GENERALIZE OR REMEMBER? RETHINKING GENERATIVE RECOMMENDATION FROM A DATA-CENTRIC PERSPECTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern generative recommendation (GR) is largely built around generalization: semantic IDs (SIDs) allow related items to share discrete tokens and transfer knowledge across items. Yet generalization alone is insufficient for recommendation: when interaction patterns recur across users, effectively reusing such recurring knowledge is equally important. This raises a key question: when should a generative recommender generalize, and when should it remember? We revisit GR from a data-centric perspective by distinguishing interaction instances according to whether their target transitions are supported by recurring cross-user behaviors, yielding memory and generalization subsets. Our analysis shows that memory specialization improves performance on the memory subset at the cost of generalization and overall performance, while gradients from the two subsets frequently conflict. To address this trade-off, we propose **MemOPSD**, a memory-driven on-policy self-distillation framework. MemOPSD trains a memory-specialized teacher on recurring interactions and selectively transfers its knowledge only to memory samples while retaining the original GR objective on all training instances, strengthening memory without indiscriminately imposing memory-oriented supervision on generalization data. Since conventional distillation relies on fixed prefixes that differ from student-generated states at inference time, MemOPSD further performs on-policy distillation along student-generated SID trajectories, transferring memory knowledge at the states actually visited by the student. Extensive experiments on three benchmark datasets and four GR backbones demonstrate consistent improvements in overall and memory recommendation performance. The code is available at https://anonymous.4open.science/r/MemOPSD-0627.

1 INTRODUCTION

Generative recommendation (GR) approaches (Hou et al., 2023; Rajput et al., 2023; Wang et al., 2024) have recently emerged as a promising paradigm for next-item recommendation. Unlike conventional sequential recommendation (SR) methods (Kang & McAuley, 2018; Qiu et al., 2022; Zhou et al., 2020), which represent each item with an independent ID embedding, SID-based GR encodes an item as a semantic ID (SID), i.e., an ordered tuple of discrete tokens drawn from a shared vocabulary. By allowing semantically related items to share SID tokens or prefixes, this representation facilitates knowledge transfer across items while reducing reliance on item-specific embedding tables. Recent studies (Deng et al., 2025; Ding et al., 2026; Hou et al., 2026b; Liu et al., 2026a;b) have explored improvements to SID construction, generative architectures, and optimization strategies. Shared semantic structures provide an inductive bias for generalization, allowing knowledge learned from observed items and interactions to transfer to related ones.

However, semantic generalization is only one side of recommendation. User interaction histories also contain recurring item transitions across users, where previously observed knowledge can be effectively reused (Lin et al., 2026a). We examine memory and generalization from a data-centric perspective by distinguishing instances according to recurring cross-user transitions. Specifically, an instance is labeled as memory data if the transition from its last observed item to its target item appears in another user's training history; otherwise, it belongs to the generalization subset (Fig. 6).

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

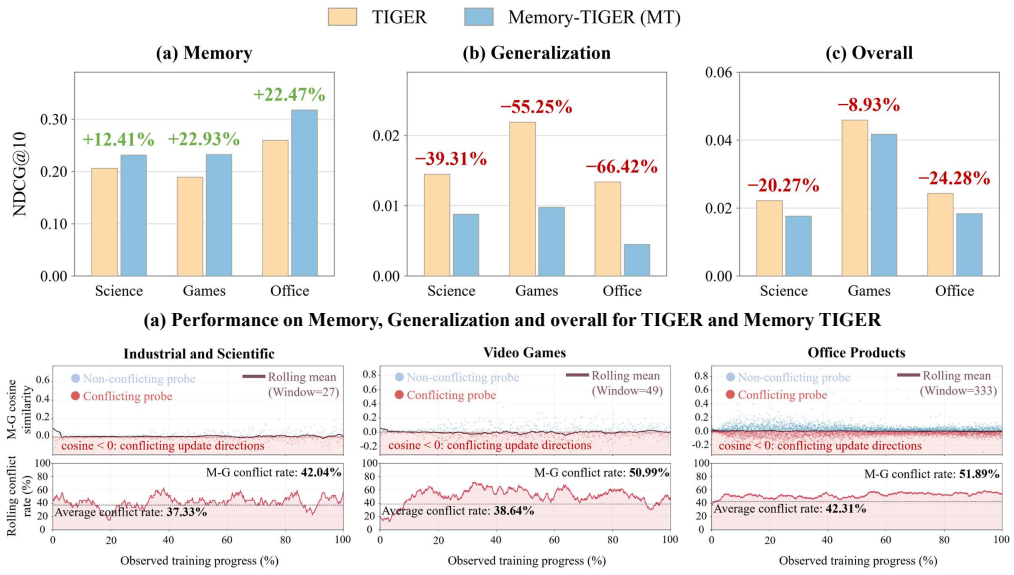


Figure 1: **Memory specialization and gradient conflict analysis.** (a) NDCG@10 on the Memory and Generalization subsets and overall for TIGER (Rajput et al., 2023) trained on the full training set and Memory-TIGER (MT), an identically structured model trained exclusively on memory data. (b) Gradient conflicts between memory and generalization data during GR training (Appendix F.5).

As shown in Fig. 1(a), we compare TIGER trained on the full training set with Memory-TIGER (MT), an identically structured model trained exclusively on memory data. MT achieves higher NDCG@10 on the Memory subset across all three datasets, but lower performance on the Generalization subset and overall. These results reveal an opportunity to improve recommendation for recurring interactions through memory specialization, while also exposing its cost to generalization. This trade-off motivates selectively transferring the memory teacher's strengths while limiting the influence of its weaker generalization performance (Appendix B).

Turning these observations into an effective learning strategy raises two key challenges. **First, memory enhancement must limit adverse effects on generalization.** Memory and generalization samples update the same model parameters, so strengthening recurring interaction knowledge may interfere with generalization learning. Fig. 1(b) shows that gradients induced by the two subsets frequently point in conflicting directions. Together with the specialization trade-off in Fig. 1(a), these gradient conflicts motivate restricting memory-teacher guidance to memory samples while retaining the original GR objective on all training instances. **Second, memory knowledge must be transferred under the autoregressive states visited by the student.** Conventional off-policy knowledge distillation (KD) (Hinton, 2014; Kim & Rush, 2016) typically provides supervision under ground-truth or teacher-generated prefixes, whereas inference conditions on the student's own generated SID tokens. Early SID errors can therefore lead to prefix states poorly represented during distillation and affect subsequent token predictions.

To address these challenges, we propose MemOPSD, a memory-driven on-policy self-distillation framework for generative recommendation. Its key principle is to use recurring interaction evidence to determine which samples receive memory-specific supervision, and student-generated trajectories to determine where that supervision is applied during autoregressive generation. For the first challenge, we introduce **Selective Memory Specialization and Transfer**. We train a same-architecture teacher exclusively on memory data, allowing it to specialize in recurring interaction knowledge. During post-training, the teacher provides additional supervision only to memory samples, while the original GR objective is retained over the complete training set. For the second challenge, we introduce **on-policy SID self-distillation** as the transfer mechanism. The student samples complete SID trajectories from its current policy, and the frozen memory teacher provides token-level guidance conditioned on the same student-generated prefixes. Memory knowledge is thus transferred at autoregressive states visited by the student, reducing the mismatch between fixed-prefix supervision and student-generated prefixes. Our main contributions are summarized as follows:

- We revisit generative recommendation from a data-centric perspective and analyze its behavior on memory and generalization data, identifying opportunities for memory enhancement and the trade-offs introduced by memory specialization.
- We propose MemOPSD, which combines selective memory specialization and transfer with on-policy SID self-distillation. The former determines where memory knowledge is learned and which samples receive additional supervision, while the latter transfers this knowledge along student-generated autoregressive trajectories.
- Extensive experiments on three benchmark datasets and multiple GR backbones demonstrate consistent improvements in memory and overall recommendation performance. Further analyses across data subsets, model scales, language-model families, and distillation configurations validate the effectiveness and generality of MemOPSD.

2 Related Work

Generative Recommendation Generative recommendation (GR) (Hou et al., 2026a; Ju et al., 2025; Liu et al., 2025; Rajput et al., 2023) has recently emerged as a highly promising paradigm for next-item recommendation. GRs first tokenize items into discrete identifiers (e.g., semantic IDs) and then train a generative model to autoregressively produce the identifier of the next item. Early work such as P5 (Geng et al., 2022) formulated recommendation as a language-generation task, whereas TIGER (Rajput et al., 2023) further introduced semantic-ID-based generative retrieval by quantizing item representations into sequences of codewords. The first research line focuses on semantic-ID construction (He et al., 2026; Hou et al., 2023; Shi et al., 2025; Wan et al., 2026; Wang et al., 2024) and representation optimization (Bai et al., 2026; Li et al., 2026b; Wang et al., 2025). VQ-Rec (Hou et al., 2023) maps textual item representations into discrete codes via product quantization, demonstrating the transferability of semantic encodings to recommendation. GRID (Ju et al., 2025) provides a unified implementation and evaluation of multiple SID tokenization methods, enabling systematic comparisons of quantization algorithms, encoders, and codebook configurations. The second line of research focuses on the architecture (Gao et al., 2026; Hou et al., 2026b; Zhai et al., 2024; Zheng et al., 2024) and optimization (Deng et al., 2025; Lin et al., 2025; Team et al., 2026) of generative recommenders. HSTU (Zhai et al., 2024) introduces a scalable sequential transduction architecture for high-cardinality user behaviors. OneRec (Deng et al., 2025) further unifies retrieval and ranking through a MoE-based encoder-decoder, session-wise generation, and preference alignment. More recently, MemGen-GR (Ding et al., 2026) analyzes the memorization–generalization trade-off and adaptively ensembles ID-based and generative recommenders.

LLMs Post-Training and Distillation As a widely used pre-training method, supervised fine-tuning (SFT) (Chen et al., 2025; Zhang et al., 2025) applies maximum-likelihood learning to instruction–demonstration data, enabling stable instruction following and basic reasoning formats. However, Some studies (Chu et al., 2025) indicates that SFT often memorizes training examples, leading to weak generalization to out-of-distribution settings, motivating reinforcement learning (RL) (Guo et al., 2025; Rafailov et al., 2023; Yu et al., 2026)–based post-training for better domain generalization. Proximal policy optimization (PPO) (Schulman et al., 2017) improves a policy by maximizing expected reward while limiting drift from a reference model, and is effective for strengthening reasoning (Chen et al., 2025). Building on this RL foundation, DPO (Rafailov et al., 2023) learns from paired preferences, avoiding explicit reward modeling and in-policy rollouts as in PPO, which improves stability and efficiency. GRPO (Guo et al., 2025) follows PPO's on-policy RL setting but computes group-wise relative advantages from multiple samples under the same prompt, which better supports reasoning-oriented optimization. Beyond reinforcement learning and preference optimization, knowledge distillation (KD) (Gu et al., 2024; Hinton, 2014; Yang et al., 2025; 2026) provides a complementary post-training paradigm for transferring capabilities from a stronger teacher to a student model. For autoregressive sequence models, sequence-level KD (Kim & Rush, 2016) trains the student on teacher-generated sequences. With the rise of LLMs, conventional off-policy KD cannot align training with the student's inference distribution, motivating on-policy distillation over student-generated trajectories. GKD (Agarwal et al., 2024) generalizes this formulation with flexible divergence objectives and mixtures of on-policy and off-policy data. Rethinking OPD (Li et al., 2026a) identifies compatible teacher–student reasoning patterns and novel teacher capabilities as key conditions, and proposes practical recipes for recovering failed training.

3 METHOD

3.1 PROBLEM FORMULATION

Let $\mathcal{U}$ and $\mathcal{V}$ denote the user and item sets, respectively. For each user u, the chronological interaction sequence is $S_u = (v_1^u, \ldots, v_{T_u}^u)$. Each instance consists of a history $x_{u,t} = (v_1^u, \ldots, v_{t-1}^u)$ and a target item $y_{u,t} = v_t^u$. In generative recommendation, each item v is represented by an L-token semantic ID (SID), $\mathbf{z}^v = (z_1^v, \ldots, z_L^v)$, which is generated autoregressively:

$$p_\theta(v \mid x) = \prod_{l=1}^{L} p_\theta(z_l^v \mid x, \mathbf{z}_{<l}^v). \tag{1}$$

Motivated by transition-based analyses of memory and generalization (Ding et al., 2026), we distinguish interaction instances according to cross-user support for their final adjacent item transitions. Let $\mathcal{R}_{\mathrm{tr}}^{(-u)}$ contain all adjacent item transitions from the training histories of users other than u. An instance is labeled as memory if its final transition appears in this reference set:

$$m_{u,t} = \mathbb{I}\left[(v_{t-1}^u, v_t^u) \in \mathcal{R}_{\mathrm{tr}}^{(-u)}\right]. \tag{2}$$

The same criterion applies to training, validation, and test instances, with the reference sets constructed only from training histories. For the training set $\mathcal{D}$, instances with $m_{u,t} = 1$ form $\mathcal{D}_{\mathrm{mem}}$, and the remainder form $\mathcal{D}_{\mathrm{gen}}$. This partition measures cross-user support for the final transition; the Generalization subset does not necessarily imply compositional or out-of-distribution generalization. The labels are used for training supervision and offline subset evaluation. Inference uses a single student model and does not require the target-dependent memory labels. Given a student θ_0 trained on $\mathcal{D}$ and a same-architecture teacher ϕ trained on $\mathcal{D}_{\mathrm{mem}}$, our goal is to improve recommendation through selective memory distillation while retaining the backbone objective on the complete training set. MemOPSD first determines where memory knowledge should be learned and applied, and then transfers this knowledge along trajectories generated by the current student.

3.2 SELECTIVE MEMORY SPECIALIZATION AND TRANSFER

As illustrated in Fig. 2, we introduce selective memory specialization and transfer to determine the source and scope of memory supervision. A teacher trained on memory data specializes in recurring interaction patterns and provides additional supervision only to memory samples, while the student retains the backbone objective on the complete training set.

Student Training and Memory Specialization. For a batch $\mathcal{B}$, let $\mathcal{L}_{\mathrm{GR}}(\mathcal{B})$ denote the original backbone's supervised training loss, preserving its target-token scope, normalization scheme, and auxiliary loss terms where applicable. Its token-level supervision is computed using ground-truth target SIDs and their prefixes. Given the complete training set $\mathcal{D} = \mathcal{D}_{\mathrm{mem}} \cup \mathcal{D}_{\mathrm{gen}}$, we train the student and the memory-specialized teacher by minimizing this objective on their respective training data. The student is trained on the complete dataset $\mathcal{D}$ to obtain the initial parameters θ_0, allowing it to learn interaction patterns from both memory and generalization data. The teacher ϕ is trained exclusively on $\mathcal{D}_{\mathrm{mem}}$ to specialize in recurring interaction patterns.

The teacher and student use the same backbone architecture and SID vocabulary. We refer to knowledge transfer in this same-architecture teacher–student setting as self-distillation. The teacher is a memory-specialized counterpart of the student and remains frozen throughout post-training.

Selective Knowledge Transfer. During post-training, we retain the original backbone's supervised objective on the complete training set and apply additional teacher supervision only to memory instances. Specifically, memory instances receive both GR supervision and distillation, whereas generalization instances receive only GR supervision. This design strengthens recurring interaction knowledge where cross-user support exists, while avoiding directly forcing generalization samples to match the memory teacher's distribution.

For an original training pair (x_i, y_i), let m_i denote its memory label under the criterion above. This label remains fixed across all sampled trajectories, regardless of their generated SID tokens. Thus, the scope of teacher supervision is determined by the observed training instance, while the prefixes at which knowledge is transferred are generated by the student, as described next.

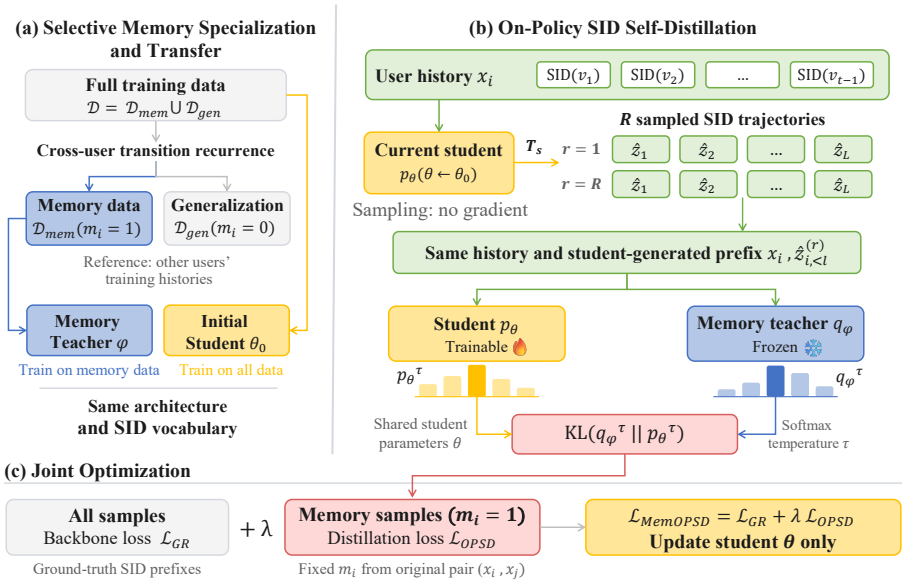


Figure 2: **Overview of MemOPSD.** A memory-specialized teacher provides token-level guidance on student-generated SID prefixes. The student retains the backbone objective on all training samples and receives additional on-policy distillation only on memory samples.

3.3 On-Policy SID Self-Distillation

Fixed-prefix knowledge distillation conditions the student on ground-truth or teacher-generated SID prefixes, whereas the student relies on its own predictions during autoregressive inference. This discrepancy can expose the student to prefix states not covered by fixed-prefix distillation, allowing early SID errors to affect subsequent generation. To address this issue, we follow on-policy distillation (Agarwal et al., 2024) and transfer memory knowledge directly along trajectories sampled from the student's current rollout policy.

Student Rollout. Using i to index a context–target pair (x_i, y_i), the student independently samples R complete SID trajectories given the context x_i:

$$\hat{\mathbf{z}}_i^{(r)} = \left(\hat{z}_{i,1}^{(r)}, \ldots, \hat{z}_{i,L}^{(r)}\right) \sim \pi_{\theta,T_s}(\cdot \mid x_i), \qquad r = 1, \ldots, R, \tag{3}$$

where π_{θ,T_s} denotes the autoregressive rollout policy induced by the student's temperature-scaled token distributions. Specifically, at generation step l, the next SID token is sampled as

$$\hat{z}_{i,l}^{(r)} \sim \mathrm{Cat}\left[\mathrm{Softmax}\left(\frac{\mathbf{g}_\theta\left(x_i, \hat{\mathbf{z}}_{i,<l}^{(r)}\right)}{T_s}\right)\right], \tag{4}$$

where $\mathbf{g}_\theta$ denotes the student logits and T_s is the rollout temperature. The sampled token is appended to the current prefix, and this process repeats until all L SID tokens are generated. The R trajectories are sampled independently, although duplicate trajectories are allowed.

The rollout policy is determined by the current student parameters and therefore evolves as the student is updated. Here, on-policy refers to this temperature-scaled sampling policy. Training rollouts use stochastic sampling, while evaluation retains the backbone's original inference-time decoding procedure.

Same-Prefix Teacher Guidance. The teacher does not generate additional SID sequences. Instead, we repeat each context R times and use the student rollouts as teacher-forcing sequences. At each position l, both models predict the next token conditioned on the same student-generated prefix $\hat{\mathbf{z}}_{i,<l}^{(r)}$. Their softened distributions are defined as

$$q_\phi^\tau\left(\cdot \mid x_i, \hat{\mathbf{z}}_{i,<l}^{(r)}\right) = \mathrm{Softmax}\left(\mathbf{g}_\phi\left(x_i, \hat{\mathbf{z}}_{i,<l}^{(r)}\right)/\tau\right), \tag{5}$$

and

$$p_\theta^\tau\left(\cdot \mid x_i, \hat{\mathbf{z}}_{i,<l}^{(r)}\right) = \mathrm{Softmax}\left(\mathbf{g}_\theta\left(x_i, \hat{\mathbf{z}}_{i,<l}^{(r)}\right)/\tau\right), \tag{6}$$

where $\mathbf{g}_\phi$ and $\mathbf{g}_\theta$ denote the teacher and student logits, respectively, and τ is the distillation temperature. The rollout temperature T_s controls trajectory sampling, whereas τ controls the distributions used for knowledge transfer. The teacher remains frozen and is evaluated without gradients, whereas the student performs a second forward pass with gradients enabled under the same sampled prefixes. We then minimize the token-wise discrepancy between the two distributions along the student-generated trajectories:

$$\mathcal{L}_{\mathrm{OPSD}}^{(i)} = \frac{\tau^2}{RL}\sum_{r=1}^{R}\sum_{l=1}^{L} D_{\mathrm{KL}}\left(q_\phi^\tau\left(\cdot \mid x_i, \hat{\mathbf{z}}_{i,<l}^{(r)}\right) \middle\| p_\theta^\tau\left(\cdot \mid x_i, \hat{\mathbf{z}}_{i,<l}^{(r)}\right)\right). \tag{7}$$

Gradients are propagated only through the student distribution. By querying the teacher at states actually visited by the student, this objective narrows the train–inference distribution gap and mitigates error propagation in autoregressive SID generation.

3.4 Joint Optimization

During post-training, the student is optimized using the backbone objective on all training instances and additional on-policy distillation on memory instances. Using the fixed memory labels introduced above, the memory-selective distillation loss for a batch $\mathcal{B}$ is defined as

$$\mathcal{L}_{\mathrm{OPSD}}(\mathcal{B}) = \frac{\sum_{i\in\mathcal{B}} m_i\, \mathcal{L}_{\mathrm{OPSD}}^{(i)}}{\max\left(1, \sum_{i\in\mathcal{B}} m_i\right)}. \tag{8}$$

Here, $\mathcal{L}_{\mathrm{OPSD}}^{(i)}$ is the per-instance distillation loss, averaged over sampled trajectories and SID positions, while $\mathcal{L}_{\mathrm{OPSD}}(\mathcal{B})$ further averages over memory instances in the batch. The denominator also handles batches without memory instances, for which the distillation term is zero. The student is optimized using the joint objective:

$$\mathcal{L}_{\mathrm{MemOPSD}}(\mathcal{B}) = \mathcal{L}_{\mathrm{GR}}(\mathcal{B}) + \lambda\, \mathcal{L}_{\mathrm{OPSD}}(\mathcal{B}), \tag{9}$$

where the backbone loss has a fixed coefficient of 1, and λ controls the strength of memory distillation. GR supervision is computed using ground-truth target SIDs and their prefixes, whereas KL distillation is evaluated along prefixes sampled from the student's current rollout policy. **At shared prefixes, the effective target strengthens preference for the observed token while preserving the teacher's relative preferences among alternatives (Appendix C).** During optimization, only the student parameters are updated. The teacher remains frozen, and gradients are not propagated through the discrete trajectory sampling process. Memory instances receive both backbone supervision and teacher guidance, whereas generalization instances receive only backbone supervision. This design does not guarantee conflict-free optimization or unchanged generalization performance. **The time complexity analysis is provided in Appendix D.**

4 Experiments

In this section, we conduct extensive experiments and answer the following research questions:

- **RQ1:** Does our proposed MemOPSD improve upon existing state-of-the-art recommenders across various experimental settings?
- **RQ2:** Are the key components in our MemOPSD delivering the expected performance gains?
- **RQ3:** How does MemOPSD affect SID branch probabilities and item ranking in a representative memory instance?
- **RQ4:** How do different hyperparameters affect MemOPSD?

4.1 Experiment Setup

4.1.1 Datasets

We evaluate MemOPSD on public datasets from the Amazon Reviews 2023 collection (Ding et al., 2026): Office Products, Video Games, and Industrial and Scientific. Appendix E.1 provides details.

Table 1: Overall recommendation performance on three Amazon Reviews 2023 datasets.

Method	Industrial and Scientific				Video Games				Office Products			
	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
ID-based Recommendation												
SASRec (ICDM 18)	0.0245	0.0385	0.0156	0.0201	0.0531	0.0841	0.0338	0.0437	0.0230	0.0338	0.0153	0.0188
Bert4Rec (CIKM 19)	0.0211	0.0338	0.0138	0.0178	0.0531	0.0827	0.0351	0.0446	0.0184	0.0273	0.0121	0.0149
FDSA (IJCAI 19)	0.0236	0.0360	0.0157	0.0197	0.0536	0.0831	0.0350	0.0445	0.0232	0.0332	0.0159	0.0191
S3Rec (CIKM 20)	0.0246	0.0385	0.0142	0.0187	0.0541	0.0865	0.0325	0.0430	0.0256	0.0376	0.0150	0.0189
CL4SRec (ICDE 22)	0.0211	0.0345	0.0136	0.0179	0.0528	0.0849	0.0338	0.0441	0.0225	0.0329	0.0148	0.0182
Generative Recommendation												
HSTU (ICML 24)	0.0253	0.0412	0.0140	0.0191	0.0577	0.0908	0.0330	0.0443	0.0259	0.0383	0.0152	0.0192
RPG (KDD 25)	0.0247	0.0371	0.0167	0.0207	0.0551	0.0897	0.0361	0.0466	0.0234	0.0321	0.0168	0.0197
TIGER (NeurIPS 23)	0.0265	0.0429	0.0172	0.0224	0.0532	0.0860	0.0346	0.0452	0.0288	0.0419	0.0199	0.0241
TIGER + MemOPSD	**0.0285***	**0.0441***	**0.0183***	**0.0232***	**0.0579***	**0.0915***	**0.0377***	**0.0485***	**0.0299***	**0.0432***	**0.0208***	**0.0251***
Improv. (%)	+7.55%	+2.80%	+6.40%	+3.57%	+8.83%	+6.40%	+8.96%	+7.30%	+3.82%	+3.10%	+4.52%	+4.15%
LETTER (CIKM 24)	0.0274	0.0423	0.0177	0.0225	0.0534	0.0839	0.0350	0.0448	0.0292	0.0427	0.0203	0.0246
LETTER + MemOPSD	**0.0287***	**0.0441***	**0.0186***	**0.0236***	**0.0578***	**0.0895***	**0.0377***	**0.0480***	**0.0301***	**0.0433***	**0.0209***	**0.0252***
Improv. (%)	+4.74%	+4.26%	+5.08%	+4.89%	+8.24%	+6.67%	+7.71%	+7.14%	+3.08%	+1.41%	+2.96%	+2.44%
CARE (WWW 26)	0.0273	0.0426	0.0179	0.0228	0.0572	0.0894	0.0375	0.0478	0.0300	0.0433	0.0207	0.0250
CARE + MemOPSD	**0.0285***	**0.0451***	**0.0187***	**0.0240***	**0.0588***	**0.0918***	**0.0388***	**0.0493***	**0.0306***	**0.0439***	**0.0213***	**0.0255***
Improv. (%)	+4.40%	+5.87%	+4.47%	+5.26%	+2.80%	+2.68%	+3.47%	+3.14%	+2.00%	+1.39%	+2.90%	+2.00%
LatentR3 (ICLR 26)	0.0274	0.0423	0.0176	0.0224	0.0547	0.0868	0.0353	0.0456	0.0299	0.0432	0.0205	0.0248
LatentR3 + MemOPSD	**0.0285***	**0.0446***	**0.0186***	**0.0238***	**0.0591***	**0.0916***	**0.0385***	**0.0490***	**0.0307***	**0.0441***	**0.0213***	**0.0256***
Improv. (%)	+4.01%	+5.44%	+5.68%	+6.25%	+8.04%	+5.53%	+9.07%	+7.46%	+2.68%	+2.08%	+3.90%	+3.23%

The superscript * indicates a statistically significant improvement ($p < 0.05$).

4.1.2 BASELINES

We compare MemOPSD with representative ID-based and generative recommendation methods, and we further instantiate MemOPSD on top of four generative recommendation backbones. The full list of baselines and their implementation details are described in Appendix E.2.

4.1.3 IMPLEMENTATION DETAILS

All models are implemented within the MemGen-GR (Ding et al., 2026) framework using consistent data preprocessing, sequence splitting, and evaluation protocols. We tune MemOPSD hyperparameters on the validation set and provide complete model and training configurations in Appendix E.3.

4.2 PERFORMANCE COMPARISONS (RQ1)

4.2.1 OVERALL PERFORMANCE

Table 1 shows the performance improvements of MemOPSD on three public datasets. We further discuss interesting observations in the results and provide possible explanations. All experimental results are averaged over five runs.

- **ID-based Recommendation.** ID-based methods generally underperform the evaluated SID-based generative backbones. For example, on Office Products, the best ID-based method achieves 0.0191 NDCG@10, whereas generative backbones achieve 0.0241–0.0250.
- **Generative Recommendation.** By incorporating semantic representations and autoregressive generation, generative recommenders generally outperform ID-based methods. However, no backbone consistently dominates across all metrics. Their ability to exploit recurring interaction knowledge is examined through the memory and generalization subset results in Appendix F.1.
- **Advantages of MemOPSD.** MemOPSD consistently improves the four generative backbones, with relative gains ranging from 1.39% to 9.07%. Improvements in both Recall and NDCG show that it enhances target-item retrieval and ranking quality. Its consistent gains across TIGER, LETTER, CARE, and LatentR3 demonstrate its effectiveness across the four evaluated GR backbones.

4.2.2 MEMORY ENHANCEMENT AND SCALING STUDY

We further evaluate memory and generalization performance in Appendix F.1 and the effect of model size on MemOPSD in Appendix F.2.

Table 2: Ablation study on three Amazon Reviews 2023 datasets.

Method	Industrial and Scientific				Video Games				Office Products			
	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
TIGER (NeurIPS 23)	0.0265	0.0429	0.0172	0.0224	0.0532	0.0860	0.0346	0.0452	0.0288	0.0419	0.0199	0.0241
TIGER w/o GR	0.0257	0.0409	0.0166	0.0215	0.0557	0.0871	0.0361	0.0462	0.0277	0.0408	0.0191	0.0230
TIGER w/o OPSD	0.0278	0.0435	0.0177	0.0226	0.0538	0.0865	0.0351	0.0456	0.0290	0.0418	0.0200	0.0242
TIGER w/ KD	0.0278	0.0435	0.0179	0.0229	0.0542	0.0865	0.0354	0.0458	0.0287	0.0419	0.0199	0.0241
TIGER + MemOPSD	**0.0285***	**0.0441***	**0.0183***	**0.0232***	**0.0579***	**0.0915***	**0.0377***	**0.0485***	**0.0299***	**0.0432***	**0.0208***	**0.0251***
LETTER (CIKM 24)	0.0274	0.0423	0.0177	0.0225	0.0534	0.0839	0.0350	0.0448	0.0292	0.0427	0.0203	0.0246
LETTER w/o GR	0.0274	0.0424	0.0178	0.0227	0.0551	0.0847	0.0359	0.0454	0.0280	0.0410	0.0197	0.0236
LETTER w/o OPSD	0.0281	0.0438	0.0183	0.0233	0.0567	0.0880	0.0369	0.0468	0.0295	0.0428	0.0205	0.0247
LETTER w/ KD	0.0278	0.0436	0.0181	0.0230	0.0560	0.0876	0.0365	0.0467	0.0294	0.0426	0.0203	0.0246
LETTER + MemOPSD	**0.0287***	**0.0441***	**0.0186***	**0.0236***	**0.0578***	**0.0895***	**0.0377***	**0.0480***	**0.0301***	**0.0433***	**0.0209***	**0.0252***
CARE (WWW 26)	0.0273	0.0426	0.0179	0.0228	0.0572	0.0894	0.0375	0.0478	0.0300	0.0433	0.0207	0.0250
CARE w/o GR	0.0263	0.0426	0.0172	0.0225	0.0571	0.0888	0.0374	0.0476	0.0288	0.0416	0.0195	0.0243
CARE w/o OPSD	0.0283	0.0441	0.0186	0.0237	0.0573	0.0895	0.0375	0.0479	0.0300	0.0431	0.0207	0.0248
CARE w/ KD	0.0268	0.0420	0.0175	0.0224	0.0567	0.0891	0.0371	0.0475	0.0298	0.0428	0.0205	0.0247
CARE + MemOPSD	**0.0285***	**0.0451***	**0.0187***	**0.0240***	**0.0588***	**0.0918***	**0.0388***	**0.0493***	**0.0306***	**0.0439***	**0.0213***	**0.0255***
LatentR3 (ICLR 26)	0.0274	0.0423	0.0176	0.0224	0.0547	0.0868	0.0353	0.0456	0.0299	0.0432	0.0205	0.0248
LatentR3 w/o GR	0.0267	0.0415	0.0173	0.0221	0.0565	0.0873	0.0369	0.0468	0.0287	0.0418	0.0198	0.0240
LatentR3 w/o OPSD	0.0277	0.0435	0.0178	0.0230	0.0565	0.0893	0.0365	0.0470	0.0299	0.0432	0.0206	0.0249
LatentR3 w/ KD	0.0279	0.0438	0.0180	0.0232	0.0575	0.0895	0.0373	0.0475	0.0297	0.0431	0.0204	0.0247
LatentR3 + MemOPSD	**0.0285***	**0.0446***	**0.0186***	**0.0238***	**0.0591***	**0.0916***	**0.0385***	**0.0490***	**0.0307***	**0.0441***	**0.0213***	**0.0256***

4.3 Ablation Study (RQ2)

4.3.1 Impact of key components

In this section, we assess the impact of key components in MemOPSD and provide potential explanations in Table 2. The following model variants are evaluated:

- **MemOPSD w/o GR:** Removes the original backbone loss.
- **MemOPSD w/o OPSD:** Removes the KL distillation loss.
- **MemOPSD w/ KD:** Replaces student-generated prefixes with ground-truth SID prefixes while retaining the same memory-specialized teacher, distillation scope, and loss normalization.

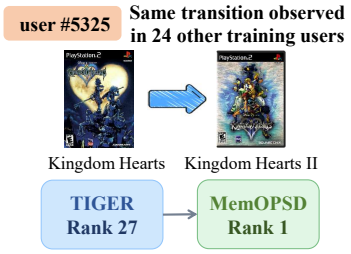
First, removing the backbone loss consistently reduces performance. On Office Products, NDCG@10 decreases by 4.71%–8.37%, highlighting the importance of retaining the backbone's original objective during distillation. Second, removing the KL distillation loss reduces NDCG@10 across all four backbones on Video Games and Office Products by 2.50%–5.98% and 1.98%–3.59%, respectively, demonstrating the benefit of teacher supervision beyond the backbone objective alone. Third, replacing on-policy self-distillation with ground-truth-prefix KD generally reduces performance and can even underperform the original backbone. For example, on Industrial and Scientific, CARE achieves an NDCG@10 of 0.0224 with ground-truth-prefix KD, compared with 0.0228 for the original backbone and 0.0240 with MemOPSD. This comparison supports the effectiveness of on-policy SID self-distillation over ground-truth-prefix KD under the evaluated settings. **Overall, retaining the backbone objective and incorporating on-policy KL distillation jointly contribute to the effectiveness of MemOPSD.**

4.3.2 Additional Ablation Studies

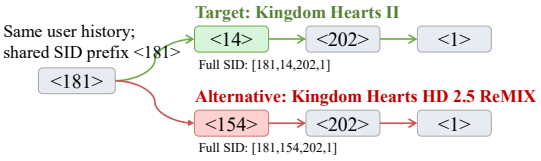
We further examine teacher training data and distillation scope in Appendix F.3, and evaluate different language model families and parameter scales in Appendix F.4.

4.4 Case Study (RQ3)

Fig. 3 examines a memory instance from Video Games (sample #5325). The user's last interaction is *Kingdom Hearts*, and the target next item is *Kingdom Hearts II*. The same adjacent transition appears in the training sequences of 24 other users, providing evidence of a recurring interaction pattern. TIGER ranks the target item 27th, whereas MemOPSD raises it to 1st. Under the same user history and SID prefix $\langle 181 \rangle$, the target and the alternative item, *Kingdom Hearts HD 2.5 ReMIX*, diverge at the next tokens $\langle 14 \rangle$ and $\langle 154 \rangle$, respectively. TIGER assigns probabilities of 11.25% and 56.54% to these two branches, while the memory-specialized teacher favors the target with probabilities of 34.94% and 8.90%. MemOPSD increases the target-branch probability to 59.06% and reduces the alternative-branch probability to 10.12%. The inspected stochastic rollout still selects the alternative



(a) MemOPSD improves the target item's rank from 27 to 1.



Next Token	TIGER	Memory Teacher	MemOPSD
<14> (target)	11.25%	34.94%	59.06%
<154> (alternative)	56.54%	8.90%	10.12%

(b) MemOPSD favors the target SID branch over the alternative.

Figure 3: **Case study on Video Games.**

branch, so the observed gains concern conditional probabilities and ranking. **Overall, this case illustrates how memory-guided on-policy distillation can help distinguish related items at an early SID branch and improve recommendation for recurring interaction patterns.** In addition, Appendix F.5 provides a case study of memory–generalization gradient conflicts during MemOPSD post-training and their association with recommendation performance.

4.5 Hyperparameter Sensitivity (RQ4)

We investigatethe distillation weight $\lambda \in \{0.1, 0.2, 0.3\}$ and rollout count $R \in \{1, 2, 4\}$. Fig. 4 shows that increasing λ generally benefits TIGER, CARE, and LatentR3 on Video Games, while LETTER changes little. Smaller weights generally favor TIGER, LETTER, and CARE on Industrial and Scientific and Office Products. LatentR3 remains relatively stable on Industrial and Scientific and typically performs best at $\lambda = 0.2$ on Office Products. Fig. 5 shows that increasing the rollout count does not consistently improve performance. CARE benefits from additional rollouts on Video Games, while LatentR3 performs best with $R = 1$ on Industrial and Scientific. For TIGER on Industrial and Scientific, $R = 1$ and $R = 4$ yield the same NDCG@10, while $R = 4$ achieves higher Recall@10. On Office Products, the preferred count varies across backbones and metrics. **The best hyperparameter settings selected on the validation set are reported in Table 5.**

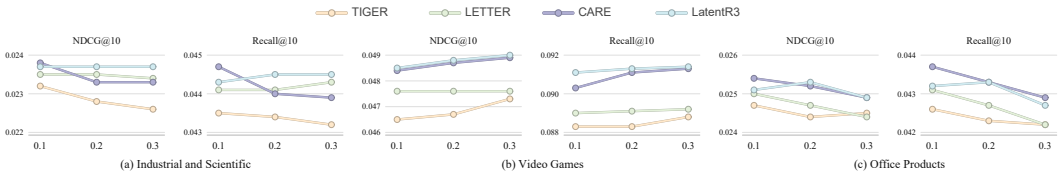


Figure 4: Effect of the distillation weight λ on recommendation performance across all datasets.

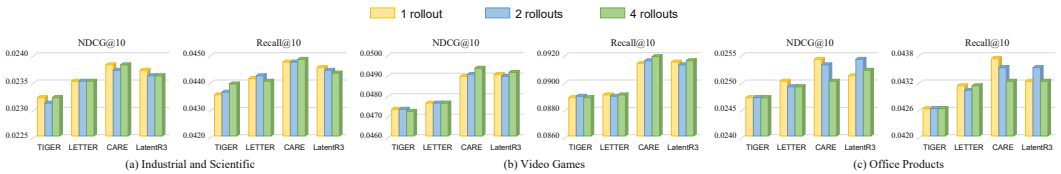


Figure 5: Effect of the rollout count R on recommendation performance across all datasets.

5 Conclusion

We revisit generative recommendation from a data-centric perspective, distinguishing memory and generalization instances by cross-user recurrence of target transitions. Motivated by their distinct learning behaviors, **MemOPSD** applies on-policy self-distillation from a memory-specialized teacher along student-generated SID trajectories only to memory samples, retaining the backbone objective on all samples. Experiments on three datasets and four GR backbones show consistent overall and memory improvements while largely preserving generalization performance.

6 AI USE STATEMENT

We used AI tools to assist with code writing and to improve the grammar, clarity, and readability of text written by the authors. AI tools were not used to create figures, generate synthetic data, or draft original sections of the manuscript. The authors take full responsibility for the final manuscript and the associated code, including all AI-assisted contributions.

ETHICS STATEMENT

This work studies generative recommendation through offline experiments on three publicly available subsets of the Amazon Reviews 2023 dataset. User interaction data may reflect existing popularity and exposure biases, and strengthening the learning of recurring interaction patterns may reinforce such biases. The reported accuracy improvements do not establish fairness or privacy guarantees. Deployment beyond the evaluated benchmark settings should therefore include additional assessments of user privacy, exposure fairness, and potential feedback loops.

REPRODUCIBILITY STATEMENT

We provide our code at `https://anonymous.4open.science/r/MemOPSD-0627`. The released code is fully consistent with the methods and experimental settings described in this paper and enables full reproduction of the reported results.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*, pp. 21246–21263, 2024.

Yimeng Bai, Chang Liu, Yang Zhang, Dingxian Wang, Frank Yang, Andrew Rabinovich, Wenge Rong, and Fuli Feng. Bi-level optimization for generative recommendation: Bridging tokenization and generation. In *SIGIR*, pp. 53–64, 2026.

Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. Sft or rl? an early investigation into training r1-like reasoning large vision-language models. *arXiv preprint arXiv:2504.11468*, 2025.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. In *ICML*, 2025.

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

Yijie Ding, Zitian Guo, Jiacheng Li, Letian Peng, Shuai Shao, Wei Shao, Xiaoqiang Luo, Luke Simon, Jingbo Shang, Julian McAuley, et al. On the memorization and generalization of generative recommendation. In *KDD*, pp. 921–932, 2026.

Tianqi Gao, Chengkai Huang, Zihan Wang, Cao Liu, Ke Zeng, and Lina Yao. Factorized latent reasoning for llm-based recommendation. In *SIGIR*, pp. 392–402, 2026.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Recsys*, pp. 299–315, 2022.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *ICLR*, pp. 32694–32717, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Yingzhi He, Yan Sun, Junfei Tan, Yuxin Chen, Xiaoyu Kong, Chunxu Shen, Xiang Wang, An Zhang, and Tat-Seng Chua. Reasoning over semantic ids enhances generative recommendation. In *KDD*, pp. 1638–1649, 2026.

G Hinton. Distilling the knowledge in a neural network. In *Deep Learning and Representation Learning Workshop in Conjunction with NIPS*, 2014.

Min Hou, Le Wu, Yuxin Liao, Yonghui Yang, Zhen Zhang, Yu Wang, Changlong Zheng, Han Wu, and Richang Hong. A survey on generative recommendation: Data, model, and tasks. *AI Open*, 2026a.

Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. Learning vector-qantized item representation for transferablesequential recommenders. In *WWW*, pp. 1162–1171, 2023.

Yupeng Hou, Jiacheng Li, Ashley Shin, Jinsung Jeon, Abhishek Santhanam, Wei Shao, Kaveh Hassani, Ning Yao, and Julian McAuley. Generating long semantic ids in parallel for recommendation. In *KDD*, pp. 956–966, 2025.

Yupeng Hou, Haven Kim, Clark Mingxuan Ju, Eduardo Escoto, Neil Shah, and Julian McAuley. Expressiveness limits of autoregressive semantic id generation in generative recommendation. *arXiv preprint arXiv:2605.06331*, 2026b.

Clark Mingxuan Ju, Liam Collins, Leonardo Neves, Bhuvesh Kumar, Louis Yufeng Wang, Tong Zhao, and Neil Shah. Generative recommendation with semantic ids: A practitioner's handbook. In *CIKM*, pp. 6420–6425, 2025.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *ICDM*, pp. 197–206, 2018.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *EMNLP*, pp. 1317–1327, 2016.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. In *ICML 2026 Workshop on Foundations of Deep Generative Models: Understanding Memorization, Generalization, and Reasoning*, 2026a.

Zhao Li, FengYang Qi, Chuanyu Xu, Tao Zhang, Chengfu Huo, and Peng Zhang. Lsig: Long semantic ids for generative recommendation. In *WWW*, pp. 7779–7788, 2026b.

Xinyu Lin, Haihan Shi, Wenjie Wang, Fuli Feng, Qifan Wang, See-Kiong Ng, and Tat-Seng Chua. Order-agnostic identifier for large language model-based generative recommendation. In *SIGIR*, pp. 1923–1933, 2025.

Xinyu Lin, Zhuosong Jiang, Zixiao Suo, Siqin Wang, Hanqing Zeng, Hanchao Yu, Yinglong Xia, Jiang Zhang, Aashu Singh, Fei Liu, et al. Self-evolving memory for generative recommendation. *arXiv preprint arXiv:2609.15598*, 2026a.

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

Enze Liu, Bowen Zheng, Cheng Ling, Lantao Hu, Han Li, and Wayne Xin Zhao. Generative recommender with end-to-end learnable item tokenization. In *SIGIR*, pp. 729–739, 2025.

Enze Liu, Zhen Tian, and Wayne Xin Zhao. Collaborative memory augmentation for generative recommendation. In *KDD*, pp. 3164–3175, 2026a.

Jingzhe Liu, Liam Collins, Jiliang Tang, Tong Zhao, Neil Shah, and Clark Mingxuan Ju. Understanding generative recommendation with semantic ids from a model-scaling view. In *KDD*, 2026b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *ACL Findings*, pp. 1864–1874, 2022.

Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. Contrastive learning for representation degeneration problem in sequential recommendation. In *WSDM*, pp. 813–823, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, volume 36, pp. 53728–53741, 2023.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. In *NeurIPS*, volume 36, pp. 10299–10315, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Teng Shi, Chenglei Shen, Weijie Yu, Shen Nie, Chongxuan Li, Xiao Zhang, Ming He, Yan Han, and Jun Xu. Llada-rec: Discrete diffusion for parallel semantic id generation in generative recommendation. *arXiv preprint arXiv:2511.06254*, 2025.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, pp. 1441–1450, 2019.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, volume 30, 2017.

Qiang Wan, Ze Yang, Dawei Yang, Ying Fan, Xin Yan, Siyang Liu, Yicong Liu, Chenwei Zhang, Wei Xu, Jiahao Qin, et al. R3-vae: Reference vector-guided rating residual quantization vae for generative recommendation. *arXiv preprint arXiv:2604.11440*, 2026.

Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation. In *CIKM*, pp. 2400–2409, 2024.

Yu Wang, Lei Sang, Yi Zhang, and Yiwen Zhang. Intent representation learning with large language model for recommendation. In *SIGIR*, pp. 1870–1879, 2025.

Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. Contrastive learning for sequential recommendation. In *ICDE*, pp. 1259–1273, 2022.

Chuanpeng Yang, Yao Zhu, Wang Lu, Yidong Wang, Qian Chen, Chenlong Gao, Bingjie Yan, and Yiqiang Chen. Survey on knowledge distillation for large language models: Methods, evaluation, and application. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 16(6):1–27, 2025.

Shenzhi Yang, Guangcheng Zhu, Bowen Song, Haobo Wang, Mingxuan Xia, Xing Zheng, Yingfan Ma, Zhongqi Chen, Weiqiang Wang, and Gang Chen. Oprd: On-policy representation distillation. *arXiv preprint arXiv:2606.06021*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113222–113244, 2026.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. In *ICML*, pp. 58484–58509, 2024.

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

Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. Collm: Integrating collaborative embeddings into large language models for recommendation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 37(5):2329–2340, 2025.

Yang Zhang, Wenxin Xu, Xiaoyan Zhao, Wenjie Wang, Fuli Feng, Xiangnan He, and Tat-Seng Chua. Reinforced latent reasoning for llm-based recommendation. In *ICLR*, 2026.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *ICDE*, pp. 1435–1448, 2024.

Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM*, pp. 1893–1902, 2020.

Appendix

A Memory–Generalization Data Partition

For each user's chronological interaction sequence, we reserve the last interaction for testing, the second-to-last for validation, and the remaining interactions for training. We then classify each instance by its final adjacent transition, from the last history item to the target: it belongs to the **memory subset** if the same ordered item pair appears in another user's training sequence, and to the **generalization subset** otherwise. As illustrated in Fig. 6, a phone-to-charger transition found in another user's training sequence is classified as memory data, whereas a football-to-cap transition absent from other users' training sequences is classified as generalization data.

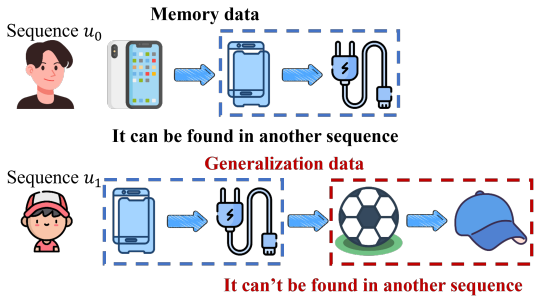


Figure 6: **Illustration of the memory–generalization data partition.**

B Memory Teacher Diagnostics

Evaluation setup. We compare standard TIGER trained on the full training set with Memory-TIGER (MT), a same-architecture model trained exclusively on memory data. Both models are evaluated on the same full test sets using identical subset labels, SID mappings, and decoding settings. All comparisons in this section use paired diagnostic predictions on the same test instances. The TIGER scores in this diagnostic evaluation differ slightly from those reported in Table 7; the differences between TIGER and MT reported here are computed within the diagnostic evaluation.

Specialization trade-off. As shown in Fig. 1(a), memory specialization improves NDCG@10 on the Memory subset but reduces Generalization and Overall performance across all three datasets. This comparison motivates selective transfer of the memory teacher's strengths, rather than uniform imitation across all training instances.

Differences in retrieved target items. We compare the distinct memory target items successfully retrieved by each model. A target item is counted once if it is ranked within the Top-10 predictions for at least one Memory test instance in which it is the ground-truth next item. Fig. 7 partitions the union of the two models' hit-item sets into targets hit by TIGER only, both models, and MT only. MT-only targets outnumber TIGER-only targets across all datasets, indicating that MT successfully retrieves a larger set of distinct memory target items. Nevertheless, both models have exclusive hits, so neither hit-item set contains the other. This analysis characterizes differences in target-item coverage; shared hits do not necessarily correspond to the same successfully predicted test instances.

C Theoretical Analysis

We analyze mean token-level cross-entropy (CE) with unit temperature and fixed token counts. Let $b = |B|$, $n_M = \sum_i m_i > 0$, and $\rho = n_M/b$. Denote ground-truth and sampled-prefix states by s^*_{it} and $\hat{s}_{irl}$, each including the history, and the target-token one-hot vector by $e_{y_{it}}$. CE covers K positions and distillation covers L SID positions; for TIGER, $K = L+1$ includes EOS. The teacher and sampled trajectories are fixed during differentiation.

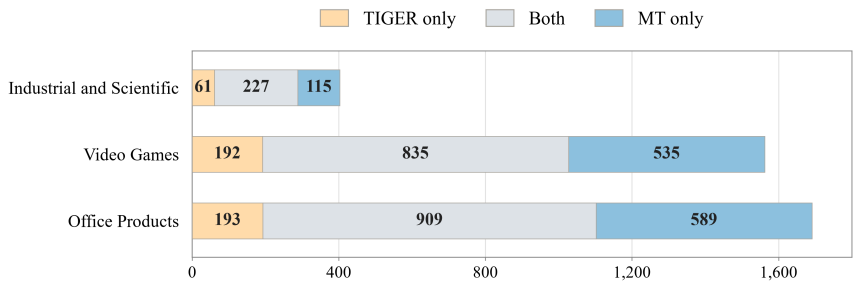


Figure 7: Overlap of distinct memory target items successfully retrieved at Top-10. A target item is counted once per model if it is correctly retrieved for at least one Memory test instance. Each bar partitions the union of the two models' hit-item sets into targets hit by TIGER only, both models, and Memory-TIGER (MT) only.

Parameter gradient. Let $J_\theta(s) = \partial g_\theta(s)/\partial\theta$, and abbreviate the next-token distribution at s as $p_\theta(s)$. The logit gradients of CE and temperature-scaled KL are $p - e_y$ and $\tau(p^\tau - q^\tau)$, respectively. The chain rule gives:

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\text{MemOPSD}} = \frac{1}{bK}\sum_{i,t} J_\theta(s^*_{it})^\top \left[p_\theta(s^*_{it}) - e_{y_{it}}\right] \\ + \frac{\lambda\tau}{n_M RL}\sum_i m_i \sum_{r,l} J_\theta(\hat{s}_{irl})^\top \left[p^\tau_\theta(\hat{s}_{irl}) - q^\tau_\phi(\hat{s}_{irl})\right].\end{aligned} \tag{10}$$

Thus, distillation adds a teacher-dependent correction at student-visited memory states, whereas CE retains supervision at ground-truth states. Their Jacobians generally differ. For $n_M = 0$, the KD term is zero; backbone auxiliary losses contribute their own gradients.

An effective target at shared prefixes. Set $\tau = 1$, as in our experiments, and assume deterministic forward passes. For one memory instance, consider a ground-truth SID prefix visited by N_l of the R rollouts. Write $\hat{a}_l = N_l/R$ and suppress the state in p, q. Its CE and KD coefficients satisfy:

$$\alpha = \frac{1}{bK}, \qquad \beta = \frac{\lambda N_l}{n_M RL}, \qquad c = \frac{\beta}{\alpha} = \frac{\lambda K}{\rho L}\hat{a}_l. \tag{11}$$

The corresponding local contribution can be rewritten exactly as:

$$\begin{aligned}\ell_s &= \alpha H(e_y, p) + \beta D_{\text{KL}}(q\|p) = (\alpha+\beta)H(t,p) + C, \\ t &= \frac{e_y + cq}{1+c}, \qquad \nabla_g \ell_s = (\alpha+\beta)(p-t),\end{aligned} \tag{12}$$

where H denotes cross-entropy and C is independent of p. MemOPSD therefore combines the observed label with the memory teacher's distribution, with a relative weight determined by memory normalization and prefix visitation rather than by λ alone.

Label anchoring and teacher structure. If this conditional distribution is optimized independently on the probability simplex, its optimum is $p^* = t$. For $c > 0$ and positive teacher probabilities, this implies:

$$p^*_y - q_y = \frac{1-q_y}{1+c} \geq 0, \qquad \frac{p^*_u}{p^*_v} = \frac{q_u}{q_v} \quad (u, v \neq y). \tag{13}$$

Hence, the local optimum can assign more probability to the correct token than the teacher while retaining the teacher's relative preferences among alternatives. Merely rescaling CE does not encode these teacher-dependent preferences. Moreover,

$$p^*_y - p^*_v = \frac{1 + c(q_y - q_v)}{1+c}. \tag{14}$$

Even when the teacher prefers a rival ($q_v > q_y$), the effective target still favors the observed token whenever $c < 1/(q_v - q_y)$. Retaining CE therefore permits correction of teacher preferences instead of unrestricted imitation. **Overall, this analysis highlights advantage in selectively transferring memory knowledge at student-visited states while retaining ground-truth supervision, which can counteract incorrect teacher preferences at shared prefixes.**

D TIME COMPLEXITY ANALYSIS

We analyze model computation after SID construction and memory labeling. Let $N = |\mathcal{D}|$, $N_{\text{mem}} = |\mathcal{D}_{\text{mem}}| \leq N$, n be the maximum encoder input length in tokens, L the SID length, and R the rollout count. Model width, depth, token-vocabulary size, and auxiliary tokens per SID position are fixed.

SFT Model Training. Here, SFT refers to initial student and teacher training using the corresponding backbone objective $\mathcal{L}_{\text{GR}}$. With dense attention, teacher forcing and backpropagation cost $\mathcal{O}(n^2 + nL + L^2)$ per instance. Training the student and memory-specialized teacher for E_s and E_t epochs, respectively,

$$T_{\text{SFT}} = \mathcal{O}\big((E_s N + E_t N_{\text{mem}})(n^2 + nL + L^2)\big)\,. \tag{15}$$

MemOPSD Training. Each instance incurs GR supervision, R student rollouts, and same-prefix student/teacher passes. The implementation recomputes decoder prefixes without a key–value cache, yielding cubic dependence on L. Memory masking occurs after full-batch computation, so cost scales with N, not N_{mem}. Over E_p post-training epochs,

$$T_{\text{MemOPSD}} = \mathcal{O}\big(E_p N R(n^2 + nL^2 + L^3)\big)\,. \tag{16}$$

For bounded n and L, each post-training epoch costs $\mathcal{O}(NR)$, and the full schedule costs $\mathcal{O}(E_s N + E_t N_{\text{mem}} + E_p NR)$, hence $\mathcal{O}(N)$ for fixed epoch counts and R. Reusing a student checkpoint avoids repeating its SFT cost.

Inference. Only the optimized student is used, with unchanged architecture and decoding procedure; inference complexity therefore matches the backbone.

Empirical Runtime. Table 3 reports per-epoch times and stage totals calculated from the corresponding epoch counts. The selected $R = 1$ MemOPSD runs require 5–26 post-training epochs, compared with 108–300 epochs for student SFT. Teacher training time is excluded from this comparison. **Overall, MemOPSD preserves linear training complexity in the dataset size under fixed model dimensions, sequence lengths, epoch counts, and rollout count, without increasing inference complexity. In the reported $R = 1$ runs, the post-training stage takes less time than initial student training; this comparison excludes teacher training.**

Table 3: Runtime of initial student training and MemOPSD post-training.

Dataset	Backbone	SFT			MemOPSD ($R = 1$)		
		s/epoch	Epochs	Total (min)	s/epoch	Epochs	Total (min)
Industrial and Scientific	TIGER	29	108	52	82	15	21
	LETTER	28	144	67	81	8	11
	CARE	33	116	64	85	5	7
	LatentR3	43	140	100	125	10	21
Video Games	TIGER	60	128	128	242	9	36
	LETTER	58	120	116	241	21	84
	CARE	67	166	185	254	9	38
	LatentR3	89	155	230	253	26	110
Office Products	TIGER	125	300	625	354	15	89
	LETTER	123	300	615	349	10	58
	CARE	144	300	720	369	25	154
	LatentR3	187	300	935	548	10	91

E DETAILS OF EXPERIMENT SETUP

E.1 DATASETS

We conduct experiments on three public datasets from the Amazon Reviews 2023 collection (Ding et al., 2026), including Office Products (Office), Video Games (Games), and Industrial and Scientific (Science). The processed statistics are reported in Table 4. Office contains 223,309 users, 77,552 items, and 1,800,878 interactions. Games contains 94,763 users, 25,613 items, and 814,586 interactions. Science contains 50,986 users, 25,849 items, and 412,947 interactions.

Their average sequence lengths are 8.06, 8.60, and 8.10, respectively. The sparsity is computed as $1 - \#\text{Interactions}/(\#\text{Users} \times \#\text{Items})$. We adopt the standard leave-last-out data split, using the last and second-to-last items of each user sequence for testing and validation, respectively. Memory labels are assigned after data splitting using only training transitions, with the current user's contributions excluded for all splits. Validation and test targets are queried only for offline labeling and are never added to the reference index. Following MemGen-GR, we apply 5-core filtering and report Recall@K and NDCG@K for $K \in \{5, 10, 20, 50\}$. The checkpoint with the highest validation NDCG@10 is used for testing.

Table 4: Statistics of the processed Amazon Reviews 2023 datasets.

Dataset	#Users	#Items	#Interactions	Avg. Length	Sparsity
Office Products	223,309	77,552	1,800,878	8.06	99.99%
Video Games	94,763	25,613	814,586	8.60	99.97%
Industrial and Scientific	50,986	25,849	412,947	8.10	99.97%

E.2 Baselines

- **ID-based Recommendation.** SASRec (Kang & McAuley, 2018) models the user's historical behavior sequence using a self-attention mechanism for next-item prediction. Bert4Rec (Sun et al., 2019) adopts a bidirectional Transformer with masked item prediction to learn sequential user preferences. FDSA (Zhang et al., 2019) separately models item transitions and feature transitions to capture both sequential and side signals. S3Rec (Zhou et al., 2020) introduces self-supervised pre-training objectives to enhance sequential recommendation. CL4SRec (Xie et al., 2022) applies contrastive learning to sequential recommendation by augmenting user behavior sequences.
- **Generative Recommendation.** HSTU (Zhai et al., 2024) designs a hierarchical sequential transduction architecture to model behavior sequences for GR models. RPG (Hou et al., 2025) reformulates sequential recommendation as a generative retrieval process to improve item generation.
- **Backbones.** TIGER (Rajput et al., 2023) uses semantic item identifiers and a generative sequence-to-sequence framework for next-item recommendation. LETTER (Wang et al., 2024) enhances generative recommendation by learning transferable item representations for sequential user preference modeling. CARE (Lin et al., 2026b) incorporates collaborative and semantic signals into generative recommendation to improve sequential item generation. LatentR3 (Zhang et al., 2026) models user preferences in a latent generative space to support robust sequential recommendation.

E.3 Implementation Details

All models are implemented with MemGen-GR (Ding et al., 2026). For a fair comparison, we adopt lightweight Transformer configurations for both GR and ID-based methods. Specifically, the four GR backbones in the main experiments employ an encoder-decoder architecture with a two-layer encoder and a two-layer decoder. Table 1 reports the performance under this standard setting, followed by a detailed analysis of the main experimental observations. We encode item text using sentence-t5-base (Ni et al., 2022) and reduce the 768-dimensional embeddings to 32 dimensions with PCA (Wang et al., 2025). TIGER, CARE, and LatentR3 use three residual-quantization codebooks of size 256 and append a collision token, yielding four-token semantic IDs. LETTER retains its own identifier module with the same number and size of codebooks. In the main experiments, all four backbones use a lightweight T5 (Rajput et al., 2023) with two encoder and decoder layers, hidden size 128, feed-forward size 1,024, six attention heads, and dropout 0.1. The maximum history length and beam size are both 50. Detailed baseline implementations, including architectures, attention masks, model-specific objectives, and training procedures, are provided in our code.

Following the general setting (Ding et al., 2026; Hou et al., 2025; Rajput et al., 2023), we train the backbones with AdamW (Loshchilov & Hutter, 2019) using a learning rate of 10^{-3}, weight decay 0.05, train/validation/test batch sizes of 512/256/256, and gradient clipping at 1.0. Training lasts at most 300 epochs, with validation every four epochs, early-stopping patience 20, and random seed 2026. For MemOPSD, the backbone trained on the complete training set serves as the initial student, while its same-architecture counterpart trained exclusively on memory samples serves as the memory-specialized teacher. The backbone objective is retained on all training samples, with

additional on-policy distillation applied only to memory samples. We train for at most 30 epochs with patience 3, no warm-up, and learning rate 2×10^{-5}. The backbone loss coefficient, distillation temperature τ, and rollout temperature T_s are 1, 1.0, and 0.7, respectively. The distillation weight λ and rollout count R are selected on the validation set, with the selected values reported in Table 5. Their sensitivity is analyzed in Figs. 4 and 5.

Table 5: Hyperparameters for MemOPSD on three datasets.

Backbone	Industrial and Scientific		Video Games		Office Products	
	weight λ	rollouts R	weight λ	rollouts R	weight λ	rollouts R
TIGER	0.1	4	0.3	2	0.1	1
LETTER	0.1	2	0.3	1	0.1	1
CARE	0.1	4	0.3	4	0.1	1
LatentR3	0.2	1	0.3	4	0.2	2

F Extension of the main experiment

F.1 Memory and Generalization Performance

As shown in Table 7, MemOPSD improves Memory and Overall NDCG@10 across all four backbones and three datasets. For TIGER, Memory NDCG@10 increases from 0.2063 to 0.2241, from 0.1853 to 0.2085, and from 0.2595 to 0.2742, yielding relative gains of 8.63%, 12.52%, and 5.66% on Industrial and Scientific, Video Games, and Office Products, respectively. Generalization performance improves across all four backbones on Industrial and Scientific. On Video Games, it improves for LETTER and LatentR3, remains unchanged for TIGER at the reported precision, and decreases for CARE. On Office Products, it increases by 2.27% for TIGER, remains unchanged for LETTER at the reported precision, and decreases by 2.21% and 3.68% for CARE and LatentR3, respectively. These results suggest that memory enhancement can involve a trade-off with generalization performance in some settings. **Overall, MemOPSD consistently improves memory and overall performance, while its effects on generalization vary across datasets and backbones.**

Table 6: Statistics of the processed test sets.

Category	Industrial and Scientific		Video Games		Office Products	
	#Instances	Ratio	#Instances	Ratio	#Instances	Ratio
Memory	2,031	3.98%	13,599	14.35%	9,904	4.44%
Generalization	48,954	96.02%	81,163	85.65%	213,404	95.56%

Table 7: Overall, Memory, and Generalization NDCG@10 on the three datasets.

Model	Industrial and Scientific				Video Games				Office Products			
	Overall	Memory	General	M/G	Overall	Memory	General	M/G	Overall	Memory	General	M/G
SASRec	0.0201	0.2461	0.0107	23.00×	0.0437	0.1940	0.0185	10.49×	0.0188	0.2382	0.0086	27.70×
TIGER	0.0224	0.2063	0.0148	13.94×	0.0452	0.1853	0.0217	8.54×	0.0241	0.2595	0.0132	19.66×
+ MemOPSD	0.0232	0.2241	0.0149	15.04×	0.0485	0.2085	0.0217	9.61×	0.0251	0.2742	0.0135	20.31×
LETTER	0.0225	0.2117	0.0147	14.40×	0.0448	0.1847	0.0214	8.63×	0.0246	0.2614	0.0136	19.22×
+ MemOPSD	0.0236	0.2230	0.0153	14.58×	0.0480	0.1971	0.0230	8.57×	0.0252	0.2751	0.0136	20.23×
CARE	0.0228	0.2163	0.0148	14.61×	0.0478	0.1971	0.0228	8.64×	0.0250	0.2700	0.0136	19.85×
+ MemOPSD	0.0240	0.2340	0.0153	15.29×	0.0493	0.2095	0.0225	9.31×	0.0255	0.2892	0.0133	21.74×
LatentR3	0.0224	0.2157	0.0144	14.98×	0.0456	0.1851	0.0222	8.34×	0.0248	0.2665	0.0136	19.60×
+ MemOPSD	0.0238	0.2282	0.0153	14.92×	0.0490	0.2058	0.0227	9.07×	0.0256	0.2957	0.0131	22.57×

M/G denotes the ratio of Memory to Generalization performance.

F.2 Scaling study

Table 8 examines how model capacity affects the four backbones and their MemOPSD-enhanced variants. Increasing the encoder and decoder depths generally improves recommendation performance, but the gains are neither monotonic nor uniform across datasets. The 4.6M configuration

is often more effective on Industrial and Scientific and Video Games, whereas the 9.1M configuration tends to perform best on Office Products. Further increasing the depth can instead lead to saturation or slight degradation, as observed for LETTER and LatentR3 on Industrial and Scientific and for LatentR3 on Video Games. Nevertheless, when comparing the best score across the two larger configurations with the 2.4M setting, the MemOPSD-enhanced variants improve on most dataset–metric combinations. More importantly, at matched model sizes, MemOPSD generally outperforms its corresponding backbone, indicating that its improvements cannot be attributed solely to an increased parameter count. **Overall, MemOPSD remains effective across the evaluated model sizes, while increasing model capacity does not consistently improve performance across datasets and backbones.**

Table 8: Scaling study on three Amazon Reviews 2023 datasets.

Model	Size	Industrial and Scientific				Video Games				Office Products			
		R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
TIGER	2.4M	0.0265	0.0429	0.0172	0.0224	0.0532	0.0860	0.0346	0.0452	0.0288	0.0419	0.0199	0.0241
	4.6M	0.0286	0.0440	0.0185	0.0235	0.0554	0.0871	0.0362	0.0464	0.0315	0.0456	0.0218	0.0263
	9.1M	0.0257	0.0426	0.0164	0.0218	0.0556	0.0879	0.0360	0.0464	0.0317	0.0457	0.0221	0.0266
Improv. (%)		+7.92%	+2.56%	+7.56%	+4.91%	+4.51%	+2.21%	+4.62%	+2.65%	+10.07%	+9.07%	+11.06%	+10.37%
MemOPSD	2.4M	0.0285	0.0441	0.0183	0.0232	0.0579	0.0915	0.0377	0.0485	0.0299	0.0432	0.0208	0.0251
	4.6M	0.0290	0.0465	0.0190	0.0246	0.0600	0.0926	0.0390	0.0495	0.0317	0.0461	0.0219	0.0266
	9.1M	0.0281	0.0440	0.0180	0.0231	0.0572	0.0901	0.0370	0.0476	0.0323	0.0463	0.0226	0.0271
Improv. (%)		+1.75%	+5.44%	+3.83%	+6.03%	+3.63%	+1.20%	+3.45%	+2.06%	+8.03%	+7.18%	+8.65%	+7.97%
LETTER	2.4M	0.0274	0.0423	0.0177	0.0225	0.0534	0.0839	0.0350	0.0448	0.0292	0.0427	0.0203	0.0246
	4.6M	0.0267	0.0424	0.0175	0.0225	0.0519	0.0833	0.0338	0.0439	0.0314	0.0456	0.0219	0.0264
	9.1M	0.0259	0.0420	0.0167	0.0219	0.0550	0.0864	0.0359	0.0460	0.0314	0.0453	0.0219	0.0264
Improv. (%)		-2.55%	+0.24%	-1.13%	0.00%	+3.00%	+2.98%	+2.57%	+2.68%	+7.53%	+6.79%	+7.88%	+7.32%
MemOPSD	2.4M	0.0287	0.0441	0.0186	0.0236	0.0578	0.0895	0.0377	0.0480	0.0301	0.0433	0.0209	0.0252
	4.6M	0.0290	0.0465	0.0190	0.0246	0.0554	0.0881	0.0364	0.0469	0.0315	0.0457	0.0219	0.0266
	9.1M	0.0272	0.0439	0.0173	0.0227	0.0586	0.0907	0.0384	0.0487	0.0323	0.0462	0.0225	0.0271
Improv. (%)		+1.05%	+5.44%	+2.15%	+4.24%	+1.38%	+1.34%	+1.86%	+1.46%	+7.31%	+6.70%	+7.66%	+7.54%
CARE	2.4M	0.0273	0.0426	0.0179	0.0228	0.0572	0.0894	0.0375	0.0478	0.0300	0.0433	0.0207	0.0250
	4.6M	0.0281	0.0443	0.0183	0.0235	0.0572	0.0896	0.0375	0.0479	0.0316	0.0461	0.0221	0.0267
	9.1M	0.0272	0.0435	0.0178	0.0230	0.0571	0.0884	0.0373	0.0473	0.0325	0.0470	0.0226	0.0273
Improv. (%)		+2.93%	+3.99%	+2.23%	+3.07%	0.00%	+0.22%	0.00%	+0.21%	+8.33%	+8.55%	+9.18%	+9.20%
MemOPSD	2.4M	0.0285	0.0451	0.0187	0.0240	0.0588	0.0918	0.0388	0.0493	0.0306	0.0439	0.0213	0.0255
	4.6M	0.0294	0.0457	0.0192	0.0244	0.0618	0.0955	0.0406	0.0514	0.0318	0.0461	0.0221	0.0267
	9.1M	0.0291	0.0466	0.0197	0.0247	0.0610	0.0946	0.0401	0.0509	0.0332	0.0481	0.0232	0.0280
Improv. (%)		+3.16%	+3.33%	+5.35%	+2.92%	+5.10%	+4.03%	+4.64%	+4.26%	+8.50%	+9.57%	+8.92%	+9.80%
LatentR3	2.4M	0.0274	0.0423	0.0176	0.0224	0.0547	0.0868	0.0353	0.0456	0.0299	0.0432	0.0205	0.0248
	4.6M	0.0266	0.0432	0.0174	0.0227	0.0561	0.0888	0.0365	0.0471	0.0314	0.0461	0.0218	0.0266
	9.1M	0.0269	0.0425	0.0175	0.0225	0.0514	0.0814	0.0331	0.0428	0.0317	0.0455	0.0220	0.0265
Improv. (%)		-1.82%	+2.13%	-0.57%	+1.34%	+2.56%	+2.30%	+3.40%	+3.29%	+6.02%	+6.71%	+7.32%	+7.26%
MemOPSD	2.4M	0.0285	0.0446	0.0186	0.0238	0.0591	0.0916	0.0385	0.0490	0.0307	0.0441	0.0213	0.0256
	4.6M	0.0279	0.0447	0.0182	0.0236	0.0604	0.0936	0.0397	0.0504	0.0318	0.0456	0.0222	0.0266
	9.1M	0.0278	0.0445	0.0179	0.0232	0.0567	0.0891	0.0369	0.0473	0.0329	0.0473	0.0231	0.0277
Improv. (%)		-2.11%	+0.22%	-2.15%	-0.84%	+2.20%	+2.18%	+3.12%	+2.86%	+7.17%	+7.26%	+8.45%	+8.20%

Red rows denote the 2.4M baselines, green highlights indicate improvements or ties, and yellow highlights indicate performance degradation. The 2.4M configuration uses a Transformer (Vaswani et al., 2017) with a two-layer encoder and a two-layer decoder; the 4.6M configuration doubles both depths to four layers, and the 9.1M configuration doubles them again to eight layers. Improv. reports the relative change from the 2.4M setting to the best score across the two larger configurations for each metric.

F.3 Ablation of Distillation Configurations

We conduct further ablation experiments to examine teacher training data, supervision scope, loss normalization, and teacher guidance versus additional memory CE supervision. All configurations retain the original backbone objective on all training samples:

- **Group A:** Trains the teacher on all samples and applies distillation to all samples, averaging the distillation loss over the full batch.
- **Group B:** Trains the teacher on all samples and applies distillation only to memory samples, averaging the distillation loss over memory samples.
- **Group C:** Trains the teacher only on memory samples and applies distillation to all samples, averaging the distillation loss over the full batch.
- **Group D:** Uses the same teacher as Group C but restricts distillation to memory samples while retaining the full-batch denominator.

- **Group E:** Replaces teacher-guided distillation with additional ground-truth CE supervision on memory samples, averaging the additional loss over memory samples.
- **Group F (MemOPSD):** Trains the teacher only on memory samples and applies distillation only to memory samples, averaging the distillation loss over memory samples.

Table 9 compares these six configurations using TIGER as the backbone. Groups A and B yield limited and inconsistent improvements over TIGER, whereas Groups C, D, and F improve both Overall and Memory performance across all three datasets. Group F achieves the highest Overall and Memory scores in both NDCG@10 and Recall@10.

Comparing Groups C and D examines the effect of restricting distillation to memory samples while retaining the full-batch denominator. Group D improves Generalization performance on all three datasets and improves Overall and Memory performance on Industrial and Scientific and Video Games, but slightly reduces the latter two on Office Products. Thus, restricting distillation scope alone does not uniformly improve recommendation performance. Comparing Groups D and F further examines normalization: averaging over memory samples improves Overall NDCG@10 by 0.87%, 1.25%, and 3.29% on the three datasets, respectively, with consistent improvements in Memory performance. These results indicate that both supervision scope.

Group E tests whether additional supervision on memory samples can account for the improvements without teacher guidance. Group F outperforms Group E in both Overall and Memory performance across all datasets and both metrics, with Overall NDCG@10 gains of 7.41%, 4.53%, and 8.19%, respectively. This comparison supports the benefit of teacher-guided on-policy distillation beyond the evaluated memory CE reweighting baseline.

Nevertheless, Group F does not uniformly maximize Generalization performance. On Video Games, Group E achieves the highest Generalization NDCG@10, whereas Group D achieves the highest Generalization Recall@10. Compared with Group D, Group F increases Memory NDCG@10 from 0.1988 to 0.2085 but decreases Generalization NDCG@10 from 0.0226 to 0.0217. On Office Products, Group F achieves the highest Generalization NDCG@10, but its Generalization Recall@10 is slightly lower than that of Group D.

Overall, MemOPSD achieves the strongest overall and memory performance among the evaluated configurations, with gains beyond additional memory CE supervision, while generalization trade-offs remain dependent on the dataset and metric.

Table 9: Ablation of distillation configurations on three Amazon Reviews 2023 datasets.

Metric	Method	Industrial and Scientific			Video Games			Office Products		
		Overall	Memory	General	Overall	Memory	General	Overall	Memory	General
NDCG@10	TIGER	0.0224	0.2063	0.0148	0.0452	0.1853	0.0217	0.0241	0.2595	0.0132
	Group A	0.0220	0.2055	0.0144	0.0455	0.1861	0.0219	0.0242	0.2585	0.0134
	Group B	0.0219	0.2048	0.0143	0.0451	0.1843	0.0218	0.0242	0.2578	0.0133
	Group C	0.0228	0.2212	0.0146	0.0470	0.1933	0.0225	0.0244	0.2700	0.0130
	Group D	0.0230	0.2213	0.0148	0.0479	0.1988	0.0226	0.0243	0.2603	0.0134
	Group E	0.0216	0.2102	0.0138	0.0464	0.1878	**0.0227**	0.0232	0.2576	0.0123
	Group F	**0.0232**	**0.2241**	**0.0149**	**0.0485**	**0.2085**	0.0217	**0.0251**	**0.2742**	**0.0135**
Recall@10	TIGER	0.0429	0.3486	0.0302	0.0860	0.3386	0.0437	0.0419	0.4039	0.0252
	Group A	0.0422	0.3494	0.0295	0.0873	0.3372	0.0454	0.0420	0.4031	0.0252
	Group B	0.0420	0.3479	0.0293	0.0865	0.3367	0.0446	0.0419	0.4030	0.0252
	Group C	0.0433	0.3647	0.0300	0.0893	0.3452	0.0464	0.0424	0.4223	0.0248
	Group D	0.0437	0.3663	0.0303	0.0905	0.3530	**0.0466**	0.0421	0.4067	0.0252
	Group E	0.0414	0.3467	0.0287	0.0877	0.3437	0.0448	0.0409	0.4040	0.0240
	Group F	**0.0441**	**0.3767**	**0.0303**	**0.0915**	**0.3677**	0.0452	**0.0432**	**0.4296**	**0.0252**

F.4 Ablation of Language Model Backbones

Table 10 compares TIGER and its MemOPSD-enhanced variant using four language model families, with five configurations ranging from 270M to 1.7B parameters. MemOPSD improves all reported Recall and NDCG metrics across the five configurations and three datasets, with relative NDCG@10 gains ranging from 2.43% to 5.58%. For Qwen3-1.7B, NDCG@10 increases from 0.0233 to 0.0246, from 0.0470 to 0.0491, and from 0.0253 to 0.0262 on Industrial and Scientific, Video Games, and Office Products, respectively. Increasing Qwen3 from 0.6B to 1.7B improves both TIGER and its MemOPSD-enhanced variant across all reported metrics. However, a larger parameter count does

not consistently lead to better performance across model families. For example, Gemma3-270M outperforms BitCPM4-1B across all reported metrics in both settings, despite its smaller parameter count. Some language model configurations nevertheless underperform standard TIGER in Table 1. We use mixed-precision training to accelerate fine-tuning. **Overall, MemOPSD consistently improves performance across the evaluated language model families and parameter scales.**

Table 10: Performance comparison of different language model backbones on three datasets.

Backbone	Size	Method	Industrial and Scientific				Video Games				Office Products			
			R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
Qwen3	0.6B[1]	TIGER	0.0250	0.0408	0.0163	0.0214	0.0501	0.0809	0.0327	0.0427	0.0274	0.0397	0.0188	0.0227
		+MemOPSD	0.0268	0.0410	0.0172	0.0220	0.0529	0.0840	0.0350	0.0446	0.0284	0.0409	0.0197	0.0237
	1.7B[2]	TIGER	0.0276	0.0450	0.0181	0.0233	0.0558	0.0891	0.0360	0.0470	0.0302	0.0436	0.0208	0.0253
		+MemOPSD	0.0284	0.0459	0.0190	0.0246	0.0574	0.0916	0.0382	0.0491	0.0310	0.0449	0.0216	0.0262
MiniCPM4	0.5B[3]	TIGER	0.0243	0.0396	0.0157	0.0206	0.0487	0.0783	0.0316	0.0418	0.0266	0.0383	0.0181	0.0220
		+MemOPSD	0.0260	0.0398	0.0167	0.0211	0.0517	0.0808	0.0337	0.0433	0.0272	0.0397	0.0190	0.0231
BitCPM4	1B[4]	TIGER	0.0238	0.0388	0.0156	0.0201	0.0478	0.0771	0.0311	0.0407	0.0261	0.0376	0.0180	0.0218
		+MemOPSD	0.0256	0.0389	0.0164	0.0209	0.0505	0.0808	0.0333	0.0426	0.0268	0.0388	0.0189	0.0228
Gemma3	270M[5]	TIGER	0.0269	0.0438	0.0175	0.0228	0.0541	0.0877	0.0352	0.0461	0.0294	0.0426	0.0203	0.0247
		+MemOPSD	0.0287	0.0447	0.0185	0.0238	0.0570	0.0902	0.0378	0.0486	0.0306	0.0439	0.0211	0.0257

[1] Qwen3-0.6B: `https://huggingface.co/Qwen/Qwen3-0.6B`.
[2] Qwen3-1.7B: `https://huggingface.co/Qwen/Qwen3-1.7B`.
[3] MiniCPM4-0.5B: `https://huggingface.co/openbmb/MiniCPM4-0.5B`.
[4] BitCPM4-1B: `https://huggingface.co/openbmb/BitCPM4-1B`.
[5] Gemma3-270M: `https://huggingface.co/google/gemma-3-270m`.

F.5 Case Study: Gradient Analysis

Experimental setup. We examine TIGER-based MemOPSD on Industrial and Scientific, Video Games, and Office Products, using the cross-user transition criterion to distinguish memory and generalization instances. During post-training, we probe the gradient directions associated with the two subsets. Let $\mathbf{g}_M$ and $\mathbf{g}_G$ denote their gradients over the same probed parameter set. We measure their directional agreement by:

$$c = \frac{\mathbf{g}_M^\top \mathbf{g}_G}{\|\mathbf{g}_M\|_2 \|\mathbf{g}_G\|_2}, \tag{17}$$

where $c < 0$ indicates conflicting directions.

Reading the figure. The horizontal axis indicates normalized progress through the observed training interval. In the upper panels, each point represents a gradient probe: red points indicate negative cosine similarity, and blue points indicate nonnegative similarity. The horizontal dotted line marks zero, while the dark curve shows the rolling mean, with window sizes of 5, 5, and 13 probes, respectively. The lower panels show rolling conflict rates, with the reported M–G conflict rates annotated in each panel. The horizontal dashed lines mark the separately reported average conflict rates.

Observed behavior. Fig. 8 presents gradient conflict diagnostics during MemOPSD post-training, with M–G conflict rates of 40.32%, 46.43%, and 45.08% on Industrial and Scientific, Video Games, and Office Products, respectively. Compared with the TIGER results in Fig. 1, the conflict rate decreases from 50.99% to 46.43% on Video Games and from 51.89% to 45.08% on Office Products, corresponding to reductions of 4.56 and 6.81 percentage points. Meanwhile, Table 7 shows that Overall NDCG@10 improves from 0.0452 to 0.0485 and from 0.0241 to 0.0251 on these two datasets, respectively. Memory performance also improves, while generalization performance remains unchanged at the reported precision on Video Games and increases on Office Products. These comparisons suggest that MemOPSD improves recommendation performance alongside a reduction in the M–G gradient conflict observed in Fig. 1.

Our hypothesis. We hypothesize that selective memory specialization and transfer helps incorporate recurring interaction knowledge while limiting interference with generalization learning. The memory-specialized teacher provides additional supervision only to memory samples, avoiding direct imposition of its distribution on generalization samples, while the backbone objective remains active on both subsets. On-policy SID self-distillation further supplies teacher guidance at student-generated prefixes, which may help the student learn useful memory-related updates under its current

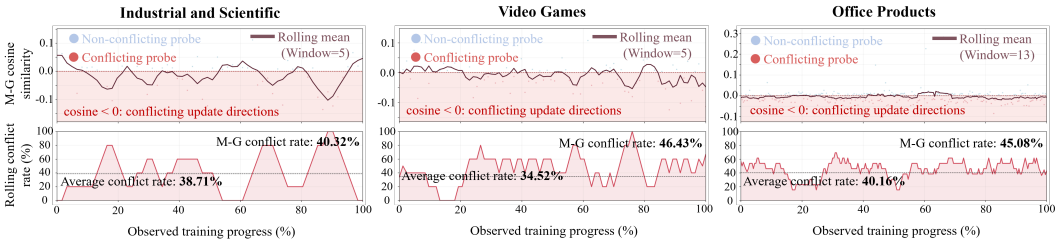


Figure 8: **Gradient conflict during MemOPSD post-training.**

generation behavior. Together, these mechanisms may contribute to a more compatible combination of memory and generalization updates.

Nevertheless, the observed reductions do not establish that lower conflict causes the performance gains. Conflict frequency measures directional disagreement but does not capture gradient magnitudes or the resulting changes in item rankings. Moreover, conflicts remain frequent, indicating that selective supervision does not eliminate interference in the shared parameter space.

Overall, this case provides empirical evidence that MemOPSD can improve overall and memory recommendation performance alongside reduced M–G gradient conflict, while largely preserving generalization performance.