an LLM-based consolidator with GRPO [13] to merge the selected candidates into a single query. The consolidator is optimized with an NDCG-based reward, so the learned query is encouraged to improve item ranking rather than merely produce a coherent summary. Unlike Premature Intent Collapse, which learns only from the final output signal, PRISMIC exposes intermediate intent candidates and learns how to combine them based on retrieval utility. This decomposition-and-consolidation process enables the model to search for a more effective semantic target before representation learning.

Importantly, PRISMIC does not rely on LLM generation at inference time. The consolidated query is encoded and used as a semantic supervision target for training a lightweight user encoder. Thus, the LLM serves as an offline constructor of retrieval-aware supervision rather than an online recommender. At inference time, PRISMIC performs recommendation only with the trained encoder and projection module, enabling efficient LLM-free retrieval while preserving the benefit of LLM-guided intent consolidation. PRISMIC is therefore a supervised offline target-construction method,

not a zero-shot prompting method: the next item is used only as a training-user label to score candidate queries and define retrieval rewards, never inserted into LLM prompts and never accessed at validation, test, or inference time.

We evaluate PRISMIC under full-item-pool retrieval across four real-world datasets. PRISMIC consistently outperforms representative baselines, and ablations show that the gains arise from the combination of decomposition, learned consolidation, and encoder distillation rather than from any single component. Additional analysis further shows that PRISMIC is particularly effective when user histories contain conflicting but recoverable intent signals.

Our contributions are summarized as follows:

- We characterize *Premature Intent Collapse* as a recurring failure mode in LLM-based sequential recommendation, where heterogeneous histories are compressed into a single semantic query before competing intent signals are explicitly evaluated.
- We instantiate a decomposition-and-consolidation framework for LLM-assisted sequential recommendation, where multiple intent candidates are first generated and then selectively consolidated according to retrieval utility.
- We propose **PRISMIC**, a supervised offline semantic-target construction framework that generates candidate queries from different historical items, filters them by training-only retrieval effectiveness, and trains an LLM-based consolidator with GRPO under an NDCG-based reward.
- We introduce an offline distillation pipeline that converts the consolidated query into a semantic supervision target for user encoder training, enabling efficient LLM-free inference.
- We empirically show that PRISMIC outperforms representative baselines and that its improvements arise from the combination of decomposition, learned consolidation, and encoder distillation, rather than from any single component.

2 Related Work

2.1 Sequential and Denoising Recommendation

Sequential recommendation models predict the next item from ordered user interactions. Representative methods such as GRU4Rec [1], SASRec [2], and BERT4Rec [14] capture temporal dependencies with recurrent or self-attention architectures, while FDSA [15] and S^3-Rec [16] further exploit item features and self-supervised objectives. A related line of work studies denoising and robust recommendation, where noisy or unreliable interactions are removed, reweighted, or corrected during training. A separate line of multi-interest recommenders (MIND [7], ComiRec [8], Re4 [9]) explicitly model that a history may reflect multiple intents, but they obtain multi-interest representations through fixed routing, attention, or contrastive operators in the CF embedding space rather than multi-signal consolidation.

2.2 LLM-based Recommendation

LLMs support recommendation through reranking [17], classification [18], semantic encoding, and query generation. Several methods further align CF embeddings with LLM latent spaces to inject collaborative signals [19–21]. Others use LLMs to generate retrieval queries or item identifiers; for example, BIGRec [4] fine-tunes an LLM to generate search queries for full-item-pool retrieval.

Despite their promise, existing LLM-based recommenders often rely on one-shot semantic compression: a heterogeneous user history is summarized into a single query, representation, or reasoning path. This can lead to overly general outputs when multiple plausible intents coexist. Furthermore, methods that invoke LLMs during inference face substantial cost and latency limitations. More recently, LLMDiRec [22] injects LLM-derived semantic representations into intent-aware diffusion, showing that LLM semantics can improve latent intent modeling. However, such intents remain implicit in a representation or diffusion space, whereas PRISMIC explicitly decomposes a heterogeneous history into multiple retrieval-testable textual signals, consolidates them with retrieval-grounded feedback, and distills the resulting signal into an encoder-only retriever.

2.3 RL for LLM-based Recommendation

Reinforcement learning has been widely used to align LLMs with task-specific objectives. Compared with PPO [23], GRPO [13] reduces computational overhead by estimating relative advantages within sampled groups without a separate critic, making it attractive for verifiable reward settings. Recent LLM-based recommendation methods adopt GRPO-style objectives: Rec-R1 [5] optimizes generated queries with rank-based rewards, while RecZero [6] applies GRPO to rating prediction.

However, directly applying sparse retrieval rewards to full-item-pool sequential recommendation is difficult. When the initial policy rarely retrieves the target item, rewards can collapse to near-zero, producing sparse learning signals. PRISMIC instead uses GRPO to learn retrieval-grounded consolidation of multiple signals, then uses the consolidated query as a semantic supervision target for encoder distillation. Thus, RL is used for offline target construction rather than inference-time LLM recommendation.

3 Method: PRISMIC

PRISMIC learns a retrieval-oriented representation of user intent by moving LLM reasoning to an offline training stage. Instead of asking an LLM to directly compress the full interaction history into a single query, PRISMIC first decomposes the history into multiple item-conditioned intent signals, evaluates which intent signals are useful for retrieving the next item, and trains an LLM to consolidate the strongest evidence into a unified query. The resulting query is then used as a semantic supervision target for a lightweight recommendation encoder. At test time, recommendation is performed only with the trained encoder and projection layer, without invoking an LLM. A formal motivation for the decomposition-and-consolidation design under a retrieval reward, including the regime in which decomposition strictly improves over direct one-shot compression and why an RL-trained consolidator is no worse than fixed selection, is given in Appendix D.

The method consists of three stages, matching the three subsections of this section:

1. **Multi-query generation, evaluation, and selection** (Section 3.1). For each history item, an anchor-conditioned query is generated to describe a plausible next-purchase intent related to that item; each query is then evaluated by full-pool retrieval against the target item, and the top-K semantic queries are kept as the consolidator's evidence set.
2. **GRPO-based query consolidation** (Section 3.2). A small LLM is trained with GRPO under an NDCG-based retrieval reward to merge the selected intent signals into a single retrieval-effective query.
3. **Encoder training** (Section 3.3). The consolidated query embedding is used as an offline supervision target for training a lightweight user encoder, together with a contrastive ranking objective.

3.1 Multi-Query Generation and Retrieval Evaluation

A user history often contains multiple plausible intents, and asking an LLM to summarize them in one shot collapses these competing signals before their predictive utility can be distinguished. We therefore generate one query per history item so that each item-conditioned query acts as a separate intent signal whose retrieval usefulness can be measured independently.

For each user u and each history item $i \in H_u$, we generate an anchor-conditioned query using a frozen instruction-tuned LLM π_{base}. The prompt asks the model to infer a plausible next-purchase intent by foregrounding item i while still considering the full history. This produces a set of intent signals:

$$\mathcal{Q}_u = \{q_{u,i} \sim \pi_{\text{base}}(\cdot \mid H_u, i) \ : \ i \in H_u\}\,. \tag{1}$$

Each query is encoded with the frozen text encoder and used to retrieve items from the full candidate pool. We record the target rank obtained by each query and use it as an offline estimate of how useful the corresponding anchor is for explaining the next item. A lower rank indicates that the anchor-conditioned query provides a stronger semantic route to the target. The target item is used only as a training label for offline query evaluation. It is never included in the prompt, never appears inside the generated candidate queries by construction, and is not used when constructing representations

for validation or test users. This makes the selection step target-aware during training but not target-conditioned at generation or inference time.

3.2 Query Selection and GRPO-Based Consolidation

Once each query has a measured retrieval rank, the question is how to combine the selected queries into a single query that best aligns with the next item. Naive concatenation or averaging treats all selected queries equally, so we instead train an LLM consolidator with GRPO under an NDCG-based reward, which encourages the policy to place more weight on signals that improve the rank of the training target and less weight on signals that hurt retrieval. The expressive-space argument behind this choice, namely that a learned consolidator can copy any selected candidate as a special case but can also synthesize a new query from complementary fragments, is formalized in Appendix D.

After evaluating all anchor-conditioned queries, we construct a compact evidence set $\mathcal{Q}_u^{\text{top}}$ for each user by selecting the top-K queries with the best semantic retrieval ranks. We additionally require each selected query to retrieve the target within rank ≤ 100, so that the consolidator only sees evidence that is at least *somewhat* retrieval-effective; queries that fail this rank threshold are dropped from $\mathcal{Q}_u^{\text{top}}$. We use $K = 3$ throughout the experiments. We train the consolidator only on training users whose best candidate obtains a non-zero NDCG@K reward, so that GRPO receives at least one informative comparison within each sampled group. This filtering is applied only to the training split and only affects which users provide GRPO updates; all validation and test users are evaluated under the same full-item-pool protocol without filtering.

Given H_u and $\mathcal{Q}_u^{\text{top}}$, a small LLM policy π_θ generates a consolidated retrieval query:

$$q_u^{\text{con}} \sim \pi_\theta(\cdot \mid H_u, \mathcal{Q}_u^{\text{top}}). \tag{2}$$

The input contains the selected semantic queries together with anchor item metadata. The output is constrained to a consolidated query. The objective is not to produce a fluent user summary, but to synthesize a query whose embedding retrieves the target item well.

For each user, GRPO samples a group of candidate consolidated queries and scores each candidate with an NDCG-based retrieval reward. The reward is the NDCG@K score of the candidate query against the target item under full-pool retrieval, so candidates that rank the target item higher receive higher reward:

$$R(q, u) = \text{NDCG@}K(\pi_\phi(q),\, t_u \mid \mathcal{I} \setminus H_u)\,, \tag{3}$$

where $\pi_\phi(q)$ is the ranking induced by retrieving with the encoded query $\phi(q)$ over the candidate pool. GRPO normalizes rewards within each sampled group and updates the policy with a KL-regularized objective against the initial policy. After training, the optimized policy generates consolidated queries for the training set. These consolidated queries are then used as offline semantic supervision for encoder training.

3.3 Encoder Distillation and Inference

LLM-generated queries are informative but too expensive to invoke per request at deployment, so we distill the consolidated query embedding into a lightweight encoder that captures the same intent signal without an online LLM. We additionally pair the distillation target with an InfoNCE auxiliary loss so that the learned representation, beyond being semantically aligned, remains discriminative against negative items.

The consolidated queries from Section 3.2 are used purely as offline supervision; the LLM is not invoked at test time. For each training user u, the consolidated query q_u^{con} is encoded with the frozen text encoder ϕ to form a semantic target, and a lightweight user encoder f_ψ with a projection g_ω produces a user representation in the same space. We train the encoder with two complementary objectives: an MSE term that aligns the user representation with the consolidated query embedding, and an auxiliary InfoNCE term that contrasts the target item against in-batch negatives. The MSE term transfers LLM-derived transition intent, while the InfoNCE term sharpens item-level discrimination. Full training settings are deferred to Appendix F.

At inference time, PRISMIC does not generate multi-queries or invoke any LLM. Given a test user history H_u, the trained encoder produces a user representation $\mathbf{z}_u = g_\omega(f_\psi(H_u))$, and items are ranked by cosine similarity over $\mathcal{I} \setminus H_u$. PRISMIC thus preserves the benefit of LLM-guided intent reasoning during training while keeping inference as efficient as standard embedding-based retrieval.

Table 1: Comparison of PRISMIC with baseline methods. We report Hit@10 and NDCG@10 for all baselines across four datasets. PRISMIC uses only the trained encoder at test time, without LLM inference. The best result in each column is bolded, and the strongest baseline is underlined.

		Sports		Beauty		Toys		Yelp	
Family	Method	H@10	N@10	H@10	N@10	H@10	N@10	H@10	N@10
CF-based	GRU4Rec	0.0204	0.0110	0.0283	0.0137	0.0176	0.0084	0.0263	0.0134
	SASRec	0.0350	0.0192	0.0605	0.0318	0.0675	0.0374	0.0390	0.0194
	BERT4Rec	0.0191	0.0099	0.0347	0.0170	0.0203	0.0099	0.0090	0.0045
	FDSA	0.0288	0.0156	0.0407	0.0208	0.0381	0.0189	0.0544	<u>0.0293</u>
	S^3-Rec	0.0385	0.0204	0.0647	0.0327	0.0700	0.0376	0.0341	0.0168
Multi-Interest	ComiRec-SA	0.0095	0.0042	0.0510	0.0294	0.0464	0.0306	0.0131	0.0066
	ComiRec-DR	0.0123	0.0062	0.0314	0.0164	0.0263	0.0154	0.0249	0.0164
	MIND	0.0085	0.0041	0.0318	0.0171	0.0218	0.0122	0.0218	0.0132
	Re4	0.0142	0.0071	0.0236	0.0128	0.0121	0.0076	0.0094	0.0048
LLM-Embedding	AlphaRec (MLP)	0.0268	0.0135	0.0456	0.0238	0.0407	0.0221	0.0073	0.0034
	AlphaRec (LGCN)	0.0328	0.0177	0.0432	0.0242	0.0179	0.0091	0.0070	0.0031
	LLMEmb	0.0389	<u>0.0205</u>	0.0754	<u>0.0398</u>	0.0838	0.0458	0.0205	0.0103
	DLLM2Rec	0.0292	0.0143	0.0496	0.0242	0.0589	0.0315	0.0224	0.0112
	LEARN	0.0188	0.0098	0.0286	0.0136	0.0349	0.0181	0.0096	0.0042
	LLM-SRec	<u>0.0429</u>	0.0171	<u>0.0767</u>	0.0358	0.0682	0.0327	<u>0.0588</u>	0.0267
LLM-Generation	BIGRec (SFT)	0.0110	0.0059	0.0280	0.0151	0.0425	0.0241	0.0146	0.0100
	BIGRec (DFT)	0.0161	0.0084	0.0276	0.0146	0.0404	0.0222	0.0250	0.0164
	Rec-R1 (RL)	0.0171	0.0089	0.0343	0.0180	0.0380	0.0196	0.0185	0.0103
	RecZero (RL)	0.0283	0.0154	0.0585	0.0337	<u>0.0842</u>	<u>0.0478</u>	0.0156	0.0089
LLM-Agentic	ReAct	0.0117	0.0056	0.0220	0.0106	0.0315	0.0152	0.0150	0.0088
	Plan-Solve	0.0099	0.0045	0.0186	0.0085	0.0281	0.0136	0.0165	0.0092
	Self-Refine	0.0098	0.0046	0.0172	0.0080	0.0244	0.0112	0.0156	0.0086
	Direct-QG	0.0142	0.0065	0.0298	0.0134	0.0424	0.0193	0.0194	0.0115
	Query-AGG	0.0251	0.0168	0.0247	0.0114	0.0464	0.0215	0.0356	0.0260
Ours	PRISMIC	**0.0462**	**0.0242**	**0.0868**	**0.0472**	**0.0984**	**0.0554**	**0.0769**	**0.0469**
	Rel. Improv.	+7.69%	+18.05%	+13.17%	+18.59%	+16.87%	+15.90%	+30.78%	+60.07%

4 Experiments

We evaluate PRISMIC with four research questions aligned with the design choices in Section 3:

- **RQ1.** Does PRISMIC improve recommendation accuracy over representative baselines?
- **RQ2.** Does decomposing user histories into multiple intent signals alleviate premature intent collapse compared with direct history compression?
- **RQ3.** Does *learning and optimizing* the consolidation mechanism, including how candidates are selected and how they are integrated, outperform fixed selection or heuristic aggregation?
- **RQ4.** Does the consolidated query embedding provide more effective supervision for the final user encoder than alternative targets?

4.1 Experimental Setup

Setup. We evaluate on four standard recommendation benchmarks: Amazon-Sports, Amazon-Beauty, Amazon-Toys, and Yelp, under the leave-one-out protocol with full-item-pool retrieval, and report Hit@10 and NDCG@10. The retrieval encoder ϕ is a frozen `mxbai-embed-large-v1` and the consolidation policy π_θ is `Gemma-3-1B-it`. Detailed prompts, hyperparameters, and decoding settings are in the appendix; the full implementation is available at `https://github.com/venzino-han/prismic`.

Baselines. We compare against five families of baselines: sequential CF, multi-interest, LLM-derived embeddings, LLM generation, and agentic methods. The sequential CF family includes GRU4Rec [1], SASRec [2], BERT4Rec [14], FDSA [15], and S^3-Rec [16]. The multi-interest family includes ComiRec-SA [8], ComiRec-DR [8], MIND [7], and Re4 [9], which extract multiple interest vectors per user and rank items against max- or sum-similarity over those vectors in CF embedding space. The LLM-embedding family includes AlphaRec [24], LLMEmb [3], DLLM2Rec [25], LEARN [26], and LLM-SRec [21]. The LLM-generation family includes BIGRec [4], Rec-R1 [5], RecZero [6], and a direct query-generation variant. The agentic family includes ReAct [27], Plan-Solve [28], Self-Refine [29], along with two diagnostic baselines we construct, Query-AGG and Direct-QG, that probe whether naive aggregation of multi-query embeddings or one-shot LLM

Table 2: Ablation study. We remove or replace one design choice at a time and group rows by the research question they address.

		Sports		Beauty		Toys		Yelp	
Group	Variant	H@10	N@10	H@10	N@10	H@10	N@10	H@10	N@10
	PRISMIC	**0.0462**	**0.0242**	**0.0868**	**0.0472**	**0.0984**	**0.0554**	**0.0769**	**0.0469**
Decomposition (RQ2)	w/o decomposition (LLM)	0.0142	0.0065	0.0298	0.0134	0.0424	0.0193	0.0194	0.0115
	w/o decomposition (Enc)	0.0243	0.0122	0.0506	0.0268	0.0722	0.0388	0.0429	0.0267
Consolidation (RQ3)	w/o RL, top-1 query	0.0231	0.0113	0.0482	0.0241	0.0633	0.0321	0.0548	0.0382
	w/o RL, top-2 query	0.0281	0.0140	0.0555	0.0281	0.0667	0.0349	0.0543	0.0381
	w/o RL, top-3 query	0.0320	0.0159	0.0620	0.0322	0.0762	0.0403	0.0541	0.0381
	Mean-pooling consolidation	0.0319	0.0163	0.0597	0.0324	0.0773	0.0420	0.0569	0.0444
	Consolidator w/o RL	0.0184	0.0092	0.0395	0.0205	0.0445	0.0239	0.0426	0.0288
	Naive LLM consolidation	0.0251	0.0068	0.0247	0.0114	0.0464	0.0215	0.0356	0.0260
	Best-candidate SFT	0.0249	0.0137	0.0482	0.0277	0.0669	0.0376	0.0500	0.0405
Target / Loss (RQ4)	Target item only	0.0334	0.0174	0.0783	0.0436	0.0894	0.0498	0.0435	0.0269
	Consolidation target only (w/o InfoNCE)	0.0233	0.0115	0.0842	0.0443	0.0951	0.0523	0.0545	0.0379
	InfoNCE only (w/o consolidation target)	0.0352	0.0183	0.0534	0.0284	0.0627	0.0341	0.0512	0.0331

rewriting can explain PRISMIC's gain; we treat these two as ablations of design choices rather than representatives of fully optimized prompting frameworks. The full per-method comparison appears in Table 1, with extended details in the appendix. All reported methods are evaluated with the same leave-one-out split and full-item-pool ranking protocol; for methods requiring text retrieval, item embeddings are constructed from the same item metadata fields to avoid giving PRISMIC additional catalog information.

4.2 RQ1: Overall Recommendation Performance

Table 1 compares PRISMIC against five families of baselines: sequential CF, multi-interest, LLM-embedding, LLM-generation, and agentic. PRISMIC attains the highest Hit@10 and NDCG@10 on all four datasets. The LLM-generation family in Table 1 (BIGRec, Rec-R1, RecZero) feeds the user history directly into an LLM to produce a textual output that is then grounded back to items; none of these methods first decomposes the history into multiple signals. PRISMIC outperforms each of them on every dataset, including Rec-R1 and RecZero (the mechanically closest competitor, also using GRPO with a rank-based reward), and an approximate isolation in Appendix K suggests that "decomposition first" already accounts for a substantial part of the gap, even before learned consolidation is added.

The agentic prompting rows in Table 1 (ReAct, Plan-Solve, Self-Refine) indicate that additional inference-time deliberation alone is insufficient under our retrieval setup. The diagnostic baselines we construct, Direct-QG (one-shot LLM rewrite) and Query-AGG (mean-pool of multi-query embeddings), are designed to isolate two naive alternatives to retrieval-grounded decomposition and consolidation; they are not intended to represent fully optimized prompting or agentic recommenders, and we use them primarily as ablations of design choices. PRISMIC first improves over the strong retrieval-oriented and published LLM-based recommenders (Rec-R1, RecZero, BIGRec) on every dataset. The gap to the diagnostic baselines further suggests that the gain is not obtained by simply prompting an LLM once or by averaging its query embeddings. We do not interpret this as evidence against the entire class of LLM-based recommenders, but as evidence that retrieval-grounded decomposition and consolidation matter beyond what one-shot generation or naive aggregation can recover.

4.3 RQ2: Decomposition Mitigates Premature Intent Collapse

The central motivation of PRISMIC is that compressing a heterogeneous user history into a single LLM-generated query collapses competing intents before their predictive utility is distinguished. Several rows directly test this hypothesis. We test the decomposition hypothesis at two levels. The "*w/o decomposition (LLM)*" row in Table 2 replaces the multi-signal pipeline with a single direct-from-history LLM query that is used directly as the retrieval query (no encoder distillation), and produces one of the largest performance drops in the table. The "*w/o decomposition (Enc)*" row distills that same one-shot query into the user encoder. Even after adding the distillation, both

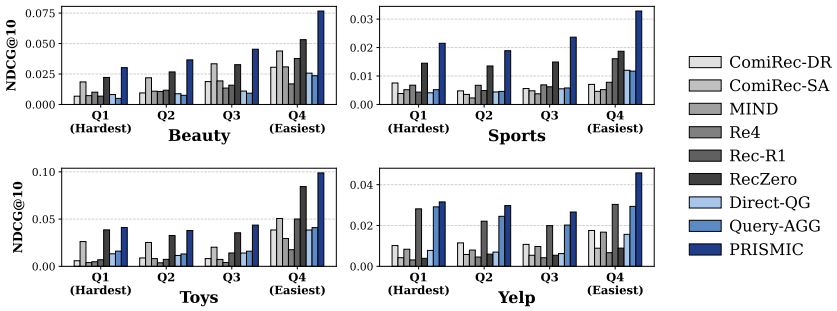


Figure 2: Hard-user analysis. Users are split into four groups by mean pairwise history similarity (Q1 = most heterogeneous, Q4 = most coherent). PRISMIC leads on Q1 across datasets, and its Q1→Q4 slope is gentler than the baselines.

variants remain consistently below PRISMIC, indicating that explicitly decomposing heterogeneous intent signals is critical when user histories contain mixed and competing evidence.

4.4 RQ3: Learned and Optimized Consolidation

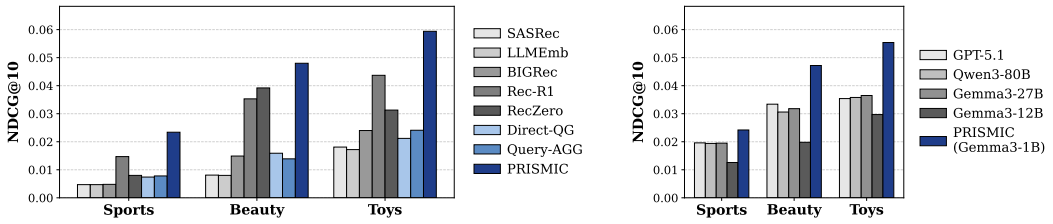
PRISMIC both *selects* which decomposed signals to use and *learns* how to integrate them under a retrieval-grounded reward. We isolate each axis using two row groups in Table 2. The Selection group ("*w/o RL, top-K query*", $K \in \{1, 2, 3\}$) keeps retrieval-grounded selection but uses the top-K queries directly as encoder targets without a consolidator: increasing K from 1 to 3 improves performance on Sports, Beauty, and Toys while Yelp is roughly flat across K; yet even top-3 still trails PRISMIC substantially on every dataset, so selection alone is necessary but insufficient. The Consolidation group fixes the same evidence set and replaces only the integration step: *Mean-pooling consolidation* loses 20% to 35% on Sports/Beauty/Toys (fixed aggregation is brittle); *Consolidator w/o RL* keeps the same prompt and distillation pipeline but skips the GRPO step (frozen base policy) and produces one of the largest drops, showing that prompting alone is insufficient and the consolidator must be *trained*; *Naive LLM consolidation* asks a separate frozen LLM to rewrite the queries into a paragraph used directly as the retrieval query, and underperforms top-1 selection on Beauty, Toys, and Yelp (it is roughly comparable on Sports Hit@10). Together these results pin the gain on *learning and optimizing* the integration mechanism: retrieval-grounded selection feeds the right candidates into an RL-trained consolidator, and removing either side collapses performance.

4.5 RQ4: Supervision Target for the Encoder

A key claim of PRISMIC is that the consolidated query embedding and InfoNCE provide complementary supervision for the downstream encoder. The Target / Loss group of Table 2 compares the full method against three alternatives: directly distilling the target-item embedding, using only the consolidated-query target, and training only with InfoNCE. All three variants underperform PRISMIC, showing that neither raw item identity, semantic alignment, nor contrastive discrimination alone is sufficient. The target-item variant provides a useful but limited item-level signal; the consolidation-only variant captures LLM-derived transition intent but lacks enough discrimination among competing items; and the InfoNCE-only variant improves item separation but misses the semantic direction provided by the consolidated query. Overall, the results suggest that the consolidated query supplies a transition-intent target that InfoNCE alone cannot recover, while InfoNCE sharpens retrieval against nearby candidates, making their combination the most effective supervision strategy across datasets.

4.6 Analysis

We close the experiments with three analyses beyond aggregate accuracy: hard users with heterogeneous histories, long-tail items, and compare with larger LLMs. A geometric case study with t-SNE projections is provided in Appendix H. A discussion of the scope of these analyses is given in Appendix L.



(a) Long-tail (cold-item) performance **(b) Comparison with larger LLMs**

Figure 3: (a) Cold-item NDCG@10 on Sports/Beauty/Toys (bottom 80% by frequency, averaged over three seeds for PRISMIC). (b) NDCG@10 of PRISMIC (Gemma-3-1B, offline-trained) against *zero-shot* inference of larger LLMs (GPT-5.1, Qwen3-80B, Gemma-3-27B/12B).

Hard users: robustness under heterogeneous histories (diagnostic). The decompose-then-consolidate motivation predicts that PRISMIC should help most on users whose history items span semantically distant categories, where one-shot LLM compression cannot summarize the basket from a single point. We quantify history heterogeneity by mean pairwise cosine similarity of items in H_u in the same frozen encoder space PRISMIC uses, partition users into four equal-frequency quartiles, and report NDCG@10 per bin (Q1 = most heterogeneous, Q4 = most coherent). Figure 2 shows that PRISMIC is most effective for users with heterogeneous histories. This result suggests that PRISMIC is more robust to mixed intent signals than baselines.

Long-tail items: consistent direction in the cold-item regime. Aggregate accuracy can be inflated by recommending popular items. To rule this out, Figure 3(a) reports cold-item NDCG@10 (bottom 80% of items by training-set frequency) on Sports/Beauty/Toys, comparing PRISMIC against existing methods spanning sequential CF (SASRec), LLM-embedding (LLMEmb), LLM-generation (BIGRec, Rec-R1, RecZero), and our diagnostic zero-shot query baselines (Direct-QG, Query-AGG). PRISMIC shows a consistent direction of improvement across the analyzed datasets, suggesting that the gains in Table 1 are not an artifact of head-item exposure. We do not claim a uniform long-tail advantage across all datasets and metrics; a fuller dataset-wide long-tail analysis (including Yelp and representation-independent slicing) is left as future work.

Task-specific alignment vs. zero-shot scale. A natural concern is whether PRISMIC's strength simply comes from the chosen LLM backbone. Figure 3(b) compares NDCG@10 of PRISMIC using Gemma-3-1B within our pipeline against zero-shot query generation by substantially larger LLMs, including GPT-5.1, Qwen3-80B, Gemma-3-27B, and Gemma-3-12B. PRISMIC achieves the highest NDCG@10 among these settings on every dataset shown. We emphasize that this comparison does not isolate backbone scale alone: PRISMIC differs from the larger-LLM rows in both training and inference procedure (offline RL alignment + distillation vs. zero-shot inference). The result should therefore be interpreted as evidence that task-specific offline alignment and retrieval-grounded distillation can outperform zero-shot query generation by larger general-purpose LLMs in this setting, rather than as a claim that the 1B backbone is intrinsically superior or that scaling the LLM is unhelpful in general.

4.7 Additional Analysis and Extension Directions

Offline cost and online efficiency. On Yelp, the measured pipeline across the three splits takes 210.8 minutes (3.51 GPU-hours) on a single GPU. Draft generation takes 96 minutes, query embedding 9, ranking 13, GRPO 62, encoder training 30, and projection training 0.8 (Appendix I). At deployment, the LLM is removed and no tokens are generated. The encoder forward pass takes 0.093 ms per user at batch size 256; this is an amortized batched measurement, not end-to-end single-request retrieval latency. The offline cost remains relevant when scaling to larger user populations or longer histories. Draft generation and GRPO together account for 158 of the 210.8 minutes, making text generation and policy optimization the main targets for reducing construction cost in this measurement. A more expensive offline teacher need not increase the deployed encoder's parameter count. However, the forward-pass timing excludes history preprocessing and catalog search,

and therefore does not establish end-to-end serving throughput. Deployment studies should also measure the cost of refreshing supervision as user behavior and the item catalog change.

Alternative LLM backbones. Table 3 reports recommendation NDCG@10 with two additional Qwen3.5 backbones alongside the original Gemma result. Qwen3.5-2B improves over Qwen3.5-0.8B on all four datasets, but neither Qwen model exceeds Gemma on Yelp. Qwen3.5-2B has the highest reported NDCG@10 on the three Amazon datasets, whereas Gemma retains the highest score on Yelp. This pattern does not identify why the models differ: draft coverage, consolidation quality, and the encoder's ability to absorb the resulting supervision can each affect the final score. A controlled comparison that fixes prompts, encoder inputs, and training seeds would be needed to isolate the contribution of the backbone itself.

Table 3: Backbone comparison (NDCG@10). Gemma values reproduce Table 1.

Backbone	Sports	Beauty	Toys	Yelp
Gemma-3-1B-it	0.0242	0.0472	0.0554	**0.0469**
Qwen3.5-0.8B	0.0228	0.0455	0.0525	0.0416
Qwen3.5-2B	**0.0256**	**0.0503**	**0.0575**	0.0424

Directions for extending the framework. For longer histories, an adaptive anchor budget could allocate more drafts to users with diverse interactions while limiting redundant drafts from repeated behavior. Such a selector would need to use history information alone at deployment; training-target retrieval scores could provide offline supervision rather than a test-time selection rule. For changing catalogs, selectively refreshing supervision could be explored, balancing reconstruction cost against preference drift and recommendation quality.

A further extension is to distill several consolidated intent targets into a user encoder that produces multiple vectors. This could preserve alternatives that remain incompatible with a single-vector representation, but would increase retrieval and score-fusion costs. The compression-rule diagnostic in Appendix J motivates examining what is lost during aggregation; it does not establish the benefit of this proposed extension. Evaluating such a model would require both overall ranking metrics and conditional target retention and recovery, alongside memory and serving-time measurements. These extensions remain untested in the present study.

5 Conclusion

We presented PRISMIC, a decomposition-and-consolidation framework for sequential recommendation that addresses Premature Intent Collapse by separating heterogeneous user histories into multiple intent signals before learning how to integrate them. Rather than relying on single-shot LLM compression or inference-time deliberation, PRISMIC uses retrieval-grounded selection and GRPO-based consolidation to construct a domain-aligned semantic supervision target offline. This target is then distilled into a lightweight encoder with an auxiliary InfoNCE objective, enabling LLM-free full-pool retrieval at inference time. Across four real-world benchmarks, PRISMIC consistently improves over CF-based, LLM-embedding, LLM-generation, and agentic baselines, with especially clear gains on heterogeneous users and long-tail items. These results suggest that, for recommendation, RL-style learning can be used not only for direct policy optimization, but also as a mechanism for refining high-quality semantic supervision signals that improve efficient recommenders.

Acknowledgments and Disclosure of Funding

This work was supported by the Korean government (MSIT) through grants from the Institute of Information & communications Technology Planning & Evaluation (IITP) (No. RS-2024-00445087, Enhancing AI Model Reliability Through Domain-Specific Automated Value Alignment Assessment) and the National Research Foundation of Korea (NRF) (No. RS-2022-NR068758).

Competing interests. The authors declare no competing interests.

References

[1] B Hidasi. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

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

[3] Qidong Liu, Xian Wu, Wanyu Wang, Yejing Wang, Yuanshao Zhu, Xiangyu Zhao, Feng Tian, and Yefeng Zheng. Llmemb: Large language model can be a good embedding generator for sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 12183–12191, 2025.

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

[5] Jiacheng Lin, Tian Wang, and Kun Qian. Rec-r1: Bridging generative large language models and user-centric recommendation systems via reinforcement learning. *arXiv preprint arXiv:2503.24289*, 2025.

[6] Xiaoyu Kong, Junguang Jiang, Bin Liu, Ziru Xu, Han Zhu, Jian Xu, Bo Zheng, Jiancan Wu, and Xiang Wang. Think before recommendation: Autonomous reasoning-enhanced recommender. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[7] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Pipei Huang, Huan Zhao, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. Multi-interest network with dynamic routing for recommendation at tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

[8] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

[9] Shengyu Zhang, Lingxiao Yang, Dong Yao, Yujie Lu, Fuli Feng, Zhou Zhao, Tat-Seng Chua, and Fei Wu. Re4: Learning to re-contrast, re-attend, re-construct for multi-interest recommendation. In *Proceedings of the ACM Web Conference*, 2022.

[10] Mathieu Ravaut, Aixin Sun, Nancy Chen, and Shafiq Joty. On context utilization in summarization with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2764–2781, 2024.

[11] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR, 2023.

[12] Kremena Valkanova and Pencho Yordanov. Irrelevant alternatives bias large language model hiring decisions. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6899–6912, 2024.

[13] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[14] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[15] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, Xiaofang Zhou, et al. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*, pages 4320–4326, 2019.

[16] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1893–1902, 2020.

[17] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*, pages 364–381. Springer, 2024.

[18] 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 *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1007–1014, 2023.

[19] Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pages 1395–1406, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3671931. URL https://doi.org/10.1145/3637528.3671931.

[20] Zhiang Dong, Liya Hu, Jingyuan Chen, Zhihua Wang, and Fei Wu. Comprehend then predict: Prompting large language models for recommendation with semantic and collaborative data. *ACM Trans. Inf. Syst.*, February 2025. ISSN 1046-8188. doi: 10.1145/3716499. URL https://doi.org/10.1145/3716499. Just Accepted.

[21] Sein Kim, Hongseok Kang, Kibum Kim, Jiwan Kim, Donghyun Kim, Minchul Yang, Kwangjin Oh, Julian McAuley, and Chanyoung Park. Lost in sequence: Do large language models understand sequential recommendation?, 2025. URL https://arxiv.org/abs/2502.13909.

[22] Bo-Chian Chen and Manel Slokom. Llmdirec: Llm-enhanced intent diffusion for sequential recommendation. *arXiv preprint arXiv:2601.03259*, 2025.

[23] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[24] Leheng Sheng, An Zhang, Yi Zhang, Yuxin Chen, Xiang Wang, and Tat-Seng Chua. Language representations can be what recommenders need: Findings and potentials. In *ICLR*, 2025.

[25] Yu Cui, Feng Liu, Pengbo Wang, Bohao Wang, Heng Tang, Yi Wan, Jun Wang, and Jiawei Chen. Distillation matters: Empowering sequential recommenders to match the performance of large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, RecSys '24, pages 507–517, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400705052. doi: 10.1145/3640457.3688118. URL https://doi.org/10.1145/3640457.3688118.

[26] Jian Jia, Yipei Wang, Yan Li, Honggang Chen, Xuehan Bai, Zhaocheng Liu, Jian Liang, Quan Chen, Han Li, Peng Jiang, and Kun Gai. Learn: knowledge adaptation from large language model to recommendation for practical industrial application. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i11.33291. URL https://doi.org/10.1609/aaai.v39i11.33291.

[27] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[28] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

[29] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[30] Lei Li, Yongfeng Zhang, and Li Chen. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1348–1357, 2023.

[31] Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. On the generalization of sft: A reinforcement learning perspective with reward rectification. *arXiv preprint arXiv:2508.05629*, 2025.

[32] Xianming Li and Jing Li. AoE: Angle-optimized embeddings for semantic textual similarity. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1825–1839, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.101. URL https://aclanthology.org/2024.acl-long.101/.

[33] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, and Alexandre Ramé et al. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

[34] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

A Datasets

Table 4: Statistics of the four benchmark datasets used in our experiments.

Dataset	#Users	#Items	#Reviews	Density (%)
Sports	35,598	18,357	296,337	0.0453
Beauty	22,363	12,101	198,502	0.0734
Toys	19,412	11,924	167,597	0.0724
Yelp	30,431	20,033	316,354	0.0519

We evaluate on four widely used benchmarks: *Sports*, *Beauty*, *Toys* (three Amazon product domains), and *Yelp* (a service-review domain). Each interaction record contains a user id, an item id, a rating, a review, and a timestamp. Statistics are listed in Table 4. We use the five-core variant in which both users and items have at least five interactions. For evaluation we follow the standard leave-one-out protocol used by prior sequential-recommendation work [2, 14, 16, 30], holding out the last interaction of each user as the test target, the second-to-last as validation, and the remaining interactions as the training history. All metrics are computed under full-item-pool retrieval, excluding items already present in the user's history.

B Baselines

We organize baselines into five families: sequential CF, multi-interest, LLM-embedding, LLM-generation, and agentic prompting. For each method we briefly describe how it produces its prediction on full-item-pool retrieval.

B.1 Sequential CF Baselines

- **GRU4Rec** [1] models user behavior with a gated recurrent unit and a sequence-to-one pairwise ranking objective; one of the earliest sequential recommenders.
- **SASRec** [2] replaces recurrence with self-attention so that long- and short-range dependencies in the interaction sequence are captured by stacked attention blocks.
- **BERT4Rec** [14] extends self-attention sequence modeling to bidirectional context using a Cloze-style masked-item objective, yielding richer contextual representations of the history.
- **FDSA** [15] models item-level and feature-level transitions with two parallel self-attention blocks, exploiting metadata in addition to item ids.
- **S^3-Rec** [16] pre-trains a self-attentive backbone with four self-supervised objectives over items, attributes, and subsequences before fine-tuning on the recommendation task; especially effective under data sparsity.

B.2 Multi-Interest Baselines

These baselines explicitly extract multiple interest vectors from the same user history and rank items against a max- or sum-similarity over those vectors. Unlike PRISMIC, they operate purely in the ID/CF embedding space and do not exploit textual item semantics.

- **MIND** [7] extracts multiple interest capsules from the user behavior sequence with dynamic routing, and selects the closest interest capsule to a candidate item at retrieval time.
- **ComiRec-SA** [8] produces multiple interest embeddings using a self-attention extractor; the controllable framework supports retrieval by max-similarity over the resulting interest vectors.
- **ComiRec-DR** [8] replaces the self-attention extractor with dynamic capsule routing inherited from MIND, while keeping the controllable retrieval head; we report it as a separate variant for completeness.
- **Re4** [9] augments multi-interest learning with three auxiliary objectives (re-contrast, re-attend, re-construct) that explicitly disentangle the per-interest representations and reduce redundancy across interests.

B.3 LLM-Embedding Baselines

- **AlphaRec** [24] replaces ID embeddings with item representations extracted from a pre-trained language model and trains a lightweight head on top. We evaluate two variants used in the original paper: an MLP head (AlphaRec MLP) and a Light Graph Convolutional Network head (AlphaRec LGCN).
- **LLMEmb** [3] produces semantically rich item embeddings from an LLM through supervised contrastive fine-tuning and a recommendation-adaptation step that aligns the LLM-derived embeddings with collaborative signals.
- **DLLM2Rec** [25] distills LLM knowledge into a lightweight sequential recommender, combining importance-aware ranking distillation with a collaborative embedding distillation that fuses semantic and CF signals.
- **LEARN** [26] couples a frozen LLM item encoder with a twin-tower architecture so that open-world textual semantics are aligned with user-item interactions while avoiding catastrophic forgetting.
- **LLM-SRec** [21] feeds CF-derived user and item representations into an LLM, producing a hybrid recommender that combines LLM semantics with collaborative signals from a pre-trained sequential model.

B.4 LLM-Generation Baselines

- **BIGRec** [4] fine-tunes an LLM to autoregressively generate the textual representation of the next item, then grounds the generated text back to the catalog by similarity search over item text embeddings. We evaluate both the supervised fine-tuning variant (BIGRec SFT) and a dynamic fine-tuning [31] variant (BIGRec DFT).
- **Rec-R1** [5] applies reinforcement learning to a generative LLM recommender, using a rank-based reward to optimize the generated query for retrieval performance.
- **RecZero** [6] prepends an explicit reasoning step to LLM-based recommendation; the policy is trained with reinforcement learning so that the LLM "thinks before recommending" instead of producing direct outputs.

B.5 Agentic Baselines

- **ReAct** [27] interleaves reasoning steps and discrete actions in a single LLM trajectory. For recommendation, the LLM alternates between reasoning over the user history and emitting a query, then uses the query for retrieval.
- **Plan-Solve** [28] prompts the LLM to first *plan* a sequence of subtasks and then *solve* each one, producing a more structured chain of thought before arriving at a final query.
- **Self-Refine** [29] generates an initial query, asks the same LLM to critique it, and revises iteratively; the final revised query is used for retrieval.
- **Query-AGG** (ours, baseline construction). The LLM produces multiple history-conditioned queries; their text-encoder embeddings are then averaged and the result is used as a single retrieval query. This isolates whether simple aggregation of multiple LLM queries already suffices, without learned consolidation.
- **Direct-QG** (ours, baseline construction). The LLM is prompted once with the full user history and emits a single query that is used directly for retrieval. This is the canonical one-shot LLM query-generation setup and serves as the lower bound on multi-signal performance.

Query-AGG and Direct-QG are not from prior work; we constructed them to expose the failure modes of simple aggregation and single-shot generation under the same Gemma-3-1B backbone used by PRISMIC.

C Implementation Details

The text-generation steps of the multi-query and consolidation stages are implemented with vLLM[1]. The frozen text encoder ϕ is `mxbai-embed-large-v1` [32] accessed through SentenceTransform-

[1] `https://docs.vllm.ai/`

ers[2]. LLM-based query generation uses a single Gemma-3-1B-it [33] model for both the anchor-conditioned query generator π_{base} and the consolidation policy π_θ, so the entire training-time LLM pipeline runs on the same 1B backbone. All experiments run on a single NVIDIA RTX A6000 PRO GPU (96 GB VRAM) under Ubuntu 22.04.

The pipeline is trained in three stages. (1) *Multi-query generation*: for each training user we generate one query per history item with π_{base}, decoded greedily (temperature 0.01) with maximum 512 tokens. (2) *GRPO consolidation*: we sample $G = 8$ candidate consolidated queries per user, score each with the NDCG@K retrieval reward (Eq. 3), and update π_θ for one epoch with batch size 16, learning rate $5{\times}10^{-5}$, and a KL coefficient of 0.04 against the initial policy. (3) *Encoder distillation*: the projection layer is trained for 30 epochs with batch size 128, learning rate $1{\times}10^{-4}$, 99 uniformly sampled negatives per positive, InfoNCE temperature $\tau = 0.07$, and InfoNCE weight $\lambda_{\text{NCE}} = 0.1$ (full loss in Section F).

We do not claim PRISMIC is target-free during training; as in standard supervised sequential recommendation, the next item provides the training signal. The distinction is that this signal is used only as a label to score candidate queries (Stage 1) and to compute the GRPO reward (Stage 2), not to condition the text seen by the LLM or the deployed encoder. The next-target item is never inserted into LLM prompts, never used when constructing validation or test representations, and never available at inference time, where the encoder produces $\mathbf{z}_u$ from the history alone with no LLM call. A complete list of hyperparameters and exact decoding settings is released with the code at the public repository [3].

Seeds and statistical significance. We report mean results across three independent training runs with seeds $\{22, 42, 62\}$ for each dataset and metric. The same seed controls Stage 2 (GRPO consolidation) and Stage 3 (encoder distillation); Stage 1 is deterministic and shared across runs. For every (dataset, metric) cell we run a paired two-sided t-test of PRISMIC against the strongest baseline of that column over the three seeds, and confirm $p < 0.05$ in every cell.

D Theoretical Motivation

We provide a simple theoretical view of why decomposing a heterogeneous user history into multiple intent hypotheses and learning their consolidation with a retrieval reward can improve sequential recommendation. The goal of this analysis is not to prove unconditional dominance, but to characterize the regime in which PRISMIC is expected to outperform direct one-shot compression: histories contain multiple competing signals, at least one decomposed signal preserves the decisive next-intent direction, and the consolidator is optimized with respect to the same retrieval objective used for evaluation.

D.1 User History as a Mixture of Latent Intents

Let $H_u = \{i_1, \ldots, i_n\}$ denote the interaction history of user u, and let $t_u \in \mathcal{I}$ be the next item to be retrieved from the candidate pool $\mathcal{I} \setminus H_u$. We assume that the history contains multiple latent intent signals

$$\mathcal{Z}_u = \{z_{u,1}, \ldots, z_{u,m}\}, \tag{4}$$

where each $z_{u,j}$ corresponds to a possible semantic direction induced by a subset of historical interactions. Among them, let $z_u^\star$ denote the decisive intent that best explains the next item t_u.

A direct LLM-based query generator compresses the entire history into a single query:

$$q_u^{\text{dir}} \sim \pi_{\text{base}}(\cdot \mid H_u). \tag{5}$$

Let $\phi(q) \in \mathbb{R}^d$ be the embedding of query q, and let $e_i \in \mathbb{R}^d$ be the embedding of item i. Retrieval is performed by a similarity score

$$s(q, i) = \langle \phi(q), e_i \rangle, \tag{6}$$

and the retrieval reward is defined as

$$R(q, u) = \text{NDCG@K}\big(\text{rank}_{\phi(q)}(t_u)\big). \tag{7}$$

[2] https://sbert.net/

[3] https://github.com/venzino-han/prismic

When the history contains heterogeneous or competing intents, a direct query can be viewed as a mixed representation:

$$\phi(q_u^{\mathrm{dir}}) \approx \sum_{j=1}^{m} \alpha_j v_{u,j}, \qquad \sum_{j=1}^{m} \alpha_j = 1, \tag{8}$$

where $v_{u,j}$ is the semantic direction associated with intent $z_{u,j}$. We assume that one intent direction $v_u^\star$ is the most retrieval-aligned with the next item, i.e., $\langle v_{u,j}, e_{t_u} \rangle < \langle v_u^\star, e_{t_u} \rangle$ for all $j \neq \star$ (otherwise the direct query is already at least as good as the best decomposed signal and decomposition cannot improve retrieval). Under this assumption, if the direct query assigns non-trivial mass to irrelevant, stale, or exploratory intents, the target score is diluted:

$$\langle \phi(q_u^{\mathrm{dir}}), e_{t_u} \rangle = \sum_{j=1}^{m} \alpha_j \langle v_{u,j}, e_{t_u} \rangle < \langle v_u^\star, e_{t_u} \rangle. \tag{9}$$

This gives a geometric interpretation of premature intent collapse: the model compresses multiple signals into a single point before it has identified which signal is retrieval-effective.

D.2 Benefit of Decomposition: Expanding the Candidate Intent Space

PRISMIC avoids direct compression by generating one anchor-conditioned query per historical item:

$$Q_u = \{ q_{u,i} \sim \pi_{\mathrm{base}}(\cdot \mid H_u, i) : i \in H_u \}. \tag{10}$$

Each $q_{u,i}$ acts as a separate intent hypothesis. Instead of immediately forcing all signals into a single query, decomposition exposes multiple semantic directions whose retrieval utility can be evaluated independently.

Assume that the decomposed candidate set has ϵ-coverage of the decisive intent. That is, for each user u, there exists a candidate query $q_u^\star \in Q_u$ such that

$$\|\phi(q_u^\star) - v_u^\star\| \leq \epsilon. \tag{11}$$

Also assume that the retrieval reward is locally L-Lipschitz with respect to the query embedding around the decisive intent:

$$\|\phi(q) - v_u^\star\| \leq \epsilon \quad \Rightarrow \quad R(q,u) \geq R_u^\star - L\epsilon, \tag{12}$$

where $R_u^\star$ is the reward achieved by an ideal query perfectly aligned with the decisive intent. Then the best decomposed query satisfies

$$\max_{q \in Q_u} R(q,u) \geq R(q_u^\star, u) \geq R_u^\star - L\epsilon. \tag{13}$$

By contrast, suppose direct compression incurs an intent-collapse penalty Δ_u:

$$R(q_u^{\mathrm{dir}}, u) \leq R_u^\star - \Delta_u. \tag{14}$$

Then decomposition provides a strictly better candidate whenever

$$\Delta_u > L\epsilon. \tag{15}$$

Thus, decomposition is expected to help most when the history is heterogeneous enough that direct compression incurs a large penalty, but the relevant intent is still recoverable from at least one anchor-conditioned candidate.

D.3 Benefit of Consolidation: Beyond Fixed Selection

After evaluating the decomposed queries, PRISMIC selects the top-K candidates according to their training-time retrieval utility:

$$Q_u^{\mathrm{top}} = \mathrm{TopK}_{q \in Q_u} R(q,u). \tag{16}$$

A simple selection method would choose the best candidate:

$$q_u^{\mathrm{sel}} = \arg\max_{q \in Q_u^{\mathrm{top}}} R(q,u). \tag{17}$$

However, selection can only reuse one of the existing candidates. It cannot merge complementary evidence across multiple candidates or suppress distracting phrases inside otherwise useful queries.

PRISMIC instead trains a consolidation policy

$$q_u^{\mathrm{con}} \sim \pi_\theta(\cdot \mid H_u, Q_u^{\mathrm{top}}), \tag{18}$$

which generates a new query conditioned on the selected intent hypotheses. The consolidator has a strictly more expressive feasible space than fixed selection: it can copy any selected candidate, but it can also synthesize a new query by combining useful parts from multiple candidates. Therefore, if copying is allowed as a special case, the optimal consolidator is no worse than the optimal selector:

$$\max_{\pi_\theta} \mathbb{E}_{q\sim\pi_\theta(\cdot|H_u,Q_u^{\mathrm{top}})}[R(q,u)] \geq \max_{q\in Q_u^{\mathrm{top}}} R(q,u). \tag{19}$$

This inequality explains why learned consolidation can improve over top-K selection or mean-pooling: the policy searches over a richer space of semantic targets while still retaining the ability to fall back to the best selected candidate.

D.4 Why Reinforcement Learning is Appropriate for Consolidation

To align the generation process with recommendation objectives, we adopt GRPO [13]. Specifically, we use *Dr.GRPO*, a variation of GRPO that provides more stable optimization [34]. Unlike standard RL approaches that rely on single-trajectory feedback, Dr.GRPO leverages the relative quality of outputs within a sampled group to estimate the baseline, while mitigating the optimization biases (*i.e.*, length bias and difficulty bias) inherent in the original GRPO formulation.

Formally, let x denote the input prompt constructed from the user history H_u and the selected evidence set $\mathcal{Q}_u^{\mathrm{top}}$, and let π_θ be the consolidation policy. Following Section 3.2, $\mathcal{Q}_u^{\mathrm{top}}$ contains only queries that satisfy two joint conditions: they are among the top-K candidates by retrieval rank ($K = 3$ in our experiments) *and* they retrieve the target within rank ≤ 100, so the consolidator is trained on prompts whose evidence set is at least somewhat retrieval-effective. For each input x, we sample a group of G outputs $\{o_1, o_2, \ldots, o_G\}$ from π_θ and compute the retrieval-based reward $R(x, o_g)$ for each output, where $R(\cdot)$ is the NDCG@K retrieval reward defined in the main text. To reduce variance and estimate the advantage, Dr.GRPO computes the advantage $\hat{A}_g$ by centering the rewards within the group:

$$\hat{A}_g = R(x, o_g) - \frac{1}{G}\sum_{j=1}^{G} R(x, o_j). \tag{20}$$

Unlike the original GRPO, we omit standard-deviation normalization to prevent a difficulty bias that overweights low-variance tasks [34].

The objective of Dr.GRPO maximizes the advantage-weighted likelihood while enforcing a trust-region constraint at the token level. Specifically, with the per-token probability ratio

$$\rho_{g,t} = \frac{\pi_\theta(o_{g,t} \mid x, o_{g,<t})}{\pi_{\theta_{\mathrm{old}}}(o_{g,t} \mid x, o_{g,<t})}, \tag{21}$$

the loss is

$$\mathcal{L}_{\text{Dr.GRPO}}(\theta) = -\frac{1}{G}\sum_{g=1}^{G}\sum_{t=1}^{|o_g|}\Big[\min\Big(\rho_{g,t}\,\hat{A}_g,\ \mathrm{clip}(\rho_{g,t}, 1-\epsilon, 1+\epsilon)\,\hat{A}_g\Big) - \beta\,\mathbb{D}_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}})\Big]. \tag{22}$$

where $|o_g|$ is the length of output sequence o_g, π_{ref} is the reference policy (the initial π_θ before training), ϵ is the clipping parameter, and $\mathbb{D}_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}})$ is the KL divergence between the current policy and the reference policy with regularization coefficient β.

Therefore, among candidate consolidations for the same user, queries that produce higher NDCG receive positive relative advantage and become more likely under π_θ, while queries that hurt retrieval receive lower or negative advantage and are suppressed. This gives a direct mathematical reason for using RL in PRISMIC: the model learns how to consolidate decomposed intent signals under the actual retrieval reward, rather than under a surrogate objective such as textual coherence, averaging, or next-token likelihood.

E Design Rationale and Novelty Clarification

This appendix collects the design choices behind PRISMIC and clarifies how each contributes to the novelty of the framework.

Why generate multiple queries? A user history often contains multiple plausible intents. Generating one query per anchor item prevents the model from collapsing the entire history into a single average preference too early. Each query acts as a separate intent signal and is later evaluated by retrieval, so the consolidator receives an explicitly enumerated and filtered evidence set rather than an entangled history.

Why use GRPO for consolidation? Naively concatenating or averaging selected queries does not teach the model which signals matter. GRPO directly optimizes retrieval quality, allowing the consolidator to emphasize useful anchors and suppress distracting ones. Unlike methods that apply RL to LLM-based recommendation as an inference-time tool, PRISMIC uses RL only to construct a better training-time supervision target, so the cost of GRPO does not propagate to deployment.

Why train an encoder after consolidation? LLM-generated queries are informative but expensive. By distilling consolidated query embeddings into a lightweight encoder, PRISMIC converts offline LLM reasoning into a deployable recommendation representation. This separation is the source of the inference-time efficiency claim: the LLM and multi-query pipeline appear only during target construction, while the deployed model is a standard encoder-cosine-similarity retriever.

Why add InfoNCE? MSE aligns the encoder with the semantic query target, but recommendation also requires distinguishing the target item from many negatives. The InfoNCE objective adds this item-level discrimination so that a semantically rich representation is also a discriminative one.

Summary of novelty. The novelty of PRISMIC does not lie in any single component but in the combination of three design choices: (i) decomposing a user history into multiple retrieval-evaluable intent signals rather than a single direct query; (ii) learning a consolidator with an NDCG-based reward that selects and merges these signals by retrieval utility; and (iii) using the consolidated query purely as offline supervision for a lightweight encoder, so that LLM reasoning is captured during training without paying for it at inference.

F Encoder Training Details

This appendix expands the encoder distillation step described in Section 3.3.

Targets and representations. For each training user u, the consolidated query q_u^{con} produced by the GRPO policy is encoded with the frozen text encoder ϕ to obtain a semantic target $\mathbf{y}_u$. The user encoder f_ψ and projection g_ω produce a user representation $\mathbf{z}_u$ in the same space:

$$\mathbf{y}_u = \phi(q_u^{\mathrm{con}}), \qquad \mathbf{z}_u = g_\omega(f_\psi(H_u)). \tag{23}$$

Item embeddings $\mathbf{e}_i = \phi(x_i)$ are computed once with the same frozen encoder over the item's title, brand, and category text and cached; they are not updated during encoder training.

Combined loss. The encoder is trained with an MSE distillation loss against the consolidated query target and an InfoNCE auxiliary loss against negative items:

$$\begin{aligned} \mathcal{L}_{\mathrm{enc}} &= \mathcal{L}_{\mathrm{MSE}} + \lambda_{\mathrm{NCE}}\mathcal{L}_{\mathrm{NCE}}, \\ \mathcal{L}_{\mathrm{MSE}} &= \|\mathbf{z}_u - \mathbf{y}_u\|_2^2, \\ \mathcal{L}_{\mathrm{NCE}} &= -\log \frac{\exp(\mathrm{sim}(\mathbf{z}_u, \mathbf{e}_{t_u})/\tau)}{\sum_{j\in\{t_u\}\cup\mathcal{N}_u} \exp(\mathrm{sim}(\mathbf{z}_u, \mathbf{e}_j)/\tau)}. \end{aligned} \tag{24}$$

Here t_u is the next-target item, $\mathcal{N}_u$ is a sampled negative set (size 99, drawn uniformly from $\mathcal{I} \setminus (H_u \cup \{t_u\})$), $\tau = 0.07$ is the InfoNCE temperature, and sim is cosine similarity. The InfoNCE weight λ_{NCE} is set to 0.1 by default.

Inference. At test time, items are ranked by $s(u, i) = \mathrm{sim}(\mathbf{z}_u, \mathbf{e}_i)$ for $i \in \mathcal{I} \setminus H_u$, and the top-ranked items are returned as recommendations.

Implementation. Hidden dimensions and other architectural choices follow standard sequential-encoder defaults; full hyperparameter values, optimizer settings, and dataset-specific batch sizes are released with the code at the public repository linked in Section 4.1.

G Prompt Formats

This appendix lists the LLM prompt templates used by PRISMIC. All prompts are designed for an instruction-tuned chat model and are formatted with the model's chat template at decoding time.

G.1 Multi-Query Generation Prompt

For each user u and each history item $i \in H_u$, we generate one anchor-conditioned query with π_{base} (Gemma-3-1B-it). The system prompt is fixed; the user message is filled with the user's chronological history, with the anchor item placed last so that it serves as the focal item for the next-purchase query.

Multi-query generation: prompt template

You are an intelligent assistant that analyzes a user's recent preferences based on their chronological purchase history. Given a user's history together with one designated anchor item from that history, infer a plausible next-purchase intent that foregrounds the anchor while still considering the rest of the history, and write the result as a single coherent paragraph.

Rules:

- Use the user's history as the primary evidence and treat the anchor item as the focal next-intent direction.
- Write the response as one detailed paragraph (max 512 words), not as a short phrase or keyword list.
- Emphasize recent purchases and recurring interests.
- Do not include personal opinions or assumptions beyond what is stated in the data.
- Output only one final query paragraph.

`[User History]`
{user_history}

`[Anchor Item]`
{anchor_item}

The `[User History]` block is the user's chronological purchase list with date, title, brand, and categories per item. The `[Anchor Item]` block specifies which history item should be foregrounded as the focal next-intent direction; the prompt is run once per anchor, producing a set $\mathcal{Q}_u$ of up to $|H_u|$ queries. This prompt is run once per anchor item, producing a set $\mathcal{Q}_u$ of up to $|H_u|$ queries. For each $q_{u,i}$ we record the rank of the target item under $\phi(q_{u,i})$ and select the top-K queries by retrieval rank as the consolidator's evidence set $\mathcal{Q}_u^{\text{top}}$ (Section 3.2).

G.2 GRPO Consolidation Prompt

The consolidator π_θ (also Gemma-3-1B-it) receives the top-K anchor-conditioned queries from $\mathcal{Q}_u^{\text{top}}$ along with their anchor item metadata, and is trained with GRPO to produce a single retrieval-effective consolidated query enclosed by `<query>...</query>` tags.

GRPO consolidation: prompt template

You are given a user's purchase history and several strong candidate queries. Write one improved final retrieval query as a detailed descriptive paragraph that better represents the user's next-item intent.

Rules:

- Use the user's history as the primary evidence.

- Write the final query as one detailed paragraph, not as a short phrase or keyword list.
- Include relevant items, user preferences, recurring interests, and likely purchase tendencies when supported by the history.
- Keep the paragraph specific, informative, and retrieval-oriented.
- Avoid overly generic summaries.
- Output only one final query paragraph.

`[User History]`
{user_history}

`[Sample Queries]`
{sample_queries}

The `[User History]` block is the user's chronological purchase list with date, title, brand, and categories per item. The `[Sample Queries]` block contains the top-K anchor-conditioned queries from $\mathcal{Q}_u^{\text{top}}$ together with the rank at which each retrieves the target on the training split. The consolidator is trained to produce a single descriptive paragraph that integrates the history and the candidate-query evidence into a retrieval-effective query. At training time, GRPO samples a group of $G = 8$ candidate consolidated queries per user; each candidate is encoded with ϕ and scored by NDCG@K against the target item under full-pool retrieval (Eq. 3), and the policy is updated using group-relative advantages with a KL penalty against the initial policy. At target-construction time, the optimized policy emits one consolidated query per training user, which becomes the encoder's distillation target.

H Case Studies

To probe *why* PRISMIC retrieves the target item when zero-shot Gemma-3-1B baselines fail, we project user representations together with a random sample of 50 catalog candidates and the user's most-recent purchased items into a shared 2D space using cosine-distance t-SNE. For each dataset we report three hand-picked test users; every user is plotted as a separate panel inside a single 1×3 figure. A unified color and marker scheme is used throughout: $\times$ target item, ★ PRISMIC, ▲ Direct-QG, ■ Query-AGG, ♢ recent purchased history (up to eight items), and small light-gray dots for the 50 random catalog candidates. PRISMIC is the only blue marker, the ground-truth target is the only orange marker, and every other element belongs to the gray family so that proximity between the blue star and the orange cross is immediately visible.

In every case the PRISMIC representation lands in the same neighborhood as the target item, while Direct-QG and Query-AGG drift toward the bulk of unrelated candidates, providing a qualitative view of when multi-query consolidation places the user point near the target.

Scope of these visualizations. The figures below are intended as *qualitative illustrations*, not as quantitative evidence. They are computed on hand-picked test users where PRISMIC out-ranks both diagnostic baselines, and t-SNE is sensitive to sampling, perplexity, and initialization. We therefore do not draw claims about global geometric superiority from these plots; they merely visualize representative cases where decomposition prevents the consolidated query from collapsing toward a misleading part of the history. A representation- independent quantitative version of this analysis is left as future work.

I Compute Reporting

The Yelp timing study uses a single GPU, reported as an NVIDIA RTX A6000 PRO in the experimental setup. Table 5 reports measured stage-wise runtimes for the pipeline across the three Yelp splits, replacing the approximate stage-level estimates. The dataset contains 30,431 users. These measurements describe this benchmark run rather than a scaling guarantee for larger catalogs or user populations.

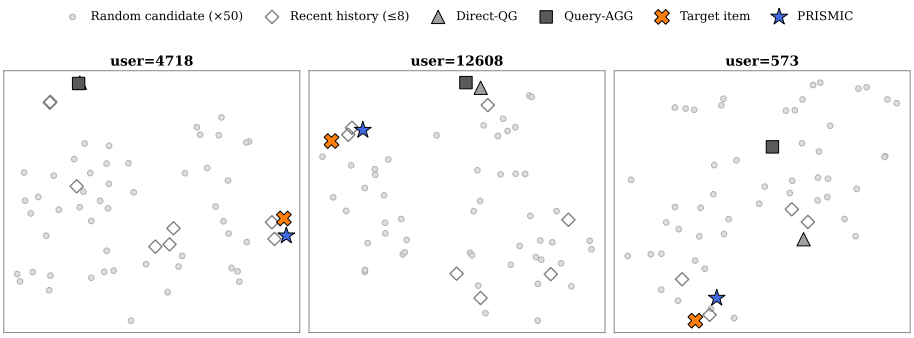


Figure 4: Case studies on Beauty.

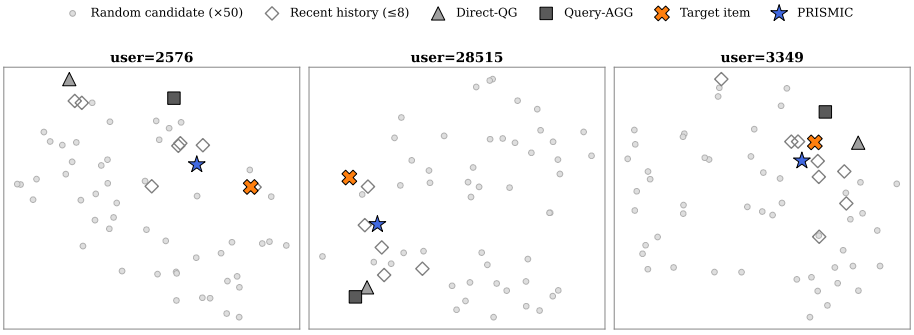


Figure 5: Case studies on Sports.

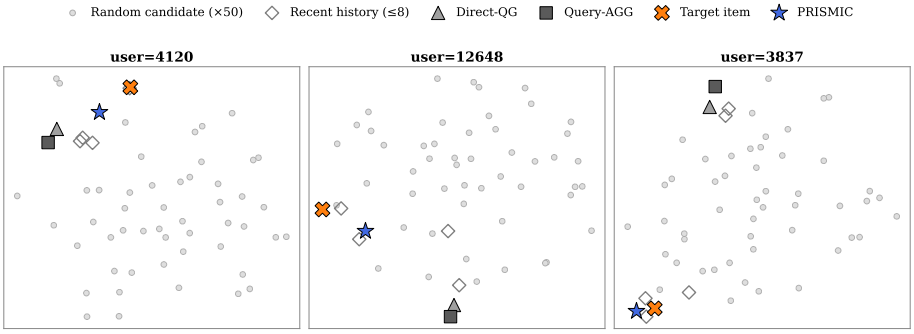


Figure 6: Case studies on Toys.

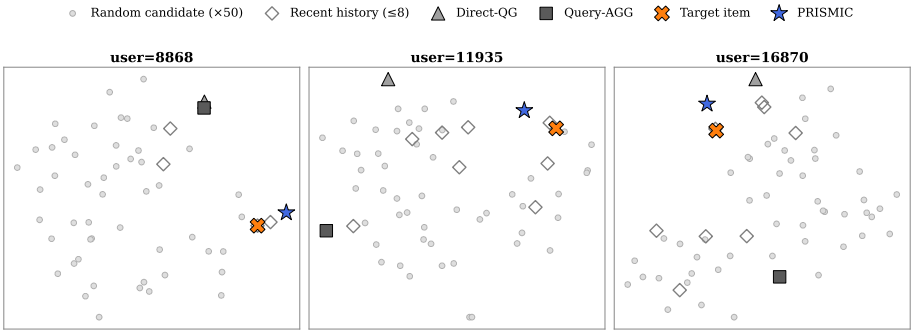


Figure 7: Case studies on Yelp.

Offline construction. Draft generation and GRPO account for most of the measured cost. The total is approximately 3.51 GPU-hours. LLM computation is confined to offline construction; the deployed user encoder does not generate draft or consolidated queries.

Table 5: Measured offline pipeline cost on Yelp.

Stage	Time (minutes)	Share (%)
LLM draft generation	96.0	45.5
Query embedding	9.0	4.3
Ranking (encoder + MIPS)	13.0	6.2
GRPO training	62.0	29.4
Encoder training	30.0	14.2
Projection training	0.8	0.4
Total	210.8	100.0

Online inference. The measured encoder forward time is 0.093 ms per user at batch size 256. This is an amortized per-user time under batched execution, not a batch-size-one or end-to-end service latency measurement. Item retrieval uses the user and item embeddings; the reported forward time should not be interpreted as including the complete retrieval service, such as index lookup, data loading, or network overhead.

J Retrieval-based Diagnosis of Intent Loss

We measure *LostTarget@10* over users whose target is retrieved in the top 10 by at least one anchor-conditioned draft: the fraction for which the target falls outside the top 10 after compression. The diagnostic draft bank contains queries from up to three recent history anchors per user. Table 6 compares compression rules on this recoverable-target subset; lower is better. One-shot full-history queries lose 68.8–86.0% of these targets. CF-weighted consolidation reduces this loss most on the three Amazon datasets, whereas uniform pooling performs best on Yelp. These rows evaluate alternative compression rules, not the final distilled PRISMIC encoder. This diagnostic supports evaluating separate hypotheses before compression; it does not directly measure semantic intent diversity or imply that single-vector compression is eliminated.

Table 6: LostTarget@10 for alternative compression rules (↓).

Compression rule	Sports	Beauty	Toys	Yelp
One-shot full-history query	0.860	0.791	0.688	0.857
Direct LLM aggregation	0.810	0.797	0.663	0.391
Uniform draft pooling	0.570	0.497	0.473	**0.228**
CF-weighted consolidation	**0.411**	**0.377**	**0.303**	0.241

K Comparison with LLM-Generation Baselines: the "Decomposition First" Effect

The LLM-generation family in our baseline set (BIGRec [4], Rec-R1 [5], RecZero [6]) all share a common recipe: the user history is fed to an LLM as a single input, and the LLM produces text that is then grounded back to items. BIGRec uses supervised next-item generation (SFT/DFT variants), while Rec-R1 and RecZero add reinforcement learning under rank- or reasoning-based rewards. None of them decomposes the history into multiple anchor-conditioned queries before the LLM is invoked. The comparison with this family therefore isolates the contribution of *decomposition first* on top of a shared "one-shot LLM over the whole history" recipe.

Full-system gap over the LLM-generation family. From Table 1, PRISMIC improves H@10 over the strongest LLM-generation baseline on each dataset by sizable margins: $+63\%$ on Sports (vs. RecZero), $+48\%$ on Beauty (vs. RecZero), $+17\%$ on Toys (vs. RecZero, the closest competitor), and $+208\%$ on Yelp (vs. BIGRec-DFT). The gap to Rec-R1, the mechanically closest competitor (it also uses GRPO with a rank-based reward), is larger still ($+170\%$ on Sports, $+153\%$ on Beauty, $+159\%$ on Toys, $+316\%$ on Yelp). These full-system gaps reflect three differences (decomposition, retrieval-grounded selection, and encoder distillation), so they do not attribute the gain to decomposition alone.

Approximate isolation of the decomposition step. To approximately isolate decomposition, we compare the LLM-generation family against the *w/o RL, top-1 query* row of Table 2, which shares with these baselines a single (best) anchor-conditioned query but applies the same encoder-distillation pipeline as PRISMIC on top of decomposition. Even without any RL on the consolidator, this decomposition-only configuration already exceeds the LLM-generation baselines on most cells: it surpasses Rec-R1 on every dataset (e.g., Beauty H@10 0.0482 vs. 0.0343; Yelp H@10 0.0548 vs. 0.0185) and surpasses BIGRec on every dataset, while RecZero is competitive on Toys but trails on Sports/Beauty/Yelp. This pattern suggests that retrieval-grounded decomposition itself, even before learned consolidation is added, accounts for a substantial portion of the gap. The remaining gap up to the full PRISMIC (e.g., Beauty H@10 0.0868, Yelp H@10 0.0769) is then attributable to learned consolidation and the joint Stage-3 supervision, consistent with the RQ3 ablation in Section 4.4.

Caveats. The methods differ in more than the decomposition step (training-time hyperparameters, prompt formats, supervision targets), and the top-1 query is selected with retrieval ranks computed from training labels. We therefore use the comparison as suggestive evidence that "decomposition first" contributes meaningfully to the gain over the LLM-generation family, not as an exact decomposition versus RL trade-off measurement.

L Limitations

We discuss limitations of the present study.

Scope of the analyses. We interpret the analyses in Section 4.6 cautiously. Direct-QG and QueryAGG are diagnostic baselines we constructed to isolate naive alternatives to retrieval-grounded decomposition and consolidation, rather than fully optimized representatives of all prompting-based recommenders. The hard-user, long-tail, larger-LLM, and qualitative case-study analyses are intended to provide complementary evidence about *when* PRISMIC helps, not standalone mechanistic proofs. Taken together with the main comparisons against retrieval-oriented and published LLM-based recommenders in Table 1 and the ablations in Table 2, these results support that retrieval-grounded decomposition and consolidation are useful for handling heterogeneous user histories under our setting, while broader intent-aware baselines and representation-independent hardness measures remain valuable directions for future evaluation.

Domain coverage. Three of our four benchmarks (Sports, Beauty, Toys) are Amazon product domains drawn from a single review platform, and Yelp is a service-review domain with a known popularity-bias structure. Generalization to more heterogeneous corpora (e.g., movies, music, news, conversational logs) is not directly tested.

English-only metadata. The retrieval encoder is an English-trained sentence transformer and the consolidation policy is an English-instruction-tuned LLM. Performance on non-English catalogs is not characterized.

Training-time LLM cost. Although inference is LLM-free, Stage-1 multi-query generation is the most expensive stage and scales linearly in the number of training users times the average history length. Domains with very long histories or large user counts will require further engineering (e.g., quantized π_{base}, sub-sampling history items) to keep Stage-1 tractable.

Backbone coverage. Section 4.7 extends the original Gemma-3-1B-it experiments to Qwen3.5-0.8B and Qwen3.5-2B. This comparison covers two model families and a limited capacity range; it does not establish generality across all LLMs or isolate the effects of draft generation and consolidation backbones. The retrieval encoder ϕ remains fixed, and broader backbone and encoder sweeps remain future work.

Hardness measure is not representation-independent. The hard-user analysis (Figure 2) bins users by mean pairwise cosine similarity of their history items in the same frozen `mxbai-embed-large-v1` space the encoder distillation operates on. The hardness score is therefore not orthogonal to the representation family PRISMIC exploits, and the analysis should be read as

a diagnostic of robustness under semantically heterogeneous histories rather than a representation-independent mechanistic proof. Representation-independent hardness measures (raw category Jaccard, SASRec-space diversity) are left as future work.

Larger-LLM comparison does not isolate scale. Figure 3(b) compares an offline-trained 1B PRISMIC pipeline against *zero-shot inference* of much larger LLMs. The two settings differ in both training and inference procedure, so this panel is evidence for task-specific offline alignment beating zero-shot generation in our setting, not for the 1B backbone being intrinsically superior or for scaling LLMs being unhelpful in general.

M Broader Impacts

Positive. PRISMIC replaces an online LLM call with a single encoder forward pass at inference time, substantially reducing the latency, dollar, and energy footprint of LLM-augmented recommendation, and making such recommendation feasible at the request volumes typical of production retrieval systems. The decompose-then-consolidate framework also exposes intermediate intent signals (the per-anchor queries) that can be inspected for debugging or explanation.

Risks and mitigations. LLM-derived semantic similarity is a known channel for popularity-bias inheritance, and the Q1 leftmost bars in Figure 2 show that the same backbone family recommends popular items more aggressively for hard users. Although PRISMIC mitigates this on the cold-item axis (Figure 3(a)), deployments should monitor exposure distributions over time to avoid filter-bubble effects. The training stage incurs a non-trivial energy cost (approximately 4 to 5 GPU-hours per dataset on a single A6000 PRO, see Section I); batching across catalogs and reusing precomputed multi-query generations across re-training cycles can reduce this in practice.

Misuse considerations. The method targets standard sequential-recommendation benchmarks that contain no personally identifying information and no sensitive demographic attributes. The same pipeline could in principle be applied to histories that contain such information; in that case operators should follow domain-specific data-handling guidelines and consult the relevant regulators before deployment.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state three contributions of PRISMIC (decomposition, GRPO-based consolidation, and encoder distillation), which are the components evaluated empirically in Sections 4.2 to 4.5; we do not claim deployment, theoretical guarantees, or generalization beyond the four benchmark datasets reported.

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss scope (four standard benchmarks under full-item-pool retrieval, single Gemma-3 backbone family) and offline training cost (multi-query generation scales linearly with $|H_u|$) in Sections 4.1, 3, and the appendix; we do not claim the framework removes the need for retrieval-encoder text alignment or that gains transfer to non-retrieval recommendation settings.

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: The paper is empirical; it introduces no new theorems or formal proofs.

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: Method details are in Section 3; benchmarks, splits, and full-item-pool protocol are in Section 4.1; encoder loss, negative sampling, hyperparameters, and training stages are in Appendices A, C, and F; a public code repository is linked in Section 4.1.

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

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

   Answer: [Yes]

   Justification: A public code repository (URL in Section 4.1) contains training, evaluation, and data-preparation scripts. The four benchmarks (Amazon-Sports/Beauty/Toys, Yelp) are publicly available, and we use the standard five-core leave-one-out preprocessing as described in Appendix A.

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: Splits, optimizer, learning rate, batch size, GRPO group size, KL coefficient, encoder InfoNCE temperature and weight, and evaluation cutoffs are listed in Appendices A, C, and F.

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: Statistical significance is verified with three seeds ($\{22, 42, 62\}$); details and the per-cell paired t-test ($p < 0.05$) are in Appendix C.

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: Appendix C reports the GPU type (single NVIDIA RTX A6000 PRO, 96 GB), the operating system, and the per-stage training configuration. Per-stage runtime is dominated by Stage 1 multi-query generation and Stage 2 encoder distillation; representative wall-clock figures are listed in the appendix.

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work uses publicly available recommendation benchmarks, does not involve human-subject experimentation, does not release pretrained model weights, and does not collect or redistribute user-identifying data.

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Appendix M discusses both potential positive impacts (LLM-free inference reduces deployment cost and latency for recommender services) and negative ones (reinforcement of popularity bias on the underlying retrieval encoder, reliance on English-trained text encoders limiting non-English deployments).

    Justification: The paper is foundational methodological research on retrieval-quality of sequential recommenders. It does not introduce a deployed system, a new dataset, or an application targeted at end users; potential downstream impacts (e.g., recommendation echo effects) are common to the broader recommender-systems literature and not specific to this work.

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: We release no high-risk artefacts: no pretrained model weights, no scraped datasets, and no generative system accessible to end users. The released code reproduces evaluation on existing public benchmarks.

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: Datasets (Amazon Reviews, Yelp Open Dataset), the text encoder (`mxbai-embed-large-v1`), and the language models (Gemma-3 family) are cited in Sections 4.1 and C via their original publications and model cards. We use each asset within the terms specified by the original release.

13. **New assets**

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

    Answer: [Yes]

    Justification: The single new asset is the implementation accompanying this paper; it is provided in the public repository linked in Section 4.1 together with run scripts, configuration files, and a README that documents the four-stage pipeline.

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

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

    Answer: [N/A]

    Justification: The paper involves no crowdsourcing and no research with human subjects.

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

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

    Answer: [N/A]

    Justification: The paper involves no research with human subjects.

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: LLMs were used as an auxiliary tool to support parts of the experimental implementation and manuscript writing process, such as drafting code snippets, debugging implementation details, and improving the clarity of paper text. All methodological decisions, experimental design, analyses, and final manuscript content were reviewed and verified by the authors.