HIERARCHICAL CONDITIONAL TRANSITION MEMORY FOR GENERATIVE RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative recommendation (GR) with Semantic-ID (SID) represents each item as a short sequence of semantic tokens. These tokens introduce dual-role dependencies: fine-grained token transitions capture reusable structures shared across instances, whereas next-item selection demands dynamic contextual inference over user history. Standard dense Transformers entangle both roles within uniform computations, lacking an explicit retrieval pathway for recurrent structural patterns. Our empirical investigation reveals that existing GR models fail to fully exploit such reusable transitions, hindering their representational capacity. To address this, we propose Hierarchical Conditional Transition Memory (HCTM), which decouples generation into retrieval-augmented transition lookup and context-dependent preference inference. HCTM integrates an encoder-side Item-Internal Token Memory with a decoder-side Prefix-Conditioned Transition Memory, fusing retrieved memories into hidden states via context-conditioned residual gating. Extensive experiments across three benchmark datasets demonstrate that HCTM consistently outperforms competitive baselines. In-depth analyses further confirm that the performance gains stem directly from selective transition retrieval, while the memory substantially accelerates training convergence. The anonymized code repository is available at `https://anonymous.4open.science/r/HCTM-C3B5/`.

1 INTRODUCTION

Generative recommendation (GR) formulates next-item prediction as autoregressive sequence generation (Geng et al., 2022; Cui et al., 2022; Rajput et al., 2023). Instead of relying on atomic item identifiers, SID-based approaches represent each item as a compact sequence of discrete tokens encoding semantic features (Rajput et al., 2023; Ju et al., 2025). This representation supports compositional generalization across items (Singh et al., 2024; Wang et al., 2024). Consequently, modeling SID sequences requires capturing hierarchical dependencies, both internally within individual items and sequentially across historical interactions.

Dependencies in SID-based recommendation serve fundamentally distinct roles during generation. On one hand, structural regularities recur consistently across users and sequences, encompassing both intra-item token compositions and inter-item transition patterns. Because these regularities are invariant, they can be shared across prediction instances. On the other hand, the next item cannot be determined by static structures alone. Conditioned on the user history and current decoding context, the model must dynamically disambiguate among plausible item candidates and weight contextual signals accordingly (Kang & McAuley, 2018; Sun et al., 2019). Consequently, an effective generative recommender necessitates both structure-aligned retrieval and context-dependent inference.

Transformers (Vaswani et al., 2017) have been widely adopted in generative recommendation due to their exceptional capacity for contextual computation. However, standard Transformer architectures do not expose an explicit lookup primitive for reusable patterns (Cheng et al., 2026). In generative recommendation, this architectural constraint entangles reusable structural regularities and dynamic, context-dependent preference signals within identical dense computations. Consequently, the model is burdened with reconstructing invariant structures from scratch while simultaneously inferring which candidate aligns with historical signals and the current generation state. These two processes follow fundamentally divergent paradigms. Structural reuse relies on identifying previ-

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.

ously observed SID structures and reliably retrieving their representations across instances. Conversely, contextual inference requires adaptation to personalized user states and evolving generation contexts. Conflating both operations within shared representations forces structural memorization and contextual inference to compete for the same representational capacity.

Recently, memory-augmented sequence modeling (Cheng et al., 2026; Zhou et al., 2026; Zheng et al., 2026) has emerged as a compelling paradigm that augments Transformer architectures with conditional memory, introducing an explicit lookup pathway for recurrent structural patterns. By conditioning on token prefixes as addressing keys, it decouples pattern retrieval from dense computation, allowing models to reuse previously observed structures with high efficiency. Despite its effectiveness in language modeling, directly adapting such conditional memory to Semantic-ID-based recommendation faces a fundamental challenge in structural granularity and context conditioning. While standard conditional memory treats sequences as flat token n-grams, Semantic-ID sequences inherently exhibit a hierarchical dichotomy: tightly-coupled intra-item compositions versus open-ended, user-dependent inter-item transitions. Crucially, observed transition patterns cannot be deterministically replayed; they present multiple plausible continuations that must be selectively filtered and dynamically disambiguated against the user's historical signals. Therefore, an effective memory mechanism for generative recommendation must transcend flat prefix matching by hierarchically organizing transition patterns and conditioning their retrieval on dynamic user states.

Motivated by these challenges, we conduct an empirical study to investigate reusable transitions in SID-based recommendation. Our investigation yields four key findings. First, SID sequences exhibit pronounced transitions at dual scales—intra-item token transitions and inter-item transitions across consecutive items—demonstrating that recurrent patterns at both levels serve as viable memory entries. Second, an explicit transition lookup conditioned solely on observed SID structures yields strong predictive signals for next-item prediction, confirming that addressed retrieval effectively complements dense computation. Third, while dense Transformers capture the general frequency trend of repeated transitions, a noticeable performance gap persists for frequent transitions where explicit transition statistics provide highly reliable signals. Fourth, a given transition prefix frequently maps to multiple valid successors, demonstrating that transition lookup alone is insufficient and that context-dependent inference is essential to disambiguate among those candidate paths. Collectively, these findings motivate a conditional memory framework that augments generative recommendation with reusable structural signals while preserving dynamic contextual inference.

We propose **Hierarchical Conditional Transition Memory (HCTM)**, a structure-aligned conditional-memory framework for SID-based generative recommendation. Specifically, an **Item-Internal Token Memory** captures recurring intra-item token patterns, while a **Prefix-Conditioned Transition Memory** retrieves successor-prefix predictive signals for candidate items. Each memory component is seamlessly incorporated into the Transformer backbone via a context-conditioned residual gate. This gating mechanism enables the model to dynamically blend retrieved structural signals with the user's ongoing interaction history and the current generation state.

Extensive experiments on three widely-adopted benchmarks demonstrate that HCTM consistently outperforms competitive baselines across all evaluation metrics. Beyond accuracy gains, comprehensive diagnostic analyses elucidate the underlying mechanisms of HCTM from two critical perspectives. First, fine-grained evaluations reveal that HCTM provides monotonically increasing benefits on recurring item transitions, successfully closing the performance gap on frequent patterns identified in our empirical study. Second, training dynamic analyses demonstrate that explicitly decoupling transition retrieval significantly accelerates model convergence. By relieving the Transformer backbone from memorizing invariant transitions via gradient descent, HCTM frees up critical representational capacity, thereby facilitating faster and more effective context-dependent inference.

2 Problem Formulation: Generative Recommendation

Let $u \in \mathcal{U}$ denote a user and $\mathcal{H}_{u,<t} = (v_1, \ldots, v_{t-1})$ represent their chronological interaction history prior to step t. Following the SID paradigm, each item v_t is uniquely represented as a compact tuple of discrete semantic tokens (Rajput et al., 2023):

$$\mathbf{s}(v_t) = (s_{t,1}, s_{t,2}, s_{t,3}, s_{t,4}) = (c_{t,1}, c_{t,2}, c_{t,3}, d_t), \tag{1}$$

where the first three components $c_{t,1:3}$ denote hierarchical quantized semantic codes, and d_t is a deduplication suffix ensuring collision-free identification for items sharing identical codes (Ju et al.,

2025). We refer to the complete tuple $\mathbf{s}(v_t)$ as a *semantic item* and each constituent element $s_{t,\ell}$ as a *semantic token*. Interaction sequences are flattened into a continuous stream of semantic tokens while strictly preserving item-level boundaries.

Conditioned on the historical interactions $\mathcal{H}_{u,<t}$, the generative recommender autoregressively predicts the target semantic item token by token (Rajput et al., 2023):

$$p(v_t \mid \mathcal{H}_{u,<t}) = \prod_{\ell=1}^{4} p(s_{t,\ell} \mid \mathcal{H}_{u,<t}, s_{t,<\ell}) \,, \tag{2}$$

where $s_{t,<\ell} = (s_{t,1}, \ldots, s_{t,\ell-1})$ denotes the prefix generated up to position ℓ. The backbone model is optimized end-to-end using the standard autoregressive cross-entropy loss over all target tokens.

3 Preliminary: Missing Memory Pathway in Dense Transformers

Transformers serve as the standard backbone for generative recommendation owing to their capacity for modeling sequential interactions (Vaswani et al., 2017; Kang & McAuley, 2018). However, standard architectures lack an explicit lookup primitive for recurrent patterns, conflating invariant structural regularities and dynamic context-dependent signals within identical dense computations (Cheng et al., 2026). Rather than indexing recurring patterns from dedicated memory, the model repeatedly reconstructs them across layers, entangling structural knowledge with behavioral inference. Recent language modeling works address this limitation via conditional memory; for instance, DeepSeek-Engram (Cheng et al., 2026) retrieves pre-indexed representations via prefix addressing and injects them into hidden states.

Extending explicit memory to SID-based recommendation remains an open challenge. Unlike classical recommenders that employ memory primarily for long-term user profiles (Pi et al., 2019; 2020; Wang et al., 2025), retrieving reusable SID structures introduces unique non-determinism. Specifically, unlike deterministic phrase completions in language, an identical SID prefix often branches into divergent item continuations conditioned on contextual signals. To investigate whether explicit lookup benefits this setting, we conduct an empirical study in the following sections.

4 An Empirical Study of SID Transition Patterns

This empirical study systematically investigates whether SID sequences contain reusable structural patterns suitable for explicit retrieval, and whether such retrieval fundamentally complements dense Transformer architectures. Specifically, our analysis addresses four sequential research questions: (1) the empirical existence of recurrent transitions across different granularities; (2) the predictive efficacy of explicit transition lookup; (3) the extent to which standard dense Transformers successfully utilize these recurrent structures; and (4) the architectural reasons why static lookup alone cannot resolve exact next-item prediction. For conciseness, we present representative diagnostic findings on the *Toys* dataset in the main text, while providing comprehensive results for *Beauty* and *Sports* in Appendix H. Detailed experimental protocols are deferred to Appendix H.1.

SID sequences exhibit concentrated and reusable transition patterns. SID interaction sequences exhibit pronounced structural concentration amenable to explicit memory indexing. On *Toys*, we analyze two structural families: intra-item token bigrams and inter-item prefix transitions (source item prefix to target item). As shown in Figure 1, the empirical distribution is heavily skewed: the top 5% most frequent patterns cover 30.50% of intra-item and 13.50% of inter-item occurrences (11.31% and 5.35% for the top 1%). While confirming abundant structural reuse, high frequency alone does not ensure reliable predictive signals or effective Transformer utilization, motivating our subsequent analysis.

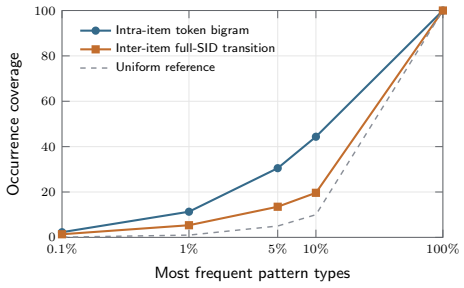

Figure 1: Cumulative occurrence coverage of frequency-ranked SID patterns in Toys.

Explicit transition retrieval reveals predictive signals from SID structures. To evaluate whether explicit transitions provide informative priors, we compare TIGER (Rajput et al., 2023) against a non-parametric lookup based on training-set transition counts. The lookup indexes the preceding item's SID to predict the next item autoregressively, acting as a parameter-free diagnostic for explicit structural signals. As shown in Figure 2, the simple lookup achieves NDCG@10 comparable to TIGER across *Beauty*, *Sports*, and *Toys*, trading the lead between datasets. This competitive performance demonstrates that recurrent SID transitions carry potent predictive signals directly extractable via explicit lookup, even without parameter learning or full historical context.

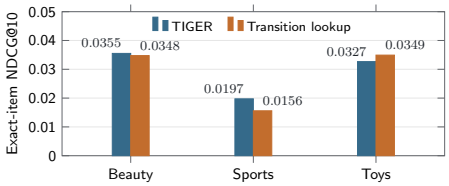

Figure 2: Exact-item NDCG@10 for TIGER and the full-SID transition lookup.

Dense Transformers capture but do not fully exploit reusable transition structures. To assess whether standard architectures exploit these recurrent patterns, we stratify test transitions by their training frequencies and evaluate teacher-forced next-item accuracy. Across benchmarks, TIGER's accuracy scales monotonically with transition frequency and exhibits positive Spearman correlation with empirical probabilities (Appendix H.3), confirming that self-attention partially internalizes frequency biases. However, a marked performance gap remains between TIGER and the non-parametric lookup, widening substantially on frequent transitions where empirical counts provide strong predictive signals. This shortfall shows that dense Transformers perceive transition patterns but lack the structural capacity to fully exploit high-confidence, recurrent transitions without explicit memory mechanisms.

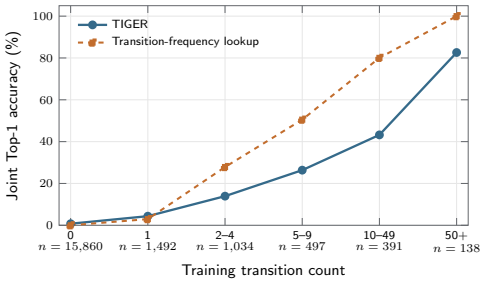

Figure 3: TIGER utilization of repeated transitions on *Toys*.

Pattern lookup alone is insufficient due to one-to-many transition ambiguity. While explicit lookup provides plausible transition candidates, the source pattern cannot independently determine the exact successor item. Conditioning training transitions on the complete source SID (Figure 4), the single most probable continuation accounts for merely 12.93% of the conditional probability mass. This reveals that the primary bottleneck is not a lack of pattern coverage, but the high entropy of one-to-many transition branches originating from a single source prefix. Consequently, explicit lookup effectively circumscribes plausible structural candidates, leaving final disambiguation among branching paths to context-dependent inference over the broader interaction history.

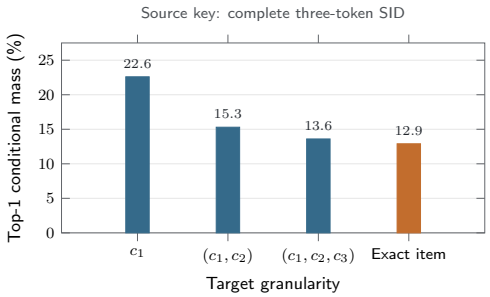

Figure 4: Conditional probability mass distribution of top candidate targets given the complete source SID on *Toys*.

Summary. While explicit transition lookup unlocks salient predictive signals from recurring patterns, it cannot independently resolve one-to-many continuation ambiguity. These empirical insights underscore the necessity of a conditional memory architecture, which couples explicit structural retrieval with adaptive contextual inference to achieve robust next-item disambiguation.

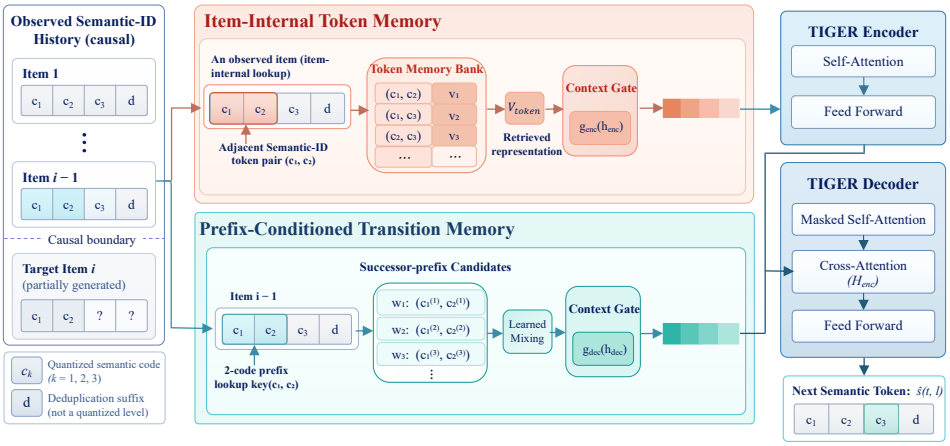

Figure 5: HCTM: Dual memory pathways retrieve invariant structural representations and inter-item transition candidates from causally visible SID contexts, which are dynamically modulated via context-conditioned residual gates before merging into the Transformer hidden states.

5 METHOD

5.1 OVERVIEW

We propose **Hierarchical Conditional Transition Memory (HCTM)**, a structure-aligned conditional memory framework for SID-based generative recommendation. HCTM operates via two complementary memory pathways tailored to distinct structural granularities. On the encoder side, the **Item-Internal Token Memory** retrieves representations for contiguous intra-item token patterns within observed histories (section 5.2). On the decoder side, the **Prefix-Conditioned Transition Memory** leverages the prefix of the most recently observed item to retrieve candidate prefixes for the target item and blends their learned representations (section 5.3). Context-dependent gating networks dynamically regulate both memory signals (section 5.4), which are seamlessly injected into the Transformer representations as hidden-state residuals (section 5.5). Detailed formulations of the causal attention masks and the end-to-end training objective are deferred to Appendix D.3.

5.2 ITEM-INTERNAL TOKEN MEMORY

SIDs possess strong hierarchical inductive biases (Singh et al., 2024; Ju et al., 2025), where contiguous intra-item tokens form highly recurrent local patterns. For a given semantic item v_i, let its token sequence be denoted as $\mathbf{s}(v_i) = (s_{i,1}, s_{i,2}, s_{i,3}, s_{i,4})$. To explicitly index these recurring local structures, the encoder-side **Item-Internal Token Memory** establishes an addressed retrieval pathway over adjacent token pairs. Specifically, for each internal token position $j \in \{2, 3, 4\}$, we construct a localized bigram address $a_{i,j}^{\text{tok}} = (s_{i,j-1}, s_{i,j})$, and retrieve the corresponding structural representation via: $\mathbf{m}_{i,j}^{\text{tok}} = \mathbf{V}_{\text{tok}}\left[h(a_{i,j}^{\text{tok}})\right]$, where $h(\cdot)$ denotes a deterministic hashing function mapping the token tuple to a discrete slot index, and $\mathbf{V}_{\text{tok}} \in \mathbb{R}^{B_{\text{tok}} \times d}$ represents a trainable memory embedding table with B_{tok} slots. The retrieved memory vector $\mathbf{m}_{i,j}^{\text{tok}}$ is integrated into the corresponding encoder hidden state, offloading the burden of reconstructing recurring intra-item compositions from dense self-attention layers and enriching the context passed to the decoder.

5.3 PREFIX-CONDITIONED TRANSITION MEMORY

While the Item-Internal Token Memory provides an addressed lookup pathway for static intra-item compositions, next-item generation introduces a distinct operational objective. Specifically, the model must not only preserve observed historical structures, but also project how the current interaction context extrapolates into plausible successor items. To satisfy this predictive requirement, we introduce the **Prefix-Conditioned Transition Memory**, which indexes observed transition structures to supply explicit predictive signals for upcoming token positions.

For a semantic item v_i, we define its structural prefix as the initial two-token tuple $p_i = (s_{i,1}, s_{i,2})$. At each item transition boundary, the memory module is queried using the prefix of the preceding item v_{i-1} as the addressing key $a_i = p_{i-1}$. Conditioned on this key, the memory retrieves a candidate set of top-K successor prefixes observed in the training corpus:

$$\mathcal{C}(a_i) = \left\{ \widehat{p}_i^{(j)} \right\}_{j=1}^{K}, \qquad \widehat{p}_i^{(j)} = \left(\hat{s}_{i,1}^{(j)}, \hat{s}_{i,2}^{(j)} \right). \tag{3}$$

To map these discrete candidate prefixes into continuous latent space, each candidate $\widehat{p}_i^{(j)}$ is associated with a trainable embedding $\mathbf{e}_j^{(\ell)} \in \mathbb{R}^d$ corresponding to SID code level $\ell \in \{1, 2\}$. The memory readout subsequently aggregates these candidate representations via a convex combination parameterized by non-negative normalized mixture weights $\pi_{a_i,j}$:

$$\mathbf{\Delta}_{i,\ell} = \sum_{j=1}^{K} \pi_{a_i,j}\, \mathbf{e}_j^{(\ell)}, \qquad \text{where} \quad \sum_{j=1}^{K} \pi_{a_i,j} = 1. \tag{4}$$

The aggregated residual vectors $\mathbf{\Delta}_{i,\ell}$ inject strong structural transition signals directly into the decoder representations, providing an explicit prior while preserving the Transformer's capacity to perform context-dependent inference over the full interaction history.

5.4 Context-Conditioned Gating

Retrieved patterns provide prior signals rather than deterministic predictions, whose utility depends on the current decoding context. We therefore employ context-conditioned gating to dynamically modulate the injected memory representations. Formally, for each memory branch $r \in \{\mathrm{tok}, \mathrm{trans}\}$ at step t, we compute:

$$\mathbf{g}_{t,r} = \sigma(\mathbf{W}_r \mathbf{h}_t + \mathbf{b}_r), \qquad \widetilde{\mathbf{m}}_{t,r} = \mathbf{g}_{t,r} \odot \mathbf{m}_{t,r}, \tag{5}$$

where $\mathbf{h}_t$ is the current hidden state, $\mathbf{W}_r$ and $\mathbf{b}_r$ are branch-specific parameters, and $\odot$ denotes elementwise multiplication. These decoupled gates allow the model to flexibly balance structural priors and dynamic contextual signals, leaving final disambiguation to the Transformer backbone.

5.5 Residual Architecture Integration

Generative recommendation models typically adopt an encoder–decoder backbone (Rajput et al., 2023): the encoder processes user history, while the decoder autoregressively generates the target SID. We structurally align each memory pathway with its functional stage. The Item-Internal Token Memory injects intra-item structural representations into the encoder to enrich historical context, whereas the Prefix-Conditioned Transition Memory supplies successor priors to the decoder to guide next-item prediction.

To balance initial representation learning with final contextual projection, we inject each modulated memory as a residual addition after the first and last Transformer layers of its respective module:

$$\mathbf{h}_t^{(\ell)} \leftarrow \mathbf{h}_t^{(\ell)} + \widetilde{\mathbf{m}}_{t,r}, \qquad \ell \in \{1, L\}. \tag{6}$$

This dual-stage placement allows structural priors to directly participate in early encoding without bypassing late context-dependent inference.

6 Experimental Setup

6.1 Datasets and Evaluation Protocol

We evaluate HCTM on three standard Amazon benchmarks: *Beauty*, *Sports*, and *Toys*, adopting the standardized SID setups from GRID (Ju et al., 2025).[1] Items are represented by textual SIDs, with user sequences chronologically partitioned into training, validation, and test splits across all baselines (Appendix A). Performance is measured by $\mathrm{NDCG@}K$, $\mathrm{HR@}K$, and $\mathrm{MRR@}K$ ($K \in \{5, 10\}$; Appendix B). For generative models, candidates are generated via prefix-tree-constrained beam search (beam width 50) over valid items without negative sampling. Checkpoints are chosen by validation NDCG@10, with final metrics reported on held-out test sets.

[1] https://github.com/snap-research/GRID

Table 1: Recommendation performance across three datasets. The best and second-best results are bolded and underlined, respectively. * denotes statistically significant ($p < 0.05$, paired t-test). Comprehensive evaluations across all metric cutoffs ($K \in \{5, 10\}$) are reported in Table 6.

Method	Beauty		Sports		Toys	
	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10
Traditional sequential baselines						
Caser	0.0035	0.0071	0.0016	0.0036	0.0011	0.0024
GRU4Rec	0.0102	0.0218	0.0072	0.0151	0.0080	0.0165
SASRec	0.0283	0.0539	0.0153	0.0304	<u>0.0353</u>	<u>0.0619</u>
Generative baselines						
LETTER	0.0306	0.0551	0.0171	0.0323	0.0257	0.0488
OneRec	<u>0.0363</u>	<u>0.0658</u>	<u>0.0201</u>	0.0367	0.0320	0.0577
TIGER	0.0355	0.0638	0.0197	<u>0.0370</u>	0.0327	0.0605
Memory augmented baselines						
DeepSeek-Engram	0.0301	0.0533	0.0164	0.0318	0.0320	0.0557
HCTM	**0.0406***	**0.0684***	**0.0220***	**0.0379**	**0.0404***	**0.0698***

6.2 Compared Methods

We evaluate baselines across three groups: (1) traditional sequential models (Caser, GRU4Rec, and SASRec); (2) generative recommendation frameworks, including TIGER (Rajput et al., 2023), OneRec (Zhou et al., 2025), and LETTER (Wang et al., 2024); and (3) architecture-side memory baselines, adapting DeepSeek-Engram (Cheng et al., 2026) to TIGER via flat local-token lookup. Unlike Engram's uniform adjacency indexing, HCTM hierarchically aligns memory retrieval with the stages of generative recommendation—reusing intra-item patterns during encoding and retrieving successor priors during decoding. Detailed implementations are provided in Appendix C.

6.3 Implementation Details

All memory-augmented variants adopt the standard training protocol of the TIGER backbone. HCTM indexes the Item-Internal Token Memory on the encoder and the Prefix-Conditioned Transition Memory on the decoder, injecting gated residuals after Layers 0 and 3 of each respective stack. Identical placement and residual parameterizations are enforced across all memory-based baselines. Further hyperparameters and training configurations are detailed in Appendix D.

7 Results

7.1 Main Results

Table 1 reports test-set ranking results across the three datasets. HCTM consistently improves over TIGER across all reported metrics. These gains demonstrate the effectiveness of incorporating structure-aligned conditional memory into generative recommendation. Item-Internal Token Memory captures recurring token structures within observed Semantic Items. On the decoder side, Prefix-Conditioned Transition Memory operates at the item-prefix level: it retrieves successor-prefix candidates from the observed history for the next item. Both memories contribute context-conditioned residuals that support TIGER's autoregressive SID generation.

7.2 Ablation Study

We evaluate three core architectural dimensions of HCTM on *Toys*: component contributions, context-conditioned gating, and memory placement. Table 2 summarizes the test-set NDCG@10 results. Here, Item and Prefix denote Item-Internal Token Memory and Prefix-Conditioned Transition Memory, respectively; in assignment variants, the order denotes encoder/decoder placement. Assignment variants use context-conditioned gates; Constant fixes gating coefficients to ones.

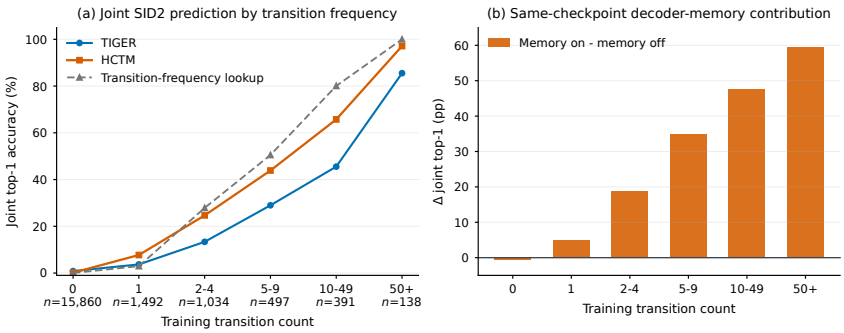

Figure 6: Transition-frequency utilization analysis on *Toys*. (a) Teacher-forced joint top-1 accuracy over the target SID prefix $(s_{*,1}, s_{*,2})$, stratified by the training frequency of the exact source-to-target prefix transition. (b) Paired performance shifts resulting from zeroing out decoder transition-memory hooks on the identical HCTM checkpoint.

Both memory components contribute to the full model. The unified HCTM configuration achieves the highest NDCG@10 among the component variants. Prefix memory accounts for most of the gain. Adding Item memory to Prefix-only yields a smaller further improvement, so the component ablation is consistent with complementary contributions but does not by itself establish them.

Context-conditioned gating improves evidence fusion. Replacing the adaptive gates with constant unit residuals lowers NDCG@10 from 0.040431 to 0.038962, indicating that state-dependent modulation is more effective than uniform injection.

Structure-aligned assignment gives the best result. The intended Item/Prefix assignment achieves the highest NDCG@10, while both same-type assignments are lower. The inverted Prefix/Item variant is competitive, so the ablation supports Item/Prefix as the best observed configuration but does not show that inversion is uniformly harmful.

Table 2: Ablation study on *Toys*.

Setting	Variant	NDCG@10
Full		
	HCTM (Item/Prefix)	0.040431
Components		
	TIGER	0.032657
	Item only	0.033105
	Prefix only	0.039444
Gating		
	Constant	0.038962
Assignment		
	Item/Item	0.034481
	Prefix/Item	0.038165
	Prefix/Prefix	0.036871

8 Analysis

8.1 Where Does Explicit Memory Help?

To determine which interaction regimes benefit most from explicit memory, we evaluate teacher-forced joint top-1 prefix accuracy $(s_{*,1}, s_{*,2})$ stratified by training transition frequency (Figure 6a). Across frequency bins, HCTM outperforms TIGER on all seen-transition bins, and zeroing out decoder transition-memory hooks isolates a marginal contribution that increases monotonically with training transition frequency (Figure 6).

Explicit memory yields monotonically increasing gains with pattern frequency. While memory incurs a negligible performance penalty on unseen transitions, its benefits scale monotonically with transition frequency, delivering substantial gains on frequent patterns. In aggregate, the dramatic improvements on observed transitions far outweigh the slight unseen degradation. This empirical trend resolves the under-utilization of recurrent transitions identified in Figure 3, validating that explicit priors reliably boost high-confidence continuations without sacrificing generalizability.

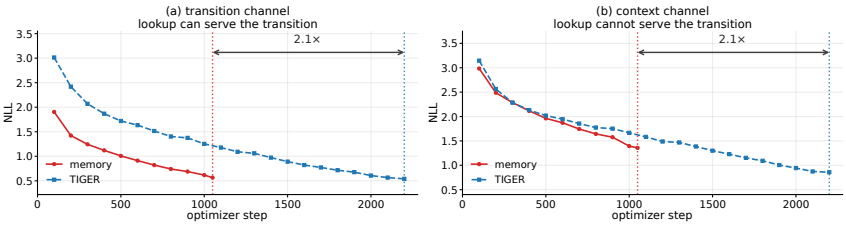

Figure 7: Teacher-forced NLL trajectories on *Toys* across memory-hit and memory-miss channels. Dotted verticals denote the early-stopping steps; arrows indicate the convergence speedup ratio.

8.2 Can Explicit Memory Accelerate Convergence?

We hypothesize that explicit lookup of recurrent transitions alleviates the parameter-fitting burden on dense attention, freeing capacity for contextual inference. We track teacher-forced NLL at checkpoints on fixed *memory-hit* and *memory-miss* training slices (Appendix G).

Explicit retrieval bypasses parametric warm-up. On the memory-hit channel (Figure 7a), HCTM exhibits an immediate initial loss drop, maintaining a persistent performance margin over TIGER from step 0. The retrieval lets HCTM reach TIGER's final loss in roughly a quarter of the steps, a $2.1\times$ speedup to early stopping.

Capacity offloading promotes generalization. Crucially, on the memory-miss channel (Figure 7b) where memory supplies no direct pattern priors, the two trajectories align initially but visibly diverge, with HCTM descending at a consistently steeper slope. Pre-indexing recurrent transitions frees parametric bandwidth, enhancing inductive capacity on novel contexts.

9 Related Work

SID-Based Generative Recommendation. Generative recommendation models items as sequences of Semantic Tokens, generating target items code-by-code via auto-regressive decoding (Rajput et al., 2023). Subsequent efforts have focused either on learning better item tokenizers via collaborative and semantic regularization (Wang et al., 2024) or refining structural decoding through prefix- and trie-constrained policies (Zhou et al., 2025; Deng et al., 2025; Ma et al., 2025; Xu et al., 2026). While these paradigms enhance token representations and generation constraints, they leave the generator without an explicit mechanism to retrieve recurrent patterns from SID histories. HCTM keeps the representation and training objective intact, directly innovating the internal information-access pathway within the generator backbone.

Memory-Augmented Sequence Modeling and Recommendation. Conditional-memory architectures augment dense Transformers with sparse lookup channels, spanning product-key memories (Lample et al., 2019; Rae et al., 2016), embedding stores (Liu et al., 2026b; Sadhukhan et al., 2026), and n-gram-indexed modules (Cheng et al., 2026; Zhou et al., 2026; Zheng et al., 2026). Although providing retrieval capacity, these generic architectures apply flat token-level indexing that neglects the compositional roles of SIDs. Separately, memory-based recommendation frameworks use external or dynamic tables to capture long-term user interests and evolving behavioral profiles (Pi et al., 2019; 2020; Wang et al., 2025; Chen, 2025; Qian et al., 2026; Liu et al., 2026a). These methods prioritize user-level behavioral history rather than directly exploiting the intrinsic syntax and transition structures of SIDs.

10 Conclusion

In this paper, we address the structure–memory mismatch in SID-based generative recommendation by proposing HCTM, a memory framework. By decoupling item-internal token memory on the encoder from prefix-conditioned transition memory on the decoder, HCTM provides structural priors via context-conditioned residual gates, letting dense Transformers focus on contextual disambiguation. Empirical evaluations demonstrate that HCTM consistently boosts ranking metrics, delivers monotonically increasing benefits on frequent transitions, and accelerates convergence.

AI USE STATEMENT

We used generative AI tools, including OpenAI Codex, for the following tasks with required disclosure: proposing and refining hypotheses; designing and providing feedback on research methodology and experiments; implementing methods and evaluation code; and interpreting experimental results. We also used these tools for recommended-disclosure tasks, including creating and editing software code and scientific figures, summarizing existing literature, brainstorming, and drafting and editing the manuscript. The remaining required-disclosure subtasks, including synthetic-data generation, translation, qualitative data analysis, and proof assistance, were not applicable to this work.

All AI-assisted code was inspected and executed as part of the reported experimental pipeline. Numerical results were extracted from recorded evaluation artifacts and checked against their source files. The authors reviewed all AI-assisted text, equations, implementation descriptions, analyses, citations, and claims, and take responsibility for the final content of this submission.

REPRODUCIBILITY STATEMENT

To support reproducibility, we will release an anonymized repository containing the model implementation, experiment configurations, data-processing procedures, evaluation scripts, and instructions for reproducing the reported results. The appendix provides dataset accounting, memory addressing and fusion details, branch-specific settings, and the training and evaluation protocol. All method comparisons use fixed data splits and report results on the held-out test set.

REFERENCES

Jiarui Chen. Memory assisted llm for personalized recommendation system. *arXiv preprint arXiv:2505.03824*, 2025.

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Zhewen Hao, Han Zhang, Yukun Li, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. Conditional memory via scalable lookup: A new axis of sparsity for large language models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2–7, 2026*, pp. 4968–4990. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.226. URL https://doi.org/10.18653/v1/2026.acl-long.226.

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

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

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.

Balazs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

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 Meeyoung Cha, Chanyoung Park, Noseong Park, Carl Yang, Senjuti Basu Roy, Jessie Li, Jaap Kamps, Kijung Shin, Bryan Hooi, and Lifang He (eds.), *Proceedings of the 34th ACM International Conference on Information and Knowledge Management, CIKM 2025, Seoul, Republic of Korea, November 10–14, 2025*, pp. 6420–6425. ACM, 2025. doi: 10.1145/3746252.3761612. URL https://doi.org/10.1145/3746252.3761612.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining*, pp. 197–206, 2018.

Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Large memory layers with product keys. *Advances in Neural Information Processing Systems*, 32, 2019.

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.

Hong Liu, Jiaqi Zhang, Chao Wang, Xing Hu, Linkun Lyu, Jiaqi Sun, Xurui Yang, Bo Wang, Fengcun Li, Yulei Qian, et al. Scaling embeddings outperforms scaling experts in language models. *arXiv preprint arXiv:2601.21204*, 2026b.

Luyi Ma, Wanjia Zhang, Kai Zhao, Abhishek Kulkarni, Lalitesh Morishetti, Anjana Ganesh, Ashish Ranjan, Aashika Padmanabhan, Jianpeng Xu, Jason HD Cho, et al. Grace: Generative recommendation via journey-aware sparse attention on chain-of-thought tokenization. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pp. 135–144, 2025.

Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. Practice on long sequential user behavior modeling for click-through rate prediction. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2671–2679, 2019.

Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 2685–2692, 2020.

Hui Qian, Changfa Wu, Chang Liu, Binbin Cao, Jian Wu, Yuliang Yan, Han Zhu, Bo Zheng, and Shiye Wang. Loopmemgr: From behavior logs to evolving memory for generative recommendation. *arXiv preprint arXiv:2607.27647*, 2026.

Jack Rae, Jonathan J Hunt, Ivo Danihelka, Timothy Harley, Andrew W Senior, Gregory Wayne, Alex Graves, and Timothy Lillicrap. Scaling memory-augmented neural networks with sparse reads and writes. *Advances in Neural Information Processing Systems*, 29, 2016.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Mahesh Sathiamoorthy. Recommender systems with generative retrieval. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10–16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/20dcab0f14046a5c6b02b61da9f13229-Abstract-Conference.html.

Ranajoy Sadhukhan, Sheng Cao, Harry Dong, Changsheng Zhao, Attiano Purpura-Pontoniere, Yuandong Tian, Zechun Liu, and Beidi Chen. Stem: Scaling transformers with embedding modules. *arXiv preprint arXiv:2601.10639*, 2026.

Anima Singh, Trung Vu, Nikhil Mehta, Raghunandan H. Keshavan, Maheswaran Sathiamoorthy, Yilin Zheng, Lichan Hong, Lukasz Heldt, Li Wei, Devansh Tandon, Ed H. Chi, and Xinyang Yi. Better generalization with semantic ids: A case study in ranking for recommendations. In Tommaso Di Noia, Pasquale Lops, Thorsten Joachims, Katrien Verbert, Pablo Castells, Zhenhua Dong, and Ben London (eds.), *Proceedings of the 18th ACM Conference on Recommender Systems, RecSys 2024, Bari, Italy, October 14-18, 2024*, pp. 1039–1044. ACM, 2024. doi: 10.1145/3640457.3688190. URL https://doi.org/10.1145/3640457.3688190.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (eds.), *Proceedings of the 28th ACM International Conference on Information*

and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019, pp. 1441–1450. ACM, 2019. doi: 10.1145/3357384.3357895. URL https://doi.org/10.1145/3357384.3357895.

Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 565–573, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Chengbing Wang, Yang Zhang, Fengbin Zhu, Jizhi Zhang, Tianhao Shi, and Fuli Feng. Leveraging memory retrieval to enhance llm-based generative recommendation. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 1346–1350, 2025.

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. *arXiv preprint arXiv:2405.07314*, 2024. URL https://arxiv.org/abs/2405.07314.

Zhenxiang Xu, Jiawei Chen, Sirui Chen, Yong He, Jieyu Yang, Chuan Yuan, Ke Ding, and Can Wang. Trie-aware transformers for generative recommendation. *arXiv preprint arXiv:2602.21677*, 2026.

Yunao Zheng, Guoyang Xia, Xiaojie Wang, and Lei Ren. Lngram: N-gram conditional memory in latent space. *arXiv preprint arXiv:2605.24869*, 2026.

Guorui Zhou, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Qiang Luo, Qianqian Wang, Qigen Hu, Rui Huang, Shiyao Wang, et al. Onerec technical report. *arXiv preprint arXiv:2506.13695*, 2025.

Wuyang Zhou, Yuxuan Gu, Giorgos Iacovides, Yuning Qiu, Qibin Zhao, and Danilo Mandic. Tensorizing engram: Sharing latents across n-gram embeddings is beneficial in llms. *arXiv preprint arXiv:2606.08347*, 2026.

A Dataset Statistics

We use the chronological `training`, `evaluation` (validation), and `testing` partitions materialized by the GRID preprocessing pipeline. Each partition contains one sequence record per user, and we use these artifacts without creating a new split or shuffling interactions across partitions. Table 3 reports the dataset statistics. "Users" is the number of distinct users in the training partition; because each split has one sequence record per user, this is also the number of training records. "Items" is the number of distinct item IDs in the union of all three partitions. "Train interactions" is the total number of item IDs in the training sequences before model-side truncation. The mean length is the training interaction count divided by the number of users.

Table 3: Statistics of the materialized GRID partitions. Interaction counts and mean sequence lengths refer to the training partition before model-side sequence truncation.

Dataset	Users	Items	Train interactions	Mean length
Beauty	22,363	12,101	153,776	6.88
Sports	35,598	18,357	225,141	6.32
Toys	19,412	11,924	128,773	6.63

B Metric Definitions

For direct-item recommenders, the item catalog is scored directly. For TIGER and its architecture-side memory methods, beam search generates SID candidates, which are ranked by their sequence

likelihood. The reported configuration uses beam width 50. We compute ranking metrics against the held-out target item in this ranked list.

Let r_u be the rank of the held-out target for test example u, with $r_u = \infty$ when the target is absent from the generated list. We use

$$\begin{aligned} \mathrm{HR@}K &= \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \mathbf{1}[r_u \le K], \\ \mathrm{NDCG@}K &= \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{\mathbf{1}[r_u \le K]}{\log_2(r_u + 1)}. \\ \mathrm{MRR@}K &= \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{\mathbf{1}[r_u \le K]}{r_u}. \end{aligned} \tag{7}$$

HR@K. HR@K is the fraction of test examples for which the held-out target item appears anywhere in the top-K list. A higher value means that the recommender successfully retrieves the correct item for more users, but HR@K does not distinguish among positions within the top-K list.

NDCG@K. NDCG@K also measures whether the target is retrieved, while assigning more weight to higher-ranked positions through the logarithmic discount. A higher value therefore means that the correct item is both retrieved more often and, on average, placed closer to the top of the recommendation list. Each test example has one held-out relevant item, so its ideal DCG is 1; the expression above is therefore already normalized by the ideal ranking.

MRR@K. MRR@K averages the reciprocal rank of the target when it appears within the top-K list. A higher value means that successful predictions tend to place the correct item near rank 1; predictions with the target outside the cutoff contribute zero.

C Baseline Descriptions

Traditional methods. These sequential recommenders use a user's recent item sequence to build a representation of preference and predict the next item directly from the item catalog. They differ in how they summarize the sequence: convolutional patterns, recurrent state, or self-attention over the history.

- **Caser** (Tang & Wang, 2018) embeds the recent interaction sequence across time and latent dimensions, then applies convolutional filters to detect local sequence patterns. Its representation combines these patterns with broader user preferences for catalog-item scoring.
- **GRU4Rec** (Hidasi et al., 2016) processes a session in order with gated recurrent units. Its hidden state summarizes the items observed so far and supports next-item ranking, making it a recurrent alternative for short interaction histories.
- **SASRec** (Kang & McAuley, 2018) uses causal self-attention to select relevant items from the preceding history when forming a prediction state. This lets the model represent both nearby transitions and longer-range dependencies without compressing the whole history into one recurrent state.

Generative baselines. Unlike direct-item recommenders, these methods predict through a generated item representation. TIGER generates a sequence of SID codes and constrains decoding to valid catalog items; the main comparison also includes additional generative baselines.

- **LETTER** (a LEarnable Tokenizer for generaTivE Recommendation) (Wang et al., 2024) treats item tokenization as a learnable component of generative recommendation. It uses RQ-VAE for hierarchical semantic regularization, a contrastive alignment loss to transfer collaborative-filtering signals into the tokenizer, and a diversity loss to reduce code-assignment and item-generation bias. LETTER can be instantiated on different generative recommender backbones and adds a ranking-guided generation loss to improve ranking performance. We compare against its reported NDCG, HR, and MRR results.

- **OneRec** (Zhou et al., 2025) replaces the decoder feed-forward network with a sparse top-2 mixture-of-experts layer. In our controlled reproduction, it retains the GRID encoder–decoder backbone, SIDs, history length, and next-4 SID generation protocol.
- **TIGER** (Rajput et al., 2023) represents each item with a SID code sequence and uses a Transformer encoder–decoder to predict the next item's codes autoregressively from the interaction history. At test time, beam search is constrained to valid catalog items.

Architecture-side memory methods. Both methods add conditional retrieval to the TIGER backbone. DeepSeek-Engram serves as a generic adjacent-token memory control; HCTM instead aligns each retrieval pathway with the hierarchy of SIDs and with the distinction between representing history and generating the next item.

- **DeepSeek-Engram** (Cheng et al., 2026) adapts Engram-style conditional n-gram lookup to Semantic Tokens. In our control, adjacent token pairs provide the lookup keys in both encoder and decoder streams, including pairs that cross Semantic-Item boundaries. Appendix D.2 gives the generic local-adjacency control and fusion details.
- **HCTM** is the proposed structure-aligned architecture. Its encoder Item-Internal Token Memory retrieves recurring within-item token patterns from the observed history. Its decoder Prefix-Conditioned Transition Memory retrieves likely successor prefixes for next-item generation. The separate pathways make the lookup key reflect both the SID structure and the role of the current prediction state.

D Implementation Details

D.1 Address Construction and Tables

The encoder Item-Internal Token Memory uses a hashed bigram table with 4,096 rows. For a Semantic Token c_t at level ℓ, the level-offset ID is $z_t = c_t + 256\ell$. For a bigram ending at position t, hashing starts at $h_0 = 17$ and applies $h_{r+1} = (1{,}000{,}003h_r + z_{t-r} + 97(r+1)) \bmod 4{,}096$ for $r = 0, 1$. Addresses are valid only for adjacent Semantic Tokens belonging to the same Semantic Item. The lookup is applied after the first and last encoder blocks.

The coarse decoder index is compiled from the training partition only. Its source key is the first two codes (c_1, c_2) of the latest complete visible history item. For each source key, the index stores at most eight successor prefixes ordered by training frequency; this candidate support is fixed after index construction. Each retrieved slot has a learnable score initialized to the log of its training successor probability, and a softmax over valid slots produces the mixture weights. Candidate values are read from separate learned embedding tables for the first and second SID levels, indexed by the candidate SID codes rather than by slot identity or decoder output-projection rows. The resulting level-specific residuals are modulated by context-conditioned gates and added after the first and last decoder blocks.

D.2 Fusion Configuration

Table 4 summarizes the memory-branch configuration at the level needed to distinguish the compared methods. The two retained branches use context-conditioned gated residual injection.

Table 4: Summary of memory branches and lookup inputs.

Branch	Role	Lookup key	Source
Item-Internal Token Memory	Within-item history patterns	Adjacent token pair	Train history
Prefix-Conditioned Transition Memory	Successor-prefix evidence	Latest complete item prefix	Train history
DeepSeek-Engram	Generic local adjacency control	Adjacent token pair	Generic control

D.3 Causal Inputs and Training Objective

The encoder processes the serialized observed history $\mathcal{H}_{u,<t}$, and its Item-Internal Token Memory is queried only from Semantic-Token patterns inside those visible items. The decoder receives the encoder representation together with the prefix already available for the target Semantic Item. Its Prefix-Conditioned Transition Memory is addressed by the most recent complete history-item prefix, which is available during both training and autoregressive inference. Thus, each lookup uses only information available when the corresponding prediction is made.

All branches are optimized end-to-end with the factorized next-token cross-entropy objective from the problem setup,

$$\mathcal{L}_{\mathrm{CE}} = -\sum_{t}\sum_{\ell=1}^{4} \log p_\theta(s_{t,\ell} \mid \mathcal{H}_{u,<t}, s_{t,<\ell}) \,.$$

D.4 Training and Evaluation

Table 5 summarizes the canonical training and evaluation configuration. All variants use the backbone's autoregressive cross-entropy objective.

Table 5: Summary of canonical training and evaluation settings.

Parameter	Value
Backbone	4-layer T5 encoder–decoder ($d_{\text{model}} = 128$, 6 heads, $d_{\text{ff}} = 1024$)
SID levels	4
Random seed	42
Optimizer	Adam, lr $= 10^{-3}$, weight decay 10^{-4}
Effective training batch	4,096
Maximum sequence length	120
Checkpoint selection	Validation NDCG@10
Test decoding	Beam width 50
Test cutoffs	5, 10

Validation is used only for checkpoint selection; all reported method comparisons use the held-out test set. TIGER-based methods rank generated SID candidates by sequence likelihood.

E Additional Test-Set Metrics

Table 6 reports test NDCG, HR, and MRR values at cutoffs 5 and 10 for all evaluated methods. Table 1 reports the corresponding NDCG@10 and HR@10 columns.

Table 6: Additional test-set metrics for the main comparison. * denotes $p < 0.05$ against TIGER.

Method	NDCG@5	NDCG@10	HR@5	HR@10	MRR@5	MRR@10
Amazon Beauty						
Caser	0.0026	0.0035	0.0041	0.0071	0.0021	0.0025
GRU4Rec	0.0070	0.0102	0.0117	0.0218	0.0055	0.0068
SASRec	0.0193	0.0283	0.0294	0.0539	0.0137	0.0169
LETTER	0.0232	0.0306	0.0357	0.0551	0.0202	0.0224
OneRec	0.0292	0.0363	0.0440	0.0658	0.0244	0.0273
TIGER	0.0289	0.0355	0.0430	0.0638	0.0243	0.0270
DeepSeek-Engram	0.0245	0.0301	0.0357	0.0533	0.0209	0.0232
HCTM	**0.0339***	**0.0406***	**0.0477***	**0.0684***	**0.0294***	**0.0321***
Amazon Sports						
Caser	0.0011	0.0016	0.0018	0.0036	0.0008	0.0011
GRU4Rec	0.0047	0.0072	0.0074	0.0151	0.0039	0.0049
SASRec	0.0102	0.0153	0.0163	0.0304	0.0078	0.0107
LETTER	0.0133	0.0171	0.0205	0.0323	0.0116	0.0126
OneRec	0.0155	0.0201	0.0238	0.0367	0.0128	0.0147
TIGER	0.0153	0.0197	0.0233	0.0370	0.0128	0.0145
DeepSeek-Engram	0.0124	0.0164	0.0194	0.0318	0.0101	0.0118
HCTM	**0.0182***	**0.0220***	**0.0260**	**0.0379**	**0.0156***	**0.0172***
Amazon Toys						
Caser	0.0008	0.0011	0.0014	0.0024	0.0006	0.0008
GRU4Rec	0.0059	0.0080	0.0098	0.0165	0.0046	0.0055
SASRec	0.0275	0.0353	0.0411	0.0619	0.0224	0.0248
LETTER	0.0201	0.0257	0.0314	0.0488	0.0164	0.0199
OneRec	0.0258	0.0320	0.0387	0.0577	0.0216	0.0241
TIGER	0.0260	0.0327	0.0397	0.0605	0.0215	0.0242
DeepSeek-Engram	0.0263	0.0320	0.0381	0.0557	0.0225	0.0248
HCTM	**0.0340***	**0.0404***	**0.0498***	**0.0698***	**0.0288***	**0.0315***

Beauty TIGER values are from the historical 5090 baseline run. Sports and Toys TIGER values and all HCTM values use the registered test-only evaluations.

F EFFICIENCY AND COMPLEXITY

HCTM has 13,761,211 trainable parameters versus 13,128,064 for the TIGER backbone; the additional parameters come from the token and transition-memory readouts. Table 7 reports a controlled single-GPU micro-benchmark on one RTX 4090. Both models use the same Toys inputs, batch size and training/inference code paths; the measurements use five warm-up steps, twenty measured training steps, and four test batches. Training throughput counts non-padding SID tokens per second, while test throughput measures the generation path used without labels.

Table 7: Single-GPU efficiency micro-benchmark. Parameters are trainable parameters. Train throughput is measured in non-padding tokens/s and test throughput in generated recommendation samples/s. Peak train GB is allocated GPU memory during training. The measured training batches contained 14,112 tokens for TIGER and 14,400 for HCTM because sequence expansion is data-dependent.

Backbone	Method	Params.	Memory branches	Train tok/s	Peak train GB	Test samples/s
TIGER	Baseline	13,128,064	None	167,821	5.21	744.5
TIGER	HCTM	13,761,211	Both	154,771	5.56	680.7

Relative to TIGER, HCTM uses 4.8% more trainable parameters, 7.8% lower training token throughput, and 8.6% lower test generation throughput. Training peak memory increases from 5.21 to 5.56 GiB, while inference peak memory is essentially unchanged (1.811 versus 1.814 GiB). This micro-benchmark isolates architectural overhead.

Memory reads use hash or indexed lookup and mix a small candidate set. For Q transition queries, hidden dimension d, and K retrieved candidates, this lookup and mixing costs $O(QKd)$; hashed token-table reads cost $O(Qd)$. This does not change the Transformer's quadratic self-attention dependence on sequence length. During beam decoding, decoder lookup work scales with the number of active beams.

G PROTOCOL FOR THE CONVERGENCE EXPERIMENT

The two runs compared in Section 8.2 share their seed, data order, optimiser, micro-batch composition, and four-GPU data-parallel setup; they differ only in whether the two memory pathways are present, and both are trained on Toys with the four-level SID and the same training-index lookup as the main model.

Evaluation uses a fixed bank of $1{,}953$ teacher-forced training subsequences taken from the first four batches of the training loader, which correspond to $1{,}613$ distinct inputs once the collate function has expanded some rows into repeated subsequences. Every saved checkpoint, one per 100 optimiser steps, is scored on the same bank, and each subsequence is attributed to a channel by its lookup: the *transition channel* holds the 865 subsequences whose target pair appears among the recorded successors of their source key, and the *context channel* the $1{,}088$ where it does not. The reported quantity is the teacher-forced NLL of the complete target Semantic Item, averaged over the subsequences of a channel.

The two runs also stop at different points under the shared early-stopping rule, which monitors validation NDCG@10 with a patience of 10 evaluations: the memory run stops at step $1{,}050$ and the TIGER run at $2{,}200$. Comparisons above are made at equal step counts, and the whole experiment uses a single seed.

H SUPPLEMENTARY EXPERIMENTS

This section collects diagnostic analyses that use training-split statistics and test-set evaluation where ranking metrics are reported. These analyses characterize the structural signal available in the data; they are not trained-model ablations.

H.1 PROTOCOLS FOR THE DIAGNOSTIC ANALYSES

This subsection documents the four analyses of Section 4 and the paragraphs that follow it. All four use training-split statistics; analyses that report ranking metrics are evaluated on the testing split, and validation data is not used. The first, second, and fourth analyses are computed from training counts alone, and the third additionally reads next-token distributions from the TIGER checkpoint used for the main comparison; no model is trained for any of the four. Ranking metrics use the cutoffs $[1, 5, 10, 20, 50]$.

Pattern frequency concentration (Figure 1, Table 8). Patterns are formed from the three leading SID levels, excluding the collision suffix. For every item occurrence in a training sequence we count two hierarchy-specific adjacent bigrams, (c_1, c_2) and (c_2, c_3), and for every adjacent pair of items we count one inter-item transition whose source is (c_1, c_2) of the preceding item and whose target is (c_1, c_2, c_3) of the next item, so counts accumulate over item occurrences in the training sequences. For a family with T distinct pattern types we sort the types by descending count and report the fraction of all occurrences covered by the $\lceil fT \rceil$ most frequent types for $f \in \{0.1\%, 1\%, 5\%, 10\%\}$; a uniform distribution over types assigns coverage f. The main text illustrates this analysis on Toys, and Table 8 reports all three datasets.

Strict transition lookup (Figure 2). The lookup is a token-autoregressive frequency model with no learned parameters. Its source key is the complete three-code SID of the latest visible history item, and the next item's SID is generated one token at a time, where the score of a candidate token is the number of training transitions in which that token follows both the source key and the target tokens already generated. We use beam width 50 and no additive smoothing, and we retain only hypotheses whose generated prefix is a prefix of at least one catalog SID. Ties are broken by higher conditional count, then lower token ID, then lower SID tuple. The generated SID is resolved to a catalog item and ranked against all catalog items.

Repeated-transition utilization (Figure 3, Table 9). For every adjacent pair of items in a test sequence we take the source prefix (c_1, c_2) of the preceding item and the target prefix (c_1, c_2) of the following item, and we group the pair by the number of times that exact source–target transition occurs in the training split, using the bins 0, 1, 2–4, 5–9, 10–49, and 50+. The model quantity is a teacher-forced joint top-1 accuracy: the ground-truth context up to and including the preceding item is given, no beam search is used, and the pair counts as correct only if the argmax of the first-level distribution and the argmax of the second-level distribution both equal the target codes. The lookup quantity uses training statistics only: candidates are the target prefixes observed after the same source in training, ordered by training count with lexicographic tie-breaking, and the pair counts as correct if the highest-ranked candidate equals the target prefix. Appendix Table 9 reports the per-bin values and bin sizes for all three datasets.

Conditional ambiguity (Figure 4). Using training transitions only, we estimate the conditional distribution of the next target given the complete three-code SID of the preceding item, and we consider four target granularities: the first code c_1, the prefix (c_1, c_2), the three leading codes (c_1, c_2, c_3), and the exact catalog item. Each source pattern defines a distribution over targets obtained by normalizing its training counts, and we average over source patterns with occurrence weighting, so a source pattern contributes in proportion to the number of training transitions that start from it. Figure 4 reports the probability mass of the most likely target (top-1), the five most likely targets (top-5), and the ten most likely targets (top-10).

H.2 PATTERN FREQUENCY CONCENTRATION ACROSS DATASETS

Figure 8 repeats the frequency-concentration analysis of Figure 1 on all three training splits, and Table 8 lists the coverage values. The qualitative profile is the same everywhere: intra-item bigrams are more concentrated than inter-item transitions, and both are far from the uniform reference. The most frequent 1% of pattern types cover 11.31%, 12.84%, and 12.91% of intra-item bigram occurrences on Toys, Beauty, and Sports, and 5.35%, 8.36%, and 4.55% of inter-item transitions.

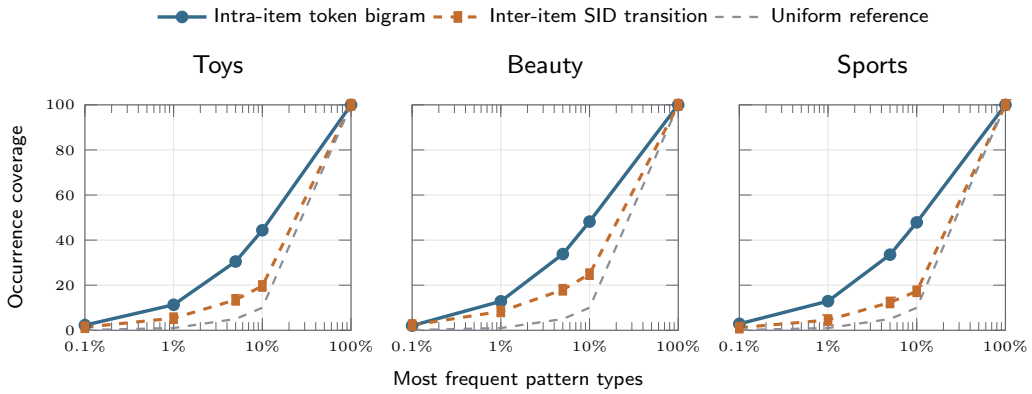

Figure 8: Occurrence coverage of frequency-ranked SID patterns on all three training splits. Curves follow the definition of Figure 1; the dashed gray line is the uniform reference.

Table 8: Cumulative occurrence coverage of frequency-ranked SID patterns in each training split, with occurrence (interaction) weighting. "Types" is the number of distinct pattern types.

Dataset	Pattern family	Types	Occurrences	@0.1%	@1%	@5%	@10%
Toys	Intra-item bigram	12,255	257,546	2.28	11.31	30.50	44.37
	Inter-item transition	97,641	109,361	1.37	5.35	13.50	19.65
Beauty	Intra-item bigram	13,163	307,552	2.06	12.84	33.81	48.21
	Inter-item transition	109,644	131,413	2.30	8.36	17.96	24.91
Sports	Intra-item bigram	16,140	450,282	2.86	12.91	33.54	47.87
	Inter-item transition	174,150	189,543	1.16	4.55	12.34	17.31

H.3 Repeated-Transition Utilization Across Datasets

Figure 9 repeats the repeated-transition analysis of Figure 3 on all three datasets, and Table 9 lists the per-bin values together with the number of test transitions in each bin. The qualitative pattern is the same everywhere: the training-frequency lookup is weak for transitions seen once or not at all, overtakes TIGER once a transition occurs at least twice, and the gap widens with frequency. The magnitude of the gap differs by dataset.

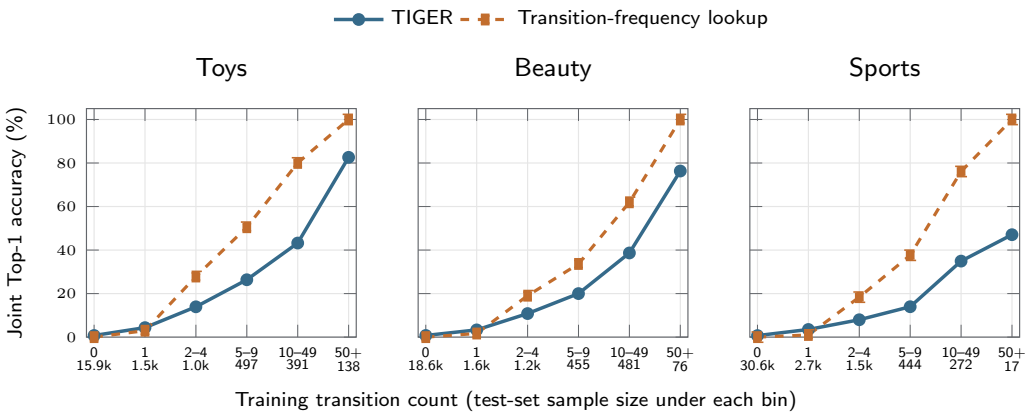

Figure 9: Joint top-1 accuracy of TIGER and of the source-conditioned transition-frequency lookup, grouped by the training transition count. Each panel repeats the analysis of Figure 3 on one dataset; the number under each bin is the number of test transitions in that bin.

Table 9: Joint top-1 accuracy (%) of TIGER and of the source-conditioned transition-frequency lookup by training transition count. n is the number of test transitions in the bin.

Dataset	Method	0	1	2–4	5–9	10–49	50+
Toys	n	15,860	1,492	1,034	497	391	138
	TIGER	0.75	4.36	13.93	26.36	43.22	82.61
	Lookup	0.00	2.95	27.85	50.50	80.05	100.00
Beauty	n	18,568	1,628	1,155	455	481	76
	TIGER	0.76	3.32	10.82	20.00	38.67	76.32
	Lookup	0.00	1.66	19.05	33.63	61.95	100.00
Sports	n	30,618	2,706	1,541	444	272	17
	TIGER	0.70	3.51	7.98	13.96	34.93	47.06
	Lookup	0.00	1.00	18.36	37.61	76.10	100.00