RecGPT-V3 Technical Report

RecGPT Team

Large language models (LLMs) are transforming recommender systems from matching co-occurrence patterns in historical behavior toward reasoning about the intent that drives it. `RecGPT-V1` pioneered this paradigm on Taobao by placing a genuine understanding of the user at the center of the pipeline, and `RecGPT-V2` scaled it through coordinated multi-agent reasoning; both have been deployed in production and yielded consistent gains in user experience and commercial interests. Yet operating the `RecGPT` series at scale exposes three fundamental challenges: (1) *stateless behavior modeling*, where each request reprocesses a user's full history from scratch, incurring redundant computation and discarding prior analysis; (2) a *tag-to-item information bottleneck*, where natural-language tags form a lossy information channel between user understanding and item grounding; and (3) *inefficient explicit reasoning*, whose lengthy chain-of-thought imposes untenable latency and computational overhead.

To address these challenges, we present `RecGPT-V3`, a stateful, hybrid-modal recommender that efficiently reasons over natural language for open-world knowledge and Semantic IDs (SIDs) for concrete item grounding. As the system engine, a *Memory Hub* maintains a structured, continually evolving user memory that distills long-horizon behavior into condensed memory units, reducing user-modeling computation by 55.8%. To bridge reasoning and retrieval, a *Hybrid-modal Foundation Model* enables the LLM to reason jointly over text-based tags and SIDs, opening a high-bandwidth channel into the item space. To render reasoning tractable under latency and compute budgets, *Latent Intent Reasoning* internalizes verbose rationales into compact learnable latent tokens that remain decodable into readable explanations, lowering output token cost by 200×. Deployed in Taobao's "Guess What You Like" feed, `RecGPT-V3` achieves consistent gains in large-scale online A/B tests, improving IPV by +1.28%, CTR by +1.00%, TC by +1.97%, and GMV by +3.97% while substantially cutting end-to-end serving resource consumption by 52.4%. These results show that LLMs can serve as the intelligent brain of industrial-scale business systems, advancing recommendation accuracy and serving efficiency in tandem.

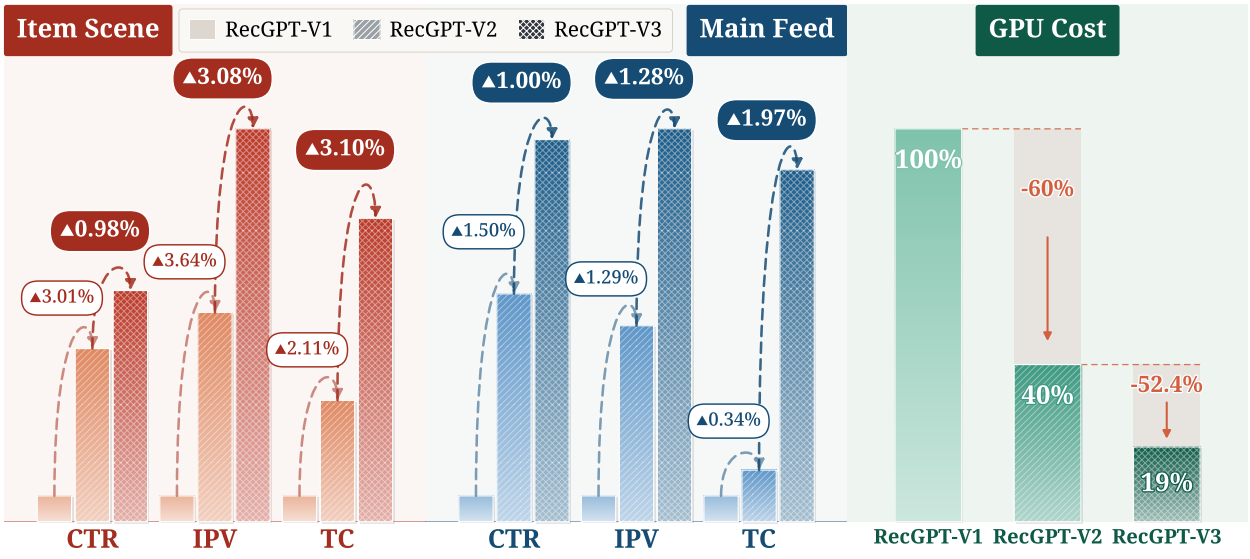

Figure 1 | Performance and efficiency of the `RecGPT` series on Taobao. Each generation brings further gains on CTR, IPV, TC, and GMV in both the Item Scene and Main Feed, while GPU cost drops steadily: `RecGPT-V3` uses only 19% of `RecGPT-V1`'s compute, a 52.4% reduction over `RecGPT-V2`.

Contents

1 Introduction **3**

2 Memory Hub **5**

2.1 Structured Behavior Compression 5

2.2 Evolving Memory Curation 6

3 Hybrid-Modal Recommendation Foundation Model **8**

3.1 Hybrid Tokenization 9

3.2 Pre-training 10

4 Latent Intent Reasoning **12**

4.1 Reasoning Internalization 13

4.2 Post-training 14

5 Experiments **16**

5.1 Online A/B Test Results 16

5.2 Memory Hub Evaluation 17

5.3 Foundation Model Evaluation 18

5.4 Latent Reasoning Evaluation 20

5.5 Further Analysis 21

6 Conclusion **24**

References **24**

A Contributors **27**

B Intent-to-Item Retrieval **27**

B.1 Architecture 27

B.2 Training Recipe 28

C Latent Reasoning Reconstruction **29**

1. Introduction

An ideal recommender system should grasp what a user actually wants and surface the items that meet that underlying need. In practice, however, the models that power today's platforms largely reduce this goal to predicting the next interaction from statistical regularities in historical logs. From matrix factorization (Koren et al., 2009; Rendle et al., 2009) to deep sequential networks (Kang and McAuley, 2018; Tang et al., 2026), these models have grown increasingly expressive, yet they share a common limitation: they match co-occurrence patterns in past behavior, capturing which items appear together but not the intent behind them. Confined to such correlational signals, they amplify what a user has already revealed, narrowing exposure (Chen et al., 2023), reinforcing filter bubbles (Nguyen et al., 2014), and underserving long-tail content (Gao et al., 2023), which gradually erodes user experience and the health of the recommendation ecosystem.

The emergence of large language models (LLMs) opens a promising path beyond this limitation. By drawing on broad world knowledge and deliberative reasoning (Guo et al., 2025; Zhao et al., 2023), LLMs can interpret user behavior in context and infer the motivation and potential interest behind each action, rather than extrapolating co-occurrence alone. This capability recasts recommendation as a problem of reasoning about user intent, a direction we have pursued across our work. **RecGPT-V1** (Yi et al., 2025b) rebuilt the industrial pipeline around this principle, delegating interest mining, item retrieval, and explanation generation to the LLM so that recommendations follow from understanding why a user acts, not from fitting what they did before. Building on this foundation, **RecGPT-V2** (Yi et al., 2025a) coordinated a planner and specialized multi-expert sub-agents to analyze user intent collaboratively, broadening its coverage while sharply lowering cost. Both systems have been deployed at scale on Taobao, yielding consistent gains for users, merchants, and the platform and showing that LLM-driven user modeling is promising and practical in production.

Despite these gains, operating the **RecGPT** series at scale has exposed three key challenges that constrain further progress.

C1 **Stateless behavior modeling.** Most existing LLM-based recommenders (Team et al., 2026; Wang et al., 2026; Zhou et al., 2025) reason over a user's entire behavior history in a one-shot manner. As interactions accumulate, each request reprocesses the full history from scratch, without reusing any earlier analysis. This stateless design incurs redundant computation that grows with the behavior sequence and, more fundamentally, prevents the system from carrying forward what it has already learned about the user.

C2 **Tag-to-Item information bottleneck.** Both **RecGPT-V1** and **RecGPT-V2** have the LLM predict natural-language tags of the items a user is likely to interact with next, which a downstream model then uses to ground concrete items. This textual interface forms a lossy channel between reasoning and retrieval: a coarse tag (e.g., "*rotating adjustable badminton racket stand*") corresponds to a broad and heterogeneous set of candidate items and cannot convey the fine-grained, ID-level evidence that the item space relies on, leaving a unbridgeable information gap between the intent the LLM predicts and the items ultimately retrieved.

C3 **Inefficient explicit reasoning.** Explicit chain-of-thought helps the LLM understand users and uncover their latent interests (Li et al., 2026; Liu et al., 2025; Zhang et al., 2026), but autoregressively generating lengthy rationales (averaging $\sim 3,000$ tokens in our task) incurs considerable inference latency. At billion-user scale and under high QPS, this cost is prohibitive, naturally raising a central question: how to cut latency while retaining the explainable reasoning ability that the **RecGPT** series has upheld as a core design philosophy.

To address these challenges, we improve **RecGPT-V3** along three dimensions, namely user model-

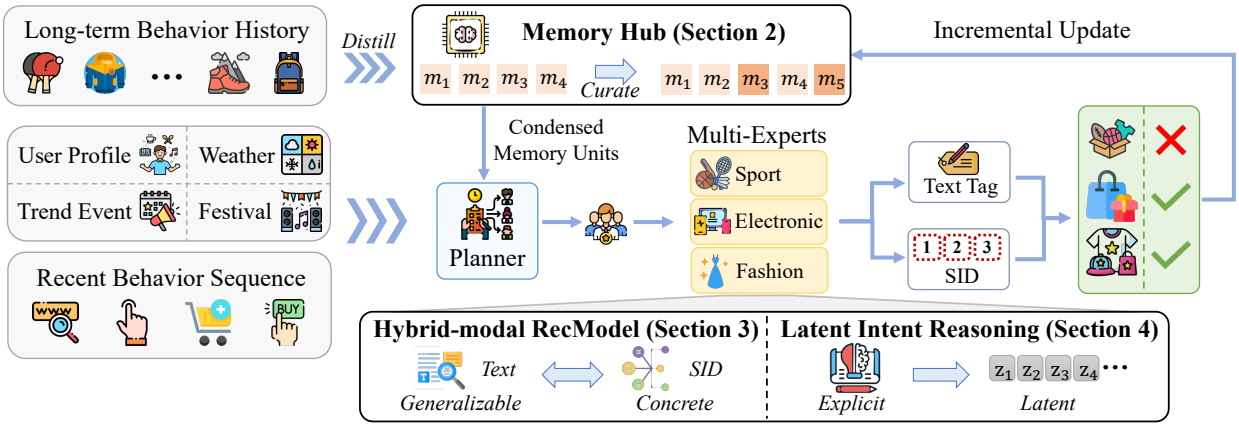

Figure 2 | Overview of `RecGPT-V3`. A *Memory Hub* distills a user's long-term behavior into incrementally curated memory units; conditioned on these units and recent behaviors, a planner and its multi-expert modules reason with the *Hybrid-modal Foundation Model* (natural language and Semantic IDs) and *Latent Intent Reasoning* (compact latent tokens) to predict the next item to interact with.

ing, intent representation, and intent reasoning, with the goal of jointly improving recommendation accuracy and efficiency. Figure 2 illustrates the overall `RecGPT-V3` framework, which embodies the following three key technical innovations.

S1 **Memory Hub** (§2). We move from **stateless, one-shot user modeling** to a **stateful, memory-augmented recommender** that continually learns each user's personalized preferences, placing a structured user memory at the center of the pipeline as its engine. This memory distills each user's long-horizon behavior history into succinct memory units, compressing tokens by **94.5%** (***condensed***), links every unit back to its originating behaviors for provenance (***traceable***), and curates them incrementally as new interactions arrive (***evolvable***), so that understanding of the user accrues over time, no longer rediscovered from scratch. The model thus draws on the consolidated memory together with a recent behavioral increment instead of the raw full history, reducing user-modeling computation by **55.8%** while preserving essential behavioral contexts.

S2 **Hybrid-modal Recommendation Foundation Model** (§3). To overcome the **Tag-to-Item** bottleneck, we equip the LLM to reason in ***Semantic IDs (SIDs)***, discrete codes that encode item semantics, as a second modality alongside natural language. The two modalities serve complementary purposes: natural language expresses intent through rich, open-ended world knowledge (***generalizable***), while SIDs ground that intent in semantic item representations enriched by collaborative signals (***concrete***). The model therefore generates intent representations natively compatible with the downstream model, opening a high-bandwidth channel from intent reasoning to product retrieval that closes the gap left by the language-only interface.

S3 **Latent Intent Reasoning** (§4). We internalize **lengthy explicit chain-of-thought** into **compact learnable latent-tokens**, collapsing a verbose rationale into a few dense steps of latent deliberation. We introduce dedicated latent `<CoT>` slots that encode an entire reasoning trace, cutting its token cost by **200×** (***efficient***). Crucially, these latent tokens decode back into a readable rationale on demand (***explainable***), reconciling low-latency inference with the interpretability that production recommendation demands.

`RecGPT-V3` has been deployed on the "Guess What You Like" scenario of the Taobao homepage, one of the world's largest e-commerce platform surfaces serving hundreds of millions of daily active

users. Through large-scale online A/B testing against the live `RecGPT-V2` system, `RecGPT-V3` delivers strong gains across engagement and business metrics, including **+1.28%** IPV, **+1.00%** CTR, **+1.97%** TC, and **+3.97%** GMV, while simultaneously reducing end-to-end serving compute by **52.4%**. More broadly, `RecGPT-V3` shows that large language models can power the real-world industrial pipeline at low cost, returning recommendation application to its founding purpose: to faithfully understand and serve what every user truly needs.

2. Memory Hub

In the RecGPT-V2 pipeline, the Global Planner analyzes a user's behavioral history, decomposes it into intent personas, and dispatches them to multi-expert modules for item tag prediction and retrieval. Since it reprocesses the full behavioral sequence at every pass (~**55K** tokens for highly active users), the Planner alone accounts for ~**95%** of the computational cost in agentic intent analysis, while each pass redundantly rediscovers patterns such as repurchase cycles, brand loyalty, and seasonal shifts that could instead persist and update across sessions. To address this, we introduce the **Memory Hub**, a persistent, structured memory layer that turns the Planner's input from stateless full-sequence encoding into stateful memory-driven reasoning, achieving three properties simultaneously:

- **Condensed**: consolidates the full behavioral sequence into a compact set of schema-defined memory units, achieving **80%** token reduction while retaining only coherent, high-confidence behavioral patterns.
- **Traceable**: preserves provenance links from each memory unit to its source behaviors, enabling interpretability while maintaining downstream recommendation quality.
- **Evolvable**: incrementally incorporates new behaviors through selective updates and new pattern extraction, without full re-encoding.

The memory hub builds and maintains this memory through two mechanisms: **Structured Behavior Compression** (§2.1) performs an initial consolidation of each user's full historical behavior sequence into structured memory units, while **Evolving Memory Curation** (§2.2) keeps this memory current by periodically integrating newly observed behaviors. Together, they reduce overall Global Planner compute by **55.8%** while preserving recommendation quality. Figure 3 provides an overview of the memory hub architecture.

2.1. Structured Behavior Compression

Structured behavior compression consolidates the user's full behavioral sequence $\mathcal{B} = \{b_1, b_2, \ldots, b_N\}$ into a compact set of structured memory units $\mathcal{M} = \{m_1, m_2, \ldots, m_K\}$ with $K \ll N$. Formally, we define this process as an LLM-based memory construction function:

$$\mathcal{F}_\phi : \mathcal{B} \longrightarrow \mathcal{M}, \qquad \mathcal{M} = \mathcal{F}_\phi(\mathcal{B}) = \{m_1, m_2, \ldots, m_K\}, \tag{1}$$

It runs once as an initial pass, producing a persistent memory that serves as the foundation for subsequent incremental updates (§2.2). Each unit centers on two core fields that together characterize a behavioral pattern. The first, the **behavior pattern identifier** p_k, is a categorical label drawn from a predefined taxonomy of hundreds of patterns spanning the major e-commerce behavioral archetypes, with controlled extension permitted when no existing pattern captures a high-confidence cluster. The second, the **preference summary** s_k, describes in natural language the user's preferences, trends, and consumption characteristics within that pattern. Complementing these two fields, each unit further records supporting metadata: representative behavior indices for provenance, preferred brands, a temporal activity profile, and a timestamp. Table 1 shows a complete example.

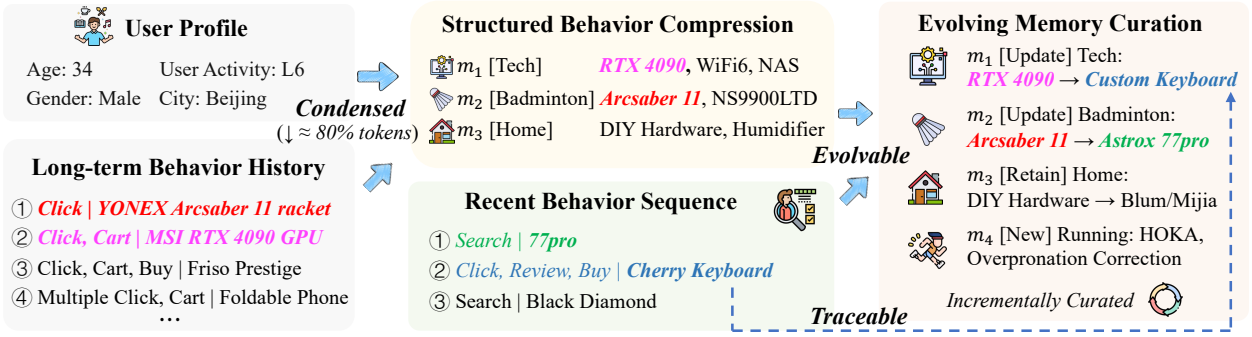

Figure 3 | Overview of the Memory Hub. Structured Behavior Compression distills a user's long-term behavioral history into a compact set of structured memory units, reducing token usage by 94.5%. Evolving Memory Curation then keeps this memory current by selectively updating existing units and extracting new patterns from unmatched behaviors, yielding a condensed, traceable, and evolvable user representation for downstream recommendation.

Table 1 | Example of a structured memory unit.

Field	Example
Behavior Pattern	K-pop Fandom
Preference Summary	Dedicated fan of a K-pop girl group. Full-lifecycle engagement from album pre-orders to merchandise collection and live concert participation. Strong focus on collection completeness and item preservation.
Representative Indices	549, 553, 558, 563, 564, 565, ...
Preferred Brands	SingBA, 时代良品, 珍琢, ...
Temporal Activity	First appeared March 2023; peaked June and Aug–Oct 2023.
Timestamp	Created: 2023-10-31

2.2. Evolving Memory Curation

Structured Behavior Compression (§2.1) establishes the initial memory state, but user interests drift over time as preferences emerge, decay, and shift with life stages and seasons. A static memory would progressively diverge from the user and erode recommendation relevance. **Evolving Memory Curation** counters this drift through periodic incremental updates that fold newly observed behaviors into the existing memory. The Global Planner then reasons over this compact memory together with the recent behavioral delta rather than re-encoding the full behavioral sequence at each pass, thereby substantially reducing user-modeling computation.

Specifically, let $\mathcal{M}^{(t)} = \{m_1^{(t)}, \ldots, m_K^{(t)}\}$ denote the memory state at time t produced by initial compression or a previous curation cycle, and let $\Delta\mathcal{B}^{(t,t+\delta)} = \{b_{N+1}, \ldots, b_{N+\Delta N}\}$ denote the newly accumulated behavioral interactions during the interval $[t, t+\delta)$. Evolving Memory Curation defines the incremental update function $\mathcal{G}$ as follows:

$$\mathcal{G} : \left(\mathcal{M}^{(t)},\ \Delta\mathcal{B}^{(t,t+\delta)}\right) \longrightarrow \left(\mathcal{M}^{\text{update}},\ \mathcal{M}^{\text{new}}\right), \tag{2}$$

where $\mathcal{M}^{\text{update}}$ denotes the set of existing memory units after selective update or retention, and $\mathcal{M}^{\text{new}}$ denotes the set of newly created memory units extracted from previously unmatched behaviors. The final curated memory is then obtained by merging these two parts. By design, $\mathcal{G}$ operates without

Table 2 | Incremental curation of memory list from $\mathcal{M}^{(t)}$ to $\mathcal{M}^{(t+\delta)}$. Row colors indicate operation type: Update (summary refreshed with new evidence), Retain (unit left unchanged for lack of relevant new behaviors), and New (unit created from behaviors unmatched by any existing unit).

Unit	Pattern (p_k)	Summary at t	Summary at $t+\delta$	Operation
m_1	Infant Care	Frequent buyer of formula, diapers, and baby clothing for 0–6 months. $\cdots$	$\cdots$ Shifted to **6–12 months**. Added **toddler toys and walkers**.	(1) Update
m_2	Outdoor Running	Running shoes, GPS watches, and marathon nutrition supplements. $\cdots$	(Unchanged)	(1) Retain
m_3	Home Cooking	Budget-friendly kitchen gadgets and baking supplies. $\cdots$	$\cdots$ Shifted toward **premium cookware and air fryer accessories**.	(1) Update
m_4	Photography	Mirrorless camera accessories and lens filters. $\cdots$	(Unchanged)	(1) Retain
m_5	*Pet Parenting (new)*	—	Pet food and grooming supplies. First-time pet owner.	(2) New

access to the full sequence $\mathcal{B}$, drawing only on the compressed memory $\mathcal{M}^{(t)}$ and the new delta $\Delta\mathcal{B}^{(t,t+\delta)}$.

(1) Selective Update of Existing Units. For each existing memory unit $m_k^{(t)}$, the model determines whether the new behavioral delta contains interactions relevant to this pattern:

$$m_k^{(t+\delta)} = \begin{cases} \text{Update}(m_k^{(t)},\ \Delta\mathcal{B}_k^{\text{rel}}), & \text{if } \Delta\mathcal{B}_k^{\text{rel}} \neq \emptyset, \\ m_k^{(t)}, & \text{otherwise,} \end{cases} \tag{3}$$

where $\Delta\mathcal{B}_k^{\text{rel}} \subseteq \Delta\mathcal{B}^{(t,t+\delta)}$ denotes the subset of new behaviors semantically related to $m_k^{(t)}$, and $\Delta\mathcal{B}^{\text{rel}} = \bigcup_{k=1}^{K} \Delta\mathcal{B}_k^{\text{rel}}$ denotes the union of all new behaviors assigned to existing memory units. When triggered, the Update operation refreshes the unit's summary s_k and all supporting metadata; otherwise, the unit is retained without modification. This selective policy follows the principle of ***updating with evidence and retaining without disturbance***. The resulting updated-or-retained memory set is:

$$\mathcal{M}^{\text{update}} = \{\, m_k^{(t+\delta)} \,\}_{k=1}^{K}. \tag{4}$$

(2) New Pattern Extraction. The interactions that the selective update leaves unassigned to any existing unit form the novel behavior set:

$$\Delta\mathcal{B}^{\text{novel}} = \Delta\mathcal{B}^{(t,t+\delta)} \setminus \Delta\mathcal{B}^{\text{rel}}. \tag{5}$$

New pattern extraction is performed within the same curation function $\mathcal{G}$ in Eq. 2. Given the previous memory $\mathcal{M}^{(t)}$ and the new behavioral delta $\Delta\mathcal{B}^{(t,t+\delta)}$, $\mathcal{G}$ identifies coherent unmatched behaviors and writes them into new memory units:

$$\mathcal{M}^{\text{new}} = \text{Extract}\left(\Delta\mathcal{B}^{\text{novel}}\right). \tag{6}$$

If no coherent new pattern is identified, $\mathcal{M}^{\text{new}} = \emptyset$.

The final memory combines the existing units retained or updated by the selective update with these newly extracted ones:

$$\mathcal{M}^{(t+\delta)} = \underbrace{\mathcal{M}^{\text{update}}}_{\text{updated or retained}} \cup \underbrace{\mathcal{M}^{\text{new}}}_{\text{newly extracted}} . \tag{7}$$

In practice, the two operations share a single model forward pass that jointly updates the existing units and extracts the new ones, sparing the pipeline from multiple sequential calls.

Semantic Continuity. Each selective update re-synthesizes the affected unit into a coherent snapshot of the user's *current* state rather than appending new behaviors to the old summary, preventing stale and emerging preferences from accumulating within one unit. As Table 2 shows, m_1 shifts from "0–6 months" to "6–12 months" and m_3 toward premium cookware as interests mature, while unaffected units (m_2, m_4) remain unchanged and a first-time interest forms the new unit m_5.

To summarize, the memory hub transforms each user's raw behavioral sequence into a structured, continually evolving memory that serves as the engine of the recommendation pipeline. Instead of re-deriving user understanding from the full interaction history at every request, the system reasons over this compact, high-signal memory together with a recent behavioral delta, so that its understanding of the user accrues and refines over time. In production, incremental curation runs every two months, reducing overall user-modeling computation by **55.8%**. Detailed efficiency analysis and quality validation are presented in §5.2.

3. Hybrid-Modal Recommendation Foundation Model

Text-only LLM experts interface with the discrete item space through a single natural-language channel: they predict free-form item tags that a downstream retriever grounds to in-domain items. This indirection makes the tag a **lossy bottleneck** between reasoning and retrieval: a coarse tag such as "lightweight running shoes for marathon training" matches a broad, heterogeneous set of items and cannot convey the fine-grained, item-level evidence that retrieval depends on, loosening the coupling between the intent the model expresses and the items ultimately surfaced.

To overcome this bottleneck, we propose a **Hybrid-Modal Recommendation Foundation Model** that extends the Qwen3-14B (Yang et al., 2025) backbone with ***Semantic IDs (SIDs)***—discrete codes that encode item semantics—as a second modality alongside natural language. Recommendation inherently requires reasoning about user intent while simultaneously grounding it in concrete items, a dual demand that no single representation satisfies in isolation. The two modalities are thus complementary, each addressing one facet of this requirement:

- ***Generalizable***: natural language expresses intent through rich, open-ended world knowledge.
- ***Concrete***: SIDs ground that intent in item representations enriched by collaborative signals.

By incorporating both modalities into a unified vocabulary, the model emits retrieval-compatible identifiers rather than coarse tags, thereby opening a high-bandwidth channel into the item space while preserving its full natural-language capability. Figure 4 illustrates the overall pipeline, comprising two components: a hybrid tokenization scheme that constructs semantically grounded SIDs (§3.1), followed by a two-stage training procedure that grounds these tokens in language and equips the model to reason jointly over SIDs and text (§3.2).

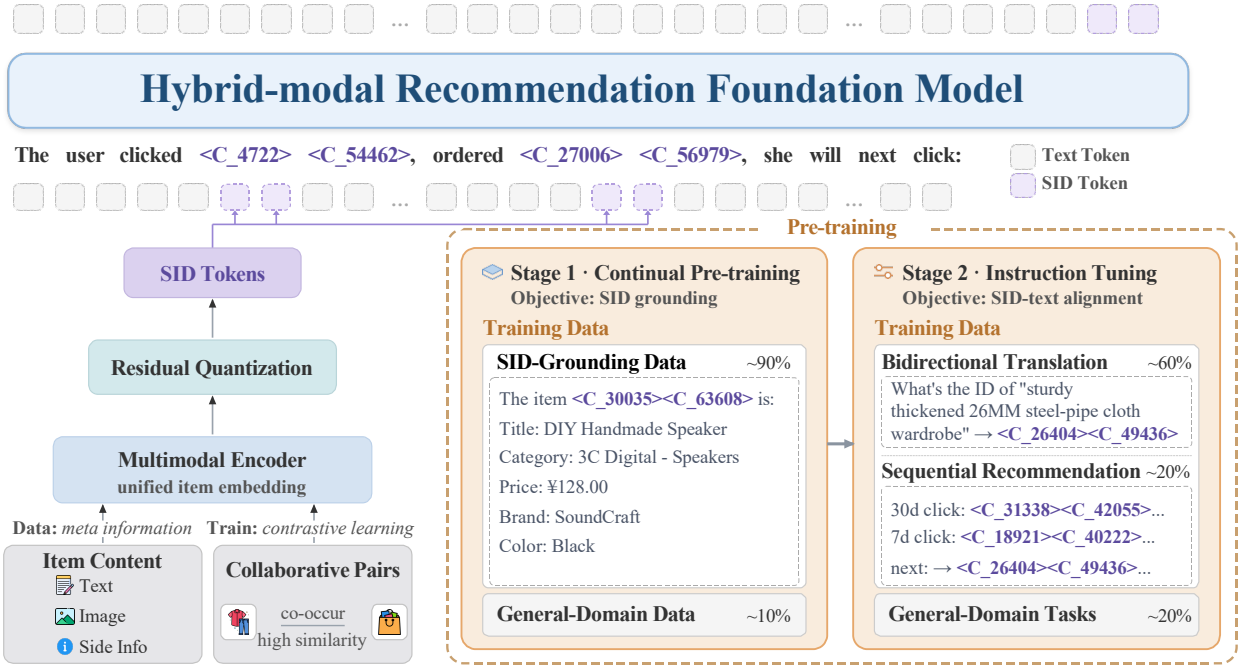

Figure 4 | Overview of the Hybrid-modal Foundation Model. Multimodal item features are quantized into 65,536 SID tokens via CN-CLIP and a two-level RQ-VAE, extending the Qwen3-14B vocabulary. Continual pre-training (Stage 1) and instruction tuning (Stage 2) then align these tokens with language, with general-domain data mixed into both stages.

3.1. Hybrid Tokenization

Hybrid tokenization augments the LLM's native text vocabulary with Semantic ID tokens derived from item content, so that semantically similar items receive similar codes. Constructing these SIDs proceeds in two stages: we first learn a discriminative multimodal item representation via contrastive learning, then quantize it into discrete hierarchical codes via Residual Quantization (RQ-VAE) (Chen et al., 2026; Fu et al., 2026; Lee et al., 2022). We detail each below.

Multimodal Item Representation. The goal of this stage is to obtain a unified, discriminative embedding for each item by fusing its text, image, and side information. As no explicit labels define item similarity, we mine positive pairs directly from behavior: items that frequently co-occur across the behavioral logs form candidate pairs, from which we discard those whose multimodal similarity is low, guarding against spurious co-occurrences. The surviving pairs supervise contrastive learning.

Specifically, for each item i, we extract its text description I^i_{text}, image information I^i_{image}, and auxiliary side information I^i_{side} (*e.g.*, item attributes and metadata). We employ CN-CLIP (Yang et al., 2022), denoted $\mathcal{E}$, to encode each modality, with the input determining which tower is invoked: $\mathcal{H}^i_{\text{text}} = \mathcal{E}(I^i_{\text{text}})$ and $\mathcal{H}^i_{\text{image}} = \mathcal{E}(I^i_{\text{image}})$, while side information is first converted to textual form and then encoded, yielding $\mathcal{H}^i_{\text{side}} = \mathcal{E}(I^i_{\text{side}})$. These modality-specific features are fused through Q-Former (Li et al., 2023), denoted $\mathcal{F}$, to produce the unified item representation:

$$\mathcal{H}^i = \mathcal{F}\left(\mathcal{E}(I^i_{\text{text}}),\ \mathcal{E}(I^i_{\text{image}}),\ \mathcal{E}(I^i_{\text{side}})\right) \tag{8}$$

For training, we use item i^+ as a positive sample and other items within the same batch as negatives

i^-, optimizing an InfoNCE-based loss that operates at both the fused and modality-specific levels:

$$\mathcal{L}_{\text{InfoNCE}} = f(\mathcal{H}^i, \mathcal{H}^{i^+}, \mathcal{H}^{i^-}) + f(\mathcal{H}^i_{\text{text}}, \mathcal{H}^{i^+}_{\text{text}}, \mathcal{H}^{i^-}_{\text{text}}) + f(\mathcal{H}^i_{\text{image}}, \mathcal{H}^{i^+}_{\text{image}}, \mathcal{H}^{i^-}_{\text{image}}) \tag{9}$$

where f denotes the InfoNCE loss function (Radford et al., 2021). This contrastive objective is what makes the subsequent quantization stable: the well-separated embedding space it produces prevents the codebook collapse to which RQ-VAE is otherwise prone.

Residual Quantization. Given the multimodal item representation $\mathcal{H}^i$, we apply RQ-VAE to map each item to a sequence of discrete codes $\{c_1, c_2, \ldots, c_L\}$, which constitute its Semantic ID. In our final configuration, we employ a two-level codebook structure where each level has a vocabulary size of 32,768. This results in a total of 65,536 new tokens (denoted as `<C_0>` through `<C_65535>`) being added to the model's vocabulary. A complete SID for an item consists of two codebook entries, e.g., `<C_18921><C_40222>`, where the first token represents the coarse-grained semantic cluster and the second provides fine-grained discrimination within that cluster. This two-level design keeps the added vocabulary size manageable while preserving fine-grained item distinctions: because related items share their coarse code, semantic similarity in the item space surfaces as shared prefixes in SID space, a structure the model can exploit to generalize across related items.

3.2. Pre-training

The new SID tokens are appended to the original Qwen3-14B vocabulary, with their embeddings randomly initialized and learned during training. To teach the model to understand and generate this hybrid vocabulary, we adopt a two-stage pipeline: **Continual Pre-Training** (§3.2.1), which grounds the SID tokens in item semantics, followed by **Instruction Tuning** (§3.2.2), which teaches the model to apply them across diverse recommendation tasks. A central concern throughout is injecting domain knowledge without eroding the model's pre-existing general capabilities, a balance we strike through careful data construction and strategic mixing.

3.2.1. Continual Pre-training

Continual pre-training pursues two objectives: ❶ grounding the newly added SID tokens in natural-language space so the model can understand and generate them in context, and ❷ retaining the backbone's pre-existing general capabilities. Our corpus therefore combines two complementary components: **SID-grounding data**, which associates each SID with the content of the item it denotes, teaching the model to read item semantics directly from these tokens, and **general-domain data**, which guards against catastrophic forgetting.

SID-Grounding Data. We construct each SID-grounding sample as a natural-language statement that pairs an item's Semantic ID with its textual attributes, such as title and category, so the model learns to recover an item's content from its SID alone. For example, the following sample associates the SID `<C_18921><C_40222>` with the item's title and category:

Example: SID-Grounding Sample

The item with Semantic ID `<C_18921>` `<C_40222>` has the following information:

Title: Insulated Lock Rod Insulated Construction Scaffolding Electrified Insulation High Voltage Electric Power Frame.

Category: Scaffolding.

Table 3 | SID-related alignment tasks and their input–output mappings. The upper block covers SID–text bidirectional translation; the bottom row is SID-based sequential recommendation.

Task	Mapping	Description	Proportion
sid2title	sid → title	Generate the item title given an SID.	13.8%
title2sid	title → sid	Predict the SID given an item title.	14.7%
sid2tag	sid → tag	Generate a natural-language tag given an SID.	11.5%
tag2sid	tag → sid	Predict candidate SIDs given a text tag.	6.2%
sid2cmd	sid → cmd	Predict the commodity category given an SID.	13.8%
sid2sid	sid → sid	Predict next-click SIDs given a user's click history.	20.0%

This stage gives the model a working semantics for the SID tokens, so that instruction tuning can operate on identifiers it already understands rather than opaque symbols.

General-Domain Data. Training predominantly on SID-grounding data risks catastrophic forgetting of the general-purpose competence that downstream deployment relies on. We therefore blend approximately **10% general-domain text** into the corpus, spanning mathematical reasoning, code, scientific literature, medical text, and instruction-following data.

3.2.2. Instruction Tuning

Following continual pre-training, we perform instruction tuning to teach the model to apply SID tokens across the diverse alignment tasks. We organize these tasks into three categories: ❶ bidirectional translation between SIDs and text, ❷ sequential recommendation carried out directly in SID space, and ❸ general-domain tasks that preserve the model's pre-existing capabilities. Table 3 summarizes the SID-related tasks of the first two categories. In the following, we detail each task type in detail.

SID-Text Bidirectional Translation. This family maps SIDs to and from their textual descriptions, linking item titles (title ↔ sid), search queries (tag ↔ sid), and commodities (sid → cmd). Covering both directions teaches the model to read an item's semantics from its SID and, conversely, to emit the right SID for a given description.

Sequential Recommendation. The sid → sid task casts sequential recommendation fully within the SID space. Given a user's click history organized by multi-granularity temporal windows (*e.g.*, 3-day, 2-day, 1-day, 12-hour, 1-hour), the model predicts the SIDs the user will click next. Operating purely on SID tokens with no textual item descriptions, it provides the most direct evidence that the model has internalized collaborative-filtering signals through these tokens.

General-Domain Tasks. To keep the model's general text abilities intact, roughly **20% of the instruction tuning data** consists of open-domain supervised tasks such as natural language reasoning, entity extraction, and e-commerce-related tasks. This share is calibrated to preserve instruction-following fidelity, particularly producing well-formed JSON outputs and complying with complex multi-constraint prompts, without diluting the recommendation-specific training signal.

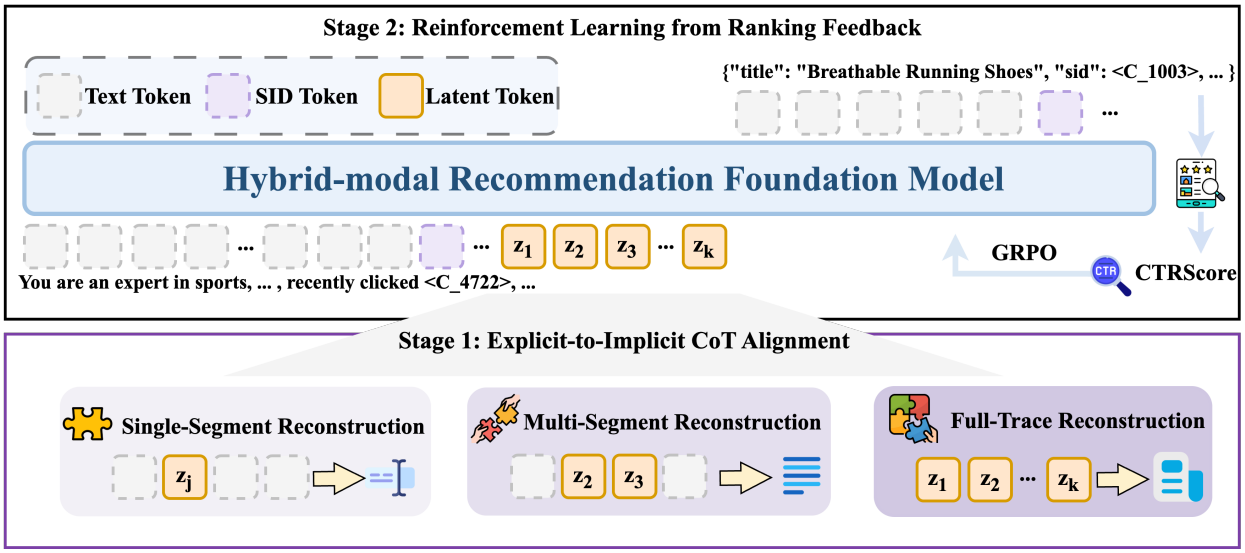

Figure 5 | Overview of Latent Intent Reasoning. Three reconstruction-based alignment tasks compress an explicit chain-of-thought trace into a short sequence of learnable latent tokens z that faithfully encode it. A two-stage post-training pipeline then internalizes this reasoning into the model: *Explicit-to-Implicit CoT Alignment* first distills teacher traces into the latent tokens, and *Reinforcement Learning from Ranking Feedback* then refines the policy against online business rewards.

4. Latent Intent Reasoning

Explicit chain-of-thought reasoning has become a common recipe for LLM-based recommenders to interpret user behavior and surface implicit interests, effectively improving prediction accuracy through multi-step deliberation. Yet this quality comes at a steep price: autoregressively generating a rationale of thousands of tokens, far longer than the final output, incurs latency and compute costs that become prohibitive at billion-user scale and high QPS. This raises a natural question: ***can the gains of deliberative reasoning be retained without paying its token cost?***

We answer this question with **Latent Intent Reasoning**, which internalizes the verbose trace into a short sequence of learnable latent tokens with two properties:

- ***Efficient***: compressing a rationale of thousands of sequentially decoded steps into a handful of latent tokens shifts the cost from slow autoregressive decoding to parallelizable prefill, reducing reasoning token cost by 200× and sharply lowering inference latency.
- ***Explainable***: unlike opaque latent representations, these tokens can be decoded back into a faithful, human-readable rationale whenever needed, preserving the interpretability that latent reasoning typically sacrifices for speed.

To endow these supervision-free tokens with reasoning semantics, we first formulate the latent reasoning mechanism: a compression scheme that condenses a lengthy explicit trace into a few latent tokens, together with multi-task alignment objectives that supervise these tokens to faithfully encode the reasoning they replace (§4.1). We then realize this internalization through a complete two-stage post-training pipeline that distills reasoning from a strong teacher and refines it with reinforcement learning driven by online feedback (§4.2). Figure 5 provides an overview.

Table 4 | The three alignment tasks on a running example (a badminton-equipment expert), each a choice of the masked set $\mathcal{J}$. Latent tokens `<cot>` replace the segments they encode, and the model reconstructs the masked segments $R_{\mathcal{J}}$ from the surrounding context. The gray anchors x and y mark the input and output, whose specific content is omitted here.

Task	**Input: context** $c(\mathcal{J})$	**Target: masked segments** $R_{\mathcal{J}}$
Single-Segment $\mathcal{J} = \{2\}$	x *Serious badminton player who favors full-carbon rackets and premium grips.* `<cot2>` *Keep the 10 most specific tags and drop broad category terms.* y	`<cot2>` : *From recent clicks, enumerate rackets, strings, grips, and court shoes.*
Multi-Segment $\mathcal{J} = \{1, 3\}$	x `<cot1>` *From recent clicks, enumerate rackets, strings, grips, and court shoes.* `<cot3>` y	`<cot1>` : *Serious badminton player who favors full-carbon rackets and premium grips.* `<cot3>` : *Keep the 10 most specific tags and drop broad category terms.*
Full-Trace $\mathcal{J} = \{1, 2, 3\}$	x `<cot1>` `<cot2>` `<cot3>` y	*Serious badminton player who favors full-carbon rackets and premium grips. From recent clicks, enumerate rackets, strings, grips, and court shoes. Keep the 10 most specific tags and drop broad category terms.*

4.1. Reasoning Internalization

The reasoning model takes as input x, the *persona* assigned by the Global Planner together with the user's recent click history, in which each clicked item is represented by its Semantic ID and short title. Conditioned on x, an explicit-CoT model (*e.g.*, DeepSeek-V3.2 (Liu et al., 2024) in our experiments) first generates a reasoning trace R of N newline-delimited steps and then the output item tags y, factorizing the joint distribution as

$$p_\theta(\boldsymbol{R}, y \mid x) = p_\theta(\boldsymbol{R} \mid x)\; p_\theta(y \mid x, \boldsymbol{R}). \tag{10}$$

Yet the trace R that carries these reasoning gains is also where the cost concentrates: its N steps are decoded autoregressively and typically far exceed the output y in length, making R the dominant efficiency bottleneck. We therefore internalize R into a short learnable latent slots $z = (z_1, \ldots, z_K)$ with $K \ll N$, introduced as new vocabulary tokens whose embeddings are randomly initialized. This replaces the explicit factorization in Eq. (10) with its latent counterpart,

$$p_\theta(\boldsymbol{z}, y \mid x) = p_\theta(\boldsymbol{z} \mid x)\; p_\theta(y \mid x, \boldsymbol{z}). \tag{11}$$

It remains to specify how z relates to the implicit derivation trace. We adopt a deterministic positional partition: the trace R is split into K contiguous, non-overlapping segments $R_1, R_2, \ldots, R_K$ of at most C steps each, with latent token z_j encoding segment R_j. The number of latent tokens follows:

$$K = \min\left(\left\lceil \frac{N}{C} \right\rceil,\; K_{\max}\right) \tag{12}$$

Here C sets the granularity of each latent token, with a larger C folding more reasoning steps into a single token, while $K_{\max}$ limits the sequence length and thereby bounds inference cost. When the trace spans fewer than $C \cdot K_{\max}$ steps, the trailing latent tokens cover shorter or empty segments; when it exceeds this budget, coverage stops at the first $K_{\max}$ segments.

Latent Token Warm-up. The randomly initialized latent embeddings lag far behind the pretrained token embeddings: they lie outside the pretrained embedding distribution and carry no semantics yet. To calibrate them into a representation space compatible with the language tokens, we randomly

select one or more contiguous spans of the trace R, replace each with latent tokens to form a mixed trace $\tilde{R}$, and optimize the standard autoregressive objective over the mixed sequence $w = (x, \tilde{R}, y)$,

$$\mathcal{L}_{\text{warm}} = -\sum_{t \in \mathcal{T}} \log p_\theta(w_t \mid w_{<t}), \tag{13}$$

where $\mathcal{T}$ indexes the surviving text tokens and the latent positions contribute only as context. Predicting each masked span from the surrounding text pulls the latent embeddings into the pretrained representation space and endows them with coarse contextual semantics, yielding a well-conditioned initialization for the subsequent segment-level alignment.

Multi-granularity Alignment. The warm-up stage calibrates the latent embeddings at the representation level, but it does not determine what information each latent slot should preserve. The positional partition assigns segment R_j to latent token z_j only as a structural correspondence; this correspondence must be converted into a learnable information constraint. We instantiate this constraint through masked reconstruction at multiple granularities. Given a masked index set $\mathcal{J} \subseteq \{1, \ldots, K\}$, we form a mixed sequence that keeps unmasked segments as text while replacing each masked one with its latent token, which is formulated as follows:

$$c(\mathcal{J}) = \big(x,\ e_1, \ldots, e_K,\ y\big), \qquad e_j = \begin{cases} z_j, & j \in \mathcal{J}, \\ R_j, & j \notin \mathcal{J}, \end{cases} \tag{14}$$

and, cued by a task-specific instruction prompt $\mathcal{P}$ that indicates which positions to recover, train the model to reconstruct the masked segments $R_{\mathcal{J}} = \{R_j\}_{j \in \mathcal{J}}$ from this context:

$$\mathcal{L}(\mathcal{J}) = -\log p_\theta\big(R_{\mathcal{J}} \mid c(\mathcal{J}),\ \mathcal{P}\big). \tag{15}$$

The prompt template for each task is provided in Appendix C. The three tasks differ only in how $\mathcal{J}$ is drawn, widening from a single latent token to the full sequence (as illustrated in Table 4):

- **Single-Segment Reconstruction** ($\mathcal{J} = \{j\}$). We hide a single segment R_j behind its latent token z_j while leaving the rest of the trace in text, and the model reconstructs R_j. The rich surrounding context makes this reconstruction straightforward, so minimizing $\mathcal{L}(\{j\})$ grounds each latent token in its own segment and yields a direct per-position encoding signal.
- **Multi-Segment Reconstruction** ($\mathcal{J} \subsetneq \{1, \ldots, K\}$, $|\mathcal{J}| \geq 2$). We hide several segments behind their latent tokens simultaneously, and the model restores only the masked text $R_{\mathcal{J}}$. As the masked spans may straddle segment boundaries, minimizing $\mathcal{L}(\mathcal{J})$ drives each token to remain informative alongside the surrounding text rather than in isolation.
- **Full-Trace Reconstruction** ($\mathcal{J} = \{1, \ldots, K\}$). We mask every segment, so the context reduces to $c = (x, z, y)$ and the objective becomes $\mathcal{L}(\{1, \ldots, K\}) = -\log p_\theta(R \mid x, z, y, \mathcal{P})$. This compels the latent sequence to be sufficient to recover the full reasoning chain from (x, z, y) alone.

Beyond compression, the reconstruction objective preserves the decodability of latent reasoning. Because z is trained as a sufficient encoding of R, the model can recover a human-readable rationale from (x, z, y), retaining the transparency of explicit CoT at a much lower token cost. Appendix C provides cases showing how the latent tokens correspond to the explicit reasoning chain.

4.2. Post-training

This section details the post-training pipeline that equips **`RecGPT-V3`** with latent reasoning, carried out in two stages: (1) **Explicit-to-Implicit CoT Alignment** (§4.2.1) distills reasoning from a strong

teacher into explicit traces and then compresses them into latent tokens through the multi-task curriculum of §4.1; (2) **Reinforcement Learning from Ranking Feedback** (§4.2.2) optimizes the latent model directly against online business objectives.

4.2.1. Stage 1: Explicit-to-Implicit CoT Alignment

In this training stage, we first distill reasoning from a strong teacher into explicit CoT traces and then compress them into latent tokens through the multi-task curriculum of §4.1.

(1) Reasoning Distillation. We distill reasoning from a strong teacher by fine-tuning the model on its explicit CoT outputs. These thinking processes are effective yet inefficient, averaging roughly 2,300 tokens and thus dominating overall inference cost.

Table 5 | Training data mixture in Stage 1.

Task	Prop.
❶ **Reasoning alignment**	21.43%
Single-segment recon.	6.45%
Multi-segment recon.	6.56%
Full-trace recon.	8.42%
❷ **General reasoning**	69.58%
❸ **Tag prediction**	8.43%

(2) Latent Internalization. We then compress each trace into latent tokens via the alignment curriculum of §4.1; with $C = 20$ and $K_{\max} = 10$, a trace maps to at most 10 latent tokens, a roughly 200:1 reduction. Training draws on a multi-task mixture (Table 5) spanning three roles: ❶ the three **Reasoning-Alignment** tasks that internalize the trace into latent tokens, ❷ **General Reasoning** data that guards against catastrophic forgetting, and ❸ **Tag Prediction** that matches the deployment format, on which we supervise only the output tokens to spare the latent positions from collapse.

4.2.2. Stage 2: Reinforcement Learning from Ranking Feedback

The supervised stages only imitate teacher trajectories and cannot optimize business objectives. Existing RecGPT-V2 (Yi et al., 2025a) adopts an accuracy reward based on **HitRate**, the fraction of a user's held-out ground-truth items recovered among the candidates its outputs retrieve, yet this offline proxy exhibits two limitations. **(1)** ***Reward sparsity***: under a GRPO-style objective, rollouts within a group can receive the same reward, collapsing their advantages to zero and yielding no learning gradient. **(2)** ***Pipeline inconsistency***: since outputs must traverse the serving pipeline before reaching users, a reward computed outside it may diverge from actual downstream performance.

To address both limitations, we propose **Reinforcement Learning from Ranking Feedback (RLRF)**, whose core idea is to draw the reward directly from the production ranking model that governs what users ultimately see, yielding a signal that is both dense and consistent with the serving pipeline. Specifically, we replace the **HitRate** accuracy reward with a **CTRScore** read from this downstream ranking model. For a generated output $y = \{t_1, \ldots, t_m\}$ of m item tags, we retrieve candidate items with y, score each with the production ranking model, and average the top-K scores:

$$r_{\text{ctr}}(y) = \frac{1}{K} \sum_{k=1}^{K} s_k, \tag{16}$$

where s_k is the k-th largest **CTRScore** among the retrieved items ($K = 100$ in our experiments). This design counters both limitations: averaging over the top-K ranking scores turns the sparse **HitRate** signal into a dense one, while reading the reward from the production ranking model aligns training with the pipeline the outputs actually traverse. An output thus earns a high reward precisely when the items it retrieves are favored by the downstream ranker, steering the policy toward candidates the pipeline is more likely to surface to users.

For the remaining reward terms, we retain the ***Constrained Reward Shaping (CRS)*** framework of RecGPT-V2, which treats an *alignment score*, a *diversity score*, and a *length score* as quality constraints on the primary reward: r_{ctr} propagates only when all three thresholds are met. The alignment score is produced by a reward model trained on human-preference pairs, rating how well each predicted tag meets human quality standards. The final reward is formulated as follows:

$$\mathcal{R}(y) = r_{\text{ctr}}(y)\,\mathbb{1}\left[\text{align}(y) \geq \tau_{\text{align}}\right]\mathbb{1}[\text{div}(y) \geq \tau_{\text{div}}]\,\mathbb{1}[\text{len}(y) \geq \tau_{\text{len}}]\,, \tag{17}$$

where τ_{align}, τ_{div}, and τ_{len} are the alignment, diversity, and length thresholds; we refer readers to RecGPT-V2 for the definitions of these terms. We optimize the policy with GRPO (Shao et al., 2024), computing group-relative advantages over a group of outputs sampled per input.

5. Experiments

We evaluate RecGPT-V3 on Taobao's homepage recommendation scenario, serving hundreds of millions of daily active users. Our evaluation covers five dimensions: (1) overall online A/B test results against the RecGPT-V2 baseline (§5.1); (2) memory hub evaluation through human annotation and compute efficiency analysis (§5.2); (3) recommendation foundation model evaluation on general-capability preservation, SID-text alignment, and downstream recommendation quality (§5.3); (4) latent reasoning evaluation through post-training effectiveness and inference efficiency analysis (§5.4); and (5) further analysis on SID modality and case studies (§5.5).

5.1. Online A/B Test Results

Experimental Setup. We deploy RecGPT-V3 in Taobao's homepage "Guess What You Like" scenario and conduct online A/B tests under the following configuration:

- **Traffic Allocation:** The experimental and control groups each receive 1% of total platform traffic, yielding statistically significant results while keeping deployment risk to a minimum.
- **Baseline:** RecGPT-V2 serves as the control group, enabling a direct assessment of the improvements introduced by RecGPT-V3.
- **Evaluation Scenarios:** We evaluate performance separately under two complementary scopes:
  - *Item Scenario:* Direct item recommendations presented in a grid layout.
  - *Feed Scenario:* A mixed-content recommendation stream in the main feed, comprising items, advertisements, live streams, and other content types.

Evaluation Metrics. We assess system performance along two dimensions:

User Engagement:

- ***IPV (Item Page Views):*** Number of item detail page visits, reflecting user interest.
- ***CTR (Click-Through Rate):*** Ratio of clicks to impressions, measuring recommendation relevance.
- ***PV (Page Views):*** Total number of page views, indicating overall browsing activity.
- ***DAU (Daily Active Users):*** Number of unique active users per day, measuring platform activity.

Business Transaction:

- ***TC (Transaction Count):*** Number of completed transactions, reflecting conversion effectiveness.
- ***GMV (Gross Merchandise Value):*** Total monetary value of transactions, measuring overall business value.

Table 6 | Online A/B test results comparing RecGPT-V3 against RecGPT-V2 baseline across item and feed scenarios. All metrics show relative percentage improvements (% omitted).

Scenario	User Engagement				Business Transaction	
	IPV	CTR	PV	DAU	TC	GMV
Item	+3.08	+0.98	+2.02	–	+3.10	+7.51
Feed	+1.28	+1.00	+0.83	+0.56	+1.97	+3.97

Note: – indicates metrics not applicable in the given scenario.

Table 7 | Human evaluation of memory unit quality. "Behavior Pattern" measures whether the assigned pattern identifier correctly categorizes the user's behavioral cluster; "Behavior Index" measures whether the representative indices correctly point to interactions belonging to the identified pattern.

Evaluation Target	Annotations	Accuracy
Behavior Pattern	2,514	82.89%
Behavior Index	21,268	95.27%

Results and Analysis. Table 6 summarizes the online A/B test results. RecGPT-V3 consistently outperforms RecGPT-V2 across all metrics in both evaluation scenarios.

♦ In the item scenario, the improvements are more pronounced: +3.08% IPV, +0.98% CTR, +2.02% PV, +3.10% TC, and +7.51% GMV. This pattern suggests that RecGPT-V3 improves item-level recommendation beyond simply attracting more clicks, as the relatively stronger gains in IPV, TC, and especially GMV indicate that users are more likely to enter product detail pages, complete transactions, and generate higher transaction value after exposure. This implies the retrieved candidates are not only more relevant at the click stage, but also better aligned with users' purchase-oriented intent.

♦ In the feed scenario, which reflects the overall platform experience, RecGPT-V3 delivers +1.28% IPV and +1.00% CTR, indicating improved recommendation relevance and user engagement. Browsing activity rises by +0.83% PV, and platform retention shows a healthy gain of +0.56% DAU, confirming that the improvements extend across a broad user base rather than concentrating on a narrow subset. Transaction metrics also improve substantially, with GMV increasing by +3.97% and TC by +1.97%, demonstrating that stronger intent understanding translates directly into purchase activity.

5.2. Memory Hub Evaluation

We evaluate the memory hub introduced in §2 through two complementary analyses: human annotation to validate memory representation accuracy (§5.2.1) and compute efficiency analysis (§5.2.2). In all experiments, the memory hub condition replaces the Global Planner's full-sequence input with the compressed memory representation plus the recent behavioral delta, while maintaining identical downstream components.

5.2.1. Memory Quality Assessment

We conduct a large-scale human annotation study on the memory units produced by Structured Behavior Compression (§2.1) to evaluate the factual accuracy of the memory hub's structured representations. Annotators assess two complementary aspects of each memory unit, as shown in Table 7. Across 2,514 annotated behavior patterns, accuracy reaches 82.89%, indicating that the LLM-based

compression reliably identifies coherent behavioral archetypes from noisy interaction sequences. Behavior index accuracy is substantially higher at 95.27% across 21,268 annotations, confirming that once a pattern is correctly identified, the supporting evidence is accurately attributed. This is particularly important because each memory unit stores only integer pointers into the original sequence (§2.1), and the high index accuracy ensures these pointers remain faithful references for downstream reasoning. Together, the two metrics validate that the memory hub produces reliable structured representations.

5.2.2. Compute Efficiency

The memory hub reduces the Global Planner's compute cost by **55.80%** relative to the RecGPT-V2 baseline (Table 8). The dominant saving comes at inference: each pass now conditions on the compressed memory and the recent behavioral delta rather than re-encoding the full behavioral sequence, bringing the per-pass cost down to **33.43%**. Periodic incremental curation of the memory (§2.2) contributes a further **10.77%**, a modest overhead relative to the saving it enables. Overall, the memory hub converts a recurring per-inference expense into an amortized memory maintenance cost, yielding a compute profile that scales favorably with request volume.

Table 8 | User-modeling compute cost with and without the memory hub, expressed relative to the RecGPT-V2 baseline.

System	Component	Cost
RecGPT-V2	Full sequence	100%
RecGPT-V3	Per-inference	33.43%
	Memory curation	10.77%
	Total	**44.20%**

5.3. Foundation Model Evaluation

We evaluate the hybrid-modal foundation model along two questions: (1) Does the model preserve strong general language capabilities despite domain-specific training (§5.3.1)? (2) Does the model achieve accurate cross-modal SID–text alignment and translate it into strong downstream recommendation quality (§5.3.2)? To disentangle the contribution of general-domain data mixing, we compare two model variants throughout:

- *w/ General-Domain Data*: Initialized from Qwen3-14B and trained with both SID-grounding data and general-domain data.
- *w/o General-Domain Data*: Same architecture and SID-grounding data, but with all general-domain data removed.

5.3.1. General Capability Preservation

To answer whether the hybrid-modal foundation model preserves general language capabilities, we evaluate its accuracy across four representative benchmarks spanning mathematical reasoning (GSM8K), broad knowledge (MMLU), Chinese language understanding (CMMLU), and instruction following (IFEval).

As shown in Figure 6, general-domain data mixing preserves the vast majority of the backbone's capabilities. GSM8K drops only 1.66% (94.31% → 92.65%), MMLU 2.65%, CMMLU 4.49%, and IFEval falls from 81.52% to 75.60%. Removing general-domain data instead causes catastrophic collapse. The *w/o General-Domain Data* variant falls to 4.70% on GSM8K, 0.12% on MMLU, 0.01% on CMMLU, and 23.29% on IFEval, and loses nearly all reasoning, knowledge, and instruction-following ability. General-domain data is therefore essential. Without the regularization it provides during

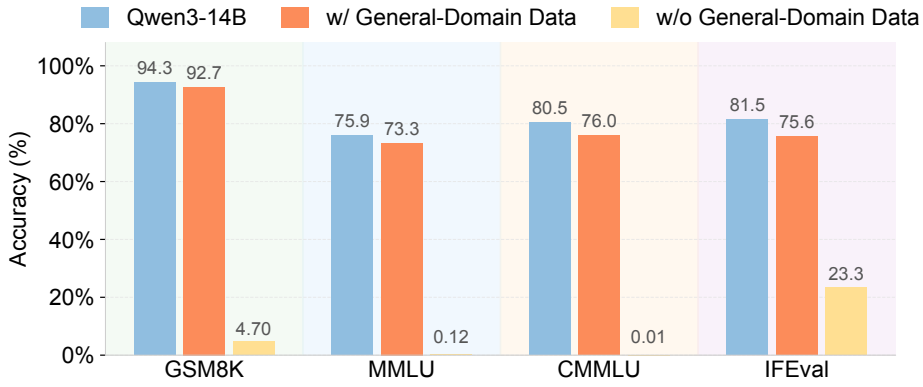

Figure 6 | General capability evaluation across four benchmarks. The hybrid-modal foundation model (*w/ General-Domain Data*) preserves most of the backbone's capabilities, while removing general-domain data (*w/o General-Domain Data*) leads to catastrophic collapse.

continual pre-training and instruction tuning, the model overfits to domain-specific patterns and loses the general capabilities needed for real-world deployment.

5.3.2. SID Alignment and Recommendation Quality

We assess the foundation model along two coupled dimensions: cross-modal alignment between SIDs and text, and downstream recommendation quality. Alignment is measured across four bidirectional tasks (*sid2title*, *sid2tag*, *title2sid*, *tag2sid*), with ROUGE-L for the two generation tasks and HR@30 (SID) for the two prediction tasks. Downstream quality is measured on three axes: tag diversity (average unique tags per user), category coverage (distinct item categories after mapping tags to the Taobao taxonomy), and category-level HR@30 (hit rate against actual clicks over two weeks).

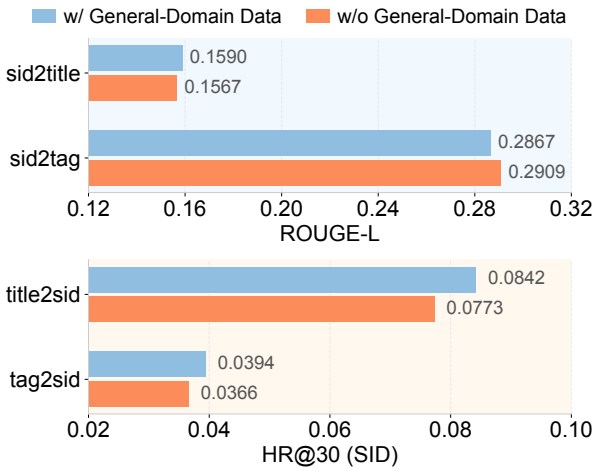

Figure 7 | SID–text semantic alignment across four bidirectional translation tasks.

Beyond mitigating catastrophic forgetting of the backbone's general capabilities (§5.3.1), general-domain data preserves SID–language alignment and lifts downstream recommendation quality. Figure 7 shows that all four alignment metrics remain essentially unchanged between the two variants, indicating that mixing general-domain data does not compromise the SID-to-text mapping ability learned from SID-grounding data. Table 9 further shows that this data mixing lifts category-level HR@30 by 26.2% (0.2250 → 0.3050) alongside comparable gains in tag diversity and category coverage, translating the retained general capabilities into materially stronger downstream performance through content-based and context-aware user intent understanding. Overall, these results justify the deliberate integration of general-domain data: it preserves the semantic alignment advantage, retains the backbone's general knowledge, and converts both into stronger downstream performance.

Table 9 | Downstream recommendation quality comparison.

Model	Avg. #Tags	Avg. #Categories	HR@30 (Category)
w/ General-Domain Data	28.77	41.73	0.3050
w/o General-Domain Data	23.88	32.49	0.2250

5.4. Latent Reasoning Evaluation

We evaluate the latent reasoning module (§4) from two perspectives: (1) post-training effectiveness, examining how each training stage and RL contribute to recommendation quality; and (2) inference efficiency, comparing the computational cost of explicit versus latent reasoning.

5.4.1. Post-training Effectiveness

We progressively add training stages to examine their individual contributions to recommendation quality. Table 10 reports two metrics: HR@30 (Category), the category-level hit rate among top-30 predictions, and CTR, the average CTR of top-100 items retrieved by the model's outputs. We compare two groups of model variants: the first is based on Qwen3-14B, assessing whether native reasoning alone improves recommendation without domain-specific training (CTR is not applicable as these variants are not connected to the online feedback pipeline); the second builds upon the hybrid-modal foundation model (§3), progressively adding reasoning components.

- *Qwen3-14B*: the base language model without reasoning, directly generating outputs from prompts.
  - *+ Native Reasoning*: enabling Qwen3's built-in chain-of-thought during inference.
- *Hybrid-modal Foundation Model*: the foundation model after pre-training, without reasoning.
  - *+ Explicit CoT (SFT)*: supervised fine-tuning with reasoning traces distilled from DeepSeek-V3.2.
  - *+ Latent Reasoning*: internalizing explicit reasoning into 10 latent tokens via the multi-task curriculum (§4.1).
  - *+ RL*: applying RL on top of latent reasoning (the full RecGPT-V3).

On Qwen3-14B, enabling native chain-of-thought yields only a marginal HR@30 improvement (0.2347 vs. 0.2276), confirming that reasoning without domain-specific training provides limited benefit. On the hybrid-modal foundation model, explicit CoT improves HR@30 from 0.3050 to 0.3508, demonstrating that structured reasoning enables more precise intent understanding. Latent reasoning achieves comparable quality (HR@30 0.3462, CTR 0.0649) while compressing ~2,700 reasoning tokens into just 10 latent tokens. With RL, performance further improves to 0.3693 on HR@30 and 0.0679 on CTR, surpassing explicit CoT on both metrics.

5.4.2. Inference Efficiency

A key practical concern with explicit chain-of-thought is its computational overhead: generating ~2,840 reasoning tokens per sample significantly increases inference latency, as each token requires sequential autoregressive decoding. Latent reasoning addresses this by compressing the reasoning process into 10 latent tokens, shifting the computational cost from slow sequential decoding to parallelizable prefill computation. Table 11 quantifies this advantage on a 1,000-sample benchmark under identical hardware. The latent reasoning model reduces per-sample output length from

Table 10 | Post-training effectiveness comparison. The upper group evaluates reasoning on the base language model; the lower group evaluates on the hybrid-modal foundation model. "–" indicates metrics not applicable to configurations outside the online feedback pipeline.

Setting	HR@30 (Category)	CTR
Qwen3-14B	0.2276	–
+ *Native Reasoning*	0.2347	–
Hybrid-modal Foundation Model	0.3050	0.0624
+ *Explicit CoT (SFT)*	0.3508	0.0638
+ *Latent Reasoning*	0.3462	0.0649
+ *RL*	**0.3693**	**0.0679**

Table 11 | Inference efficiency comparison between explicit CoT and latent reasoning on 1,000 samples under identical hardware. "Output Length" denotes the average number of tokens generated per sample, including reasoning and final output; "Input/Output TPM" denotes the tokens-per-minute throughput during prefill and decoding, respectively; "Total Time" denotes the wall-clock time to process all samples.

Mode	Output Length.	Input TPM	Output TPM	Total Time (s)
Explicit CoT	2,840	166K	531K	1,020
Latent Reasoning	122	498K	66.7K	295 (↓71.1%)

2,840 to 122 tokens (a 95.7% reduction), yielding a 3.46× end-to-end speedup (1,020s → 295s). Meanwhile, input throughput increases from 166K to 498K tokens per minute, confirming that the bottleneck has effectively shifted from output decoding to input prefill. Combined with the comparable recommendation quality demonstrated in Table 10, latent reasoning achieves a favorable trade-off between effectiveness and efficiency for production deployment.

Serving Cost Analysis. Using semantic IDs (SIDs) in both input and output, together with latent intent reasoning, increases the context length and thus introduces a 15% computational overhead for expert model compared with the RecGPT-V2 expert. However, this local overhead is outweighed by the system-level savings from the Global Planner, whose computational cost is approximately 20 times that of an expert model in the current deployment. With the memory mechanism reducing the planner-side computation by 55.8%, the weighted overall resource saving reaches **52.4%**.

5.5. Further Analysis

Beyond the quantitative evaluations above, we provide two further analyses to offer deeper insights. We first examine the complementary roles of text tags and SIDs as dual retrieval modalities (§5.5.1), then present case studies illustrating how the core modules collaborate in practice (§5.5.2).

5.5.1. SID Modality Analysis

The hybrid-modal foundation model generates both text tags and SIDs as complementary intent representations (§3): text tags provide generalizable category-level coverage through open-ended world knowledge, while SIDs ground user interests in concrete collaborative semantics. We validate this design from three perspectives: the coverage breadth of text tags versus the user specificity of SIDs, their diversity in the embedding space, and their complementarity in end-to-end retrieval.

Coverage Breadth vs. User Specificity. In each inference, the expert model generates N text tags $\{t_1, t_2, \ldots, t_N\}$ for a given user, along with K SIDs $\{\text{SID}_1, \ldots, \text{SID}_K\}$ per tag. We map each text tag t_i to an item category set $C_{\text{tag}}(t_i)$, and aggregate its co-generated SID categories as $C_{\text{sid}}(t_i) = \bigcup_{k=1}^{K} C(\text{SID}_k)$. We define the average category breadth $\tilde{C}$ and cross-user overlap J to characterize each signal type:

$$\tilde{C}_{\text{tag}} = \frac{1}{N}\sum_{i=1}^{N} |C_{\text{tag}}(t_i)|, \qquad \tilde{C}_{\text{sid}} = \frac{1}{N}\sum_{i=1}^{N} |C_{\text{sid}}(t_i)|, \tag{18}$$

$$J_{\text{tag}} = \mathbb{E}_{(u,v)}\left[\frac{|C_{\text{tag}}^{(u)} \cap C_{\text{tag}}^{(v)}|}{|C_{\text{tag}}^{(u)} \cup C_{\text{tag}}^{(v)}|}\right], \qquad J_{\text{sid}} = \mathbb{E}_{(u,v)}\left[\frac{|C_{\text{sid}}^{(u)} \cap C_{\text{sid}}^{(v)}|}{|C_{\text{sid}}^{(u)} \cup C_{\text{sid}}^{(v)}|}\right], \tag{19}$$

where $\tilde{C}$ measures the average number of distinct categories covered, and J measures the expected overlap in category sets between randomly sampled user pairs (u, v). Results are reported in Table 12. On category breadth, $\tilde{C}_{\text{tag}} = 1.40$ while $\tilde{C}_{\text{sid}} = 0.84$. Even when aggregating K SIDs per tag, the combined category set remains narrower than that of a single text tag. This confirms the broader coverage of text tags, which span diverse interest scopes, against the narrower focus of SIDs on specific category clusters. On cross-user overlap, $J_{\text{tag}} = 11.36\%$ compared to $J_{\text{sid}} = 4.61\%$. The substantially lower overlap of SIDs reflects their stronger user specificity, as collaborative semantics capture user-specific signals, whereas the higher overlap of text tags stems from their reliance on shared world knowledge.

Table 12 | Comparison between text tags and SIDs on category-level statistics.

Metric	Text Tag	SID
Category Breadth	1.40	0.84
Cross-User Overlap	11.36%	4.61%

Embedding-Space Visualization. We further examine the diversity of items retrieved through text tags versus SIDs. For each text tag and its co-generated SIDs, we sample an equal number of relevant items and obtain their embeddings from a pre-trained retrieval model. As shown in Figure 8, we project these embeddings to 2D via PCA. Tag-matched items show greater spatial dispersion across diverse regions, while SID-matched items form tighter clusters around specific neighborhoods. This pattern visually reinforces the world-knowledge-driven coverage of text tags versus the collaboration-driven specificity of SIDs.

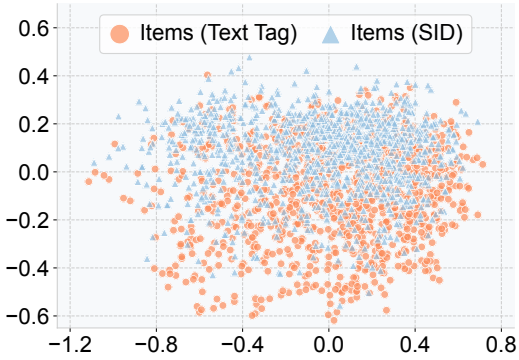

Figure 8 | PCA visualization of item embeddings retrieved by text tags and SIDs.

Table 13 | Item-level hit rate under text tag, SID, and hybrid retrieval configurations.

Configuration	HR@500	HR@1000
Tag	0.1503	0.2044
SID	0.1539	0.2144
Hybrid	**0.1571**	**0.2168**

Retrieval Complementarity. Finally, we quantify the end-to-end retrieval benefit of combining both modalities. Table 13 reports item-level hit rate on an offline test set under three configurations.

Figure 9 | Case study. The memory hub compresses a user's raw behavior sequence into structured preference units and incrementally updates them with recent behaviors. Based on the curated memory, RecGPT-V3 performs latent intent reasoning with only 10 tokens, reconstructs an explainable rationale on demand, and generates memory-driven badminton recommendations grounded by SIDs.

On HR@500, SID retrieval reaches 0.1539 compared to 0.1503 for text tag retrieval, and the hybrid configuration further improves to 0.1571. A consistent trend holds on HR@1000 (0.2168 vs. 0.2144 vs. 0.2044), confirming the complementarity of the two modalities. The hybrid retrieval model (§B) consumes both modalities jointly, combining the broad coverage of text tags with the collaborative precision of SIDs.

5.5.2. Case Study

Figure 9 presents an end-to-end case study of RecGPT-V3 on an anonymized Taobao user. The memory hub first compresses the raw behavior sequence into structured preference units, such as technology, badminton, and baby-care interests. When recent behaviors arrive, it selectively updates related memories: customized-keyboard interactions refresh the technology unit, Astrox-related searches and clicks update the badminton unit, baby-care interests evolve toward solid food and early education, stable home-improvement preferences are retained, and a new running preference is created.

Based on the curated memory, RecGPT-V3 performs Latent Intent Reasoning with only 10 latent

tokens. These latent tokens replace the long explicit CoT during inference, but can still be decoded to reconstruct a readable explicit rationale on demand. In this case, the reconstructed rationale identifies the user as a serious badminton player with an offensive playing style and infers needs for full-carbon rackets, strings, maintenance tools, and related badminton equipment. These inferred intents are then grounded into concrete memory-driven recommendations with SIDs. This case shows how RecGPT-V3 accumulates user understanding through evolving memory, preserves explainable reasoning via latent-to-explicit CoT reconstruction, and generates precise recommendations from the inferred intent.

6. Conclusion

In this paper, we presented **RecGPT-V3**, an LLM-based recommender that addresses three challenges facing LLM-based recommendation at industrial scale: stateless behavior modeling, the tag-to-item information bottleneck, and inefficient explicit reasoning. **RecGPT-V3** introduces a *Memory Hub* that maintains a structured, continually evolving user memory in place of one-shot full-history encoding, a *Hybrid-modal Foundation Model* that reasons jointly over natural language and Semantic IDs to ground intent directly in the item space, and *Latent Intent Reasoning* that compresses explicit chain-of-thought into compact, decodable latent tokens. Deployed in Taobao's "Guess What You Like" feed, **RecGPT-V3** achieves consistent gains in large-scale online A/B tests (+1.28% IPV, +1.00% CTR, +1.97% TC, and +3.97% GMV) while reducing end-to-end serving compute by 52.4%. These performance gains show that prediction accuracy and serving efficiency can improve together, and suggest that LLMs can serve as the reasoning brain of industrial-scale recommendation pipelines.

References

J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems*, 41(3):1–39, 2023.

J. Chen, T. Zhang, M. Lin, D. Huang, T. Shi, H. Fu, M. Li, X. Zhang, C. Zhang, X. Lu, et al. Shopx: A foundation model for intent-to-item fulfillment in agentic shopping. *arXiv preprint arXiv:2606.31693*, 2026.

K. Fu, T. Zhang, S. Xiao, Z. Wang, X. Zhang, C. Zhang, Y. Yan, J. Zheng, Y. Li, Z. Chen, et al. FORGE: Forming semantic identifiers for generative retrieval in industrial datasets. In *The 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2026.

C. Gao, K. Huang, J. Chen, Y. Zhang, B. Li, P. Jiang, S. Wang, Z. Zhang, and X. He. Alleviating matthew effect of offline reinforcement learning in interactive recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, pages 238–248, 2023.

D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

D. Lee, C. Kim, S. Kim, M. Cho, and W.-S. Han. Autoregressive image generation using residual quantization. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11513–11522, 2022.

J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

Y. Li, Z. Zhang, M. Liang, K. Asadi, J. Xu, J. Kim, C. Bai, J. Zhang, H. Xie, P. Agrawal, et al. GR2 technical report. *arXiv preprint arXiv:2606.31984*, 2026.

A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Z. Liu, S. Wang, X. Wang, R. Zhang, J. Deng, H. Bao, J. Zhang, W. Li, P. Zheng, X. Wu, et al. Onerec-think: In-text reasoning for generative recommendation. *arXiv preprint arXiv:2510.11639*, 2025.

T. T. Nguyen, P.-M. Hui, F. M. Harper, L. Terveen, and J. A. Konstan. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web*, pages 677–686, 2014.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461, 2009.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

J. Tang, S. Dai, T. Shi, J. Xu, X. Chen, W. Chen, J. Wu, and Y. Jiang. Think before recommend: Unleashing the latent reasoning power for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2026.

O. Team, B. Yang, B. Ding, C. Chu, D. Zang, F. Pan, H. Li, H. Jiang, H. Bao, H. Wang, et al. Onereason technical report. *arXiv preprint arXiv:2606.06260*, 2026.

H. Wang, H. Lu, Z. Feng, J. Huang, Y. Amir, G. Hinkson, B. Most, Z. Zhao, Y. K. Cui, R. Zhang, et al. Llm-based user personas for recommendations at scale. *arXiv preprint arXiv:2606.12198*, 2026.

A. Yang, J. Pan, J. Lin, R. Men, Y. Zhang, J. Zhou, and C. Zhou. Chinese clip: Contrastive vision-language pretraining in chinese. *arXiv preprint arXiv:2211.01335*, 2022.

A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

C. Yi, D. Chen, G. Guo, J. Tang, J. Wu, J. Yu, M. Zhang, W. Chen, W. Yang, Y. Luo, et al. Recgpt-v2 technical report. *arXiv preprint arXiv:2512.14503*, 2025a.

C. Yi, D. Chen, G. Guo, J. Tang, J. Wu, J. Yu, M. Zhang, S. Dai, W. Chen, W. Yang, et al. Recgpt technical report. *arXiv preprint arXiv:2507.22879*, 2025b.

Y. Zhang, M. Liang, J. Yang, R. Jin, W.-Y. Chen, Y. Han, H. Li, B. Zhang, L. Luo, L. Simon, et al. Reasonrec: A reasoning-augmented multimodal agent for unified recommendation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 7960–7980, 2026.

W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2):1–124, 2023.

G. Zhou, H. Bao, J. Huang, J. Deng, J. Zhang, J. She, K. Cai, L. Ren, L. Ren, Q. Luo, et al. Openonerec technical report. *arXiv preprint arXiv:2512.24762*, 2025.

Appendix

A. Contributors

Core Contributors

Bowen Zheng
Chao Yi
Dian Chen
Gaoyang Guo
Han Zhu
Jiakai Tang
Jian Wu
Mao Zhang
Wen Chen
Yifan Lu
Yujie Luo
Yuning Jiang
Zhujin Gao

Contributors

Bo Zheng
Dixuan Wang
Hao Fang
Jiancai Liu
Jing Yu
Ke Chen
Kewei Zhu
Mingke Xu
Wenjun Yang
Xunke Xi
Zile Zhou

The listing of authors is in alphabetical order based on their first names.

B. Intent-to-Item Retrieval

The Hybrid-modal Foundation Model (§3) enables the multi-expert modules to express user intent through two channels: natural-language tags and Semantic IDs. The two play distinct roles: tags convey intent through open-ended world knowledge (***generalizable***), whereas SIDs ground that intent in concrete item representations enriched by collaborative signals (***concrete***). Turning these hybrid signals into accurate item retrieval, however, requires a dedicated downstream model that can (1) consume both channels jointly, and (2) balance click likelihood against semantic relevance. Earlier `RecGPT` versions retrieved items through a text-only pathway, confining the system to a single channel and leaving the collaborative signal carried by SIDs unexploited.

We therefore introduce a **hybrid retrieval model** that takes both text tags and SIDs as input and learns a preference ordering over candidate items, favoring those a user is likely to click while preserving semantic alignment with the upstream intent predictions. The remainder of this section presents its hybrid input architecture (§B.1) and joint training objective (§B.2).

B.1. Architecture

The retriever takes the text tags and SIDs as input and assigns a relevance score to each candidate item. The two channels play distinct roles: a set of text tags $\{t_1, \ldots, t_M\}$ represents semantic intent, while a set of SIDs $\{d_1, \ldots, d_M\}$ grounds that intent in the item space. Each tag t_i and its paired SID d_i capture two sides of the same user intent: the former expresses "what the user wants" in natural language, while the latter encodes "which items satisfy that want" in the discrete item space.

To fuse these two channels, we employ a dual-embedding architecture. Each tag t_i is mapped to a dense vector via a text embedding layer $\mathbf{e}_t^{(i)} = \mathrm{Emb}_{\mathrm{tag}}(t_i)$, and each SID d_i is independently mapped via a SID embedding layer $\mathbf{e}_d^{(i)} = \mathrm{Emb}_{\mathrm{sid}}(d_i)$. The two embeddings are concatenated and projected

through a linear transformation to produce a unified intent representation:

$$\mathbf{q}_i = W_{\text{proj}} \cdot [\mathbf{e}_t^{(i)} \| \mathbf{e}_d^{(i)}] + \mathbf{b} \tag{20}$$

where $[\cdot\|\cdot]$ denotes concatenation and $W_{\text{proj}} \in \mathbb{R}^{d \times 2d}$ is a learnable projection matrix. The resulting intent vectors $\{\mathbf{q}_1, \ldots, \mathbf{q}_M\}$ serve as queries in a Target-Attention mechanism, attending over the candidate item representations to compute their retrieval scores. This design preserves channel-specific information in the early layers while enabling cross-channel interaction through the shared attention computation.

The dual-channel design equips the retriever with collaborative evidence that a text-only model cannot access. Through the contrastive pre-training of §3.1, the SID embedding space encodes item-level signals such as co-purchase patterns, popularity dynamics, and item-to-item affinity, which natural language cannot convey. The two channels also span different regions of the item space (§5.5.1), and together they let the retriever pair the semantic breadth of language with the collaborative precision of discrete item representations.

B.2. Training Recipe

Click signals alone provide insufficient supervision for the retriever. Because click propensity is confounded with item popularity, a purely click-driven objective tends to favor globally popular items over those faithful to the conditioning intent, producing retrievals that are clickable yet inconsistent with the tags and SIDs predicted upstream. This misalignment weakens the intent grounding that the pipeline is designed to preserve. We therefore cast retrieval training as a joint objective that reconciles click utility with semantic relevance under an explicit preference hierarchy over candidate items.

Preference Ordering. We define a hierarchical preference over candidate items that integrates both utility and relevance signals:

$$\textsc{Clicked} \cap \textsc{Relevant} \;>\; \textsc{Unclicked} \cap \textsc{Relevant} \;>\; \textsc{Unclicked} \cap \textsc{Irrelevant}$$

This ordering treats utility (click likelihood) as the primary objective and, among items of comparable utility, relies on relevance as the discriminating signal.

Joint Learning Objective. The preference ordering translates into a joint training objective that couples two goals: a utility objective governing the primary click signal, and a relevance objective that, among items of comparable utility, resolves the ordering by semantic relevance. Optimizing the two jointly aligns click utility with intent grounding within a single objective.

In specific, the **utility loss** optimizes click prediction through a standard softmax objective over the item candidate set:

$$\mathcal{L}_{\text{util}} = -\frac{1}{|C|} \sum_{i \in C} \log \frac{\exp(s_i)}{\exp(s_i) + \sum_{j \notin C} \exp(s_j)} \tag{21}$$

where C denotes the set of clicked items and s_i is the retrieval score for item i.

While the utility loss separates clicked items from the rest, it cannot order candidates that share the same click status. The **relevance loss** supplies this missing discrimination, enforcing the preference ordering through a multi-level contrastive formulation:

$$\mathcal{L}_{\text{rel}} = -\frac{1}{N} \sum_{i=1}^{N} \frac{1}{|\mathcal{P}(i)|} \sum_{j \in \mathcal{P}(i)} \log \frac{\exp(s_j)}{\exp(s_j) + \sum_{k \in \mathcal{N}(i,j)} \exp(s_k)} \tag{22}$$

where $\mathcal{P}(i)$ denotes the relevance-positive set for input intent i, constructed at progressively coarser matching criteria, and $\mathcal{N}(i, j)$ is the corresponding negative set. To keep the relevance objective from conflicting with the utility signal, we assign each candidate a priority by interaction depth, *clicked* > *exposed* > *non-exposed*, and restrict $\mathcal{N}(i, j)$ to candidates of priority no higher than the positive j.

The final training objective is a weighted sum of the utility and relevance losses:

$$\mathcal{L} = \alpha \, \mathcal{L}_{\text{util}} + \beta \, \mathcal{L}_{\text{rel}}, \tag{23}$$

where the coefficients α and β balance the two objectives (we set $\alpha = 1$ and $\beta = 0.5$ in our production experiments). This joint objective drives the retriever to favor products that are click-worthy and semantically aligned with the upstream intent, coupling click utility with intent grounding within an end-to-end optimization framework.

C. Latent Reasoning Reconstruction

The three alignment tasks of §4.1 rely on a reconstruction prompt $\mathcal{P}$ that cues the model to decode the natural-language reasoning compressed into latent `<cot>` tokens. We use two prompt variants, chosen by how many segments are being reconstructed: for **single- and multi-segment reconstruction** ($|\mathcal{J}| < K$), the prompt asks the model to explain the sentence each `<cot>` token likely represents; for **full-trace reconstruction** ($\mathcal{J} = \{1, \ldots, K\}$), it asks the model to recover the entire chain of thought. Each reconstruction sample places the prompt in the system turn. The user turn then contains the original recommendation context (expert persona, user profile, and click history) followed by the `<think>` block, in which latent tokens replace the masked reasoning segments, and the model's final output. Given this input, the reconstruction model explains each masked `<cot>` token in natural language. Below we present one representative production case per task. Personally identifiable details such as user location are redacted and shown as a blank bar (▬▬).

Case 1: Single-Segment Reconstruction ($\mathcal{J} = \{1\}$)

Input

[System] You are a semantic interpretation model skilled at explaining the meaning of `<cot>` tokens in a sentence. Output the sentences that the `<cot>` tokens in the following text likely represent, so that the resulting chain of thought is coherent.

[User] You are a bags and accessories expert. User profile: [redacted], residing in [redacted], who needs to balance fashion and practicality in daily work and life, and prefers a comfortable, minimalist yet design-oriented style. Interest pattern analysis: - Repeatedly explores and purchases backpacks, soft-leather bags, and bags with rhinestone/diamond details; favors novel designs such as neo-Chinese vintage backpacks. - Values both function and appearance: bags emphasize capacity (large capacity, dual-shoulder) and comfortable carrying, while also valuing durability features such as water resistance and abrasion resistance; likes visual accents such as rhinestones, color-blocking, and embroidery. - Prefers practical bag styles suitable for switching across multiple occasions, ideal for urban commuting and short trips. ... Based on the user's historical preferences, please recommend women's bag products, and provide purchasing rationale considering design, practicality, and daily use. ...

`<think>` `<cot1>`
Possible recommended products: 1. neo-Chinese vintage backpack (directly from click history) 2. rhinestone soft-leather backpack (combining preferences) 3. rhinestone embroidered crossbody bag 4. waterproof large-capacity backpack 5. color-blocked handbag 6. vintage-style shoulder bag 7. comfortable-carry backpack 8. rhinestone chain bag 9. neo-Chinese embroidered bag 10. soft-leather rhinestone wallet
But we must ensure the tags are specific and relevant. From the click history, the user clicked "neo-Chinese vintage backpack", so similar styles should be recommended. ...
Ensure tags are specific: each tag has a core word such as "backpack" or "crossbody bag", with modifiers such as "neo-Chinese", "rhinestone", "waterproof", etc.
Finally, check relevance: all recommendations are based on user preferences and click history. `</think>`

```
[{"t": "neo-Chinese embroidered bag"}, {"t": "soft-leather abrasion-resistant backpack"}...]
```

Reconstruction Output

The meaning of the `<cot1>` token is as follows: { `<cot1>` : "First, the user is [redacted], residing in [redacted], who needs to balance fashion and practicality in daily work and life, and prefers a comfortable, minimalist yet design-oriented style. Her interest pattern includes: repeatedly exploring and purchasing backpacks, soft-leather bags, and bags with rhinestone/diamond details, favoring novel designs such as neo-Chinese vintage backpacks. ...
Task requirement: based on the above information, recommend the products the user is most likely to click next. Product tags must be specific and not too broad ... output at least 3 and at most 10.
Now, analyze the user's click history:
- Clicked products include: a waterproof sun-protective windbreaker, a stand-collar plus-size wool coat ... but also some bag-related ones, such as a waterproof large-capacity satchel, a handheld small square crossbody bag, an Oxford-cloth minimalist crossbody bag, a neo-Chinese vintage backpack, etc.
- Specific bag-related clicks: waterproof large-capacity satchel, handheld small square crossbody bag, Oxford-cloth minimalist crossbody bag, neo-Chinese vintage backpack. This indicates the user has a sustained interest in bags, especially backpacks and crossbody bags, with design elements such as neo-Chinese and rhinestones.
From the click history, the user's preferences are:
- Bag types: backpack, crossbody bag, handbag.
- Design elements: neo-Chinese, vintage, rhinestone, embroidery, color-blocking.
- Practical features: waterproof, large capacity, comfortable carrying.
- Materials: soft leather, genuine leather, sheepskin, etc.
...}

Case 2: Multi-Segment Reconstruction ($\mathcal{J} = \{1, 2\}$)

Input

[System] You are a semantic interpretation model skilled at explaining the meaning of `<cot>` tokens in a sentence. Output the sentences that the `<cot>` tokens in the following text likely represent, so that the resulting chain of thought is coherent.

[User] You are a professional fashion apparel expert, particularly skilled at coordinating outfits for middle-aged women and teenagers. User profile: [redacted] with certain requirements for apparel quality. ...
Apparel preferences analyzed from purchase history: 1. Favors a simple and elegant style: has purchased white shirts, knitwear, basic T-shirts 2. Values comfort: has purchased cotton and ice-silk garments 3. Has sun-protection needs: sun-protective clothing, long-sleeve shirts, etc. 4. Attends to the daughter's clothing: has purchased girls' clothing, school uniforms, etc.
Current environment and seasonal needs: [redacted] summer is hot and humid (29–34°C), Major Heat is approaching, with light rain, and school starts in late July. ...Please recommend apparel and accessory products, including cool and breathable summer clothing, elegant pieces suitable for middle-aged women, sun-protective functional clothing, parent-child styles that can match the daughter, and semi-formal clothing suitable for the new school term. ...

`<think>` `<cot1>` `<cot2>`
- sun-protective functional clothing - parent-child styles that can match the daughter - semi-formal clothing suitable for the new school term

Extracting trends from click history:
- Shirts: French-style blue shirt, French-style floral shirt – prefers French style and floral prints
- Dresses: printed sleeveless dress, striped waist-cinched dress, French-style gray dress – likes dresses with floral prints and waist-cinching elements
- Pants: Tencel-linen ice-silk Yamamoto pants, bohemian linen wide-leg pants, vintage floral wide-leg pants – likes loose, comfortable pants
- Shoes: mesh breathable sandals, mesh comfortable Mary Jane shoes, soft-sole non-slip nurse shoes – values comfort and breathability in shoes

... Recommended products should be based on relevance (French, vintage, minimalist), efficiency (urgently needed in summer), and diversity (covering tops, bottoms, dresses, shoes). ... `</think>`

```
[{"t": "French-style floral top"}, {"t": "waist-cinched slimming dress"}...]
```

Reconstruction Output

The meanings of the `<cot1>` and `<cot2>` tokens are as follows: { `<cot1>` : "First, the user is [redacted], with requirements for apparel quality...
Analysis from purchase history:
- Favors a simple and elegant style: white shirts, knitwear, basic T-shirts
- Values comfort: cotton and ice-silk materials
...
Current environment: [redacted] summer is hot and humid (29–34°C), the Major Heat solar term is approaching, so cool and breathable clothing is needed ...
...
List of products the user clicked in the past month:
- French-style blue shirt
- bear-print Korean-style POLO shirt
...
- soft-sole non-slip nurse shoes",

`<cot2>` : "...
- blue waist-cinched suit

From the click history, the user clicked many apparel products, including shirts, dresses, pants, shoes, etc., with styles leaning toward French, vintage, and Korean, and materials such as Tencel, linen, and ice-silk, valuing comfort and breathability. ...
Recommended products need to be:
- cool and breathable summer clothing
- elegant pieces suitable for middle-aged women"}

Case 3: Full-Trace Reconstruction ($\mathcal{J} = \{1, \ldots, 8\}$)

Input

[System] You are a semantic interpretation model skilled at recovering the chain of thought behind `<cot>` tokens in a sentence.

[User] You are a professional automotive accessories and modification expert, familiar with all kinds of car parts and decorative products. User profile: [redacted], owns a [redacted] car (imported), values quality of life. Automotive purchase records: has purchased car stickers, in-car aromatherapy, windshield decals, wiper blades, etc., showing an interest in car decoration and maintenance. Consumption traits: prefers cute cartoon-style decorations, values the combination of practicality and aesthetics. Vehicle characteristics: the [redacted] car is a compact luxury car, suitable for refined and practical modification plans. . . . Please recommend products across dimensions such as interior decoration and ambiance creation, practical functional accessories, exterior body decoration, maintenance and care products, and safety and convenience equipment. . . .

`<think>` `<cot1>` `<cot2>` `<cot3>` `<cot4>` `<cot5>` `<cot6>` `<cot7>` `<cot8>` `</think>`

```
[{"t": "cute cartoon car ornament"}, {"t": "sun-protective cartoon sunshade"}...]
```

Reconstruction Output

First, the user is [redacted], owns a [redacted] (imported) compact luxury car, and values quality of life. Purchase records show she has bought car stickers, in-car aromatherapy, windshield decals, wiper blades, etc.; she prefers cute cartoon-style decorations and values the combination of practicality and aesthetics. The vehicle is suitable for refined and practical modification plans.

Demand dimensions:
1. interior decoration and ambiance creation 2. practical functional accessories 3. exterior body decoration 4. maintenance and care products 5. safety and convenience equipment

Recommend 2–3 specific products per dimension, but the task requires adjusting the number of outputs based on the user's click history, at least 3 and at most 10. The list of products the user clicked in the past month shows she clicked many cute cartoon-style items, such as a polka-dot Miffy phone case, a new cartoon sunshade, a black-leather Kitty car ornament, a new Hello Kitty car ornament, etc. This indicates she has a strong preference for cute, cartoon-themed decorations.

From the click history, the user clicked:
- many cartoon-themed items: Miffy, Kitty, bear, etc.
- car-related items: sunshade, car ornament, car mat, etc.

. . .
Therefore, the recommended products should be car-related and incorporate a cute cartoon style, while also considering practicality and aesthetics.

Now, analyze each dimension and recommend specific products:
1. **Interior decoration and ambiance creation:**. . . Specifically: cute cartoon car ornament, aromatherapy air-vent clip, ambient lighting.
2. **Practical functional accessories:**the user clicked sunshades, car mats, etc. Recommend: sun-protective sunshade curtain, comfortable seat cushion, storage box.
3. **Exterior body decoration:**the user has purchased car stickers and windshield decals. Recommend: cartoon-pattern car stickers

. . .
. . .
All recommendations incorporate a cute cartoon style, matching the user's aesthetic preferences, ensuring tags are specific and relevant.