COIN: COLLABORATIVE INTENT FOR EFFICIENT REASONING IN NATIVE GENERATIVE RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning can improve generative recommenders (GRs) through intermediate hypotheses that guide item prediction. Reasoning is natural for LLM-powered GRs but harder to introduce into native recommenders, which lack a natural language interface. We introduce COIN (Collaborative Intent-aware Generative Recommendation), a lightweight plan-then-execute framework that grounds reasoning in a collaborative intent space. Each intent is a coarse collaborative region learned from interactions, expressed as a single plan token that provides guidance complementary to item semantics. During training, a lightweight forecaster supplies plausible intents, and ambiguity-aware transfer trains the generator to predict the target item under planning uncertainty. COIN uses only next-token prediction, without reinforcement learning, LLM supervision, or recurrent refinement. At inference, the generator predicts and executes its own plan with only one additional token. COIN serves as a plug-in for existing native GRs across generator architectures and semantic-ID tokenizers. Experiments on four recommendation datasets show consistent ranking gains with minimal training and inference overhead, establishing collaborative intent as an efficient reasoning interface for native GRs.

1 INTRODUCTION

Reasoning extends generative prediction through intermediate steps that organize evidence and guide the final output, and has been widely adopted by modern LLM systems (Wei et al., 2022; Wang et al., 2023; Guo et al., 2025; Yang et al., 2025). Recent studies introduce reasoning into generative recommenders (GRs) to better interpret user behavior and guide item selection, particularly when observed interactions reflect multiple interests, incomplete preference evidence, and uncertain future needs (Liu et al., 2025d; Hong et al., 2025; He et al., 2026a; Zhou et al., 2025).

LLM-based GRs (Geng et al., 2022; Bao et al., 2023; Liao et al., 2024; Liu et al., 2025d) offer a natural starting point for reasoning. Their pretrained language knowledge provides a natural interface for interpreting behavioral evidence, connecting items to user needs, and developing preference hypotheses. Existing methods adapt these abilities to recommendations through item–language alignment and trajectory-based training, producing intermediate rationales that inform the final item choice (Liu et al., 2025d; Hong et al., 2025; He et al., 2026a). However, large pretrained backbones remain computationally costly, while intermediate reasoning further adds inference costs, constraining use under strict latency and resource budgets (Liu et al., 2025d; Cheng et al., 2026).

In contrast, *native GRs* use compact, task-specific backbones trained from scratch on interaction data and represent items by semantic IDs (SIDs). They combine generative modeling and efficient serving with a shared vocabulary and simple pipeline (Rajput et al., 2023; Pang et al., 2025). Unlike LLM-based GRs, introducing reasoning into native GRs is less straightforward. Existing work has explored semantic abstractions for finer item generation (Guo et al., 2026b; Zhao et al., 2026), LLM reasoning amortized through reason tokens or embeddings (Cheng et al., 2026; Jiang et al., 2026e), latent thoughts and recursive refinement (Zhang et al., 2025; Yang et al., 2026b; Jiang et al., 2026d) and iterative error correction (Xing et al., 2026a;b). Table 1 summarizes these approaches and their requirements. To activate such abilities with a native backbone, they introduce additional alignment or reinforcement learning (RL) objectives, repeated evaluation and refinement during decoding, and specialized tokenizers or architectures. These requirements complicate training and deployment, weaken the core simplicity and efficiency of native models, and inject vulnerability in the generation process (Jiang et al., 2026d; Zhang et al., 2026a). This raises a natural question:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

can native generative recommenders acquire effective and reliable reasoning while preserving their simplicity and efficiency?

Answering this leads to two key challenges. **(1)** Without the linguistic foundation of pretrained LLMs, native recommenders need an alternative interface for intermediate reasoning: a *reasoning language*. To guide beyond direct item prediction while preserving efficiency, this language must *draw on task-aligned knowledge that complements the recommender's intrinsic behavior, and support inexpensive supervision and generation*. **(2)** Reasoning can hurt when misleading intermediate conclusions steer subsequent steps toward an incorrect answer (Liu et al., 2024; Gema et al., 2025; Zhang et al., 2026a). In recommendation, reasoning should facilitate item generation rather than introduce a new predictive bottleneck and cause error propagation (Zhang et al., 2026a; Penha et al., 2026). This is particularly important for online serving, where strict latency and resource budgets limit opportunities to explore alternative paths or obtain external guidance and correction. Therefore, a desired model must *internalize reasoning that supports reliable generation across multiple plausible reasoning steps,* so that reaching the target item does not rely on a unique "correct" path.

We propose COIN, a plan-then-execute framework for efficient and robust reasoning in native GRs. To address the first challenge, COIN grounds intermediate reasoning in a collaborative intent space: each intent denotes a coarse collaborative region learned from interactions, capturing information complementary to item semantics (Zheng et al., 2024; Wang et al., 2024a; Pang et al., 2025; Guo et al., 2026b). During training, a lightweight forecaster learns to predict the next item's collaborative region from history, supplying inexpensive, task-aligned supervision directly from existing interactions. Each intent becomes a single *plan token*, a compact interface guiding next-item generation with little overhead. To address the second challenge, COIN introduces *ambiguity-aware transfer*, which repeatedly samples plausible intents from the forecaster and trains the generator to reach the same target item through these alternatives. This allows the generator to internalize planning and execution through next-token prediction while reducing dependence on any single plan. The objective connects representation learning and guided decoding while implicitly controlling disagreement in item predictions across supported plans. At inference, COIN is fully self-contained, autoregressively generating and executing its own plan without external models or an offline buffer.

We evaluate COIN on four recommendation datasets, demonstrating consistent ranking improvements with limited training and inference overhead. COIN accommodates different generator architectures and SID tokenizers, making it readily applicable as a plug-in for native GRs. Beyond ranking performance, detailed analyses provide insight into how collaborative intent is learned, transferred, and used to guide item generation. These findings suggest that native recommenders can acquire effective reasoning capabilities while retaining their practical advantages, motivating future work on lightweight reasoning for recommendations. Our main contributions are summarized as follows:

- We identify *collaborative intent* as a reasoning language for native GRs, providing task-relevant guidance complementary to item semantics through a single-token interface, with inexpensive supervision from raw interactions.
- We propose *ambiguity-aware transfer*, which learns robust planning and execution under intent uncertainty using only next-token prediction, without RL or LLM supervision.
- We demonstrate consistent ranking improvements on various recommendation datasets across different generator architectures and SID tokenizers, with limited training and inference overhead and fully self-contained inference.

Table 1: Reasoning methods in native GRs. LLM rationale: LLM-derived guidance; specialized arch./tokenizer: extra generator components or a specialized item tokenizer; auxiliary losses: losses beyond next-token prediction (NTP) on SIDs; CF: collaborative filtering; RL: reinforcement learning; passes: model forward passes. †COIN uses pure next-token prediction over plan and SID tokens; CF vocabulary construction, teacher training, and supervision preparation are lightweight preprocessing.

Reasoning basis	Training requirements	Inference overhead	Specialized arch./tokenizer	Relevant works
LLM rationale	LLM fine-tuning	Store lookup / soft prompts	Yes	GPR (2026e); TGR-Reason (2026)
Semantics	Auxiliary losses	Thinking tokens / extra passes	Yes	S^2GR (2026b); HoloRec (2026)
Latent thoughts	Auxiliary losses / RL	Thinking tokens / extra passes	Yes	STREAM-Rec (2025); DiffuReason (2026d); GPR (2026e)
Recursive Refinement	Auxiliary losses / RL	Repeated passes	Yes	IBA (2026b); DiffuReason (2026d); GPR (2026e)
Error Correction	Auxiliary losses + RL	Checks + regeneration	Yes	REG4Rec (2026a); GRC (2026b)
Collaborative intent	**Pure NTP**†	One plan token	No	**COIN (ours)**

2 Related Work

Reasoning in native GRs. Native GRs introduce reasoning through intermediate signals that compact recommenders can learn without generating natural language. LLM-assisted and semantic methods supply this guidance through reason tokens, intent embeddings, or coarse-to-fine abstractions from the SID hierarchy (Cheng et al., 2026; Jiang et al., 2026e; Guo et al., 2026b; Zhao et al., 2026). Other methods learn latent thoughts or recursive updates from recommendation objectives, or improve decoding trajectories through verification and correction (Zhang et al., 2025; Yang et al., 2026b; Jiang et al., 2026d; Guo et al., 2026a; Xing et al., 2026a;b). LLM guidance requires separate preparation, while latent refinement and trajectory search add computation during generation. COIN expresses this guidance through collaborative intent, using a vocabulary separate from SIDs and one planning token trained through next-token prediction.

Reasoning-enhanced recommendation. LLM-based recommenders use textual reasoning to interpret interaction histories and form preference hypotheses that guide item prediction (He et al., 2026a; Liu et al., 2025d; Hong et al., 2025). To reduce the cost of producing these traces, latent reasoning and distillation carry intermediate computation in compact hidden states (Zhang et al., 2026b; Chen et al., 2026; Jiang et al., 2026a). Latent refinement also underlies reasoning in traditional sequential recommendation, where successive or parallel updates to user representations precede item scoring (Tang et al., 2026b; Liu et al., 2025c; Tang et al., 2026a). Intent guidance and collaborative priors further ground these updates in behavioral evidence (Shao & Zhou, 2025; Yang et al., 2026a). Appendix A provides a broader review.

3 Method

We present COIN, a plan-then-execute framework for native SID-based GRs. We first construct a collaborative intent space and train a lightweight teacher to predict plausible intents from interaction histories (Section 3.1). We then introduce ambiguity-aware transfer to jointly learn planning and execution through next-token prediction, and analyze how this objective supports collaborative representation learning and reliable, consistent item generation (Section 3.2). At inference, the native recommender predicts and executes its own plan without the teacher. The overall framework is illustrated in Figure 1.

3.1 Collaborative Planning Space and Supervision

Given an interaction history h and its next item i^+, an item tokenizer Q produces the encoded history $X = Q(h)$ and target SID $Y = Q(i^+) = (y_1, \ldots, y_L)$ of length L. Let K be the number of collaborative intents and r_k the token for intent $k \in \{1, \ldots, K\}$. For generator parameters θ and SID prefix $y_{<\ell} = (y_1, \ldots, y_{\ell-1})$, COIN factorizes the joint prediction as

$$p_\theta(r_k, Y \mid X) = p_\theta(r_k \mid X) \prod_{\ell=1}^{L} p_\theta(y_\ell \mid X, r_k, y_{<\ell}). \tag{1}$$

For compactness, the intermediate plan is a single K-ary digit, encoded by one token inserted once between X and Y. The first factor predicts the plan from X, while the product executes it through SID decisions conditioned on both X and the selected plan. This structure lets the model first form a hypothesis about future interaction, then use it alongside the full history to generate the next item. We next establish the planning space and explain how the recommender learns to reason within it and execute its plans reliably.

Collaborative intent as a reasoning language. Unlike SID digits, the plan token is not a fixed component of an item's code. It expresses an inference from the history about where the next item may lie in SID space. We seek two properties in this reasoning space: task alignment, which makes the intermediate prediction relevant to next-item generation, and heterogeneity, which exposes distinctions beyond those readily expressed by the native representation. Collaborative signals offer a natural basis for this purpose: they relate items through shared patterns of user behavior, providing a task-relevant view that complements item semantics (Zheng et al., 2024; Wang et al., 2024a). Inspired by this, COIN grounds its reasoning in a collaborative intent space, where an intent represents a

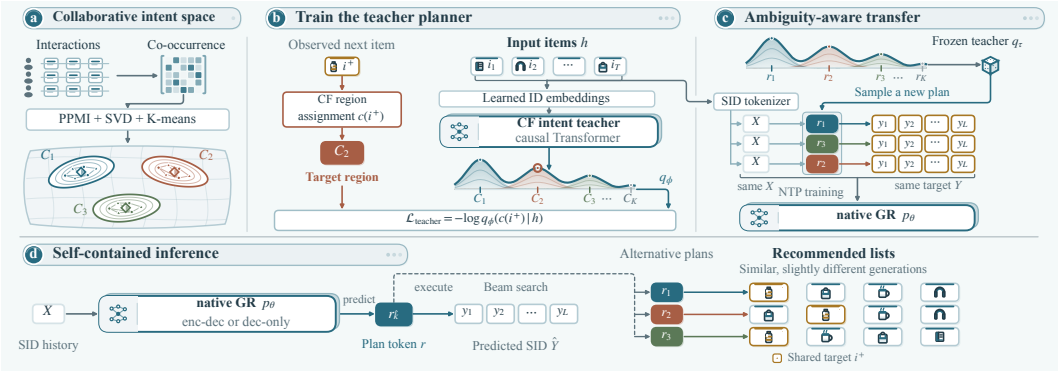


Figure 1: COIN overview. (a) Build collaborative intents from item co-occurrences. (b) Train a teacher to predict intent distributions. (c) Learn planning and execution via next-token prediction on sampled plans with a shared target. (d) Predict and execute a native plan at inference.

plausible direction of future interaction. Specifically, **(1)** we compute item co-occurrences from interaction histories (Liang et al., 2016); **(2)** we apply truncated SVD to their positive pointwise mutual information (PPMI) transform (Levy et al., 2015), obtaining collaborative embeddings e_i; and **(3)** we partition these embeddings into K regions using K-means. These regions define the intent vocabulary, with $c(i)$ denoting item i's region.

Figure 2(a) shows an example item space from Amazon Beauty with six collaborative/semantic regions. CF groups span semantic boundaries, providing a behavioral organization that cuts across the SID hierarchy. Arrows trace cross-region transitions from the last observed item to the next one within a CF group, suggesting that future interactions can follow a shared intent even when they move between semantic regions. Figure 2(b) quantifies this effect: both groupings provide information about the next item, and adding CF to semantic grouping contributes additional predictive information across datasets. These observations justify the collaborative intent space as task-aligned and heterogeneous, rather than adding another level to the same semantic hierarchy.

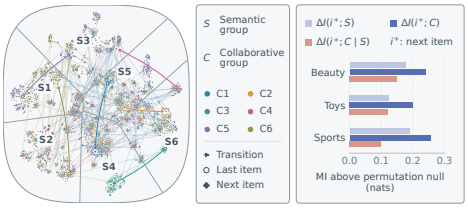


Figure 2: Collaborative intent space and mutual information. (a) Items colored by CF group (C1–C6), with semantic boundaries (S1–S6); arrows mark within-CF transitions. (b) MI gains above a permutation null.

Learning a collaborative intent teacher from interactions. The collaborative intent space provides a meaningful reasoning vocabulary. To actually activate reasoning over this space, supervision is needed to connect histories to plausible intents, following the use of intermediate demonstrations (Nye et al., 2021; Hsieh et al., 2023). Using the next item's region $c(i^+)$ directly as the intermediate token is insufficient for two reasons. First, it is a function of the target item rather than an inference from some history. A reasoning step should express what the preceding history supports; a target-tied label only moves the predictive difficulty one token earlier without realizing actual reasoning. Second, it supplies only one observed direction, while the same history can support multiple interests and plausible continuations (Li et al., 2019; Cen et al., 2020). We therefore seek a soft target: a distribution over intents that is history-dependent and preserves alternative directions beyond the top-1 choice.

To construct this supervision, we introduce an intent teacher: a lightweight causal Transformer with parameters ϕ that maps the item-ID history h to logits $z_\phi(h)$ and predicts $q_\phi(\cdot \mid h) = \mathrm{softmax}(z_\phi(h))$ over the K regions. For each observed transition (h, i^+), the target is the next item's region $c(i^+)$, and training minimizes $-\log q_\phi(c(i^+) \mid h)$, following standard next-item prediction at the coarser region level. Importantly, we use freely learned item-ID embeddings v_i as inputs, optimized jointly with the forecaster. This allows interaction evidence to shape the teacher's input features and expose behavioral distinctions that the native generator may underrepresent. To obtain useful distributional supervision, we apply strong regularization to discourage the teacher from memorizing training

histories and collapsing their forecasts onto the observed next-item region. We further control the concentration of the transferred distribution through temperature scaling:

$$q_\tau(\cdot \mid h) = \mathrm{softmax}\big(z_\phi(h)/\tau\big), \qquad \tau > 0. \tag{2}$$

Lower temperatures emphasize leading intents, while higher temperatures distribute supervision more broadly across alternatives. After training, we freeze the teacher and use $q_\tau(\cdot \mid h)$ to supervise the native recommender.

3.2 Learning to Plan and Execute

Section 3.1 defines the collaborative intent space and a teacher that supplies history-conditioned supervision. The remaining challenge is to amortize the planning ability into the native recommender while keeping item generation robust to uncertain intents. We address both goals through ambiguity-aware transfer, which learns planning and execution jointly through next-token prediction.

Whenever a transition is presented during training, we draw a fresh intent index $k \sim q_\tau(\cdot \mid h)$ while retaining the observed target SID Y. Let $Z = f_\theta(X)$ denote the native history representation, $a_\theta(k \mid Z) = p_\theta(r_k \mid X)$ the planning prediction normalized over the K plan tokens, and $p_{\theta,k}(Y \mid Z) = p_\theta(Y \mid X, r_k)$ the SID continuation. Next-token training on the sampled sequence (r_k, Y) minimizes

$$\mathcal{L}_\tau(\theta) = \mathbb{E}_{(h,Y),\, k\sim q_\tau(\cdot|h)}\big[-\log a_\theta(k \mid Z) - \log p_{\theta,k}(Y \mid Z)\big]. \tag{3}$$

For the analysis, let R denote the random intent index with conditional distribution $q_\tau(\cdot \mid h)$. For brevity, write $q_h = q_\tau(\cdot \mid h)$, $a_h = a_\theta(\cdot \mid Z)$, $p_{k,h} = p_{\theta,k}(\cdot \mid Z)$, and $\pi_h = P(Y \mid h)$. Using $R \perp Y \mid h$, the objective decomposes as

$$\mathcal{L}_\tau = \mathcal{H}_\tau + \underbrace{\mathbb{E}_h\,\mathrm{KL}(q_h \| a_h)}_{\mathcal{P}_\tau\text{: planning error}} + \underbrace{\mathbb{E}_h \sum_{k=1}^{K} q_h(k)\,\mathrm{KL}(\pi_h \| p_{k,h})}_{\mathcal{E}_\tau\text{: execution error}}. \tag{4}$$

Here, $\mathcal{H}_\tau = \mathbb{E}_h[H(\pi_h) + H(q_h)]$ is fixed w.r.t. the student parameters. The planning term matches the teacher's forecast through the student's own representation, while the execution term fits the next-item distribution under alternative plans. The following theorems connect these terms to collaborative representation learning and reliable, consistent plan execution. Proofs are provided in Appendix D.

Theorem 1 (Representation learning through planning distillation) *Let $q_{\tau,Z}(\cdot \mid Z) = \mathbb{E}[q_\tau(\cdot \mid h) \mid Z]$ for the deterministic representation $Z = f_\theta(X)$. The planning term is the teacher-to-student KL distillation error and satisfies*

$$\mathcal{P}_\tau = \underbrace{I_\tau(R; h \mid Z)}_{\mathcal{R}_\tau\text{: information not retained}} + \underbrace{\mathbb{E}_Z\,\mathrm{KL}(q_{\tau,Z}(\cdot \mid Z) \,\|\, a_\theta(\cdot \mid Z))}_{\mathcal{A}_\tau\text{: readout error}}. \tag{5}$$

For fixed Z, the minimum planning error over unrestricted heads is $\mathcal{R}_\tau$, attained by $a(\cdot \mid Z) = q_{\tau,Z}(\cdot \mid Z)$. Moreover, achieving $\mathcal{P}_\tau \le \delta$ implies $I_\tau(R; Z) \ge I_\tau(R; h) - \delta$ (retention of planning information) and $\mathcal{A}_\tau \le \delta$ (accessibility to the planning head).

Discussion. Theorem 1 shows that small planning error requires the representation to retain the teacher's history-dependent distinctions and expose them to the planning head. Their recommendation value depends on the teacher's structure: task alignment makes them relevant to next-item prediction, while heterogeneity lets them complement the student's existing representation. Section 4.3 shows greater preservation of CF structure in learned representations and improved ranking under our collaborative teacher's supervision.

Executing plans reliably. An inaccurate intermediate step can misdirect subsequent item generation (Liu et al., 2024; Gema et al., 2025; Zhang et al., 2026a). At inference, COIN predicts its own plan token and performs standard SID beam search, requiring no teacher call. Execution must accommodate an imperfect native planner and limit error propagation in autoregressive prediction (Bengio et al., 2015). This requires both reliability and consistency: accurate next-item ranking under the selected plan and similar item predictions across plausible alternative plans. The following theorem shows how the planning and execution errors control both properties.

Theorem 2 (Reliability and consistency of plan execution) *Fix the trained student, teacher, and temperature. At history h, write $q_k = q_\tau(k \mid h)$, $p_k = p_{\theta,k}(\cdot \mid Z)$, and let $\bar{p}_\tau \propto \prod_k p_k^{q_k}$ be their normalized geometric pool. The per-history execution error in Equation* (4) *satisfies*

$$\varepsilon_\tau(h) = \underbrace{\mathrm{KL}(\pi_h \| \bar{p}_\tau)}_{f_\tau(h)\text{: pooled prediction error}} + \underbrace{\sum_k q_k \,\mathrm{KL}(\bar{p}_\tau \| p_k)}_{c_\tau(h)\text{: plan disagreement}}, \tag{6}$$

with $\mathcal{E}_\tau = \mathbb{E}_h \varepsilon_\tau(h)$. Define the native selection $\hat{k} = \arg\max_k a_h(k)$, the largest teacher probability $q_{\max} = \max_k q_k$, and the per-history planning error $e_{\mathrm{plan}}(h) = \mathrm{KL}(q_h \| a_h)$, whose expectation is $\mathcal{P}_\tau$. For expected single-target Recall@B or NDCG@B, let $\mathrm{Regret}_B(h)$ be the gap to optimal eligible-item ranking under π_h, and let $\xi_{\hat{k}}(h) \geq 0$ denote the shortfall of finite SID beam search relative to exact ranking under $p_{\hat{k}}$, with $\xi_{\hat{k}}(h) = 0$ for exact conditional ranking. Then

$$\mathrm{Regret}_B(h) \leq \sqrt{\frac{2\varepsilon_\tau(h)}{q_{\hat{k}}}} + \xi_{\hat{k}}(h) \leq \sqrt{\frac{2\varepsilon_\tau(h)}{q_{\max} - \sqrt{2e_{\mathrm{plan}}(h)}}} + \xi_{\hat{k}}(h). \tag{7}$$

The second inequality requires a positive denominator. Furthermore, any two plans with $q_k, q_\ell \geq \alpha > 0$ satisfy $\mathrm{TV}(p_k, p_\ell) \leq \sqrt{2c_\tau(h)/\alpha}$ (Appendices D.3.1–D.3.2).

Discussion. The execution error jointly penalizes inaccurate pooled predictions through $f_\tau(h)$ and disagreement across supported plans through $c_\tau(h)$. When the error is small, several sufficiently supported plans can yield similar, accurate item predictions. A sufficiently small planning error then ensures that the native planner selects a plan with enough teacher support. Together, these properties support reliable rankings and consistent predictions without requiring a uniquely correct intermediate path.

Taken together, the collaborative intent space and ambiguity-aware training establish a compact framework for native planning and item generation, with supervision obtained from existing interaction logs. This design enriches the recommender's representations with collaborative distinctions and supports reliable, consistent predictions across plausible plans. Planning and execution are both amortized into the model's parameters, allowing it to generate and execute its own plans with no teacher or CF embeddings at inference.

4 Experiments

We evaluate COIN around four research questions. **RQ1: Effectiveness.** Does COIN consistently improve full-catalog ranking across datasets, generator architectures, and SID tokenizers? **RQ2: Mechanisms.** Does learning to plan preserve collaborative information, and does executing the plan guide SID decoding? **RQ3: Robustness.** Can COIN maintain ranking quality when its native plan differs from the teacher's preferred plan? **RQ4: Efficiency.** What training and inference overhead does COIN introduce? We first describe the experimental setup, then evaluate ranking performance, examine the planning and execution mechanisms, assess robustness to alternative plans, and finally measure training and deployment cost.

4.1 Experimental Setup

Datasets and baselines. We evaluate COIN on four recommendation benchmarks: Beauty, Toys & Games, and Sports & Outdoors from Amazon (McAuley et al., 2015) for product recommendation, and Yelp[1] for local business recommendation. For each user, we reserve the last interaction for testing and the penultimate interaction for validation, using all preceding interactions for training. We compare COIN with three classes of baselines: traditional recommenders, including SASRec, GRU4Rec, and LightGCN (Kang & McAuley, 2018; Hidasi et al., 2015; He et al., 2020); reasoning-enhanced recommenders, including ReaRec, LARES, and ManCAR (Tang et al., 2026b; Liu et al., 2025c; Yang et al., 2026a); and generative recommenders, including TIGER, ETEGRec, LETTER, EAGER, and DiscRec (Rajput et al., 2023; Liu et al., 2025b; Wang et al., 2024a;b; Liu et al., 2025a). Dataset statistics and preprocessing details are provided in Appendix C.1.

[1] https://www.yelp.com/dataset

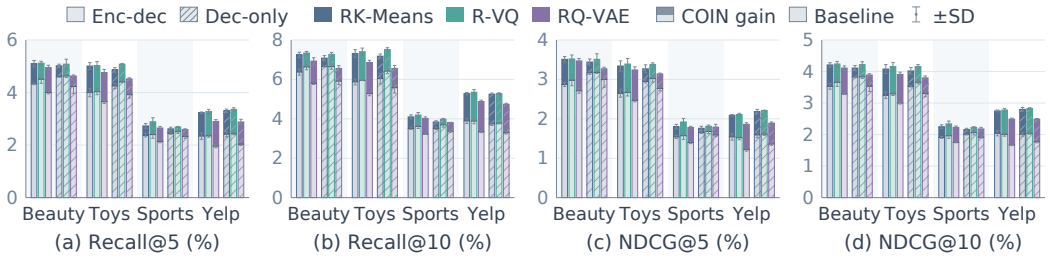


Figure 3: Native ranking performance (%). Light segments show baseline scores; darker extensions show COIN's gains. Whiskers indicate SD over three seeds.

Experiment setup. We build COIN on the GRID framework (Ju et al., 2025), using a unified SID recommendation pipeline with two generator architectures: an encoder-decoder design following TIGER (Rajput et al., 2023), and a decoder-only variant. We construct 4-digit SIDs with RK-Means, R-VQ, and RQ-VAE from FLAN-T5-XL embeddings for Amazon (Chung et al., 2024) and DIGER's precomputed LLaMA embeddings for Yelp (Fu et al., 2026). For each combination of architecture and tokenizer, the unmodified generator serves as the matched baseline. The baseline and COIN share the same SID map, backbone, training configuration, and SID decoding procedure, with COIN adding only collaborative planning. The plan vocabulary contains $K = 512$ tokens, and autoregressive SID decoding uses beam width 10 on Amazon and 20 on Yelp. We report Recall and NDCG at cutoffs 5 and 10, averaged over three random seeds. Additional experimental details are provided in Appendix C.

4.2 Overall Performance

Consistent gains across architectures and tokenizers. Figure 3 evaluates COIN on all four datasets, covering three SID tokenizers and both architectures. We plug COIN into each configuration (denoted "Baseline") through a single additional planning token, keeping everything else unchanged. COIN improves Recall and NDCG at both cutoffs in all 24 configurations, with relative gains of 6.3–46.1% in Recall@10 and 7.3–49.1% in NDCG@10, demonstrating its compatibility with different architectures and tokenizers. The stratified comparison in Appendix Table 3 further shows improvements across different popularity groups, with the largest relative gains often occurring on tail items, demonstrating the effectiveness of collaborative planning for recommending items with fewer training interactions.

Table 2: Test R@10/N@10 (%). Mean ranks: GRs / All. Bold/underline: best/second.

Method	Beauty	Toys	Sports	Yelp	Mean rank ↓ GRs	Mean rank ↓ All
Traditional						
SASRec	6.46/3.58	7.30/4.28	3.46/1.86	5.17/3.31	–	4.25
GRU4Rec	4.70/2.43	4.06/2.29	1.88/0.95	2.64/1.47	–	11.75
LightGCN	4.27/2.23	4.37/2.40	2.58/1.33	**5.94**/3.26	–	9.00
Reasoning-enhanced						
ReaRec	6.70/4.03	7.31/**4.62**	2.96/1.71	4.32/2.95	–	4.50
LARES	6.95/3.72	**7.88**/4.34	3.07/1.58	3.83/2.27	–	5.75
ManCAR	6.92/4.02	7.21/4.36	3.99/2.31	5.51/**3.43**	–	2.63
Generative						
TIGER	5.42/2.96	4.91/2.62	3.18/1.70	3.25/1.68	5.50	9.38
ETEGRec	7.04/3.97	6.53/3.57	3.73/1.99	5.11/2.67	2.00	4.38
LETTER	6.14/3.33	5.40/2.93	2.95/1.61	4.21/2.22	4.75	8.75
EAGER	5.62/3.48	5.02/3.08	2.99/1.61	3.70/1.91	4.75	8.75
DiscRec	6.41/3.50	5.78/3.24	3.40/1.83	4.29/2.27	3.00	6.50
COIN	**7.34/4.24**	7.41/4.16	**4.19/2.34**	5.35/2.78	**1.00**	**2.38**

Comparison with broader recommenders. Table 2 compares COIN with traditional, reasoning-enhanced, and generative recommenders; Appendix C.5 provides the full results with SD. COIN achieves the best mean rank both among generative recommenders and overall, outperforming every listed generative baseline across datasets and metrics. These results extend the evidence beyond the controlled comparisons in Figure 3 and show that collaborative planning remains competitive across diverse recommendation approaches.

4.3 Mechanism Study

Learning to plan enriches representations with collaborative structure. Theorem 1 identifies an incentive for the native representation to retain the teacher's history-dependent distinctions. To visualize this effect, we compare the baseline and COIN models using centered kernel alignment (CKA; Kornblith et al., 2019) and mutual information. Figure 4(a) compares layerwise representations

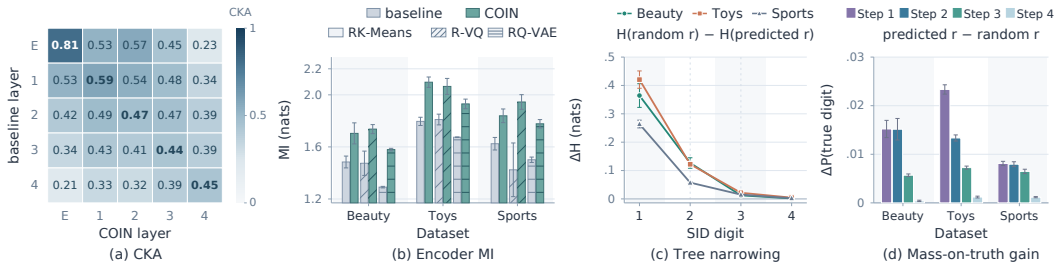


Figure 4: Collaborative planning effects. (a) Layerwise baseline–COIN CKA (E: embedding layer). (b) MI between encoder item representations and fixed CF clusters (truncated axis). (c–d) Entropy reduction and true-digit probability gain for predicted versus uniform plans with a fixed model and true SID prefix. Panels (b–d): three-seed means and SD.

of the same user histories: similarity is high at the embedding stage but lower within the encoder, suggesting that COIN changes how those histories are processed. To examine whether this change reflects richer collaborative information, we encode each item from its SID and average the encoder outputs over SID positions, then estimate mutual information with the fixed CF cluster assignment constructed from interaction data (Ross, 2014). Although both models learn from interaction sequences, Figure 4(b) shows higher estimated MI for COIN in all nine dataset–tokenizer combinations, indicating that more collaborative structure is recoverable from its item representations. Additional studies in Appendix B.1 further examine the roles of teacher input heterogeneity and regularization in providing effective planning supervision.

Executing the plan guides early SID decisions. We next examine how the learned collaborative plan guides execution within the autoregressive SID tree. For the same trained COIN model and history, we compare decoding conditioned on the model's predicted plan r with decoding conditioned on a uniformly random plan, while teacher-forcing the same true SID prefix in both cases. Figure 4(c–d) measures two complementary effects. Panel (c) reports $H(\text{random } r) - H(\text{predicted } r)$, where a positive value indicates reduced uncertainty about the next SID digit. Panel (d) reports the corresponding increase in probability assigned to the true next digit. Together, they show that executing the predicted plan concentrates the next-digit distribution while directing more probability toward the ground-truth branch. Entropy reduction is strongest at the first digit and declines sharply thereafter, while gains in target probability are concentrated in the early positions. This pattern is consistent with hierarchical SID generation: early digits select coarse semantic branches, while the growing prefix constrains later choices, leaving less uncertainty for the plan to resolve (Guo et al., 2026c; Yang et al., 2026b). A complementary ablation in Appendix B.5 shows that plan execution further improves performance on top of collaborative representations.

4.4 Robustness

Structured execution across alternative plans. The preceding section shows that a plan informed by user history can improve item prediction. We now take one step further and ask how COIN responds to different plans. Theorem 2 shows that plans with sufficient teacher support induce similar item distributions when execution disagreement is small. We examine the execution behavior by fixing the history and trained model, forcing different plans, and comparing their cosine distance with the overlap between their top-10 recommendation lists. Figure 5(a) shows a structured, smooth relationship: nearby plans typically yield largely overlapping recommendations, while overlap decreases as the plans move farther apart. The median overlap falls from roughly eight or nine shared items for nearby plans to about three at the largest distances. This pattern suggests that COIN learns a soft, smooth mapping from plans to recommendations, providing a basis for error tolerance.

Native planning preserves ranking quality. At inference, COIN generates a plan from the user history and executes it for item generation. While the planning ability is acquired from teacher supervision, there is no guarantee that COIN always emits the same plan as the teacher for a fixed history, especially given the input heterogeneity of the two models. A plan that conflicts with the user's interests can misdirect subsequent generation. Therefore, we assess whether the generator

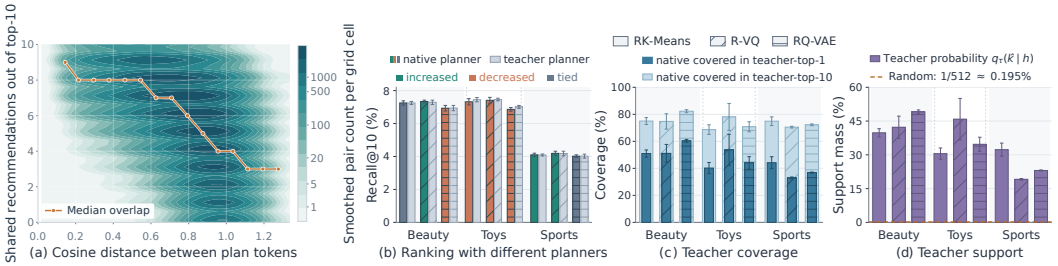


Figure 5: Robustness to alternative plans. (a) Top-10 overlap versus plan cosine distance. (b) Native versus teacher Recall@10. (c) Native-plan coverage in the teacher's top 1 and top 10. (d) Teacher probability of the native plan (uniform reference: $1/512$). Panels (b–d): whiskers are 3-seed SD.

remains reliable when guided by its own plans. We compare the *native planner*, which uses COIN's own top-one plan, with the *teacher planner*, which supplies the teacher's top-one plan for COIN to execute. Figure 5(b) shows that the two modes are effectively tied across the nine dataset–tokenizer combinations, with Recall@10 differing by at most 0.15 percentage points. The close performance coexists with substantial disagreement in plan identity: Figure 5(c) shows only 33–60% exact top-one agreement, yet the native choice falls within the teacher's top ten on 69–82% of histories. Figure 5(d) further shows that these choices receive substantially more teacher probability than uniformly random plans, suggesting that the native planner frequently selects a different alternative that remains supported by the training distribution. Appendix B.4 further shows that commitment outperforms beam-over-plans under the evaluated path-selection and no-refill policy.

4.5 Efficiency

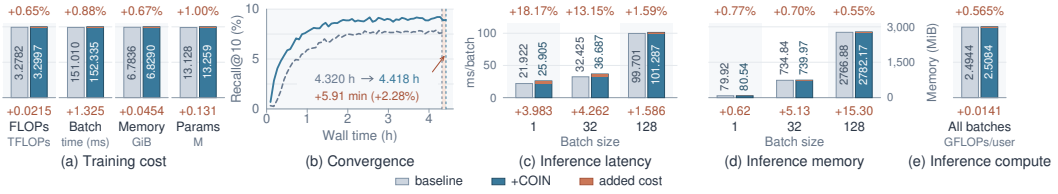


Figure 6: Training and inference efficiency. (a) Training cost and (b) validation Recall@10 versus wall time on Toys. Run times exclude preprocessing. (c–e) Native inference on Toys with RK-Means and an encoder–decoder.

Efficient planning from training to inference. We now examine the computational cost of COIN during training and inference. Both are measured on a single RTX A6000 in FP32, using the encoder–decoder backbone with RK-Means on Toys; inference uses a shared KV-cached decoder and SID beam width of 10. Figure 6(a–b) shows that training compute, batch time, and memory increase by less than 1%, while total run time increases by only 2.28%. At inference, COIN maintains nearly the same memory and compute costs as the baseline, while its latency overhead diminishes with larger batches (Figure 6(c–e)). Batching concurrent requests is a standard strategy for improving GPU throughput, indicating that COIN adds very little overhead in deployments when aggregating concurrent requests. Appendix B.7 further shows the preprocessing costs of COIN.

5 Conclusion

We propose COIN, which identifies collaborative intent as a reasoning language native to recommendation and complementary to semantic IDs. This language provides a behavioral basis for forming intermediate hypotheses about future interactions, allowing collaborative structure to guide item generation. Ambiguity-aware transfer helps the recommender internalize this ability while supporting reliable generation across plausible plans. Experiments across datasets, architectures, and tokenizers show consistent ranking improvements with low overhead and self-contained inference. These findings suggest that a compact, task-aligned reasoning language can enrich native GRs while preserving their simplicity and efficiency.

AI USE STATEMENT

We used generative AI coding agents to assist with implementation under full human supervision. All experiments and data analyses were designed, conducted, and verified by the human authors. AI use in preparing the manuscript was limited to polishing the writing and improving presentation, including figure preparation. The authors reviewed all AI-assisted code and content and take full responsibility for the methods, results, interpretations, and final manuscript.

REFERENCES

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*, pp. 1007–1014, 2023.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

Shiteng Cao, Kaian Jiang, Yunlong Gong, and Zhiheng Li. Twistar: Think fast, think slow, then act, generative recommendation with adaptive reasoning. *arXiv preprint arXiv:2605.11553*, 2026.

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 & data mining*, pp. 2942–2951, 2020.

Yiwen Chen, Fuwei Zhang, Zehao Chen, Hehan Li, Peizhi Xu, Hanmeng Liu, Shuanglong Li, Xin Pei, Fuzhen Zhuang, and Zhao Zhang. Lasar: Latent adaptive semantic aligned reasoning for generative recommendation. *arXiv preprint arXiv:2605.10207*, 2026.

Lei Cheng, Haonan Hu, Beibei Kong, Yudong Li, Zang Li, Yunsheng Pang, Hongyang Su, Jianchao Tu, Yunlong Wang, Bing Wen, et al. Tgr: Advancing industrial recommendation from generative-paradigm ranking toward unified generation and reasoning. *arXiv preprint arXiv:2609.00986*, 2026.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of machine learning research*, 25(70):1–53, 2024.

Wenhao Deng, Junchen Fu, Hanwen Du, Alexandros Karatzoglou, Ioannis Arapakis, Hangjun Guo, Kaiwen Zheng, Yongxin Ni, and Joemon M Jose. Recrec: Latent interests recursive reasoning for sequential recommendation. *arXiv preprint arXiv:2607.12945*, 2026.

Junchen Fu, Xuri Ge, Alexandros Karatzoglou, Ioannis Arapakis, Suzan Verberne, Joemon M Jose, and Zhaochun Ren. Differentiable semantic id for generative recommendation. *arXiv preprint arXiv:2601.19711*, 2026.

Aryo Pradipta Gema, Alexander Hägele, Runjin Chen, Andy Arditi, Jacob Goldman-Wetzler, Kit Fraser-Taliente, Henry Sleight, Linda Petrini, Julian Michael, Beatrice Alex, et al. Inverse scaling in test-time compute. *arXiv preprint arXiv:2507.14417*, 2025.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM conference on recommender systems*, pp. 299–315, 2022.

Chengcheng Guo, Kuo Cai, Yu Zhou, Qiang Luo, Ruiming Tang, Han Li, Kun Gai, and Guorui Zhou. Promise: Process reward models unlock test-time scaling laws in generative recommendations. *arXiv preprint arXiv:2601.04674*, 2026a.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Zihao Guo, Jian Wang, Ruxin Zhou, Youhua Liu, Jiawei Guo, Jun Zhao, Xiaoxiao Xu, Yongqi Liu, and Kaiqiao Zhan. S2gr: Stepwise semantic-guided reasoning in latent space for generative recommendation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 1498–1507, 2026b.

Zitian Guo, Yupeng Hou, Clark Mingxuan Ju, Neil Shah, and Julian McAuley. Mlps are efficient distilled generative recommenders. *arXiv preprint arXiv:2605.12617*, 2026c.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 639–648, 2020.

Yingzhi He, Yan Sun, Junfei Tan, Yuxin Chen, Xiaoyu Kong, Chunxu Shen, Xiang Wang, An Zhang, and Tat-Seng Chua. Reasoning over semantic ids enhances generative recommendation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 1638–1649, 2026a.

Yinhan He, Liam Collins, Bhuvesh Kumar, Jundong Li, Neil Shah, and Donald Loveland. Implicit reasoning for large language model-based generative recommendation. *arXiv preprint arXiv:2606.14142*, 2026b.

Tom Heskes. Selecting weighting factors in logarithmic opinion pools. *Advances in neural information processing systems*, 10, 1997.

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

Minjie Hong, Zetong Zhou, Zirun Guo, Ziang Zhang, Ruofan Hu, Weinan Gan, Jieming Zhu, and Zhou Zhao. Generative reasoning recommendation via llms. *arXiv preprint arXiv:2510.20815*, 2025.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the association for computational linguistics: ACL 2023*, pp. 8003–8017, 2023.

Hao Jiang, Peiru Du, Pengfei Yao, Mengting Li, Siyuan Lou, Kuo Cai, Sheng Yu, Qiang Luo, Jian Liang, Ruiming Tang, et al. Whisperrec: Latent reasoning for efficient foundation recommendation models. *arXiv preprint arXiv:2607.26621*, 2026a.

Jie Jiang, Yangru Huang, Zeyu Wang, Changping Wang, Yuling Xiong, Jun Zhang, and Huan Yu. Spend search where it pays: Value-guided structured sampling and optimization for generative recommendation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 2098–2109, 2026b.

Jie Jiang, Yang Wu, Qian Li, Yuling Xiong, Yihang Su, Junbang Huo, Longfei Lu, Jun Zhang, and Huan Yu. Diffureason: Bridging latent reasoning and generative refinement for sequential recommendation. *arXiv preprint arXiv:2602.09744*, 2026c.

Jie Jiang, Yang Wu, Qian Li, Yuling Xiong, Yihang Su, Junbang Huo, Longfei Lu, Jun Zhang, and Huan Yu. Diffureason: Bridging latent reasoning and generative refinement for sequential recommendation. *arXiv preprint arXiv:2602.09744*, 2026d.

Jie Jiang, Jun Zhang, Yi Li, Yue Liu, Changping Wang, Yuan Wang, Yuling Xiong, Xun Liu, Haiyang Wu, Qian Li, et al. Gpr: Towards a generative pre-trained one-model paradigm for large-scale advertising recommendation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 7477–7488, 2026e.

Clark Mingxuan Ju, Liam Collins, Leonardo Neves, Bhuvesh Kumar, Louis Yufeng Wang, Tong Zhao, and Neil Shah. Generative recommendation with semantic ids: a practitioner's handbook. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 6420–6425, 2025.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the association for computational linguistics*, 3:211–225, 2015.

Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, 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*, pp. 2615–2623, 2019.

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

Mingfu Liang, Yufei Li, Jay Xu, Kavosh Asadi, Xi Liu, Shuo Gu, Kaushik Rangadurai, Frank Shyu, Shuaiwen Wang, Song Yang, et al. Gr2: Generative reasoning re-ranker. *arXiv preprint arXiv:2602.07774*, 2026.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. Llara: Large language-recommendation assistant. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 1785–1795, 2024.

Xinyu Lin, Pengyuan Liu, Wenjie Wang, Yicheng Hu, Chen Xu, Fuli Feng, Qifan Wang, and Tat-Seng Chua. Bringing reasoning to generative recommendation through the lens of cascaded ranking. In *Proceedings of the ACM Web Conference 2026*, pp. 8939–8949, 2026.

Chang Liu, Yimeng Bai, Xiaoyan Zhao, Yang Zhang, Fuli Feng, and Wenge Rong. Discrec: Disentangled semantic-collaborative modeling for generative recommendation. *arXiv preprint arXiv:2506.15576*, 2025a.

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

Enze Liu, Bowen Zheng, Xiaolei Wang, Wayne Xin Zhao, Jinpeng Wang, Sheng Chen, and Ji-Rong Wen. Lares: Latent reasoning for sequential recommendation. *arXiv preprint arXiv:2505.16865*, 2025c.

Enze Liu, Zhen Tian, and Wayne Xin Zhao. Collaborative memory augmentation for generative recommendation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 3164–3175, 2026a.

Ryan Liu, Jiayi Geng, Addison J Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas L Griffiths. Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse. *arXiv preprint arXiv:2410.21333*, 2024.

Xiang Liu, Jingsong Su, Shuqi Zhao, Pengbo Mo, Yiming Qiu, Huimu Wang, Mingming Li, Jiao Dai, Jizhong Han, and Songlin Hu. Cogrec: Structure-cognitive fast-and-slow reasoning for generative recommendation. *arXiv preprint arXiv:2607.24402*, 2026b.

Zhanyu Liu, Shiyao Wang, Xingmei Wang, Rongzhou Zhang, Jiaxin Deng, Honghui Bao, Jinghao Zhang, Wuchao Li, Pengfei Zheng, Xiangyu Wu, et al. Onerec-think: In-text reasoning for generative recommendation. *arXiv preprint arXiv:2510.11639*, 2025d.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pp. 43–52, 2015.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Yunsheng Pang, Zijian Liu, Yudong Li, Shaojie Zhu, Zijian Luo, Chenyun Yu, Sikai Wu, Shichen Shen, Cong Xu, Bin Wang, et al. Higr: Industrial-scale hierarchical generative slate recommendation framework in tencent. *arXiv preprint arXiv:2512.24787*, 2025.

Gustavo Penha, Juan Elenter, Claudia Hauff, Hugues Bouchard, Paul Bennett, and Mounia Lalmas. The disconnect between better descriptive reasoning trace quality and recommendation effectiveness. *arXiv preprint arXiv:2608.23154*, 2026.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

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

Brian C Ross. Mutual information between discrete and continuous data sets. *PloS one*, 9(2):e87357, 2014.

Yifan Shao and Peilin Zhou. Intent-guided reasoning for sequential recommendation. *arXiv preprint arXiv:2512.14034*, 2025.

Teng Shi, Weicong Qin, Weijie Yu, Xiao Zhang, Ming He, Jianping Fan, and Jun Xu. Bridging search and recommendation through latent cross reasoning. *arXiv preprint arXiv:2508.04152*, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Jiakai Tang, Xu Chen, Wen Chen, Jian Wu, Yuning Jiang, and Bo Zheng. Parallel latent reasoning for sequential recommendation. *arXiv preprint arXiv:2601.03153*, 2026a.

Jiakai Tang, Sunhao Dai, Teng Shi, Jun Xu, Xu Chen, Wen Chen, Jian Wu, and Yuning Jiang. Think before recommend: Unleashing the latent reasoning power for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2026b.

OneRec Team, Biao Yang, Boyang Ding, Chenglong Chu, Dunju Zang, Fei Pan, Han Li, Hao Jiang, Honghui Bao, Huanjie Wang, et al. Onereason technical report. *arXiv preprint arXiv:2606.06260*, 2026.

Huanjie Wang, Xinchen Luo, Honghui Bao, Zhang Zixing, Lejian Ren, Yunfan Wu, Hongwei Zhang, Liwei Guan, and Guang Chen. Pit: A dynamic personalized item tokenizer for end-to-end generative recommendation. *arXiv preprint arXiv:2602.08530*, 2026.

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 *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 2609–2634, 2023.

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

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Tianxin Wei, Xuying Ning, Xuxing Chen, Ruizhong Qiu, Yupeng Hou, Yan Xie, Shuang Yang, Zhigang Hua, and Jingrui He. Cofirec: Coarse-to-fine tokenization for generative recommendation. *arXiv preprint arXiv:2511.22707*, 2025.

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

Haibo Xing, Hao Deng, Yucheng Mao, Lingyu Mu, Jinxin Hu, Yi Xu, Hao Zhang, Jiahao Wang, Shizhun Wang, Yu Zhang, et al. Reg4rec: Reasoning-enhanced generative model for large-scale recommendation systems. In *2026 IEEE 42nd International Conference on Data Engineering (ICDE)*, pp. 3799–3812. IEEE, 2026a.

Haibo Xing, Hao Deng, Lingyu Mu, Jinxin Hu, Yu Zhang, Xiaoyi Zeng, and Jing Zhang. Learning to reflect and correct: Towards better decoding trajectories for large-scale generative recommendation. *arXiv preprint arXiv:2602.23639*, 2026b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Kun Yang, Yuxuan Zhu, Yazhe Chen, Siyao Zheng, Bangyang Hong, Kangle Wu, Yabo Ni, Anxiang Zeng, Cong Fu, and Hui Li. Mancar: Manifold-constrained latent reasoning with adaptive test-time computation for sequential recommendation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 5998–6009, 2026a.

Shangxin Yang, Min Gao, Zongwei Wang, and Junliang Yu. Where reasoning matters: Rethinking latent reasoning in semantic id-based generative recommendation. *arXiv preprint arXiv:2607.12425*, 2026b.

Yuhao Yang, Zhi Ji, Zhaopeng Li, Yi Li, Zhonglin Mo, Yue Ding, Kai Chen, Zijian Zhang, Jie Li, LIU LIN, et al. Sparse meets dense: Unified generative recommendations with cascaded sparse-dense representations. *Advances in Neural Information Processing Systems*, 38:93746–93770, 2026c.

Junjie Zhang, Beichen Zhang, Wenqi Sun, Hongyu Lu, Wayne Xin Zhao, Yu Chen, and Ji-Rong Wen. Slow thinking for sequential recommendation. *arXiv preprint arXiv:2504.09627*, 2025.

Luankang Zhang, Yonghao Huang, Hang Lv, Xuyang Zhi, Mingjia Yin, Yuyang Ye, Wei Guo, Hao Wang, and Enhong Chen. Why thinking hurts: Diagnosing and rectifying linguistic inertia in large language models for recommendation. *arXiv preprint arXiv:2602.16587*, 2026a.

Yang Zhang, Wenxin Xu, Xiaoyan Zhao, Wenjie Wang, Fuli Feng, Xiangnan He, and Tat-Seng Chua. Reinforced latent reasoning for llm-based recommendation. In *International Conference on Learning Representations*, volume 2026, pp. 128449–128470, 2026b.

Shuqi Zhao, Jingsong Su, Xiang Liu, Xingzhi Yao, Yiming Qiu, Huimu Wang, Liang Lin, Pengbo Mo, Mingming Li, Jiao Dai, et al. Holorec: Holistic encoding and interleaved reasoning for generative recommendation. *arXiv preprint arXiv:2606.15331*, 2026.

Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *proceedings of the 30th acm international conference on information & knowledge management*, pp. 4653–4664, 2021.

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

Zaiyi Zheng, Liang Wu, Guanghui Min, Yaochen Zhu, Liangjie Hong, Chen Chen, and Jundong Li. Sapo: Step-aligned policy optimization for reasoning-based generative recommendation. *arXiv preprint arXiv:2605.17648*, 2026.

Guorui Zhou, Honghui Bao, Jiaming Huang, Jiaxin Deng, Jinghao Zhang, Junda She, Kuo Cai, Lejian Ren, Lu Ren, Qiang Luo, et al. Openonerec technical report. *arXiv preprint arXiv:2512.24762*, 2025.

Zhuang Zhuang, Zhipeng Wei, Rongfeng Guo, Shijie Li, Peng Zhao, Jie Chen, and Fei Pan. Evoreason: Self-evolving reasoning primitive-guided on-policy distillation for latent reasoning in generative recommendation. *arXiv preprint arXiv:2607.29010*, 2026.

A Related Work

Reasoning in native GRs. Native GRs generate semantic IDs (SIDs) with compact recommendation backbones trained from scratch on interaction data (Rajput et al., 2023; Pang et al., 2025). Existing methods introduce reasoning through different forms of intermediate supervision. LLM-assisted approaches convert external guidance into reason tokens or intent embeddings, allowing native generators to use LLM knowledge while adding separate adaptation and guidance preparation (Cheng et al., 2026; Jiang et al., 2026e). Semantic approaches use coarse item abstractions to guide finer SID decisions (Guo et al., 2026b; Zhao et al., 2026). Their reasoning steps follow the structure used to encode items, tying intermediate supervision to the SID hierarchy. Latent approaches learn thought tokens or recursive state updates from recommendation objectives (Zhang et al., 2025; Yang et al., 2026b; Jiang et al., 2026d). These states are trained through auxiliary supervision or reinforcement learning and introduce additional decoding or refinement steps. Related approaches refine historical evidence during SID generation through parallel latent reasoning (Lin et al., 2026). Verification and correction instead obtain feedback from candidate trajectories: process rewards guide search over SID prefixes (Guo et al., 2026a), while reflection mechanisms prune inconsistent reasoning paths or revise the generated candidates (Xing et al., 2026a;b).

Reasoning in LLM-based GRs. LLM-based GRs connect linguistic reasoning to item prediction through item–language alignment (Zhou et al., 2025; He et al., 2026a) and training on recommendation rationales and outcomes (Liu et al., 2025d; Hong et al., 2025; Team et al., 2026). The resulting reasoning trajectories interpret interaction histories and form preference hypotheses before generating items or reranking candidate lists (Liang et al., 2026). More detailed supervision connects intermediate reasoning to individual SID decisions, using interleaved reasoning segments, rewards for individual steps, or explicit transitions within the SID structure (Zheng et al., 2026; Liu et al., 2026b). However, empirical analyses report that more descriptive reasoning does not consistently improve recommendation, and textual rationales can reduce reliance on historical interaction evidence (Penha et al., 2026; Zhang et al., 2026a). Implicit reasoning reduces the cost of generating these traces by carrying intermediate computation in latent states or learned pause tokens (Zhang et al., 2026b; Chen et al., 2026; He et al., 2026b). Distillation further transfers textual reasoning into compact latent representations, moving part of the reasoning cost into supervision preparation and training (Jiang et al., 2026a; Zhuang et al., 2026). Other methods focus search on promising SID branches (Jiang et al., 2026b). Adaptive tool selection and parallel reasoning generation provide additional ways to control serving cost (Cao et al., 2026; Jiang et al., 2026c).

Reasoning in traditional sequential recommendation. Reasoning-enhanced sequential recommenders refine user representations before scoring candidate items. ReaRec and LARES extend history encoding through successive thought states or recurrent updates, allowing the model to revise its preference representation over multiple steps (Tang et al., 2026b; Liu et al., 2025c). Parallel reasoning streams and multiple latent interests maintain several preference hypotheses during this process (Tang et al., 2026a; Deng et al., 2026). Supervision also shapes what these updates retain: intent guidance and collaborative priors connect intermediate states to behavioral evidence (Shao & Zhou, 2025; Yang et al., 2026a), while latent cross-reasoning identifies useful evidence from

search histories (Shi et al., 2025). These approaches expand reasoning through its depth, the interests represented, or the evidence available for refinement. They retain the standard recommendation interface, using the refined user representation to score candidate items.

Collaborative information in GRs. Collaborative signals relate items through shared user behavior, supplying information that content similarity alone may not preserve. Existing GRs incorporate these signals into item codes through collaborative alignment, joint semantic and behavioral representations, or separate levels of a tokenization hierarchy (Wang et al., 2024a; Xiao et al., 2025; Wei et al., 2025). Learning the tokenizer together with the recommender further allows interaction feedback to shape the item encoding (Liu et al., 2025b; Wang et al., 2026). Other methods combine semantic and behavioral information during generation through separate branches or cascaded sparse and dense representations (Wang et al., 2024b; Liu et al., 2025a; Yang et al., 2026c). LLM-based GRs incorporate collaborative knowledge through alignment tasks (Zheng et al., 2024; Hong et al., 2025), while memory-augmented models retrieve relevant behavior from other users as additional context (Liu et al., 2026a). Collaborative information also appears in reasoning supervision: S^2GR derives intermediate targets from collaboratively enriched codebooks (Guo et al., 2026b), and ManCAR constrains latent reasoning with an interaction-graph prior (Yang et al., 2026a). COIN uses collaborative intent as an intermediate prediction that conditions item generation, with a planning vocabulary constructed separately from the item SID codebook.

B Additional Experiments

B.1 Teacher Analysis

Section 3.1 introduces two choices for constructing the intent teacher: freely learned item-ID embeddings and strong regularization. We examine how these choices affect the supervision transferred to the native recommender, extending the representation analysis in Section 4.3. Figure 7 relates the accessibility of teacher forecasts from the baseline representation to downstream ranking performance, then examines the effect of teacher dropout.

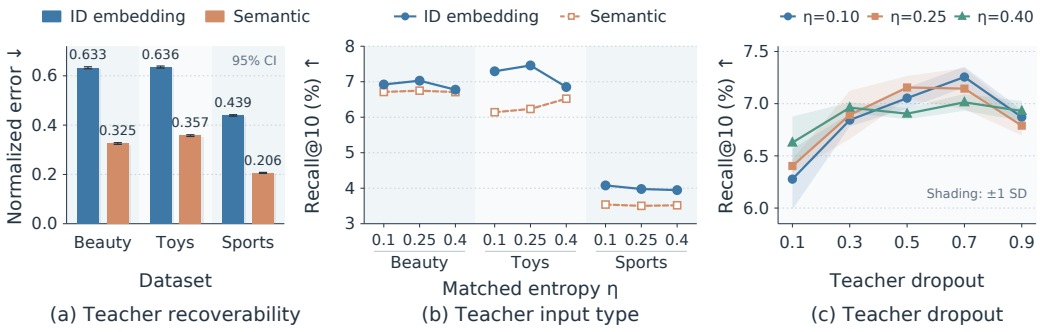


Figure 7: Teacher input and regularization. (a) Normalized linear recovery error from frozen baseline representations. (b) Student Recall@10 with ID-embedding versus semantic teachers at matched entropy. (c) Recall@10 versus dropout on Beauty (shading: ± 1 SD). η: normalized mean transfer entropy.

Teacher inputs and comparison protocol. We compare a teacher whose item-ID embeddings are learned jointly with the sequence predictor against one with fixed semantic input embeddings. Both predict the same $K = 512$ collaborative regions. To control the average concentration of their supervision, we calibrate a separate scalar temperature for each teacher to match the training-sampler-weighted mean entropy, $\eta = \mathbb{E}[H(q_\tau)]/\log K$, at $\eta \in \{0.10, 0.25, 0.40\}$. The input comparison uses RK-Means with matched student initialization and sampling within each dataset.

Input heterogeneity and forecast accessibility. We first ask how readily each teacher's forecasts can be recovered from the native generator before planning supervision. Let Z_0 denote the representation of the frozen baseline trained without planning, and let ℓ_t denote teacher t's temperature-scaled

logits, centered across the K regions. We fit a ridge regression from Z_0 to ℓ_t and measure normalized held-out reconstruction error:

$$E_t = \frac{\sum_{\text{test}} \|\ell_t - \widehat{\ell}_t\|_2^2}{\sum_{\text{test}} \|\ell_t - \overline{\ell}_{t,\text{train}}\|_2^2},$$

where $\overline{\ell}_{t,\text{train}}$ is the mean training target. The denominator is the test error of predicting this fixed training mean, so larger E_t indicates lower linear recoverability.

Features are standardized using training statistics, and the ridge coefficient is selected on validation data. We average the errors over three training-sample repeats, each containing 262,144 draws from the student-sampler distribution with multiplicity retained. Both teachers use the same sampled examples and regularization grid; validation and testing include all official held-out users. Uncertainty is estimated with 5,000 paired test-user bootstrap resamples, conditional on the fitted teachers and regression models. Scalar temperature rescaling leaves the normalized errors unchanged.

Figure 7(a) shows consistently higher recovery error for the ID-embedding teacher across the datasets. Its forecasts are therefore less accessible through a linear readout of the baseline representation, suggesting that learning input embeddings from interactions exposes distinctions that the native model does not already express as readily. This supports the heterogeneity motivation in Section 3.1. The paired transfer results determine whether these less accessible distinctions also provide useful supervision.

Input choice affects transfer beyond mean entropy. Figure 7(b) shows that the ID-embedding teacher improves student ranking at every matched entropy level across the datasets, with the clearest advantage on Toys. Read together, panels (a) and (b) suggest that the ID-embedding teacher provides useful structure that the baseline does not readily represent. This is consistent with the task-aligned complementarity described in Appendix D.2.2 and the richer collaborative representations observed in Section 4.3.

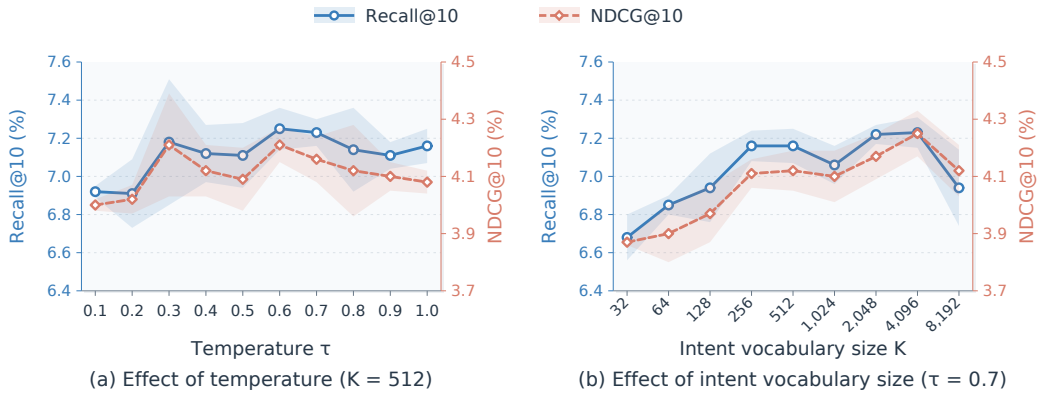
Regularization affects transfer beyond temperature scaling. Figure 7(c) shows that moving from weak dropout to strong regularization generally improves transfer, while the most aggressive dropout reduces performance. Dropout shapes the distinctions learned by the teacher, while temperature controls how those fitted forecasts are distributed across alternatives during transfer. The results support treating the two choices jointly: the teacher should learn useful behavioral structure and expose it through supervision with enough breadth to train reliable execution across plausible plans.

B.2 Hyperparameter Sensitivity

We examine the transfer temperature τ and intent vocabulary size K introduced in Section 3.1. Temperature controls the concentration of the teacher distribution used during ambiguity-aware training, while K determines the granularity of the collaborative planning space. Figure 8 reports native-planner performance on Beauty with RK-Means. We vary τ from 0.1 to 1.0 with $K = 512$, and separately vary K from 32 to 8,192 with $\tau = 0.7$.

Transfer temperature. Figure 8(a) shows that the most concentrated teacher distributions produce the weakest ranking results. Allowing more probability on alternative intents improves both metrics, after which performance remains competitive across a broad temperature range. Recall is relatively stable through much of this range, while NDCG declines modestly toward the highest temperatures. The pattern supports the motivation for ambiguity-aware transfer: plausible alternatives provide useful training variation, but continually broadening the distribution does not guarantee better guidance.

Intent vocabulary size. Figure 8(b) shows that expanding a very coarse intent vocabulary initially improves both ranking metrics. Performance then becomes less monotonic: several intermediate and larger vocabularies are competitive, but the largest tested vocabulary reduces ranking quality. The default $K = 512$ already provides effective guidance, although some larger choices yield additional gains. This pattern is consistent with a balance between the detail available in the planning space and the difficulty of learning to distinguish more intent regions. For every K, the plan remains a single token, so this choice changes the range of available intents without lengthening the planning sequence.



(a) Effect of temperature (K = 512) (b) Effect of intent vocabulary size (τ = 0.7)

Figure 8: Native-planner sensitivity on Beauty with RK-Means: (a) temperature τ at $K = 512$; (b) vocabulary size K at $\tau = 0.7$. Shading: ± 1 SD.

Table 3: Popularity-stratified NDCG@10 (three-seed means). +COIN uses native planning; parentheses show relative gains over the matched baseline.

Dataset	Tokenizer	Head		Torso		Tail	
		Baseline	+COIN	Baseline	+COIN	Baseline	+COIN
Beauty	RK-Means	.0946	.1120 (+18.4%)	.0359	.0440 (+22.5%)	.0124	.0143 (+15.5%)
	R-VQ	.0988	.1066 (+7.9%)	.0367	.0448 (+21.9%)	.0129	.0165 (+27.7%)
	RQ-VAE	.0936	.1105 (+18.0%)	.0335	.0413 (+23.5%)	.0096	.0145 (+51.9%)
Toys	RK-Means	.0583	.0686 (+17.6%)	.0399	.0518 (+29.9%)	.0176	.0231 (+31.7%)
	R-VQ	.0567	.0697 (+22.8%)	.0405	.0520 (+28.2%)	.0187	.0241 (+28.9%)
	RQ-VAE	.0526	.0668 (+27.0%)	.0383	.0502 (+31.2%)	.0158	.0213 (+34.7%)
Sports	RK-Means	.0594	.0659 (+11.0%)	.0155	.0199 (+28.0%)	.0028	.0041 (+49.7%)
	R-VQ	.0560	.0641 (+14.5%)	.0186	.0225 (+21.2%)	.0036	.0053 (+48.3%)
	RQ-VAE	.0555	.0650 (+17.1%)	.0137	.0192 (+39.9%)	.0023	.0046 (+95.6%)

B.3 Popularity-Stratified Performance

We further examine whether the gains of COIN depend on target-item popularity. Item popularity is computed exclusively from the *training* interactions. Items are sorted from most to least frequent and partitioned according to cumulative interaction mass: the first third forms the head, the second third forms the torso, and the final third forms the tail. Items with zero training count are assigned to the tail.

This partition places a small set of frequently observed items in the head and most of the catalog in the tail. Each test user is assigned to the popularity group of the target item. The resulting groups are identical across methods and seeds, allowing performance changes to be compared on the same users.

For method m, matched baseline b, popularity group s, and metric M, the reported relative improvement is computed after averaging each model over seeds:

$$\Delta^{M}_{m,s} = 100 \times \frac{\overline{M}_{m,s} - \overline{M}_{b,s}}{\overline{M}_{b,s}}, \qquad \overline{M}_{m,s} = \frac{1}{3}\sum_{q=1}^{3} M^{(q)}_{m,s}. \tag{8}$$

Averaging scores before computing the relative gain reduces the influence of unstable per-seed ratios when baseline performance is low, particularly for tail items.

Table 3 shows that adding COIN improves NDCG@10 across every dataset, tokenizer, and popularity group. The gains therefore extend throughout the catalog, including items with limited interaction evidence. Relative improvements are often larger in the torso and tail, with the clearest pattern on Sports and under RQ-VAE, although the ordering varies across settings. This is consistent

with collaborative planning supplying useful behavioral guidance across popularity levels. Tail performance remains lower in absolute terms, and its large relative gains partly reflect weaker starting baselines. The results support broad coverage of COIN's benefits while showing that sparse-item recommendation remains challenging.

B.4 Commit-One-Plan versus Beam-over-Plans

We compare two native decoding strategies for COIN. The default strategy commits to the generator's top-one plan before beam-searching the SID body. Beam-over-plans instead maintains a beam over the generator's top ten plans. Both strategies use RK-Means, $K = 512$, and full-catalog evaluation with history masking. Within each dataset, they share the same checkpoint and temperature, use SID beam width 10, and return ten output slots.

Table 4 reports ranking performance together with the duplicate-slot rate. A duplicate slot occurs when different plan paths decode to the same item. Repeated items are marked by placeholders that retain their positions in the output list without refilling, so these slots contribute no additional recommendation.

Table 4: Native commit-one-plan versus beam-over-plans decoding (%). R@10 and N@10 denote Recall@10 and NDCG@10; Dup@10 is the duplicate-slot rate. Bold marks the better result per dataset.

Dataset	Decoding	R@10 ↑	N@10 ↑	Dup@10 ↓
Beauty	Commit	**7.11**	**4.13**	**0.00**
	Beam-10	6.48	3.87	14.18
Toys	Commit	**7.47**	**4.22**	**0.00**
	Beam-10	6.68	3.89	22.00
Sports	Commit	**3.99**	**2.18**	**0.00**
	Beam-10	3.62	2.02	14.44

Commitment improves both ranking metrics across all three datasets under the evaluated path-selection and no-refill policy, although duplicate slots can account for part of the gap. Running separate SID beams in parallel for multiple plans adds serving compute, while incorporating plans into a single shared beam makes their paths compete both during search and for positions in the final ranked list. We therefore commit to one plan to preserve minimal inference overhead. The smooth structure in Figure 5(a), where nearby plans yield largely overlapping recommendations, further supports this choice by suggesting limited additional coverage from exploring similar plans.

B.5 Representation Learning and Guided Decoding

We compare the native baseline, a representation-only variant, and full COIN to examine the contributions of planning supervision and guided decoding. The two COIN variants use the identical teacher and transfer settings and differ only in whether the plan token is fed to the SID decoder. The representation-only variant retains the plan-prediction objective during training but generates SIDs without plan conditioning. Full COIN learns to predict the plan and executes its native top-one plan during SID generation. All variants use RK-Means and three random seeds. Within each dataset, both COIN variants use the same transfer temperature.

Table 5: Ablation of COIN's two effects. R@10 and N@10 are Recall@10 and NDCG@10 (%). Values are mean ± SD over three seeds. Bold marks the best mean.

Variant	Beauty		Toys		Sports	
	R@10	N@10	R@10	N@10	R@10	N@10
Baseline	6.37 ± 0.15	3.53 ± 0.09	5.87 ± 0.15	3.25 ± 0.10	3.52 ± 0.06	1.91 ± 0.03
Representation only	6.98 ± 0.10	4.04 ± 0.08	6.90 ± 0.05	3.94 ± 0.10	3.78 ± 0.08	2.11 ± 0.06
Full COIN	**7.25** ± 0.13	**4.21** ± 0.07	**7.32** ± 0.19	**4.08** ± 0.14	**4.10** ± 0.10	**2.25** ± 0.06

Table 5 shows that planning supervision improves both ranking metrics across datasets even without plan-conditioned decoding. Full COIN consistently improves further, indicating that executing the

plan adds value beyond the benefit of representation learning alone. Together, these results support the complementary roles of collaborative representation learning and guided item generation.

B.6 Sampling and Robustness to Plausible Alternative Plans

We examine whether sampling plans during training improves robustness to plausible alternatives at inference. COIN samples training plans from the teacher distribution, whereas the top-1-only variant always conditions SID generation on the teacher's highest-probability plan. The variants use identical teacher distributions and all other experimental settings. At inference, both execute either the teacher's first-ranked (top-1) or second-ranked (top-2) plan with the same SID decoding procedure.

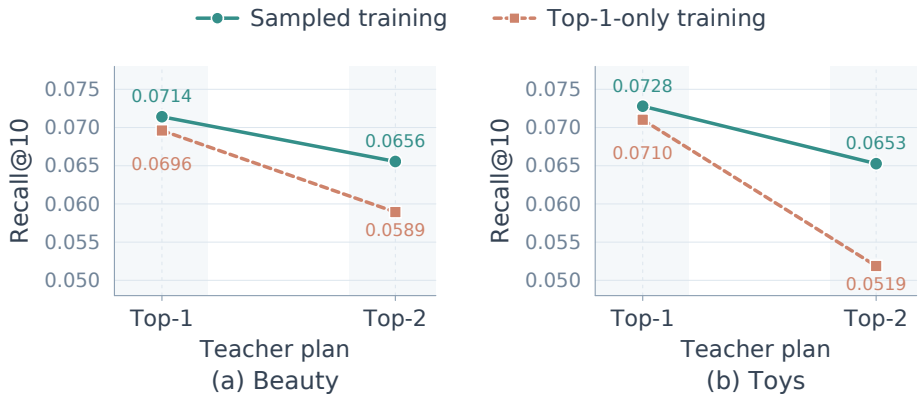


Figure 9: Robustness to a plausible alternative plan. Recall@10 under the teacher's first- and second-ranked plans after sampled or top-1-only training.

Figure 9 shows that switching from the teacher's top-1 to top-2 plan causes a smaller performance drop with sampled training, confirming that sampling helps the decoder preserve recommendation quality across plausible alternative plans.

B.7 Preprocessing Cost

We measure COIN's additional preprocessing before generator training on Toys using a single RTX A6000, repeating the complete procedure three times. The measurement covers collaborative vocabulary construction, teacher training including validation, and soft-label generation for all training windows. Table 6 reports the time and peak memory for each stage and for the complete run.

Table 6: COIN preprocessing cost on Toys. Time is mean ± SD over three repetitions. Memory values are measured peaks; the end-to-end total includes preparation beyond the five listed stages.

Stage	**Time (s)**	**Peak memory (GiB)**		
		CPU RAM	**CUDA allocated**	**CUDA reserved**
Co-occurrence and PPMI	0.37 ± 0.01	0.495	0.000	0.000
Randomized SVD (rank 64)	0.94 ± 0.41	0.507	0.000	0.000
Collaborative clusters ($K = 512$)	2.40 ± 1.09	0.998	0.000	0.002
Teacher training and validation	191.46 ± 62.27	1.236	0.280	0.357
Soft labels for all training windows	29.15 ± 1.26	19.193	0.482	0.521
End-to-end total	**259.63** ± 68.50	**19.196**	**0.482**	**0.559**

The complete preprocessing on Toys takes about 4.3 minutes on average. Teacher training dominates runtime, while soft-label generation drives peak CPU memory use; collaborative vocabulary construction is comparatively inexpensive. This preparation is completed before generator training and adds no teacher computation at inference. The end-to-end total also includes data loading, checkpointing, and input materialization. Its time SD is computed across complete runs, and its memory values are the peaks over the full procedure.

Table 7: Dataset statistics.

Dataset	Users	Items	Interactions	Avg. length
Beauty	22,363	12,101	198,502	8.88
Toys	19,412	11,924	167,597	8.63
Sports	35,598	18,357	296,337	8.32
Yelp	30,431	20,033	316,354	10.40

C Experimental Details

C.1 Dataset Statistics and Preprocessing

We use the 5-core Amazon Beauty, Toys & Games, and Sports & Outdoors benchmarks following GRID (Ju et al., 2025), and Yelp following LETTER (Wang et al., 2024a). Table 7 summarizes the processed datasets. For each user, we reserve the final interaction for testing and the penultimate interaction for validation, using the remaining interactions for training.

Interaction histories. Following GRID, Amazon training uses contiguous interaction windows with up to 30 history items, corresponding to 120 SID tokens. For Yelp, we adopt the setting of LETTER, using growing-prefix training examples with up to 20 history items. Within each matched comparison, COIN and its baseline use identical interaction sequences and preprocessing.

Semantic item identifiers. For Amazon, we extract 2,048-dimensional item representations using FLAN-T5-XL (Chung et al., 2024), truncating item text to 128 tokens and averaging the non-padding encoder outputs. RK-Means, R-VQ, and RQ-VAE produce three semantic code levels, followed by a collision-disambiguation digit. For Yelp, we use the precomputed 4,096-dimensional LLaMA item embeddings released with DIGER (Fu et al., 2026). The tokenizers produce four semantic code levels without an additional collision-disambiguation digit. Each codebook contains 256 entries.

Collaborative intent vocabulary. We construct item co-occurrences from training interactions within a window of ten positions and apply rank-64 SVD to the resulting PPMI matrix. Cosine K-means partitions the collaborative embeddings into 512 intent classes. These embeddings define the teacher's target labels; its input item-ID embeddings are learned separately.

C.2 Implementation and Evaluation

Code availability. The code for this work is available at `https://anonymous.4open.science/r/COIN_ICLR_REVIEW-8811/`.

Generator configuration. COIN and its matched baseline inherit their base implementation from GRID (Ju et al., 2025). The T5-style encoder–decoder (Raffel et al., 2020) has four encoder and four decoder layers, while the decoder-only model uses eight causal layers. Both architectures use hidden width 128, six attention heads with head dimension 64, and dropout 0.15. Following GRID, each feed-forward module is an MLP with two hidden layers of width 1,024.

Collaborative teacher. We use a causal Transformer teacher with rotary position embeddings (Su et al., 2024) and 512 intent classes. The teacher has two blocks, hidden width 64, one attention head, and dropout 0.7. It is trained on all training users using Adam (Kingma & Ba, 2014) with learning rate 10^{-3} and batch size 128 for up to 20,000 optimizer steps. Checkpoint selection monitors cluster prediction accuracy on a subset of training users. Each teacher remains frozen during student training and is reused across tokenizers and seeds.

Student optimization. Amazon models use Adam, while Yelp models use AdamW. The settings in Table 8 apply to both architectures within each dataset.

For Amazon, we retain the checkpoint with the highest validation Recall@10 over the complete 6,000-step training trajectory. For Yelp, we also select checkpoints by validation Recall@10, with

Table 8: Student optimization settings for both generator architectures.

Setting	Amazon	Yelp
Learning rate	10^{-3}	5×10^{-4}
Weight decay	10^{-4}	0.01
Learning-rate schedule	Constant	Cosine decay with 1% warmup
Microbatch	Windows from 32 user groups, capped at 512 windows	256 examples
Gradient accumulation	16 microbatches	4 microbatches
Training budget	6,000 optimizer steps	Up to 44,000 optimizer steps (200 nominal epochs)
Validation frequency	Every 100 steps	Every 220 steps

early stopping after 20 validation checks without improvement in NDCG@10. Training uses bfloat16 autocasting, and evaluation uses float32.

Transfer temperature. COIN uses $K = 512$ collaborative intent classes. We select the transfer temperature τ using validation Recall@10. The selected values are shown in Table 9, with each cell reporting encoder–decoder / decoder-only.

Table 9: Selected transfer temperatures τ (encoder–decoder / decoder-only).

Dataset	RK-Means	R-VQ	RQ-VAE
Beauty	0.6 / 0.6	0.5 / 0.9	0.5 / 0.5
Toys	0.8 / 0.6	0.5 / 0.7	0.7 / 0.7
Sports	0.5 / 0.5	0.8 / 0.8	0.7 / 0.6
Yelp	0.7 / 0.6	0.6 / 0.6	0.2 / 0.3

Evaluation. We use beam widths of 10 for Amazon and 20 for Yelp. On Amazon, duplicate, invalid, and previously observed items occupy their original ranks as misses, without promoting lower-ranked candidates. Yelp evaluation applies no history masking. We report Recall and NDCG at cutoffs 5 and 10 as means and SD across three random seeds.

C.3 Mutual Information of Semantic and Collaborative Groups

Figure 2(b) uses a discrete contingency-table calculation on held-out item transitions.

Transitions and empirical estimates. Each test user contributes one transition from the last observed item to the exact next-item identity i^+. Let S and C denote the semantic and collaborative group labels of the last observed item, respectively. Both use the fixed six-group illustrative partitions. All test users receive equal weight, independently of the arrows sampled for display in Figure 2(a). We count occurrences of (i^+, S, C) and divide by the number of users to obtain empirical joint probabilities. All marginal and conditional probabilities are derived from this same table without smoothing. We then compute

$$I(i^+; S) = H(i^+) - H(i^+ \mid S),$$
$$I(i^+; C) = H(i^+) - H(i^+ \mid C),$$
$$I(i^+; C \mid S) = H(i^+ \mid S) - H(i^+ \mid S, C).$$

Matched permutation baselines. For each statistic, we construct a matched baseline using 999 permutations and the same empirical estimator. For the marginal MIs, we shuffle paired (S, C) labels across users while keeping i^+ fixed. This preserves group frequencies and the joint distribution of the two groupings. For conditional MI, we keep (S, i^+) fixed and shuffle C only among users with the same S. This preserves the semantic–target relationship and the semantic–collaborative group counts while disrupting the additional collaborative–target association. The plotted value subtracts

Table 10: Native ranking performance (three-seed mean ± SD, %). Each cell shows the mean above its SD. R and N denote Recall and NDCG; shading marks the better result per matched pair.

Arch.	Tokenizer	Method	Beauty				Toys				Sports				Yelp			
			R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
Encoder–decoder	RK-Means	Baseline	4.33 ± 0.03	6.37 ± 0.15	2.87 ± 0.05	3.53 ± 0.09	4.00 ± 0.16	5.87 ± 0.15	2.65 ± 0.10	3.25 ± 0.10	2.38 ± 0.07	3.52 ± 0.06	1.54 ± 0.04	1.91 ± 0.03	2.34 ± 0.12	3.89 ± 0.13	1.54 ± 0.08	2.04 ± 0.08
		+COIN	5.10 ± 0.12	7.25 ± 0.13	3.51 ± 0.07	4.21 ± 0.07	5.01 ± 0.13	7.32 ± 0.19	3.34 ± 0.13	4.08 ± 0.14	2.73 ± 0.09	4.10 ± 0.10	1.80 ± 0.06	2.25 ± 0.06	3.23 ± 0.03	5.28 ± 0.06	2.09 ± 0.02	2.75 ± 0.03
	R-VQ	Baseline	4.51 ± 0.16	6.63 ± 0.14	2.98 ± 0.14	3.66 ± 0.13	4.03 ± 0.11	5.95 ± 0.01	2.67 ± 0.09	3.29 ± 0.05	2.40 ± 0.14	3.61 ± 0.10	1.57 ± 0.09	1.96 ± 0.08	2.35 ± 0.05	3.88 ± 0.13	1.52 ± 0.05	2.01 ± 0.06
		+COIN	5.12 ± 0.07	7.34 ± 0.08	3.52 ± 0.10	4.24 ± 0.08	5.03 ± 0.15	7.41 ± 0.18	3.39 ± 0.14	4.16 ± 0.13	2.89 ± 0.15	4.19 ± 0.13	1.92 ± 0.09	2.34 ± 0.09	3.28 ± 0.08	5.35 ± 0.14	2.11 ± 0.03	2.78 ± 0.05
	RQ-VAE	Baseline	4.00 ± 0.05	5.81 ± 0.07	2.71 ± 0.06	3.29 ± 0.02	3.65 ± 0.07	5.28 ± 0.11	2.47 ± 0.04	3.00 ± 0.03	2.14 ± 0.05	3.22 ± 0.01	1.40 ± 0.03	1.75 ± 0.01	1.96 ± 0.06	3.34 ± 0.04	1.22 ± 0.04	1.67 ± 0.02
		+COIN	4.95 ± 0.09	6.93 ± 0.17	3.47 ± 0.06	4.11 ± 0.07	4.76 ± 0.12	6.86 ± 0.11	3.24 ± 0.08	3.91 ± 0.07	2.65 ± 0.06	4.02 ± 0.08	1.78 ± 0.03	2.23 ± 0.04	2.90 ± 0.07	4.88 ± 0.07	1.86 ± 0.04	2.49 ± 0.03
Decoder-only	RK-Means	Baseline	4.62 ± 0.04	6.66 ± 0.03	3.17 ± 0.04	3.83 ± 0.03	4.25 ± 0.10	6.04 ± 0.27	2.95 ± 0.04	3.53 ± 0.02	2.45 ± 0.04	3.52 ± 0.06	1.65 ± 0.02	2.00 ± 0.01	2.41 ± 0.10	3.74 ± 0.08	1.59 ± 0.07	2.01 ± 0.06
		+COIN	5.02 ± 0.06	7.08 ± 0.13	3.44 ± 0.08	4.11 ± 0.08	4.86 ± 0.09	7.18 ± 0.11	3.27 ± 0.10	4.02 ± 0.11	2.62 ± 0.06	3.84 ± 0.06	1.76 ± 0.05	2.16 ± 0.04	3.31 ± 0.06	5.25 ± 0.06	2.18 ± 0.06	2.80 ± 0.06
	R-VQ	Baseline	4.63 ± 0.06	6.65 ± 0.13	3.18 ± 0.03	3.83 ± 0.05	4.41 ± 0.03	6.42 ± 0.13	3.02 ± 0.09	3.66 ± 0.05	2.51 ± 0.07	3.69 ± 0.11	1.68 ± 0.06	2.06 ± 0.07	2.43 ± 0.04	3.78 ± 0.04	1.61 ± 0.03	2.04 ± 0.03
		+COIN	5.08 ± 0.19	7.27 ± 0.10	3.51 ± 0.14	4.22 ± 0.09	5.08 ± 0.03	7.51 ± 0.10	3.38 ± 0.04	4.16 ± 0.06	2.69 ± 0.02	3.98 ± 0.04	1.81 ± 0.03	2.23 ± 0.02	3.36 ± 0.06	5.28 ± 0.04	2.21 ± 0.01	2.83 ± 0.02
	RQ-VAE	Baseline	4.23 ± 0.26	5.91 ± 0.17	3.00 ± 0.20	3.54 ± 0.17	3.92 ± 0.13	5.58 ± 0.27	2.77 ± 0.06	3.30 ± 0.10	2.33 ± 0.07	3.37 ± 0.05	1.59 ± 0.04	1.92 ± 0.03	2.04 ± 0.06	3.30 ± 0.07	1.36 ± 0.03	1.77 ± 0.03
		+COIN	4.63 ± 0.04	6.56 ± 0.14	3.27 ± 0.04	3.89 ± 0.04	4.52 ± 0.04	6.56 ± 0.15	3.14 ± 0.02	3.80 ± 0.06	2.59 ± 0.03	3.80 ± 0.02	1.79 ± 0.07	2.18 ± 0.05	2.88 ± 0.10	4.74 ± 0.05	1.89 ± 0.03	2.49 ± 0.02
Relative improvement (%)			8.7–23.8	6.3–19.3	8.5–28.0	7.3–24.9	14.4–30.4	17.0–29.9	10.8–31.2	13.7–30.3	6.9–23.8	7.9–24.8	6.7–27.1	8.0–27.4	37.3–48.0	35.7–46.1	35.7–52.5	34.8–49.1

the corresponding mean permutation estimate from the observed MI:

$$\Delta I = I_{\text{empirical}} - \frac{1}{999}\sum_{b=1}^{999} I_{\text{permuted},b}.$$

We convert both the empirical estimates and their permutation means from bits to nats by multiplying by $\ln 2$.

C.4 Full Native Ranking Results

Table 10 provides the exact Recall and NDCG scores underlying Figure 3, with SD across three seeds. The comparison covers both generator architectures, all three SID tokenizers, and all four datasets, using the matched baselines and native COIN decoding described in Section 4.1.

C.5 Comparison with Broader Baselines

In addition to the matched native-generator comparisons in the main text, we compare COIN with traditional recommenders SASRec, GRU4Rec, and LightGCN (Kang & McAuley, 2018; Hidasi et al., 2015; He et al., 2020); reasoning-enhanced recommenders ReaRec, LARES, and ManCAR (Tang et al., 2026b; Liu et al., 2025c; Yang et al., 2026a); and generative recommenders TIGER, ETEGRec, LETTER, EAGER, and DiscRec (Rajput et al., 2023; Liu et al., 2025b; Wang et al., 2024a;b; Liu et al., 2025a).

We implement the traditional baselines with RecBole (Zhao et al., 2021) and use the official repositories for the reasoning-enhanced and generative baselines. All methods use identical data splits, and all generative baselines and COIN share the same item text embeddings and evaluation protocol. For

Table 11: Broader baseline comparison: Recall@10 (R@10) and NDCG@10 (N@10), mean ± SD (%). Mean ranks use source precision across eight dataset–metric pairs: GRs compares generative recommenders and COIN; All includes every method. Bold and shading mark best results; underlining marks second best. LETTER on Beauty uses two seeds.

Method	Beauty		Toys		Sports		Yelp		Mean rank ↓	
	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10	GRs	All
Traditional recommenders										
SASRec	6.46 ± 0.13	3.58 ± 0.10	7.30 ± 0.01	4.28 ± 0.02	3.46 ± 0.02	1.86 ± 0.02	5.17 ± 0.09	3.31 ± 0.07	–	4.25
GRU4Rec	4.70 ± 0.05	2.43 ± 0.02	4.06 ± 0.04	2.29 ± 0.03	1.88 ± 0.05	0.95 ± 0.03	2.64 ± 0.08	1.47 ± 0.06	–	11.75
LightGCN	4.27 ± 0.12	2.23 ± 0.06	4.37 ± 0.06	2.40 ± 0.02	2.58 ± 0.02	1.33 ± 0.01	**5.94** ± 0.07	3.26 ± 0.02	–	9.00
Reasoning-enhanced recommenders										
ReaRec	6.70 ± 0.13	4.03 ± 0.07	7.31 ± 0.19	**4.62** ± 0.11	2.96 ± 0.10	1.71 ± 0.11	4.32 ± 0.19	2.95 ± 0.05	–	4.50
LARES	6.95 ± 0.37	3.72 ± 0.20	**7.88** ± 0.14	4.34 ± 0.09	3.07 ± 1.73	1.58 ± 0.97	3.83 ± 2.51	2.27 ± 1.53	–	5.75
ManCAR	6.92 ± 0.04	4.02 ± 0.04	7.21 ± 0.01	4.36 ± 0.02	3.99 ± 0.03	2.31 ± 0.03	5.51 ± 0.04	**3.43** ± 0.03	–	2.63
Generative recommenders										
TIGER	5.42 ± 0.06	2.96 ± 0.02	4.91 ± 0.21	2.62 ± 0.11	3.18 ± 0.10	1.70 ± 0.01	3.25 ± 0.08	1.68 ± 0.06	5.50	9.38
ETEGRec	7.04 ± 0.02	3.97 ± 0.02	6.53 ± 0.18	3.57 ± 0.09	3.73 ± 0.12	1.99 ± 0.07	5.11 ± 0.17	2.67 ± 0.11	2.00	4.38
LETTER	6.14 ± 0.11	3.33 ± 0.03	5.40 ± 0.16	2.93 ± 0.08	2.95 ± 0.09	1.61 ± 0.04	4.21 ± 0.05	2.22 ± 0.03	4.75	8.75
EAGER	5.62 ± 0.17	3.48 ± 0.08	5.02 ± 0.05	3.08 ± 0.11	2.99 ± 0.16	1.61 ± 0.07	3.70 ± 0.10	1.91 ± 0.04	4.75	8.75
DiscRec	6.41 ± 0.09	3.50 ± 0.06	5.78 ± 0.12	3.24 ± 0.04	3.40 ± 0.13	1.83 ± 0.07	4.29 ± 0.07	2.27 ± 0.02	3.00	6.50
COIN	**7.34** ± 0.08	**4.24** ± 0.08	7.41 ± 0.18	4.16 ± 0.13	**4.19** ± 0.13	**2.34** ± 0.09	5.35 ± 0.14	2.78 ± 0.05	**1.00**	**2.38**

ReaRec, we use the PRL variant because the ERL variants were empirically unstable in our pipeline. Table 11 reports Recall@10, NDCG@10, and mean rank across the four datasets.

Among the native reasoning GR baselines summarized in Table 1, we found publicly available implementations only for IBA (Yang et al., 2026b) and GPR (Jiang et al., 2026e) at the time of our experiments. Every attempted run of these implementations exceeded our 48-hour time limit without completing, so we could not obtain results for comparison. The remaining methods had no publicly released code.

The COIN row uses native encoder–decoder execution with R-VQ and matches the corresponding results in Figure 3.

C.6 Encoder Mutual-Information Probe

We directly probe whether learning to plan transfers collaborative structure into the generator's representation. The analysis covers all nine combinations of Beauty, Toys, and Sports with RK-Means, R-VQ, and RQ-VAE, using $K = 512$ and the three trained random seeds.

For each real item i, we encode the item alone using its SID and average the encoder outputs over its four SID positions:

$$Z_i = \frac{1}{L} \sum_{\ell=1}^{L} \mathbf{h}_{i,\ell}, \qquad L = 4, \tag{9}$$

where the separator position is excluded. This produces one continuous encoder representation $Z_i \in \mathbb{R}^d$ for each item.

The collaborative target variable is constructed independently of both the baseline and COIN. We first build frozen PPMI-SVD collaborative item embeddings from the training interactions. A cosine K-means codebook with 512 centroids is fitted in this collaborative space, and each item is assigned to its nearest centroid:

$$C_i = \arg\min_k \left(1 - \cos(\mathbf{e}_i, \mathbf{c}_k)\right). \tag{10}$$

The resulting discrete label vector C is fixed and identical when evaluating the baseline and COIN; only Z changes.

We then estimate

$$I(Z; C) \tag{11}$$

using a mixed continuous–discrete k-nearest-neighbor mutual information estimator (Ross, 2014) with $k = 3$ after row-normalizing Z. MI is computed independently for each dataset–tokenizer–seed model and measured in nats. Figure 4(b) reports the mean over the three seeds, with whiskers denoting the cross-seed SD.

Figure 4(b) shows higher estimated MI for COIN across all nine dataset–tokenizer combinations. The items, SIDs, collaborative labels, estimator, and architecture are shared; the comparison reflects the full COIN training objective, including both planning and execution. This item-level probe complements Theorem 1 by measuring association with collaborative regions; the theorem concerns history-dependent teacher-plan information.

C.7 Plan-Conditioned Decoding Probe

We isolate how the predicted plan guides autoregressive SID generation using RK-Means, $K = 512$, all test users, and the three trained seeds for Beauty, Toys, and Sports. For each user, the native generator first emits its top-one intent:

$$r_{\text{pred}} = r_{\arg\max_k p_\theta(r_k|X)}. \tag{12}$$

We compare this condition with a uniformly sampled random intent r_{rand}.

At SID decoding step $c \in \{1, 2, 3, 4\}$, both conditions are teacher-forced with the same ground-truth SID prefix $(y_1, \ldots, y_{c-1})$:

$$p_c^{\text{pred}} = p_\theta(\cdot \mid X, r_{\text{pred}}, y_{<c}), \tag{13}$$

$$p_c^{\text{rand}} = p_\theta(\cdot \mid X, r_{\text{rand}}, y_{<c}). \tag{14}$$

Thus, the leading intent is the only quantity that differs between the two states.

Figure 4(c) measures entropy reduction:

$$\Delta H_c = H(p_c^{\text{rand}}) - H(p_c^{\text{pred}}). \tag{15}$$

A positive value means that the native intent produces a narrower next-digit distribution than a random intent.

Figure 4(d), expressed as a probability difference, measures the corresponding gain in probability mass assigned to the ground-truth digit:

$$\Delta P_c = p_c^{\text{pred}}(y_c) - p_c^{\text{rand}}(y_c). \tag{16}$$

A positive value indicates that the predicted plan directs more probability toward the correct continuation.

Both quantities are computed within each seed and decoding position, then averaged across seeds; error bars denote the cross-seed SD. Using the same true SID prefix removes the effect of earlier decoding errors and isolates the leading plan's contribution at each step.

Figure 4(c–d) shows that predicted plans reduce uncertainty and increase true-digit probability most strongly at early SID positions. Their influence diminishes as the growing prefix constrains later choices.

D Learning collaborative plans and executing them

COIN learns a collaborative plan from the interaction history and then commits to one plan before generating the next item's semantic ID (SID). The analysis follows these two stages. We first separate planning and execution in the training objective. We then explain how planning supervision can add task-aligned structure to the student's representation. Finally, we decompose the execution error to examine reliable plan selection and overlapping item predictions across plans.

D.1 DECOMPOSING THE STUDENT OBJECTIVE

Let h denote an item history and i^+ the next item. A fixed, injective tokenizer Q maps them to the history SID sequence $X = Q(h)$ and the next-item SID $Y = Q(i^+)$. Write $\pi_h = P(Y \mid h)$ for the true next-item distribution.

A frozen teacher maps each history to a distribution over K plans:

$$q_\tau(\cdot \mid h) = \mathrm{softmax}(z_\phi(h)/\tau), \qquad \tau > 0.$$

Whenever a training example is presented, COIN samples a fresh plan $R \sim q_\tau(\cdot \mid h)$ and retains the same observed next-item target Y. The resulting training law is $P(h, Y)q_\tau(R \mid h)$, so $R \perp Y \mid h$. Thus different sampled plans for a history are all supervised using the same next-item distribution.

The student represents the history by $Z = f_\theta(X)$. Its planning head $a_\theta(k \mid Z)$ is normalized over the K plan tokens. After a plan is supplied, its executor predicts the autoregressive SID distribution $p_{\theta,k}(Y \mid Z)$. Every branch has access to Z and predicts over the full SID output space. The population training objective is

$$\mathcal{L}_\tau(\theta) = \mathbb{E}_{h,Y} \sum_k q_\tau(k \mid h) \left[\underbrace{-\log a_\theta(k \mid Z)}_{\text{predict the plan}} + \underbrace{-\log p_{\theta,k}(Y \mid Z)}_{\text{execute the plan}} \right]. \tag{17}$$

All expectations for the original objective use the law $P(h, Y)q_\tau(R \mid h)$, conditional on a fitted teacher and fixed item maps. Histories and the catalog are finite, or the stated expectations are finite. The SID output space $\mathcal{Y}$ is finite and includes invalid strings; π_h assigns probability only to valid item codes. Each $p_{\theta,k}$ is a normalized, positive distribution on $\mathcal{Y}$, and a_θ is normalized over the K plans. Logarithms are natural. Optimal readouts on the boundary are understood as infima or limits. At fixed student parameters, Z is a deterministic function of h.

Separating the entropies of the supervised targets gives the exact decomposition

$$\boxed{\mathcal{L}_\tau = \underbrace{\mathcal{H}_\tau}_{\text{fixed target entropy}} + \underbrace{\mathcal{P}_\tau}_{\text{planning error}} + \underbrace{\mathcal{E}_\tau}_{\text{execution error}}}. \tag{18}$$

Here $\mathcal{H}_\tau = H(Y \mid h) + \mathbb{E}_h H(q_\tau(\cdot \mid h))$ is fixed with respect to the student parameters. The remaining two terms describe the demands placed on the student by the two prediction stages.

The planning error is

$$\mathcal{P}_\tau = \mathbb{E}_h \, \mathrm{KL}(q_\tau(\cdot \mid h) \,\|\, a_\theta(\cdot \mid Z)). \tag{19}$$

It measures how faithfully the student predicts the teacher's forecast from its own history representation. Section D.2 separates the information preserved by that representation from the planning head's ability to read it out.

The execution error is

$$\mathcal{E}_\tau = \mathbb{E}_h \sum_k q_\tau(k \mid h) \, \mathrm{KL}(\pi_h \,\|\, p_{\theta,k}(\cdot \mid Z)). \tag{20}$$

It measures the next-item prediction error averaged over the plans used during training. A branch contributes according to its sampling weight at that history. Section D.3 decomposes this term further to distinguish collective prediction quality from disagreement across plans. Both errors in Equation (18) are nonnegative.

Write $q_k = q_\tau(k \mid h)$ and $p_k = p_{\theta,k}(\cdot \mid Z)$ at a fixed history. Separating the target entropies in the two cross-entropies gives

$$\mathbb{E}[-\log a_\theta(R \mid Z)] = \mathbb{E}_h H(q_\tau(\cdot \mid h)) + \mathcal{P}_\tau, \tag{21}$$

$$\begin{aligned} \mathbb{E}[-\log p_{\theta,R}(Y \mid Z)] &= H(Y \mid h) + \mathbb{E}_h \sum_k q_k \, \mathrm{KL}(\pi_h \| p_k) \\ &= H(Y \mid h) + \mathcal{E}_\tau. \end{aligned} \tag{22}$$

The execution identity uses $R \perp Y \mid h$, so each sampled branch is averaged against π_h. Adding the two identities proves Equation (18).

Interpretation. The objective asks the student to learn a plan and to predict the next item when executing that plan. Reducing $\mathcal{P}_\tau$ encourages the representation and planning head to express the teacher's forecast. Reducing $\mathcal{E}_\tau$ trains the sampled execution branches toward the same next-item behavior. This separation provides the structure for the remaining analysis: what the teacher helps the student represent, and how that representation supports reliable execution.

D.2 Learning collaborative plans

We first separate the representation and readout roles of $\mathcal{P}_\tau$. We then use a linear feature model to explain when preserving teacher-forecast distinctions is useful, with task alignment and heterogeneity as the two central properties.

D.2.1 Planning error separates missing information from accessibility

A planning head only observes Z. If several histories are represented by the same state, the head must make the same prediction for all of them, even when their teacher forecasts differ. The best forecast recoverable from that state is therefore the conditional average

$$q_{\tau,Z}(k \mid Z) := \mathbb{E}[q_\tau(k \mid h) \mid Z] = P_\tau(R = k \mid Z). \tag{23}$$

Here the subscript τ on probabilities and information quantities refers to the law obtained by sampling R from q_τ.

We can now separate two sources of planning error: the difference between a history's forecast and this conditional average, and the difference between the average and the head's actual prediction. To do so, insert $q_{\tau,Z}$ into the log ratio in Equation (19):

$$\log \frac{q_\tau(k \mid h)}{a_\theta(k \mid Z)} = \log \frac{q_\tau(k \mid h)}{q_{\tau,Z}(k \mid Z)} + \log \frac{q_{\tau,Z}(k \mid Z)}{a_\theta(k \mid Z)}.$$

In the second term, the log ratio depends on the history only through Z. Averaging its weight $q_\tau(k \mid h)$ conditional on Z therefore replaces that weight by $q_{\tau,Z}(k \mid Z)$. The planning error becomes

$$\mathcal{P}_\tau = \underbrace{\mathbb{E}_h \operatorname{KL}(q_\tau \| q_{\tau,Z})}_{\mathcal{R}_\tau\text{: representation floor}} + \underbrace{\mathbb{E}_Z \operatorname{KL}(q_{\tau,Z} \| a_\theta)}_{\mathcal{A}_\tau\text{: head error}}. \tag{24}$$

Arguments are suppressed in this identity: q_τ is evaluated at h, and $q_{\tau,Z}$ and a_θ at Z.

Since $P_\tau(R \mid h, Z) = q_\tau(\cdot \mid h)$ and $P_\tau(R \mid Z) = q_{\tau,Z}(\cdot \mid Z)$, the first term is $I_\tau(R; h \mid Z)$.

The representation floor $\mathcal{R}_\tau$ measures forecast distinctions absent from Z. An unrestricted head can predict $q_{\tau,Z}$ exactly, but it cannot distinguish histories that Z has merged. Its best attainable planning error is therefore $\mathcal{R}_\tau$. Equivalently, $\mathcal{R}_\tau = I_\tau(R; h \mid Z)$: it is the information about the teacher's sampled plan that remains in the history after the state is known. Since Z is a deterministic function of h, the chain rule gives $I_\tau(R; h) = I_\tau(R; Z) + \mathcal{R}_\tau$. Combining this identity with $\mathcal{A}_\tau \geq 0$ yields $I_\tau(R; Z) \geq I_\tau(R; h) - \mathcal{P}_\tau$, an information-retention consequence of Theorem 1.

The head error $\mathcal{A}_\tau$ concerns accessibility. It measures how closely the actual planning head predicts the conditional forecast already supported by Z. This error can remain positive when useful information is retained but expressed in a form that is difficult for the head to use. It can be reduced by improving the head, by reorganizing the representation to make the forecast easier to predict, or by learning both together.

At $\mathcal{P}_\tau = 0$, nonnegativity and the equality condition for KL divergence give $q_\tau(\cdot \mid h) = a_\theta(\cdot \mid Z)$ almost surely. Histories represented by the same Z must therefore have the same teacher forecast, including the probabilities assigned to alternatives. Replacing the forecast with a point mass on its top-one plan imposes only agreement on that plan. This proves the distributional-distillation conclusion in Theorem 1.

Interpretation. Optimizing $\mathcal{P}_\tau = \mathcal{R}_\tau + \mathcal{A}_\tau$ encourages the state to retain forecast distinctions and make them accessible to the head. Different targets therefore reward different representations: uniform targets require no history-specific distinctions, task-aligned targets can reward useful behavioral structure, and irrelevant targets can reward nuisance variation. Temperature changes these incentives. Very small τ can erase distinctions between histories sharing a unique winning region; very large

τ makes the target uniform and $\mathcal{R}_\tau \to 0$. Intermediate temperatures can preserve informative alternatives while distributing execution support across plans.

For fixed finite logits, q_τ converges to the uniform law as $\tau \to \infty$, hence to a history-independent law. Since $0 \le \mathcal{R}_\tau \le I_\tau(R; h) = H_\tau(R) - H_\tau(R \mid h)$, continuity of finite-alphabet entropy gives $\mathcal{R}_\tau \to 0$. As $\tau \to 0$, a unique maximizer receives probability one; histories sharing it lose distinctions carried only by alternative probabilities. With tied maxima the limit is uniform on the maximizers.

D.2.2 Task-aligned heterogeneity in ID-embedding teacher supervision

The decomposition $\mathcal{P}_\tau = \mathcal{R}_\tau + \mathcal{A}_\tau$ explains the incentive to preserve and expose teacher-forecast distinctions. Their value for recommendation depends on which distinctions the teacher supplies. We illustrate this connection with a linear model of complementary feature directions.

Let $s \in \mathbb{R}^{d_s}$ denote features already captured by a common semantic baseline $Z_0 = s$, and let $c \in \mathbb{R}^{d_c}$ denote additional behavioral features that this representation underrepresents. Assume s and c are centered, independent, and have identity covariance. The distinction concerns the representation learned under finite capacity and training. Assume that the trainable student encoder has enough capacity to preserve s and learn an additional direction in c from its SID input.

Represent the recommendation task by a scalar prediction surrogate

$$y = \underbrace{a^\top s}_{\text{signal already represented}} + \underbrace{b^\top c}_{\text{remaining task signal}} + \epsilon, \qquad \mathbb{E}[\epsilon \mid s, c] = 0, \quad \mathbb{E}[\epsilon^2] = \sigma^2. \tag{25}$$

Here b identifies the additional behavioral directions that matter for prediction. Lowercase y is the scalar surrogate used in this illustration; Y remains the next-item SID in the original objective.

For teacher t, write a linear surrogate for one direction in its planning forecast as

$$T_t = \underbrace{u_t^\top s}_{\text{overlap with the baseline}} + \underbrace{v_t^\top c}_{\text{additional teacher structure}}. \tag{26}$$

The actual supervision remains the categorical forecast $q_\tau^{(t)}$ in Equation (19). The scalar T_t isolates the feature geometry of that supervision. It makes the two properties explicit. **Heterogeneity** means $v_t \neq 0$: the teacher expresses a direction beyond the baseline. **Task alignment** means $b^\top v_t \neq 0$: this additional direction explains some of the remaining task signal.

How supervision makes the additional direction usable. In the linear illustration, an available student coordinate is trained to predict T_t with a squared auxiliary loss. At exact fit it reproduces T_t. Because the student also retains s, a linear readout can subtract $u_t^\top s$ and recover $v_t^\top c$. The additional feature is computed by the trained student from its own input. This idealized fitting step illustrates the representation effect encouraged by $\mathcal{P}_\tau$ in the original objective.

Let M_0 be the minimum mean squared prediction error using s, and let M_t be the corresponding error after the student learns this additional direction. Then

$$M_0 = \sigma^2 + \|b\|^2, \qquad \boxed{\Delta_t := M_0 - M_t = \frac{(b^\top v_t)^2}{\|v_t\|^2}} \quad (v_t \neq 0). \tag{27}$$

Set $\Delta_t = 0$ when $v_t = 0$. The result follows by projecting the missing signal $b^\top c$ onto the learned feature $v_t^\top c$. The optimal coefficient on that feature is $(b^\top v_t)/\|v_t\|^2$, so the squared size of the recovered projection is precisely Δ_t.

Under the stated assumptions, the best linear predictor based on s is $a^\top s$. Independence, identity covariance, and the condition $\mathbb{E}[\epsilon \mid s, c] = 0$ imply

$$M_0 = \mathbb{E}[(b^\top c + \epsilon)^2] = \|b\|^2 + \sigma^2.$$

For the idealized auxiliary fitting step, let the available student coordinate be $\widehat{T} = w_s^\top s + w_c^\top c$. Its squared fitting loss is

$$\mathbb{E}[(T_t - \widehat{T})^2] = \|u_t - w_s\|^2 + \|v_t - w_c\|^2. \tag{28}$$

It is minimized at $w_s = u_t$ and $w_c = v_t$. Retaining s makes the fitted coordinate equivalent, for a linear task readout, to the additional feature $v_t^\top c$. This specifies the fitting and capacity assumptions used in the illustration.

For a scalar task coefficient α on that feature, the prediction error is

$$\begin{aligned}\mathbb{E}[(y - a^\top s - \alpha v_t^\top c)^2] &= \sigma^2 + \|b - \alpha v_t\|^2 \\ &= \sigma^2 + \|b\|^2 - 2\alpha b^\top v_t + \alpha^2 \|v_t\|^2. \end{aligned} \tag{29}$$

When $v_t \neq 0$, differentiation gives $\alpha^* = (b^\top v_t)/\|v_t\|^2$. Substituting yields

$$M_t = \sigma^2 + \|b\|^2 - \frac{(b^\top v_t)^2}{\|v_t\|^2}, \qquad \Delta_t = M_0 - M_t = \frac{(b^\top v_t)^2}{\|v_t\|^2}.$$

When $v_t = 0$, the additional coordinate is redundant and $M_t = M_0$. For nonzero b and v_t, the inner-product identity gives $\Delta_t = \|b\|^2 \cos^2 \angle(b, v_t)$. This proves the gain formula and its heterogeneity and task-alignment interpretations. The matched-entropy teacher comparison in Appendix B.1 motivates the hypothesis in Equation (30).

A redundant teacher direction gives no additional gain. A heterogeneous direction orthogonal to b also gives no gain. A heterogeneous direction aligned with b supplies useful prediction structure. Thus the model identifies the value of the direction acquired through auxiliary supervision, conditional on the stated representation and fitting assumptions.

Why the ID-embedding teacher can supply a more useful direction. Both teacher variants predict the same next-item CF regions. The ID-embedding teacher learns an item-ID embedding table jointly with its sequence predictor, allowing interaction evidence to shape the input features themselves. Item-specific behavioral roles can therefore be represented directly in that table before the sequence model combines them. With fixed semantic inputs, the teacher learns these roles through its sequence model operating on an unchanged input geometry. The ID-embedding teacher provides a route to collaborative directions that are less readily expressed by the student's current semantic representation.

In the linear model, this design motivates the hypothesis

$$\Delta_{\mathrm{CF}} > \Delta_{\mathrm{sem}}, \qquad \Delta_t = \frac{(b^\top v_t)^2}{\|v_t\|^2} \text{ for } v_t \neq 0, \tag{30}$$

with the zero-direction convention from Equation (27). The comparison concerns task-aligned structure beyond the baseline. A semantic teacher can also supply such structure; the input design motivates which teacher is more likely to expose a useful additional direction under the available capacity and training.

Relation to the empirical results. The student trained with the ID-embedding teacher outperforms its semantic-input counterpart at every tested matched transfer-entropy setting. The observed ordering also persists in comparisons where the semantic teacher has comparable or slightly higher standalone teacher accuracy. These results are consistent with the proposed mechanism: standalone accuracy summarizes the teacher's predictions, whereas transfer value depends on the useful structure added beyond the student. In Equation (26), the shared component $u_t^\top s$ can contribute to the teacher's own predictions while the additional component $v_t^\top c$ determines the representational gain in Equation (27). Matching transfer entropy controls overall forecast concentration while allowing these predictive directions to differ. The empirical ranking results support the teacher choice; the linear model identifies a mechanism whose task-aligned complementarity can be examined through held-out conditional prediction probes.

Interpretation. The ID-embedding teacher is useful when it exposes a behavioral distinction that matters for recommendation and that the student has not yet learned to represent. Heterogeneity supplies an additional direction; task alignment determines its value. Planning supervision encourages the student to retain that direction and make it accessible to a readout. This connects the representation incentives in $\mathcal{R}_\tau + \mathcal{A}_\tau$ to the observed benefit of supervision from the ID-embedding teacher.

D.3 Executing a supported plan

We now fix the trained student, teacher, and temperature, and return to the execution error $\mathcal{E}_\tau$ in Equation (18). The results below compare prediction quality and agreement across execution branches

of this fixed model. Since each sampled plan is paired with the same next-item target, its branches are trained toward the same conditional law. To separate prediction quality from agreement, define their normalized geometric pool

$$\bar{p}_\tau(Y \mid h) = \frac{\prod_k p_k(Y)^{q_k}}{\sum_{Y'} \prod_k p_k(Y')^{q_k}}. \tag{31}$$

The pool summarizes the branches in the analysis. Their weighted prediction error has the exact decomposition

$$\begin{aligned} \varepsilon_\tau(h) := \sum_k q_k \,\mathrm{KL}(\pi_h \| p_k) = \underbrace{\mathrm{KL}(\pi_h \| \bar{p}_\tau)}_{f_\tau(h)\text{: pooled prediction error}} + \underbrace{\sum_k q_k \,\mathrm{KL}(\bar{p}_\tau \| p_k)}_{c_\tau(h)\text{: branch disagreement}}, \\ \mathcal{E}_\tau = \mathbb{E}_h \varepsilon_\tau(h) = \underbrace{\mathbb{E}_h f_\tau(h)}_{\mathcal{F}_\tau} + \underbrace{\mathbb{E}_h c_\tau(h)}_{\mathcal{C}_\tau}. \end{aligned} \tag{32}$$

Thus $\mathcal{F}_\tau$ measures how accurately the branches collectively predict next-item behavior, and $\mathcal{C}_\tau$ measures their departures from the pool. Both terms are nonnegative. The logarithmic-pool identity underlying this split is derived below (Heskes, 1997).

Suppress h and define $S = \sum_Y \prod_k p_k(Y)^{q_k}$ and $\bar{p}_\tau(Y) = S^{-1} \prod_k p_k(Y)^{q_k}$. Weighted arithmetic–geometric mean gives $0 < S \le \sum_Y \sum_k q_k p_k(Y) = 1$. Since $\log \bar{p}_\tau = \sum_k q_k \log p_k - \log S$,

$$c_\tau(h) = \sum_k q_k \,\mathrm{KL}(\bar{p}_\tau \| p_k) = -\log S \ge 0, \tag{33}$$

$$\begin{aligned} \varepsilon_\tau(h) &= \sum_k q_k \,\mathrm{KL}(\pi_h \| p_k) = \mathrm{KL}(\pi_h \| \bar{p}_\tau) - \log S \\ &= f_\tau(h) + c_\tau(h). \end{aligned} \tag{34}$$

Averaging proves $\mathcal{E}_\tau = \mathcal{F}_\tau + \mathcal{C}_\tau$ in Equation (32). Also $c_\tau(h) = 0$ exactly when all positively weighted branches coincide, because every weighted KL term in its definition is nonnegative. Nonnegativity of the original branch errors gives $q_k \,\mathrm{KL}(\pi_h \| p_k) \le \varepsilon_\tau(h)$. Since $R \perp Y \mid h$, an unrestricted executor with full history access minimizes each positively weighted cross-entropy at $p_k = \pi_h$.

With unrestricted readouts and full history information, the common optimum is $p_k = \pi_h$ for every positively supported branch. In a finite model, a plan prefix can organize conditional computation, while shared item supervision encourages the supported branches to fit the same behavior. We first connect their prediction error to native plan selection, then examine the overlap that agreement can produce.

D.3.1 Training support and native selection determine branch reliability

Nonnegativity of the branch errors gives

$$\mathrm{KL}(\pi_h \| p_k) \le \frac{\varepsilon_\tau(h)}{q_k} = \frac{f_\tau(h) + c_\tau(h)}{q_\tau(k \mid h)}, \qquad q_k > 0. \tag{35}$$

The sampling weight q_k is the branch's training support at that history. At a fixed combined execution error, a frequently sampled branch is controlled more tightly.

Let $u_B(\mathsf{L}, Y) \in [0,1]$ be single-target Recall@B or NDCG@B for an ordered list of distinct eligible items. Write $U_h(\mathsf{L}) = \mathbb{E}_{\pi_h} u_B(\mathsf{L}, Y)$ and $U_h^* = \max_{\mathsf{L}} U_h(\mathsf{L})$. For the list $\widehat{\mathsf{L}}_k$ decoded under plan k, define the search shortfall under its own branch as

$$\xi_k(h) = \max_{\mathsf{L}} \mathbb{E}_{p_k} u_B(\mathsf{L}, Y) - \mathbb{E}_{p_k} u_B(\widehat{\mathsf{L}}_k, Y) \ge 0. \tag{36}$$

The branch prediction error then yields the ranking bound

$$U_h^* - U_h(\widehat{\mathsf{L}}_k) \le \min\left\{1, \sqrt{\frac{2\varepsilon_\tau(h)}{q_\tau(k \mid h)}} + \xi_k(h)\right\}. \tag{37}$$

Exact conditional top-B item selection makes $\xi_k = 0$; for NDCG, the items are ordered by probability. This separates branch accuracy, training support, and the shortfall introduced by finite search.

For ranking, use a common set of ordered B-slot lists of distinct eligible items, allowing vacancies after invalid or duplicate outputs are removed. The utility is zero for invalid strings, vacancies, and ineligible targets. For single-target NDCG, an item at rank j earns $1/\log_2(j+1)$; Recall earns one for inclusion. Both utilities lie in $[0,1]$. For any such list L,

$$\left|\mathbb{E}_{\pi_h} u_B(\mathsf{L}, Y) - \mathbb{E}_{p_k} u_B(\mathsf{L}, Y)\right| \le \mathrm{TV}(\pi_h, p_k), \qquad \mathrm{TV}(p, v) = \tfrac{1}{2}\sum_Y |p(Y) - v(Y)|.$$

Let L^* maximize true utility, and let U_{p_k} denote utility under p_k. Adding and subtracting branch utilities yields

$$\begin{aligned} U_h(\mathsf{L}^*) - U_h(\widehat{\mathsf{L}}_k) &= [U_h(\mathsf{L}^*) - U_{p_k}(\mathsf{L}^*)] + [U_{p_k}(\mathsf{L}^*) - U_{p_k}(\widehat{\mathsf{L}}_k)] \\ &\quad + [U_{p_k}(\widehat{\mathsf{L}}_k) - U_h(\widehat{\mathsf{L}}_k)] \\ &\le 2\,\mathrm{TV}(\pi_h, p_k) + \xi_k(h) \\ &\le \sqrt{2\,\mathrm{KL}(\pi_h \| p_k)} + \xi_k(h) \le \sqrt{2\varepsilon_\tau(h)/q_k} + \xi_k(h). \end{aligned} \tag{38}$$

Pinsker's inequality gives the second inequality. Regret is at most one, proving Equation (37). Selecting the B highest-probability eligible items maximizes Recall; assigning them to ranks in decreasing probability maximizes NDCG by a pairwise exchange argument. These choices give $\xi_k = 0$.

The remaining question is whether the student selects a branch with adequate support. Teacher-assisted decoding (TA) chooses the teacher mode, whose support is $q_{\max}(h) = \max_k q_\tau(k \mid h)$. Native decoding (ND) chooses $\widehat{k}_{\mathrm{ND}} = \arg\max_k a_\theta(k \mid Z)$. Define $e_{\mathrm{plan}}(h) = \mathrm{KL}(q_\tau(\cdot \mid h) \| a_\theta(\cdot \mid Z))$. The planning error controls native selection through

$$\begin{aligned} \mathbb{E}_h e_{\mathrm{plan}}(h) &= \mathcal{P}_\tau = \mathcal{R}_\tau + \mathcal{A}_\tau, \\ q_\tau(\widehat{k}_{\mathrm{ND}} \mid h) &\ge q_{\max}(h) - \sqrt{2 e_{\mathrm{plan}}(h)}. \end{aligned} \tag{39}$$

When $\gamma(h) := q_{\max}(h) - \sqrt{2e_{\mathrm{plan}}(h)} > 0$, substituting this support guarantee into Equation (37) gives

$$U_h^* - U_h(\widehat{\mathsf{L}}_{\widehat{k}_{\mathrm{ND}}}) \le \min\left\{1, \sqrt{\frac{2[f_\tau(h) + c_\tau(h)]}{\gamma(h)}} + \xi_{\widehat{k}_{\mathrm{ND}}}(h)\right\}. \tag{40}$$

For native selection, fix h, abbreviate $a = a_\theta(\cdot \mid Z)$, and put $t = \arg\max_k q_k$, $s = \arg\max_k a_k$, and $d = \mathrm{TV}(q, a)$. With fixed tie-breaking, $a_s \ge a_t$, so

$$\begin{aligned} q_t - q_s &= (q_t - a_t) + (a_t - a_s) + (a_s - q_s) \\ &\le |q_t - a_t| + |a_s - q_s| \le 2d \le \sqrt{2\,\mathrm{KL}(q \| a)}. \end{aligned} \tag{41}$$

Every coordinate difference is at most d. This proves the support inequality in Equation (39); its expectation identity is the definition of $\mathcal{P}_\tau$ together with Equation (24). Substituting the positive lower bound $\gamma(h)$ for the selected branch's support in Equation (37) proves Equation (40).

The two errors from the original objective now have complementary roles: $\mathcal{E}_\tau = \mathcal{F}_\tau + \mathcal{C}_\tau$ controls branch prediction, while $\mathcal{P}_\tau = \mathcal{R}_\tau + \mathcal{A}_\tau$ controls the choice of a supported branch. These are history-specific bounds, with average losses summarizing their underlying errors.

Temperature changes both the forecast learned by the planning head and the support received by execution branches. Increasing τ makes sampling more even and reduces the teacher mode's weight. It broadens relative coverage while allocating less training to the leading plan. The useful balance retains informative planning distinctions and supports the plans used at inference; branch prediction error also changes with training.

For logits $z_i > z_j$, $q_\tau(i)/q_\tau(j) = \exp((z_i - z_j)/\tau)$ decreases toward one as τ increases. For any maximizing index t,

$$q_{\max}(\tau) = \left[\sum_j \exp\left((z_j - z_t)/\tau\right)\right]^{-1}$$

is nonincreasing in τ. These statements establish more even relative sampling and lower mode concentration. Individual non-mode probabilities and learned prediction errors can vary nonmonotonically with temperature.

Interpretation. Reliable native execution needs an accurate branch and a head that selects a sufficiently trained branch. The execution loss supplies the first part, and faithful prediction of the teacher's forecast supplies the second. Once the native head identifies such a plan, the decoder can spend its item-search budget within that branch. Temperature influences this connection by shaping both the forecast distinctions the student learns and the distribution of training across plans.

D.3.2 Agreement creates overlapping item predictions across plans

The disagreement term $\mathcal{C}_\tau$ also explains why searching several plans can repeat item predictions. At a fixed history, define the probability overlap of two distinct supported branches as

$$\Omega_{k\ell}(h) := \sum_Y \min\{p_k(Y), p_\ell(Y)\} = 1 - \mathrm{TV}(p_k, p_\ell). \tag{42}$$

It is the mass shared by their two predictive distributions, ranging from zero for disjoint support to one for identical distributions.

For normalized distributions, $\min\{a, b\} = (a + b - |a - b|)/2$ gives

$$\sum_Y \min\{p_k(Y), p_\ell(Y)\} = 1 - \tfrac{1}{2}\sum_Y |p_k(Y) - p_\ell(Y)| = 1 - \mathrm{TV}(p_k, p_\ell).$$

This proves Equation (42).

Write $d_k = \mathrm{KL}(\bar{p}_\tau \| p_k)$. Since $c_\tau(h) = \sum_j q_j d_j$, we have $d_k \le c_\tau(h)/q_k$. The triangle inequality and Pinsker's inequality give the short derivation

$$\begin{aligned} 1 - \Omega_{k\ell}(h) &= \mathrm{TV}(p_k, p_\ell) \\ &\le \mathrm{TV}(p_k, \bar{p}_\tau) + \mathrm{TV}(\bar{p}_\tau, p_\ell) \\ &\le \sqrt{d_k/2} + \sqrt{d_\ell/2} \\ &\le \sqrt{\frac{c_\tau(h)}{2q_k}} + \sqrt{\frac{c_\tau(h)}{2q_\ell}}. \end{aligned} \tag{43}$$

Thus a small local disagreement makes well-supported branches overlap strongly in probability. This is a direct consequence of the execution decomposition: agreement brings their item distributions toward the same pool.

In particular, when $q_k, q_\ell \ge \alpha > 0$, Equation (43) gives $\mathrm{TV}(p_k, p_\ell) \le 2\sqrt{c_\tau(h)/(2\alpha)} = \sqrt{2c_\tau(h)/\alpha}$. This is the consistency conclusion of Theorem 2.

The identical-branch case makes the effect on candidate lists explicit. If $c_\tau(h) = 0$, every supported branch equals $\bar{p}_\tau$. With the same eligibility rules and tie-breaking, their exact top-b item sets are the same. For m supported plans with top-b sets $\mathsf{S}_{k,b}$,

$$\left|\bigcup_{k=1}^m \mathsf{S}_{k,b}\right| = b, \qquad \sum_{k=1}^m |\mathsf{S}_{k,b}| = mb. \tag{44}$$

The mb plan–item paths therefore contain only b distinct items, with $(m-1)b$ repeated occurrences. Equation (43) describes the continuous approach to this identical-distribution case; the realized candidate overlap additionally depends on item ranks and the search rule.

For a general search retaining B distinct valid plan–item paths, let m_Y be the number of retained paths ending in item Y. Counting their distinct outputs gives the exact identity

$$\underbrace{\sum_Y \mathbf{1}\{m_Y > 0\}}_{\text{distinct items}} = B - \underbrace{\sum_Y (m_Y - 1)_+}_{\text{duplicate path slots}}, \qquad (x)_+ := \max\{x, 0\}. \tag{45}$$

With deduplication after path selection and no refill, each repeated occurrence consumes a slot that could otherwise hold another item. Committing to a single plan removes this source of cross-plan repetition; exact conditional top-B decoding yields B distinct items when that many are eligible. The branch-reliability analysis in Section D.3.1 explains the accompanying requirement that the selected plan be accurate and sufficiently supported. Item aggregation or refilling duplicate slots changes the budget accounting and can recover distinct candidates during multi-plan search.

For a retained set of B distinct valid plan–item paths, $\sum_Y m_Y = B$. For every nonnegative integer m_Y,

$$m_Y = \mathbf{1}\{m_Y > 0\} + (m_Y - 1)_+.$$

Summing proves Equation (45). The count concerns endpoint duplication among retained paths. Candidate aggregation or refill changes which paths or distinct items receive the finite budget.

Interpretation. Shared item supervision can make different plans lead to much the same recommendation distribution. Exploring more plans can then revisit items already reached under another plan. Commitment is useful when a reliable plan is available and the search budget is spent on paths that are only deduplicated afterward: it directs that budget toward distinct item continuations. The overlap derivation explains the source of this redundancy, and the counting identity makes its cost explicit.