Continual Graph Memory for Adaptive Recommendation under Intent Drift

Hao Nguyen Ngoc[1], Tung Nguyen[2,3], Nguyen Thi Hanh[1], Hoang Thai Dinh[3], Nguyen Xuan Tung[1,†]
[1]Phenikaa University, Vietnam,
[2]Hanoi University of Science and Technology, Vietnam,
[3]University of Technology Sydney, Australia
[†]Corresponding author: tung.nguyenxuan@phenikaa-uni.edu.vn

Abstract

This paper studies adaptive recommendation under intent drift, where feedback from each recommendation outcome can reveal whether the relational evidence used for ranking is useful, missing, or misleading. While Knowledge Graphs (KGs) provide essential semantic structure to handle these shifts, traditional KG-enhanced systems treat the graph as a static retrieval substrate, making it brittle to evolving intents, noisy metadata, and recurring failure patterns. This paper proposes CGM-Rec[1], a continual graph memory framework for adaptive recommendation. CGM-Rec treats the graph state as a writable memory and maintains two complementary components. Therein, a Semantic Graph Memory is updated conservatively through quality-gated typed operations for storing stable and high-confidence relational knowledge. Meanwhile, an Episodic Lesson Memory acts as a fast reactive memory that learns recent outcomes, failure cases, and corrective hints. During testing, model parameters remain frozen and adaptation occurs only through memory writes. We evaluate CGM-Rec under a frozen-parameter, one-pass reranking protocol, where encoders and prompts remain fixed during testing and adaptation occurs only through memory writes. Experiments across multiple recommendation settings show that CGM-Rec improves over evaluated neural and LLM-based baselines on most metrics. Particularly, under sampled-candidate reranking, CGM-Rec improves HR@1 by up to 29.58% over the strongest LLM baseline on Bundle, and outperforms K-RagRec on metadata-rich ML-100K with HR@5 of 0.5941 versus 0.4746.

1 Introduction

Recommendation systems are increasingly deployed in environments where user intents, item semantics, and feedback patterns change over time (Hidasi et al., 2016; Tang et al., 2023; Sun et al., 2024). In such settings, the main challenge is not only to infer preferences from past interactions, but also to decide which evidence should remain trusted as new outcomes arrive. Collaborative filtering alleviates information overload by learning latent user, item patterns, but it relies on dense interaction histories and remains vulnerable to sparsity (Koren et al., 2009; Sarwar et al., 2001). Meanwhile, neural recommenders improve prediction by modeling high-dimensional, non-linear feature interactions (He et al., 2017; Cheng et al., 2016). Nevertheless, their knowledge is encoded implicitly in static parameters, limiting rapid adaptation to intent drift and providing little explicit structure for connecting sparse items. This raises a central question: *how can a recommender preserve reusable knowledge while adapting to intent drift?*

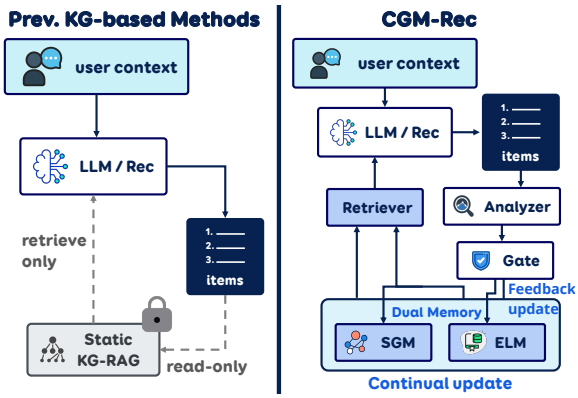

Figure 1: Comparing read-only KG-based recommendation and the continual graph memory mechanism.

Knowledge Graph (KG)-enhanced recommenders have emerged as a potential solution that incorporates external relational networks into the prediction process. By linking items to shared categories, descriptive keywords, and semantic entities, KGs provide a structured web of evidence that bridges sparse user-item interactions. Leveraging this structure, models like RippleNet (Wang et al., 2018) and KGAT (Wang

[1]https://anonymous.4open.science/r/CGM-17DD

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 8501–8517
October 24-29, 2026 ©2026 Association for Computational Linguistics

et al., 2019b) successfully enhance recommendation quality by propagating user preference signals along item-attribute paths. Graph-based session models, including SR-GNN (Wu et al., 2019), further show that graph reasoning can capture local transition patterns in short interaction sequences. However, traditional KG-based models are often constrained by predefined schemas and offline-learned embeddings, limiting their ability to interpret rich textual metadata and complex user intents. To bridge this gap, LLM-based recommenders have been widely adopted to improve semantic understanding and reasoning through prompting and in-context learning (Wang and Lim, 2023; Sun et al., 2024; Li et al., 2023; Yue et al., 2023). More recently, KG-augmented LLM recommenders have exploited graph retrieval to improve semantic reasoning and ranking (Chen et al., 2023; Wang et al., 2025). Despite these advances, most KG-enhanced recommenders treat the graph as a fixed retrieval source during inference. As illustrated in Figure 1, this limits adaptive recommendation as read-only graphs keep retrieving noisy relations and cannot add or reinforce useful relations revealed by feedback. Therefore, under intent drift, the main question is: *How to retrieve information from KG while maintaining accumulate knowledge?*

Existing adaptation methods address this issue incompletely. For instance, prompt optimization and LLM-based memory methods adapt instructions, reasoning traces, or retrieved text (Wang and Lim, 2023; Sun et al., 2024; Nguyen et al., 2025). Meanwhile, continual recommendation and self-correction methods update model behavior, error compensation states, or learned parameters (Cai et al., 2022; Zhu et al., 2023). Dynamic graph methods model evolving relational structure (Tang et al., 2023), but do not focus on frozen-parameter test-time adaptation through graph-memory writes. Consequently, the critical challenge remains: *How to update relational evidence from feedback while preserving stable semantic knowledge and avoiding noisy or hallucinated graph edits?*

Being inspired, the paper proposes **CGM-Rec** (**Continual Graph Memory for Recommendation**), a framework that treats the graph state as the primary adaptive object. Instead of retrieving from a fixed KG, CGM-Rec maintains a writable graph memory that can be updated after recommendation outcomes are observed. Specifically, the framework uses two complementary memories, including Semantic Graph Memory (SGM) and Episodic Lesson Memory (ELM). To swiftly capture emerging user intent drift, ELM acts as a fast-adapting buffer, recording recent recommendation feedback and correction hints through a quality-gated write policy acts as a strict filter. Meanwhile, SGM extracts and stores structural knowledge from data to ensure long-term robustness against noisy metadata and recurring failures. Results show that under a strict one-pass inference protocol with frozen parameters, CGM-Rec outperforms state-of-the-art baselines on most evaluated metrics. In summary, the key contributions of this paper are as follows:

- The paper formalizes adaptive recommendation under intent drift as an outcome-driven graph-memory maintenance problem, where recommendation feedback can be used to refine which relational evidence is useful, missing, or misleading for future ranking.
- We propose CGM-Rec, a continual graph-memory framework that separates feedback by timescale and trust: ELM enables low-latency reuse of recent outcome-derived lessons, while SGM consolidates only supported, low-conflict evidence as persistent relational knowledge.
- We introduce a feedback-to-graph writing mechanism that converts recommendation outcomes into typed, provenance-aware graph-edit proposals and filters them using support, confidence, conflict, recency, and memory-cost signals before updating semantic memory.
- We evaluate CGM-Rec under a frozen-parameter, one-pass sampled-candidate reranking protocol, showing improvements over evaluated neural and LLM-based baselines across multiple settings and over K-RagRec in the metadata-rich setting, with ablations supporting the complementary roles of episodic and semantic memory.

2 Related Work

2.1 Sequential and session-based recommendation.

Session-based recommendation predicts the next item from a short interaction sequence, often without persistent user identity. Representative models include GRU4Rec (Hidasi et al., 2016), NARM (Li et al., 2017), STAMP (Liu et al., 2018), and SR-GNN (Wu et al., 2019). Sequential recommendation extends this setting to longer user histories, with Transformer-based models such as SASRec and BERT4Rec (Kang and McAuley, 2018; Sun et al., 2019). These methods are effective at learn-

ing transition patterns from interaction sequences, but their adaptive state is mainly encoded in learned parameters or hidden sequence representations. As a result, they do not explicitly maintain external relational knowledge that can be corrected, consolidated, or pruned after recommendation outcomes.

2.2 KG-enhanced and KG-RAG recommendation.

Knowledge-graph-enhanced recommenders use external relational structure to enrich item representations and connect sparse user-item interactions. For example, RippleNet (Wang et al., 2018) propagates user preferences over KG paths, while KGAT (Wang et al., 2019b) performs relation-aware graph attention over collaborative knowledge graphs. Recent KG-augmented LLM recommenders, including K-RagRec (Wang et al., 2025), retrieve graph evidence to provide structured context for LLM-based generation or ranking. These methods demonstrate that graph knowledge can improve recommendation by grounding decisions in item-attribute and item-item relations. However, the graph is typically used as a retrieval source. Thus, recommendation outcomes may reveal missing, noisy, or misleading relations, but such feedback is not systematically written back into the graph as maintained memory. CGM-Rec differs by treating graph evidence as a writable memory state rather than only as retrieved context.

2.3 LLM-based adaptive recommendation.

LLMs have recently been used for recommendation through zero-shot ranking, prompt-based reasoning, and agent-style interaction. NIR (Wang and Lim, 2023) studies zero-shot next-item recommendation, PO4ISR (Sun et al., 2024) optimizes prompts for intent-driven session recommendation, and other LLM recommenders use generative ranking or multi-agent collaboration (Li et al., 2023; Yue et al., 2023; Zhang et al., 2024; Nguyen et al., 2025). Surveys and benchmarks further highlight the promise of LLM-based recommendation, while noting challenges in robustness, evaluation, memory design, and inference cost (Peng et al., 2025; Liu et al., 2025). These approaches improve semantic reasoning and can adapt through prompts, textual memories, or agent states. Nevertheless, their persistent adaptation is usually textual or prompt-level, not structured as typed graph memory. CGM-Rec instead uses the LLM as a frozen recommender and analyzer, while long-term adaptation is constrained to quality-gated graph-memory edits.

2.4 Continual, dynamic, and memory-based recommendation.

Dynamic memory-based recommendation methods aim to adapt recommenders as new interactions arrive. ReLoop (Cai et al., 2022) and ReLoop2 (Zhu et al., 2023) introduce self-correction loops, while D2K (Qin et al., 2025) turns historical data into a retrievable knowledge store. These works highlight the importance of continual adaptation and explicit memory. However, their adaptive states are error buffers, model updates, or generic retrieval stores rather than structured graph memories. Specifically, these methods do not explicitly model feedback as typed graph-edit operations to consolidate, weaken, or prune semantic relations over time. CGM-Rec is closest in spirit to this line, but its memory is organized as a graph with explicit operations for semantic promotion, relation reinforcement, suppression, and stale-edge pruning.

3 Task Definition

Let $\mathcal{I}$ denote the item set and $\mathcal{A}$ denote available item attributes, such as categories, keywords, descriptions, and other metadata. We study adaptive recommendation as a one-pass candidate reranking problem over an ordered stream of instances $\mathcal{D} = \{(x_t, \mathcal{C}_t, y_t)\}_{t=1}^{T}$, where x_t is the context, $\mathcal{C}_t \subseteq \mathcal{I}$ denotes the candidate pool, and $y_t \in \mathcal{C}_t$ is the ground-truth target item. At each step t, the system maintains a dual-timescale memory state $\mathcal{M}_t = (\mathcal{M}_{s,t}, \mathcal{M}_{e,t})$, comprising a Semantic Graph Memory ($\mathcal{M}_{s,t}$) and an Episodic Lesson Memory ($\mathcal{M}_{e,t}$) (see Section 4.2 for details). The recommender leverages this state to generate a ranking list: $\hat{\pi}_t = f_\theta(x_t, \mathcal{C}_t, \mathcal{M}_t)$. Once $\hat{\pi}_t$ is evaluated against y_t, the feedback is processed by a deterministic write policy to update the memory for the next step:

$$\mathcal{M}_{t+1} = U_\phi(\mathcal{M}_t, x_t, \mathcal{C}_t, \hat{\pi}_t, y_t), \tag{1}$$

where fixed thresholds ϕ control typed operations. During inference, both recommender parameters θ and thresholds ϕ remain strictly frozen; adaptation is driven exclusively via memory updates.

Objective. The primary objective is to maximize stream-level ranking quality:

$$\mathcal{J} = \frac{1}{T}\sum_{t=1}^{T} \mathrm{Metric@K}(\hat{\pi}_t, y_t), \tag{2}$$

where $\mathrm{Metric@K}$ denotes standard ranking indicators such as HR@K and NDCG@K (see Section 5.1). While $\mathcal{J}$ isolates ranking effectiveness, the total memory capacity is governed by a strict, finite structural budget constraint rather than a continuous soft penalty function. When the memory footprint reaches its predefined capacity limit, the write policy U_ϕ automatically activates a deterministic cache eviction protocol. This mechanism systematically purges stale, low-utility, or highly conflicting historical records, maintaining bounded storage overhead and structural stability without sacrificing streaming recommendation accuracy.

4 Methodology

4.1 Overview

Figure 2 summarizes the workflow of CGM-Rec. The framework has three stages. First, during offline preparation, training interactions and item metadata are used to construct a seed graph. A relation-aware graph encoder is trained on this seed graph and then frozen before test-time inference. The prompt templates and write-policy thresholds are fixed before testing. Second, at each test step t, the system receives recommendation context x_t and candidate set $\mathcal{C}_t$. A dual-memory retriever extracts semantic evidence from SGM and relevant textual lessons from ELM. These signals are combined into a structured prompt for a frozen LLM-based Recommender Agent, which reranks $\mathcal{C}_t$ and outputs $\hat{\pi}_t$. Third, after the target item y_t is revealed, the system analyzes the outcome. An Analyzer Agent produces textual lessons and structured graph-edit proposals. Textual lessons are stored in ELM for future retrieval, while graph-edit proposals must pass a fixed quality gate before modifying SGM. Thus, test-time adaptation is restricted to memory updates rather than parameter updates.

4.2 Dual-Memory Architecture

CGM-Rec maintains two complementary memories. Semantic Graph Memory stores stable relational knowledge as a typed graph, while Episodic Lesson Memory stores recent outcome-derived lessons and correction signals. The two memories differ in trust level and update speed: ELM reacts quickly to recent feedback, whereas SGM changes conservatively through gated graph edits.

Seed graph construction. The system first samples a small subset of the training stream, together with item metadata, to construct a seed graph $G_0 = (V_0, E_0, \mathcal{R})$, where the node set V_0 contains item and metadata nodes (categories, keywords, descriptions). The typed edge set E_0 encodes relations over the vocabulary $\mathcal{R} = \{(i, \texttt{belongs_to}, c), (i, \texttt{has_keyword}, k), (i, \texttt{has_description}, d), (i, \texttt{co_occurs}, j)$, and optionally $(i, \texttt{related_intent}, z)\}$. The seed graph serves a dual purpose: initializing the Semantic Graph Memory ($\mathcal{M}_{s,0}$) and providing the structural substrate for a relation-aware graph encoder, which is trained using D_{train} and frozen before inference. At inference time, the frozen encoder is used only for graph-evidence retrieval, not for parameter updates. Details of the encoder and its training objective are provided in Appendix A.

Semantic Graph Memory. SGM is initialized from the seed graph as $\mathcal{M}_{s,0} = (V_{s,0}, E_{s,0}, A_{s,0})$, where $V_{s,0} = V_0$, $E_{s,0} = E_0$, and $A_{s,0}$ stores edge attributes. Thus, at time t, SGM is represented as $\mathcal{M}_{s,t} = (V_{s,t}, E_{s,t}, A_{s,t})$. Each semantic edge $e = (u, r, v) \in E_{s,t}$ is associated with:

$$A_{s,t}(e) = (w_{e,t}, c_{e,t}, n_{e,t}, q_{e,t}, p_e), \qquad (3)$$

where $w_{e,t}$ is the edge weight, $c_{e,t}$ is confidence, $n_{e,t}$ is the support count, $q_{e,t}$ is a quality score, and p_e records provenance including metadata-derived, interaction-derived, or promoted-from-lesson. SGM stores relations that should change conservatively, including item-category, item-keyword, item-description, item-item, and accepted item-intent relations. During inference, SGM is updated only through accepted structured edit actions after the quality gate; raw LLM-generated text is never written directly into SGM.

Episodic Lesson Memory. ELM stores feedback-derived lessons and structured episodic signals generated after observing outcomes. Unlike the persistent, relation-typed SGM, ELM is a semi-structured store for recent outcome-specific evidence. At time t, ELM denotes $\mathcal{M}_{e,t} = \{z_j\}_{j=1}^{B_t}$, where each record z_j contains lesson and structured metadata:

$$z_j = (d_j, \kappa_j, o_j, r_j, \mathcal{E}_j, \mathcal{P}_j, u_j, \rho_j). \qquad (4)$$

Here, d_j is the lesson text, κ_j the context, o_j the outcome label, r_j the target-rank signal, $\mathcal{E}_j$ the retrieved evidence, $\mathcal{P}_j$ the edit proposals, u_j the utility or support, and ρ_j the retrieval metadata. This design separates short-term episodic adaptation from long-term semantic consolidation. ELM preserves recent success patterns, failure causes, misleading evidence, and corrective hints for future retrieval, but its records do not directly modify

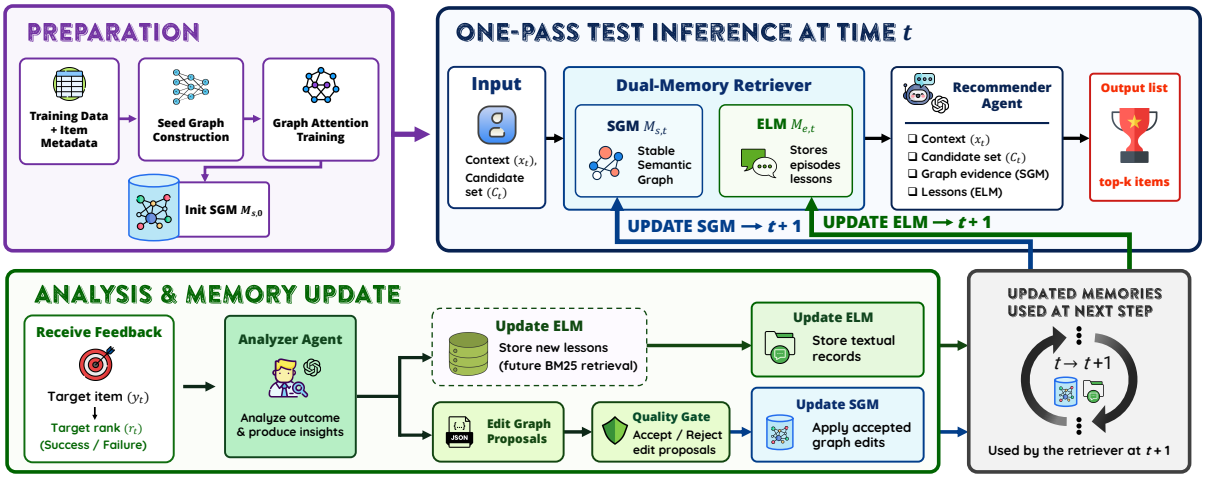

Figure 2: Workflow of CGM-Rec with continual dual-memory updates

SGM. Only structured proposals accepted by the quality-gated write policy can update SGM.

4.3 Dual-memory recommendation.

Given context x_t and candidate set C_t, the system retrieves complementary evidence from SGM and ELM before reranking. For SGM, context and candidate items are mapped to graph nodes. The retriever then expands local neighborhoods around these nodes in $\mathcal{M}_{s,t}$ and uses the frozen graph encoder to rank local edges and relational paths based on their relevance to (x_t, C_t). This produces:

$$R_t^s = \text{GraphRetrieve}(x_t, C_t, \mathcal{M}_{s,t}; \theta_g), \quad (5)$$

where θ_g denotes the frozen graph encoder. The retrieved subgraph evidence includes item–attribute relations, item–item links, intent-related relations, and edge attributes like confidence, support, and provenance. For ELM retrieval, the system constructs a query from the current context, candidate items, and available item attributes, $q_t = \text{Query}(x_t, C_t)$, and retrieves top-$k$ relevant lessons using BM25: $R_t^e = \text{TopK}_{\mathcal{M}_{e,t}}\text{BM25}(q_t, d_j, \kappa_j)$.

These complementary signals are directly integrated to populate a structured context prompt $P_t = \text{Prompt}(x_t, \mathcal{C}_t, R_t^s, R_t^e)$, fusing stable relational structures with experiences (see Appendix K for full prompt templates). This prompt is processed by the Recommender Agent, an LLM-based reranker governed by a specific task instruction. The agent is instructed to analyze the user context, infer the current intent, and leverage the retrieved SGM and ELM evidence to evaluate the candidates. Operating under strictly frozen weights without test-time fine-tuning or gradient updates, the agent outputs the optimized ranking $\hat{\pi}_t \in \Pi(\mathcal{C}_t)$:

$$\hat{\pi}_t = \text{Rerank}_\omega(\mathcal{C}_t \mid x_t, R_t^s, R_t^e), \quad (6)$$

where ω denotes the fixed LLM prompt configuration, and $\Pi(\mathcal{C}_t)$ denotes all candidate permutations.

4.4 Quality-Gated Memory Update

After the recommender outputs $\hat{\pi}_t$, the target item y_t becomes available for evaluation and memory update. Importantly, y_t is not used before the ranking is produced. The system computes the target rank $r_t = \text{rank}_{\hat{\pi}_t}(y_t)$ and assigns a binary outcome label $o_t \in \{\texttt{success}, \texttt{failure}\}$ against a predefined rank cutoff threshold K_s, where $o_t = \texttt{success}$ if $r_t \leq K_s$ and $\texttt{failure}$ otherwise. This label provides a compact feedback signal for subsequent lesson generation and memory updates.

Outcome analysis. After obtaining the target rank and outcome label, the system packages the context, retrieved evidence, and feedback into:

$$\Omega_t = (x_t, \mathcal{C}_t, \hat{\pi}_t, y_t, r_t, o_t, R_t^s, R_t^e)$$

An LLM-based Analyzer Agent (prompt in Appendix L) processes this package: $(\mathcal{L}_t, \mathcal{P}_t) = \text{Analyzer}_{\omega_a}(\Omega_t)$, where $\mathcal{L}_t$ denotes textual lessons and $\mathcal{P}_t$ denotes structured graph-edit proposals. The Analyzer is instructed to explain the observed outcome, identify useful or misleading evidence, and propose candidate memory edits in a constrained format; it cannot directly modify SGM. The lessons summarize intent cues, useful patterns, failure causes, misleading evidence, or corrective hints and are stored in ELM for future retrieval:

$$\mathcal{M}_{e,t+1} = \text{Budget}(\mathcal{M}_{e,t} \cup \mathcal{L}_t), \quad (7)$$

The $\mathrm{Budget}(\cdot)$ operator keeps ELM within a fixed capacity by retaining recent and useful lessons while removing low-utility records.

Structured edit proposals. Each proposal $p \in \mathcal{P}_t$ is represented as $p = (a, z, \delta, \xi)$, where a is the action type, z is the target object such as an edge, relation, or lesson-derived pattern, δ stores the update value or arguments, and ξ stores supporting evidence from the outcome package. The supported action types are summarized in Table 1. These proposals are treated as candidates only and can affect SGM only after passing the quality gate.

Quality-gated writing. To prevent unsupported LLM proposals from modifying SGM, structured proposals are filtered by a rule-guided quality gate U_ϕ. The gate is a deterministic write filter with predefined thresholds ϕ, rather than a learned model. For each proposal p, it checks rank, support, confidence, SGM conflict, memory cost, and recency:

$$\mathcal{S}_t(p) = \{o_t, r_t, \mathrm{supp}(p), \mathrm{conf}(p), \mathrm{conflict}(p), \mathrm{cost}(p), \mathrm{rec}(p)\}. \tag{8}$$

The quality gate filters proposals into the accepted action set $\mathcal{A}_t = U_\phi(\mathcal{P}_t, \mathcal{S}_t, \mathcal{M}_{s,t})$ using thresholds fixed before test-time inference, reinforcing useful relations while suppressing misleading ones (see Appendix B for detailed criteria).

Updating SGM. Only accepted typed actions are applied to Semantic Graph Memory:

$$\mathcal{M}_{s,t+1} = \mathrm{Apply}(\mathcal{M}_{s,t}, \mathcal{A}_t). \tag{9}$$

The $\mathrm{Apply}(\cdot)$ operator executes the accepted action types in Table 1. Reinforcement increases the support, confidence, or weight of useful relations, while suppression decreases the confidence or weight of relations associated with repeated failures or misleading evidence. Tentative edges are inserted with low confidence and must accumulate support to become stable relations. Finally, weak, unsupported, or low-utility edges are pruned to keep SGM compact and prevent noise accumulation over long streams (details in Appendix C).

5 Experiments

The experiments address four research questions: **(RQ1)** Does CGM-Rec outperform traditional, neural, and LLM-based baselines? **(RQ2)** How does CGM-Rec compare to LLM-based methods in metadata-rich contexts? (**RQ3**) How do dual memory and graph updating affect performance? (**RQ4**) How does CGM-Rec adapt under continual deployment and measurable intent shifts?

5.1 Experimental Setup

Dataset. CGM-Rec is evaluated on four datasets: Bundle (Sun et al., 2022), Games (Ni et al., 2019), and MovieLens ML-1M/ML-100K (Harper and Konstan, 2015). Bundle and Games represent short-context product and game recommendation settings, ML-1M provides user-movie interaction contexts, and ML-100K serves as a metadata-rich non-session contextual reranking benchmark with taxonomy paths, descriptions, and keywords. Each instance contains a context, a candidate set, and a ground-truth target item. Each context is an interaction sequence or a metadata-rich textual query, and the candidate set includes the ground-truth target item along with sampled items for reranking. Therefore, our evaluation focuses on constrained candidate reranking rather than full-catalog retrieval. All methods rerank the same candidate sets, and Table 5 reports dataset statistics.

Evaluation Protocol. Each dataset is split into training $\mathcal{D}_{\text{train}}$ and test stream $\mathcal{D}_{\text{test}}$. A small subset of $\mathcal{D}_{\text{train}}$, together with item metadata, is used to construct the seed graph and initialize $\mathcal{M}_{s,0}$, while $\mathcal{D}_{\text{train}}$ is used for offline graph-encoder training. This reflects a realistic deployment setting where the system starts with sparse initial knowledge and gradually accumulates structural memory from new recommendation outcomes. All write-policy thresholds are fixed before testing and are not tuned on $\mathcal{D}_{\text{test}}$. During one-pass inference, model parameters and prompt templates remain frozen. Each instance follows:

$$\text{retrieve} \rightarrow \text{rank} \rightarrow \text{feedback} \rightarrow \text{update}. \tag{10}$$

The target item is revealed after ranking to compute feedback and update memory for future instances. All main comparisons are repeated over five matched seeds $\{0, 10, 42, 625, 2023\}$, and the results are reported as mean $\pm$ standard deviation.

Leakage control. All continual evaluations follow a strict predict–reveal–update protocol. At step t, the recommendation is produced using the context and memory state available before observing the target outcome. The target and its feedback are revealed only after ranking, and any SGM or ELM update can affect subsequent steps. To prevent target leakage through episodic memory, stored lessons are sanitized to exclude target IDs and titles, candidate positions, exact ranks, raw Analyzer outputs, and executable update proposals; only attribute- and relation-level summaries are retained.

Action	Target	Function
`add_tentative_edge`	SGM	Add a low-confidence relation when feedback suggests missing evidence.
`reinforce_edge`	SGM	Increase the weight, confidence, or support count of a relation that improves ranking.
`suppress_edge`	SGM	Downweight a relation that misleads ranking or conflicts with observed feedback.
`store_lesson`	ELM	Store a textual lesson or corrective hint generated from the outcome analysis.
`promote_to_semantic`	SGM	Convert a reliable recurring lesson or edit proposal into a stable semantic relation.
`prune_stale_edge`	SGM	Remove or deactivate a low-confidence, low-utility, or stale relation.

Table 1: Typed actions used by the feedback-guided memory update mechanism.

Inference overhead. CGM-Rec averages 9.6 s per recommendation instance, including both LLM calls and memory operations, with an average of 19.93K input and 1.17K output tokens. (breakdown in Appendix E).

Metrics. Two ranking metrics are used: Hit Rate (HR@K) and Normalized Discounted Cumulative Gain (NDCG@K), with $K \in \{1, 5, 10\}$ (Sun et al., 2024). HR@K measures whether the ground-truth item appears within the top-K positions, while NDCG@K further accounts for its ranking position. Higher values indicate better performance.

5.2 Baselines

CGM-Rec is benchmarked against a wide range of baselines spanning four distinct paradigms. The first category encompasses traditional and deep learning-based single-intent recommenders: MostPop (Ji et al., 2020) for popularity-based ranking, SKNN (Jannach and Ludewig, 2017) for neighborhood similarity, FPMC (Rendle et al., 2010) for Markov chain modeling, and attention-based neural models (NARM (Li et al., 2017), STAMP (Liu et al., 2018)). This category also includes SASRec (Kang and McAuley, 2018), a self-attentive sequential recommender, and GCE-GNN (Wang et al., 2020), a session recommender employing graph neural networks to aggregate item transitions across both local sessions and a global context graph. The second category comprises advanced multi-intent frameworks, including MCPRN (Wang et al., 2019a), HIDE (Li et al., 2022), and Atten-Mixer (Zhang et al., 2023). These models explicitly route or mix sequence representations into multiple intention channels to handle diverse behavioral patterns. For continual self-adaptation, ReLoop2 (Zhu et al., 2023) is included, which adapts online through a responsive error-compensation loop without LLM components. The final category includes recent LLM-based baselines: NIR (Wang and Lim, 2023), PO4ISR (Sun et al., 2024), LLMGreenRec (Nguyen et al., 2025), and K-RagRec (Wang et al., 2025), covering zero-shot reasoning, prompt optimization, textual memory retrieval, and graph-augmented LLM recommendation. To ensure fairness, all LLM-based methods use GPT-4.1-mini as the backbone and are evaluated on the same candidate set of size 20.

6 Experiment Results

6.1 Overall Performance Comparison

Table 2 shows that CGM-Rec achieves the best results on most datasets and metrics, demonstrating strong ranking performance across the evaluated settings. Compared to traditional and neural sequence recommenders, CGM-Rec yields substantial gains because static paradigms fail to maintain adaptive states during inference. This limitation is clear on the Bundle dataset, where the strongest neural baseline, Atten-Mixer, achieves an HR@1 of 0.0630 compared to CGM-Rec's 0.3263. On ML-1M, CGM-Rec achieves the highest mean across all three reported metrics. On Games, it achieves the highest mean HR@1 and NDCG@5, while remaining competitive with ReLoop2 on HR@5; the differences on Games and ML-1M are not statistically significant. Furthermore, CGM-Rec outperforms advanced LLM baselines, improving HR@1 by 29.58% relative to LLMGreenRec on Bundle ($p < 0.01$). This demonstrates that language-driven reasoning is insufficient for robust context adaptation, whereas structured, quality-gated graph mutations offer a critical architectural advantage. These findings validate our central hypothesis: adaptive recommenders achieve superior resilience by managing graph knowledge as an active, continually maintained memory layer rather than a fixed retrieval source.

6.2 LLM-based Comparison in Metadata-rich

Table 3 shows that CGM-Rec outperforms the strongest LLM graph-augmented baseline, K-RagRec, by $+0.0971$ in HR@1 on the non-session, metadata-rich ML-100K setting, with all gains significant ($p < 0.05$). This margin stems from K-RagRec's architectural limitation: it injects static, multi-hop subgraphs directly into the prompt as dense context. Large metadata overloads the con-

Datasets	Metrics	Traditional			Single-Intent				Multi-Intent			Continual & LLM-based				
		MostPop	SKNN	FPMC	NARM	STAMP	GCE-GNN	SASRec	MCPRN	HIDE	Atten-Mixer	ReLoop2	NIR	PO4ISR	LLMGreenRec	**CGM-Rec**
ML-1M	HR@1	0.0071±.0016	0.1502±.0083	0.1254±.0109	0.0670±.0046	0.1702±.0101	0.1314±.0046	0.2018±.0136	0.0794±.0062	0.1498±.0090	0.1380±.0087	0.1132±.0076	0.0564±.0348	0.1786±.0273	0.1633±.0389	**0.2049**±.0117
	HR@5	0.0527±.0052	0.3376±.0121	0.3804±.0040	0.3054±.0066	0.5278±.0111	0.4818±.0069	0.5618±.0140	0.3374±.0094	0.5096±.0064	0.4986±.0075	0.3532±.0154	0.2314±.0994	0.5374±.0426	0.5111±.0655	**0.5786**±.0174
	NDCG@5	0.0353±.0053	0.2429±.0098	0.2554±.0052	0.1822±.0056	0.3517±.0062	0.3057±.0054	0.3692±.0110	0.2085±.0055	0.3287±.0069	0.3189±.0051	0.2338±.0088	0.1445±.0480	0.3601±.0412	0.3537±.0286	**0.3878**±.0156
Games	HR@1	0.0058±.0037	0.0498±.0078	0.0602±.0054	0.0584±.0038	0.0622±.0055	0.0590±.0016	0.0988±.0044	0.0608±.0044	0.0780±.0029	0.0594±.0011	0.2336±.0126	0.1055±.0463	0.2602±.0561	0.2739±.0402	**0.3092**±.0131
	HR@5	0.0259±.0042	0.2534±.0084	0.2544±.0062	0.2644±.0092	0.2672±.0018	0.2656±.0021	0.2920±.0117	0.2574±.0058	0.2674±.0078	0.2578±.0069	**0.5962**±.0177	0.4310±.0704	0.5474±.0616	0.5494±.0385	0.5898±.0176
	NDCG@5	0.0155±.0043	0.1481±.0070	0.1550±.0034	0.1568±.0059	0.1634±.0020	0.1592±.0015	0.1943±.0058	0.1568±.0034	0.1701±.0052	0.1558±.0029	0.4194±.0151	0.2656±.0462	0.4153±.0576	0.4236±.0340	**0.4475**±.0168
Bundle	HR@1	0.0038±.0059	0.0546±.0066	0.0261±.0035	0.0538±.0100	0.0345±.0120	0.0571±.0048	0.0261±.0075	0.0311±.0105	0.0538±.0117	0.0630±.0079	0.1454±.0114	0.0980±.0133	0.1657±.0099	0.2518±.0229	**0.3263**±.0175
	HR@5	0.0227±.0021	0.2605±.0195	0.2261±.0196	0.2630±.0189	0.2487±.0237	0.2723±.0035	0.2185±.0390	0.2025±.0137	0.2672±.0097	0.2832±.0064	0.4185±.0304	0.2882±.0901	0.3908±.0467	0.5160±.0086	**0.5716**±.0098
	NDCG@5	0.0125±.0050	0.1535±.0091	0.1199±.0082	0.1567±.0117	0.1347±.0053	0.1583±.0043	0.1174±.0246	0.1133±.0077	0.1600±.0106	0.1677±.0075	0.2831±.0198	0.1974±.0515	0.2886±.0379	0.3996±.0123	**0.4495**±.0146

Table 2: Performance comparison on three datasets, reported as mean ± SD over 5 seeds. Best in bold, runner-up underlined. Paired t-tests against the runner-up give p = 7.07e-03 (Bundle HR@1), 1.99e-03 (Bundle HR@5); differences on Games and ML-1M are not statistically significant ($p > 0.1$).

Metric	NIR	LLMGreenRec	K-RagRec	CGM-Rec	p-value
HR@1	0.0037±.0022	0.0152±.0022	0.2023±.0139	**0.2994**±.0032	1.28e-04
HR@5	0.0315±.0084	0.1318±.0083	0.4746±.0258	**0.5941**±.0087	5.12e-04
HR@10	0.1193±.0122	0.2718±.0142	0.6801±.0381	**0.7567**±.0105	1.12e-02
NDCG@1	0.0037±.0022	0.0152±.0022	0.2023±.0139	**0.2994**±.0032	1.28e-04
NDCG@5	0.0167±.0053	0.0713±.0052	0.3318±.0268	**0.4508**±.0096	6.01e-04
NDCG@10	0.0546±.0055	0.1160±.0061	0.3877±.0416	**0.5015**±.0119	3.86e-03

Table 3: Recommendation performance comparison against LLM-based baselines on the non-session, metadata-rich ML-100K benchmark. Best results are in bold and runner-up results are underlined.

text window with unstructured textual noise, obscuring subtle entity-attribute insights. In contrast, CGM-Rec maps metadata taxonomies onto its structural layers, using a quality-gated write policy to filter and compact relation paths into a dual-timescale memory. This dynamic maintenance distills metadata into a noise-free retrieval state, grounding the LLM reranker in precise semantic neighborhoods even without sequential triggers.

Candidate-pool scaling against K-RagRec at $C \in \{20, 50, 80\}$ further shows that CGM-Rec maintains higher mean HR@5 on both Games and ML-100K. (Appendix I).

6.3 Dual-Memory Ablation Study

To assess dual-memory complementarity, CGM-Rec is compared against SGM-only, ELM-only, and static variants. The full system achieves the best performance, confirming that episodic memory handles rapid intent shifts while semantic memory provides stable long-term relational evidence. Detailed ablation results are provided in Appendix G. Quality-gate sensitivity is further examined in Appendix H, where moderate threshold perturbations produce only small HR@5 changes despite substantial variation in accepted-write rates.

6.4 Continual Adaptation

This analysis evaluates continual adaptation on the chronological Bundle test stream, after initialization with a 20% warm-up subset of the training stream. After each session is ranked, the target

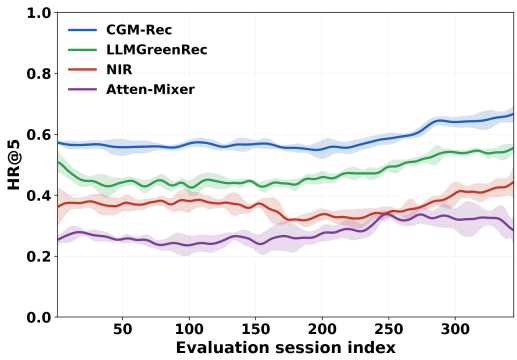

Figure 3: Continual adaptation on Bundle using 20-checkpoint rolling-mean HR@5 over five seeds, with memory updated every five sessions and shaded regions indicating ±1 SD.

item is revealed and its feedback is incorporated into the adaptive state, with memory updates applied every five sessions while model parameters remain frozen. At each evaluation snapshot, the current adaptive state is frozen and evaluated on the following test samples. The plotted curve reports the 20-checkpoint rolling mean of HR@5 over five matched seeds, with shaded regions indicating ±1 standard deviation. Thus, each point reflects how well the system performs on future samples given the accumulated memory.

Figure 3 shows that CGM-Rec outperforms all baselines throughout the stream, with its advantage exceeding the ±1 SD bands in later snapshots. Its curve also increases in later snapshots, indicating the system does not merely correct isolated failures but gradually accumulates feedback into memory. LLMGreenRec improves over time but remains below CGM-Rec, while NIR and Atten-Mixer show lower and less stable trajectories. These results highlight the continual nature of CGM-Rec: feedback from earlier sessions is converted into reusable graph memory and episodic lessons, enabling stronger adaptation.

Transition	Method	Before	Immediate	Later	Recovery
Food→Elec.	**CGM-Rec**	**0.600 ± 0.0354**	**0.520 ± 0.0274**	**0.550 ± 0.0354**	**37.5%**
	Static SGM+ELM	0.570 ± 0.0274	0.480 ± 0.0274	0.490 ± 0.0224	11.1%
	K-RagRec	0.550 ± 0.0354	0.480 ± 0.0274	0.500 ± 0.0354	28.6%
Elec.→Cloth.	**CGM-Rec**	**0.550 ± 0.0354**	**0.480 ± 0.0274**	**0.510 ± 0.0418**	**42.9%**
	Static SGM+ELM	0.490 ± 0.0224	0.430 ± 0.0274	0.440 ± 0.0224	16.7%
	K-RagRec	0.500 ± 0.0354	0.440 ± 0.0224	0.460 ± 0.0418	33.3%

Table 4: HR@5 around intent-regime shifts on Bundle, reported as 20-session window means over five matched seeds rather than full-stream averages.

6.5 Adaptation under Intent-Regime Shifts

To connect measurable intent shifts with adaptation performance, a category-blocked regime-shift analysis is conducted on the Bundle test stream. Since Bundle does not preserve persistent user identities across sessions, the observed changes are treated as stream-level intent-regime shifts rather than latent within-user preference drift. Following prior work on temporal preference dynamics and concept drift (Koren, 2009; Gama et al., 2014), observable session intent is represented by the normalized distribution $\mathbf{p}_t$ of taxonomy labels among the context items. The shift magnitude at boundary b is measured using Jensen–Shannon divergence (JSD) (Lin, 1991) between the mean intent distributions of the 20 sessions before and after the boundary:

$$D_b = \mathrm{JSD}\left(\overline{\mathbf{p}}_{b-20:b-1}, \overline{\mathbf{p}}_{b:b+19}\right). \tag{11}$$

Using base-2 logarithms, $D_b \in [0, 1]$. The Bundle test stream contains three consecutive regimes: Food (78 sessions), Electronics (77), and Clothing (83), with JSD values of 1.000 for Food→Electronics and 0.979 for Electronics→Clothing. The drift score is used only for retrospective analysis and is never provided to the recommender or memory writer.

Adaptation is evaluated over five matched seeds using fixed 20-session windows before, immediately after, and in a later window after each shift. Full CGM-Rec, Static SGM+ELM, and K-RagRec use identical contexts, candidate pools, and stream order, while memory is not reset at regime boundaries. The HR@5 recovery ratio is computed from the aggregated means as follows:

$$\mathrm{Recovery} = \frac{\mathrm{HR@5}_{\mathrm{Late}} - \mathrm{HR@5}_{\mathrm{Immediate}}}{\mathrm{HR@5}_{\mathrm{Before}} - \mathrm{HR@5}_{\mathrm{Immediate}}} \times 100\%. \tag{12}$$

As shown in Table 4, all methods experience an immediate performance decline after each transition. CGM-Rec subsequently recovers a larger proportion of the lost HR@5 than Static SGM+ELM and K-RagRec. The recovery remains partial: CGM-Rec's later HR@5 is still 0.050 and 0.040 below the pre-shift level after the two transitions, respectively. These results strengthen the empirical connection between feedback-driven writable memory and post-shift adaptation under measurable intent shifts, rather than indicating immunity to drift or within-user preference recovery.

7 Conclusion

This paper presented CGM-Rec, a continual graph-memory framework for adaptive recommendation under intent drift. The central idea is to move KG-enhanced recommendation beyond read-only graph retrieval by treating the graph state as a writable memory. CGM-Rec separates fast feedback capture from stable semantic consolidation via Episodic Lesson Memory and Semantic Graph Memory, and controls graph changes with a quality-gated writer that converts recommendation outcomes into typed, provenance-aware edits. Under a frozen-parameter, one-pass sampled-candidate reranking protocol, CGM-Rec achieves the strongest mean results on Bundle and remains competitive with strong neural and continual baselines on Games and ML-1M, while consistently outperforming K-RagRec in the metadata-rich ML-100K setting. Ablation and continual-stream analyses further suggest that episodic lessons and semantic graph memory provide complementary benefits.

8 Limitations

CGM-Rec introduces additional complexity compared with static retrieval, requiring extra storage and computation for memory maintenance. Although the quality-gate sensitivity analysis shows robustness to moderate perturbations, broader gate configurations and typed update magnitudes remain unexplored. Evaluation focuses on sampled-candidate reranking, with the $C = 50$ and $C = 80$ settings serving as larger-pool robustness tests rather than full-catalog evaluations. The controlled Bundle analysis captures stream-level intent-regime shifts rather than natural within-user preference drift. Finally, episodic-to-semantic promotion may propagate biased or noisy evidence, motivating stronger provenance safeguards.

References

Guohao Cai, Jieming Zhu, Quanyu Dai, Zhenhua Dong, Xiuqiang He, Ruiming Tang, and Rui Zhang. 2022. ReLoop: A self-correction continual learning loop for recommender systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research*

and Development in Information Retrieval, pages 2692–2697.

Qian Chen, Zhiqiang Guo, Jianjun Li, and Guohui Li. 2023. Knowledge-enhanced multi-view graph neural networks for session-based recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 352–361.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37.

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4).

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on World Wide Web*, pages 173–182.

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

Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *RecSys '17: Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 306–310.

Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2020. A re-visit of the popularity baseline in recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1749–1752.

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

Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 447–456. ACM.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1419–1428.

Jinming Li, Wentao Zhang, Tian Wang, Guanglei Xiong, Alan Lu, and Gerard Medioni. 2023. GPT4Rec: A generative framework for personalized recommendation and user interests interpretation. In *Proceedings of the 2023 SIGIR Workshop on eCommerce*.

Yinfeng Li, Chen Gao, Hengliang Luo, Depeng Jin, and Yong Li. 2022. Enhancing hypergraph neural networks with intent disentanglement for session-based recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*, pages 1997–2002, Madrid, Spain.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.

Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1831–1839.

Qijiong Liu, Jieming Zhu, Lu Fan, Kun Wang, Hengchang Hu, Wei Guo, Yong Liu, and Xiao-Ming Wu. 2025. Can LLMs outshine conventional recommenders? a comparative evaluation. In *Advances in Neural Information Processing Systems*, volume 38. Datasets and Benchmarks Track.

Hao N. Nguyen, Hieu M. Nguyen, Son Van Nguyen, and Nguyen Thi Hanh. 2025. LLMGreenRec: LLM-based multi-agent recommender system for sustainable e-commerce. In *2025 Conference on Digital Economy and Fintech Innovation (DEFI)*, pages 117–124.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197.

Qiyao Peng, Hongtao Liu, Hua Huang, Jian Yang, Qing Yang, and Minglai Shao. 2025. A survey on LLM-powered agents for recommender systems. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11574–11583.

Jiarui Qin, Weiwen Liu, Weinan Zhang, and Yong Yu. 2025. D2K: Turning historical data into retrievable knowledge for recommender systems. In *Proceedings of the ACM on Web Conference 2025*, pages 472–482.

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 811–820.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1441–1450.

Zhu Sun, Hongyang Liu, Xinghua Qu, Kaidong Feng, Yan Wang, and Yew-Soon Ong. 2024. Large language models for intent-driven session recommendations. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 324–334.

Zhu Sun, Jie Yang, Kaidong Feng, Hui Fang, Xinghua Qu, and Yew Soon Ong. 2022. Revisiting bundle recommendation: Datasets, tasks, challenges and opportunities for intent-aware product bundling. In *SIGIR '22: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2900–2911.

Haoran Tang, Shiqing Wu, Guandong Xu, and Qing Li. 2023. Dynamic graph evolution learning for recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1589–1598.

Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 417–426.

Lei Wang and Ee-Peng Lim. 2023. Zero-shot next-item recommendation using large pretrained language models. *arXiv preprint arXiv:2304.03153*.

Shijie Wang, Wenqi Fan, Yue Feng, Lin Shanru, Xinyu Ma, Shuaiqiang Wang, and Dawei Yin. 2025. Knowledge graph retrieval-augmented generation for LLM-based recommendation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27152–27168.

Shoujin Wang, Liang Hu, Yan Wang, Quan Z. Sheng, Mehmet Orgun, and Longbing Cao. 2019a. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3771–3777.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019b. KGAT: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 950–958.

Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 169–178.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 346–353.

Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. 2023. LlamaRec: Two-stage recommendation using large language models for ranking. *arXiv preprint arXiv:2311.02089*.

Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024. AgentCF: Collaborative learning with autonomous language agents for recommender systems. In *Proceedings of the ACM Web Conference 2024*, pages 3679–3689.

Peiyan Zhang, Jiayan Guo, Chaozhuo Li, Yueqi Xie, Jae Boum Kim, Yan Zhang, Xing Xie, Haohan Wang, and Sunghun Kim. 2023. Efficiently leveraging multi-level user intent for session-based recommendation via Atten-Mixer network. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining*, pages 168–176.

Jieming Zhu, Guohao Cai, Junjie Huang, Zhenhua Dong, Ruiming Tang, and Weinan Zhang. 2023. ReLoop2: Building self-adaptive recommendation models via responsive error compensation loop. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5728–5738.

A Relation-Aware Graph Encoder

Node initialization. Each node $v \in V_0$ is initialized from a textual representation x_v. For an item node, x_v concatenates available item metadata such as title, category, keywords, and description; for an attribute node, x_v is constructed from the corresponding attribute text. A frozen text encoder maps x_v into an initial normalized representation:

$$h_v^{(0)} = \frac{\mathrm{Enc}(x_v)}{\|\mathrm{Enc}(x_v)\|_2}. \tag{13}$$

The text encoder is not updated during graph training or test-time inference.

Relation-aware propagation. To incorporate typed graph structure, we use a relation-aware graph attention encoder over the seed graph G_0. For an edge (u, r, v), the attention weight from node u to neighbor v under relation r at layer l is:

$$a_{u,r,v}^{(l)} = \mathrm{softmax}_{(u,r',v') \in \mathcal{N}(u)} \left(\frac{(h_v^{(l)})^\top W_r^{(l)} h_u^{(l)}}{\tau} \right), \tag{14}$$

where $W_r^{(l)}$ is a relation-specific transformation, τ is a temperature parameter, and $\mathcal{N}(u)$ denotes the typed neighborhood of u. Node representations are updated by aggregating relation-weighted messages:

$$h_u^{(l+1)} = \sigma \left(\sum_{(u,r,v) \in \mathcal{N}(u)} a_{u,r,v}^{(l)} W_r^{(l)} h_v^{(l)} \right). \tag{15}$$

The final node representation concatenates multi-hop representations:

$$h_v^* = h_v^{(0)} \| h_v^{(1)} \| \cdots \| h_v^{(L)}. \tag{16}$$

Offline training objective. The seed graph provides the structural substrate, but the graph encoder is trained offline using the training stream D_{train}. For each training instance (x_t, C_t, y_t), the system extracts candidate-conditioned graph evidence from G_0 around the context items and candidate items. For a candidate $i \in C_t$, the encoder produces graph-based evidence features from the retrieved paths or edges, denoted by $\phi_g(i \mid x_t, G_0)$. A lightweight scoring module then computes:

$$s_\theta(i \mid x_t, G_0) = \mathrm{Score}_\theta \left(h_i^*, h_{x_t}, \phi_g(i \mid x_t, G_0) \right), \tag{17}$$

where h_i^* is the graph representation of candidate item i, and h_{x_t} is a context representation obtained from the items or textual attributes in x_t.

The encoder and scoring module are optimized with a candidate-ranking loss over D_{train}:

$$\mathcal{L}_{\text{rank}} = -\sum_{(x_t, C_t, y_t) \in D_{\text{train}}} \log \frac{\exp(s_\theta(y_t | x_t, G_0))}{\sum_{i \in C_t} \exp(s_\theta(i | x_t, G_0))}. \tag{18}$$

This objective encourages the encoder to assign higher graph-evidence scores to the ground-truth target item than to other candidates in the same candidate set.

Test-time retrieval. Before one-pass test inference, the graph encoder and scoring module are frozen. At test step t, the current SGM state $\mathcal{M}_{s,t}$ may differ from the initial seed graph because accepted memory edits can update edge weights, confidence scores, support counts, or add and prune semantic edges. Given the current context x_t and candidate set C_t, the retriever extracts local candidate-conditioned subgraphs from $\mathcal{M}_{s,t}$, including paths connecting context items, candidate items, and metadata nodes. The frozen encoder is then used only to score and rank these edges or paths; no encoder parameters are updated on the test stream. The top-ranked graph evidence is returned as R_t^s and used by the Recommender Agent.

B Quality Gate Details

The quality gate U_ϕ is a deterministic write filter, not a learned model. It uses the write signals in Eq. 8 to decide whether an Analyzer-generated proposal can update SGM. The gate can be expressed as a thresholded score:

$$\begin{aligned} \mathrm{WriteScore}_t(p) = {} & \theta_1 O_t(p) + \theta_2 Q_t(p) \\ & + \theta_3 \mathrm{Support}_t(p) + \theta_4 \mathrm{Recency}_t(p) \\ & - \theta_5 \mathrm{Conflict}_t(p) - \theta_6 \mathrm{Cost}_t(p). \end{aligned} \tag{19}$$

Here, positive terms reward useful outcomes, proposal quality, repeated support, and recency, while negative terms penalize conflict with SGM and memory cost. A proposal is accepted if

$$\mathrm{Accept}(p) = \mathbb{I}[\mathrm{WriteScore}_t(p) \geq \eta_a]. \tag{20}$$

All coefficients and thresholds are included in ϕ, fixed before inference, and not tuned on the test stream. Action-specific rules constrain the decision: reinforcement requires positive evidence, suppression requires repeated misleading evidence, promotion requires repeated support and low conflict, and pruning targets stale or low-utility relations.

C SGM Update Rules

The $\mathrm{Apply}(\cdot)$ operator updates SGM by executing the accepted typed actions in $\mathcal{A}_t$. Each edge $e = (u, r, v)$ maintains a support count $n_{e,t}$, confidence $c_{e,t}$, and weight $w_{e,t}$. These attributes are updated conservatively so that a single Analyzer proposal cannot immediately create or remove stable semantic knowledge.

For a `reinforce_edge` action, the edge receives additional support and its reliability is increased:

$$n_{e,t+1} = n_{e,t} + 1, \tag{21}$$

$$c_{e,t+1} = \min(1, c_{e,t} + \Delta c), \tag{22}$$

$$w_{e,t+1} = w_{e,t} + \Delta w. \tag{23}$$

Here, $\Delta c > 0$ and $\Delta w > 0$ are fixed update steps for confidence and edge weight. Confidence is clipped to the range $[0, 1]$, while the weight controls the strength of the relation during later retrieval.

For a `suppress_edge` action, the edge is treated as potentially misleading and its reliability is reduced:

$$c_{e,t+1} = \max(0, c_{e,t} - \Delta c), \tag{24}$$

$$w_{e,t+1} = \max(0, w_{e,t} - \Delta w). \tag{25}$$

This operation does not necessarily delete the edge immediately. Instead, it weakens relations that repeatedly conflict with observed outcomes or retrieve misleading evidence.

For `add_tentative_edge`, a new relation is inserted with low initial confidence and limited weight. Such an edge must accumulate repeated support before it can be promoted or treated as stable semantic knowledge. This prevents a single noisy observation from becoming a permanent graph relation.

Graph compaction is handled through deterministic pruning or deactivation. An edge is removed or deactivated when all of its utility indicators fall below predefined minimum thresholds:

$$w_{e,t} < \omega_{\min} \ \wedge \ c_{e,t} < c_{\min} \ \wedge \ n_{e,t} < n_{\min}.$$

This rule removes relations that are simultaneously weak, low-confidence, and insufficiently supported. It keeps SGM compact and limits the accumulation of noisy or stale edges during long one-pass evaluation streams. All update steps and pruning thresholds are fixed before test-time inference.

D Implementation Hyperparameters

Table 6 reports the default implementation configuration used in our experiments. Unless otherwise stated, the same values are shared across datasets. These settings are fixed before one-pass test-time inference and are not tuned on the test stream. During evaluation, the graph scorer, and prompt templates remain frozen; adaptation occurs only through SGM and ELM updates.

E Runtime and Token Overhead

Table 7 shows that the two LLM calls dominate runtime, while memory retrieval and update add approximately 0.2 s per instance. Token usage varies across instances with context and retrieved memory. Overall, the measurements highlight the practical trade-off of the dual-agent design: CGM-Rec avoids test-time parameter updates but requires two LLM calls per recommendation.

F Full-Catalog for Neural Baselines

To verify the fairness of the shared sampled-candidate protocol, NARM, STAMP, GCE-GNN, and Atten-Mixer are additionally evaluated on ML-1M under both sampled-20 and full-catalog ranking. The processed evaluation catalog contains 3,416 eligible items. Full-catalog evaluation ranks all eligible items, whereas sampled-20 evaluation restricts the same model score vector to the shared 20 candidates. No retraining or parameter tuning is performed between protocols; test contexts, item mappings, scoring functions, eligibility masks, and tie-breaking rules remain unchanged.

As expected, absolute ranking scores decrease under the 3,416-item catalog. Restricting each full-catalog score vector to its 20 candidates reproduces the exact sampled-20 target ranks. This consistency check confirms that the neural baselines produce candidate-independent catalog scores and that the shared 20-item protocol compares all methods on an identical reranking space. Full-catalog numbers serve only as a neural baseline sanity check, not for direct comparison with CGM-Rec.

G Detailed Ablation Study (RQ3)

Table 9 separates two complementary effects in CGM-Rec: dual-memory composition and feedback-driven semantic-graph updating. The dual-memory contribution is reflected by the difference between Static SGM+ELM and the stronger single-memory variant, whereas the write contribution is reflected by the difference between Full CGM-Rec and Static SGM+ELM. Across datasets, the static dual-memory configuration yields higher mean performance than either memory alone, while enabling feedback-driven graph updates provides a further mean gain whose magnitude varies by dataset. The write contribution is modest on Bundle and Games, more noticeable on ML-1M, and most pronounced on ML-100K, where HR@5 increases from 0.4460 to 0.5941. Overall, these results indicate that dual-memory retrieval contributes beyond either memory alone, while the complete feedback-driven write pathway provides an additional, dataset-dependent source of adaptation.

Dataset	Source interactions	Eligible items	Train episodes	Test episodes	Train context (min/mean/max)	Test context (min/mean/max)
Bundle	18,886*	14,240	146	238	2 / 5.53 / 9	3 / 6.05 / 9
Games	497,577	17,389	141	1,000	1 / 3.25 / 10	1 / 3.61 / 9
ML-1M	1,000,209	3,416	150	1,000	19 / 19.00 / 19	1 / 18.01 / 19
ML-100K	100,000	1,500	200	743	19 / 42.74 / 50	19 / 41.73 / 50

Table 5: Dataset and processed recommendation-stream statistics. Source interactions denote benchmark events before episode construction, while eligible items denote the post-filter ranking vocabulary. Train and test counts denote processed recommendation episodes rather than source interactions. Context length is reported as minimum/mean/maximum. *The Bundle source release reports 18,886 interactions; the artifact-local `user_item` snapshot contains 16,966 rows.

Component	Default configuration
Data / stream setup	Candidate size = 20 for all compared methods; random seed = 42; seed initialization = 20% of the training stream.
SGM seed graph	Keyword top-k = 5; co-occurrence window = 5; item descriptions enabled.
Graph scorer	Hidden dimension = 32; max edges per seed node = 64; optimizer = Adam.
ELM retrieval	Retrieved lessons top-k = 5; BM25 $k_1 = 1.2$, $b = 0.75$.
Analyzer feedback	Success threshold = Top-5; max success edits = 4; max failure edits = 6; reserved tentative edits = 2.
Typed edit deltas	Reinforce $(\Delta w, \Delta c, \Delta q) = (0.20, 0.05, 0.03)$; suppress $= (-0.15, -0.04, -0.03)$; tentative $= (0.10, 0.02, 0.02)$.
Quality gate	Reinforce threshold = 0.35; suppress threshold = 0.45; tentative threshold = 0.30; contribution scale = 0.25; support scale = 5.0; max conflict penalty = 0.75.
Semantic promotion	Min support = 3; min confidence = 0.50; min quality = 0.45; min unique contexts = 2; check interval = 20 steps.
LLM output control	JSON response format; max generation attempts = 3.

Table 6: Default implementation configuration for CGM-Rec. Unless otherwise stated, the same settings are used across datasets, fixed before one-pass test-time inference, and not tuned on the test stream.

Component	Avg. Latency (s)	Avg. Input	Avg. Output
Recommender	4.6	9.66K	0.78K
Analyzer	4.8	10.27K	0.39K
Memory retrieval/update	0.2	–	–
Total	**9.6**	**19.93K**	**1.17K**

Table 7: Average per-instance inference overhead of CGM-Rec. Input and output columns report token counts; memory operations do not invoke the LLM.

Model	HR@1	HR@5	NDCG@5	Med. Rank ↓
NARM	0.0000	0.0010	0.0004	1483.0
STAMP	0.0040	0.0080	0.0060	747.0
GCE-GNN	0.0010	0.0070	0.0043	1029.0
Atten-Mixer	0.0020	0.0070	0.0044	828.5

Table 8: Full-catalog sanity check for neural baselines on ML-1M with 3,416 eligible items.

H Quality-Gate Sensitivity Analysis

To examine whether CGM-Rec depends on narrowly tuned quality-gate parameters, a one-factor-at-a-time sensitivity analysis is conducted on ML-100K. The action-specific acceptance threshold η_{sup} and the outcome-signal coefficient θ_{out} are varied independently, while all other gate parameters, typed update rules, prompts, candidate sets, memory budgets, and initial memory states remain fixed. The default configuration is shown in bold.

As shown in Tables 10 and 11, varying either gate parameter substantially changes the accepted-update rate while recommendation performance remains relatively stable. Since accepted updates affect future states and retrieval, the analysis captures end-to-end sensitivity of the quality-gated update mechanism. The default configuration is retained without post-hoc selection, and the results indicate robustness to moderate gate perturbations rather than arbitrary parameter settings.

I Candidate-Pool Scaling Analysis

To evaluate robustness beyond the original 20-item reranking protocol, CGM-Rec is compared with K-RagRec under larger candidate pools on Games and ML-100K. Both methods use identical test contexts, candidate files, instance order, and five matched candidate-set seeds $\{0, 10, 42, 625, 2023\}$. The candidate sets are nested as $C_{20} \subset C_{50} \subset C_{80}$, such that larger pools preserve the original candi-

Dataset	Method	HR@1	HR@5	HR@10	NDCG@1	NDCG@5	NDCG@10
Bundle	SGM only	0.2748 ± 0.0125	0.5486 ± 0.0148	0.6842 ± 0.0163	0.2748 ± 0.0125	0.4201 ± 0.0127	0.4702 ± 0.0139
	ELM only	0.2827 ± 0.0141	0.5538 ± 0.0155	0.7105 ± 0.0149	0.2827 ± 0.0141	0.4246 ± 0.0133	0.4753 ± 0.0142
	Static SGM+ELM	0.3101 ± 0.0154	0.5634 ± 0.0126	0.7168 ± 0.0138	0.3101 ± 0.0154	0.4467 ± 0.0131	0.4859 ± 0.0137
	Full CGM-Rec	$\mathbf{0.3263 \pm 0.0175}$	$\mathbf{0.5716 \pm 0.0098}$	$\mathbf{0.7290 \pm 0.0125}$	$\mathbf{0.3263 \pm 0.0175}$	$\mathbf{0.4495 \pm 0.0146}$	$\mathbf{0.5060 \pm 0.0140}$
Games	SGM only	0.2990 ± 0.0128	0.5864 ± 0.0160	0.6985 ± 0.0180	0.2990 ± 0.0128	0.4435 ± 0.0152	0.4760 ± 0.0160
	ELM only	0.3021 ± 0.0132	0.5848 ± 0.0165	0.7106 ± 0.0171	0.3021 ± 0.0132	0.4426 ± 0.0156	0.4790 ± 0.0162
	Static SGM+ELM	0.3050 ± 0.0129	0.5877 ± 0.0168	0.7138 ± 0.0165	0.3050 ± 0.0129	0.4458 ± 0.0160	0.4822 ± 0.0158
	Full CGM-Rec	$\mathbf{0.3092 \pm 0.0131}$	$\mathbf{0.5898 \pm 0.0176}$	$\mathbf{0.7240 \pm 0.0160}$	$\mathbf{0.3092 \pm 0.0131}$	$\mathbf{0.4475 \pm 0.0168}$	$\mathbf{0.4868 \pm 0.0156}$
ML-1M	SGM only	0.1327 ± 0.0078	0.5086 ± 0.0138	0.6812 ± 0.0125	0.1327 ± 0.0078	0.3250 ± 0.0098	0.3810 ± 0.0105
	ELM only	0.1410 ± 0.0081	0.5500 ± 0.0132	0.7515 ± 0.0121	0.1410 ± 0.0081	0.3545 ± 0.0107	0.4180 ± 0.0112
	Static SGM+ELM	0.1675 ± 0.0093	0.5572 ± 0.0130	0.7582 ± 0.0118	0.1675 ± 0.0093	0.3667 ± 0.0114	0.4300 ± 0.0117
	Full CGM-Rec	$\mathbf{0.2049 \pm 0.0117}$	$\mathbf{0.5786 \pm 0.0174}$	$\mathbf{0.7715 \pm 0.0142}$	$\mathbf{0.2049 \pm 0.0117}$	$\mathbf{0.3878 \pm 0.0156}$	$\mathbf{0.4395 \pm 0.0138}$
ML-100K	SGM only	0.1185 ± 0.0105	0.3220 ± 0.0185	0.5790 ± 0.0200	0.1185 ± 0.0105	0.2210 ± 0.0148	0.3050 ± 0.0160
	ELM only	0.1372 ± 0.0118	0.4050 ± 0.0190	0.6690 ± 0.0182	0.1372 ± 0.0118	0.2695 ± 0.0165	0.3520 ± 0.0171
	Static SGM+ELM	0.1538 ± 0.0102	0.4460 ± 0.0160	0.6765 ± 0.0150	0.1538 ± 0.0102	0.2978 ± 0.0138	0.3655 ± 0.0145
	Full CGM-Rec	$\mathbf{0.2994 \pm 0.0032}$	$\mathbf{0.5941 \pm 0.0087}$	$\mathbf{0.7567 \pm 0.0105}$	$\mathbf{0.2994 \pm 0.0032}$	$\mathbf{0.4508 \pm 0.0096}$	$\mathbf{0.5015 \pm 0.0119}$

Table 9: Ablation results reported as mean±SD over five matched seeds. Static SGM+ELM retains both memory components while disabling feedback-driven semantic-graph updates.

η_{sup}	HR@1	HR@5	NDCG@5	Accept. (%)
0.25	0.2968	0.5970	0.4564	91.8
0.35	0.2992	0.6014	0.4600	89.8
0.45	**0.3005**	**0.6032**	**0.4618**	**87.3**
0.55	0.3011	0.6040	0.4625	81.7
0.65	0.2961	0.5965	0.4553	70.6

Table 10: Sensitivity to the acceptance threshold η_{sup} on ML-100K. "Accept.denotes the percentage of proposed typed updates accepted by the quality gate.

θ_{out}	HR@1	HR@5	NDCG@5	Accept. (%)
0.10	0.2955	0.5958	0.4540	72.5
0.15	0.2987	0.6009	0.4592	80.6
0.20	**0.3005**	**0.6032**	**0.4618**	**87.3**
0.25	0.3012	0.6038	0.4622	91.3
0.30	0.2972	0.5988	0.4570	94.8

Table 11: Sensitivity to the outcome-signal coefficient θ_{out} on ML-100K. The default configuration is shown in bold.

dates while introducing additional harder negatives.

Table 12 shows the expected decline in absolute ranking performance as the candidate pool becomes larger and more challenging. Nevertheless, CGM-Rec maintains higher mean performance than K-RagRec across all evaluated metrics and pool sizes. For HR@5, the relative gain on Games increases from 6.3% at $C = 20$ to 18.1% at $C = 80$, while ML-100K retains gains of 15.2–25.2% across the tested pools. The HR@5 improvements are statistically significant at every candidate size on both datasets. These results indicate that the advantage of CGM-Rec persists beyond the original 20-item setting as the sampled reranking space grows. This analysis evaluates robustness to larger sampled candidate pools rather than full-catalog recommendation or training-scale scalability.

J Discussion

The results suggest that CGM-Rec gains primarily come from explicit memory maintenance rather than test-time parameter updates. The graph encoder, LLM agents, and prompt templates remain fixed during testing; only SGM and ELM are updated. Ablations further show that the two memories are complementary: ELM reuses recent outcome-derived lessons, while SGM preserves stable relational knowledge through conservative, quality-gated edits.

The evaluation should be interpreted within its intended scope. The examples are anonymous or weakly linked, so adaptation targets transferable graph knowledge across contexts rather than lifelong per-user modeling. The candidate sets are sampled, so the results measure constrained reranking rather than full-catalog recommendation. ML-100K is treated separately as a metadata-rich contextual reranking setting rather than as a natural session stream. These constraints motivate future work on full-catalog ranking, harder negative sampling, and broader domains.

Ethical Considerations

We use established public recommendation benchmarks and do not collect new user data. The processed inputs contain no personally identifying information and are restricted to interaction/context signals and item metadata required for recommendation. Continual recommendation systems may amplify popularity bias, spurious correlations, or

C	Metric	K-RagRec	CGM-Rec	Rel. Gain	p-value
Games					
20	HR@1	0.2850 ± 0.0110	$\mathbf{0.3092 \pm 0.0131}$	+8.5%	0.0039
	HR@5	0.5550 ± 0.0139	$\mathbf{0.5898 \pm 0.0176}$	+6.3%	0.0028
	NDCG@5	0.4140 ± 0.0123	$\mathbf{0.4475 \pm 0.0168}$	+8.1%	0.0139
50	HR@1	0.2000 ± 0.0104	$\mathbf{0.2385 \pm 0.0104}$	+19.2%	< 0.001
	HR@5	0.4200 ± 0.0146	$\mathbf{0.4747 \pm 0.0106}$	+13.0%	< 0.001
	NDCG@5	0.3200 ± 0.0128	$\mathbf{0.3624 \pm 0.0107}$	+13.3%	< 0.001
80	HR@1	0.1600 ± 0.0104	$\mathbf{0.1989 \pm 0.0063}$	+24.3%	0.0045
	HR@5	0.3600 ± 0.0145	$\mathbf{0.4253 \pm 0.0075}$	+18.1%	0.0017
	NDCG@5	0.2700 ± 0.0125	$\mathbf{0.3177 \pm 0.0070}$	+17.7%	0.0034
ML-100K					
20	HR@1	0.2023 ± 0.0139	$\mathbf{0.2994 \pm 0.0032}$	+48.0%	< 0.001
	HR@5	0.4746 ± 0.0258	$\mathbf{0.5941 \pm 0.0087}$	+25.2%	< 0.001
	NDCG@5	0.3318 ± 0.0268	$\mathbf{0.4508 \pm 0.0096}$	+35.9%	< 0.001
50	HR@1	0.1276 ± 0.0197	$\mathbf{0.1491 \pm 0.0044}$	+16.9%	0.1067
	HR@5	0.3327 ± 0.0263	$\mathbf{0.3833 \pm 0.0060}$	+15.2%	0.0216
	NDCG@5	0.2259 ± 0.0237	$\mathbf{0.2683 \pm 0.0055}$	+18.8%	0.0277
80	HR@1	0.0809 ± 0.0219	$\mathbf{0.1098 \pm 0.0040}$	+35.7%	0.0610
	HR@5	0.2407 ± 0.0281	$\mathbf{0.2886 \pm 0.0066}$	+19.9%	0.0312
	NDCG@5	0.1620 ± 0.0260	$\mathbf{0.2026 \pm 0.0052}$	+25.1%	0.0388

Table 12: Candidate-pool scaling on Games and ML-100K, reported as mean±SD over five matched candidate-set seeds. Larger candidate sets are nested and preserve the original candidates while introducing additional harder negatives. Relative gains are computed with respect to K-RagRec; p-values are obtained from paired two-sided t-tests across matched seeds.

harmful feedback loops when biased outcomes are repeatedly written into memory. CGM-Rec mitigates this risk through structured evidence, conservative quality-gated updates, suppression of misleading relations, and repeated support before semantic promotion. When user-level histories or sensitive attributes are available, memory construction must respect privacy constraints, dataset licenses, and data minimization principles. This work is intended for offline research evaluation; deployment requires further auditing for fairness, privacy, robustness, and memory-induced bias.

K Recommender Agent Prompt Template

The Recommender Agent uses a fixed instruction template across datasets. At inference time, dataset-specific fields are filled with the current interaction context, candidate items, retrieved SGM evidence, and retrieved ELM lessons. The same core instruction is used across domains, while available item attributes such as category, genre, taxonomy, keywords, or descriptions are inserted when applicable.

Prompt Template: Recommender Agent

Task Instruction. This is a recommendation reranking task. Analyze the user's current interaction context and the candidate set to infer the user's preferences and current intent. Use retrieved graph evidence and episodic lessons to support the reranking decision.

Inputs

1. **User interaction context:** `{interaction_context}`
2. **Candidate set** *(rerank exactly these items):* `{candidate_set}`
3. **Retrieved graph evidence:** `{graph_evidence}`
4. **Retrieved episodic lessons:** `{episodic_lessons}`

Reasoning Instructions

1. Identify preference patterns, item combinations, or intent cues from the current interaction context.
2. Compare candidate items with the inferred intent using available attributes, such as category, genre, taxonomy, keywords, descriptions, price, or other dataset-specific metadata.
3. Use graph evidence to identify relevant semantic relations, item–attribute links, or item–item associations.
4. Use episodic lessons to account for previous success patterns, failure cases, or corrective hints from similar contexts.

5. Rerank all candidate items by their likelihood of matching the user's current intent.

Critical Rules

- Do not introduce items outside the Candidate Set.
- Return all candidate items ordered by relevance.
- The recommendations field must contain only candidate items and their corresponding indices.
- Return strictly in JSON format with no markdown.

Output Format

```
{
"reasoning": "short analysis of user
intent and ranking evidence",
"recommendations": [
{"index": 1, "item": "item name 1"},
{"index": 2, "item": "item name 2"}
]
}
```

L Analyzer Agent Prompt Template

The Analyzer Agent is invoked after the recommendation outcome is observed. It receives an outcome package containing the generated ranking, target item, outcome label, retrieved SGM evidence, and retrieved ELM lessons. The Analyzer produces reusable textual lessons and structured memory-edit proposals. These outputs are treated only as candidates and must pass the quality-gated write policy before any modification is applied to SGM.

Prompt Template: Analyzer Agent

Task Instruction. This is a post-feedback analysis task for a recommendation system. Analyze the recommendation outcome, identify useful or misleading evidence, extract reusable lessons, and propose possible memory-edit actions. All proposed updates are candidate actions and will be checked by a quality gate.

Inputs

1. **User interaction context:** `{interaction_context}`
2. **Candidate set:** `{candidate_set}`
3. **Generated ranking:** `{predicted_ranking}`
4. **Target item and rank:** `{target_item}`, `{target_rank}`
5. **Outcome label:** `{outcome_label}`
6. **Retrieved graph evidence:** `{graph_evidence}`
7. **Retrieved episodic lessons:** `{episodic_lessons}`

Analysis Instructions

1. Check whether the generated ranking matches the target item and current user intent.
2. Identify helpful evidence, such as useful item–attribute relations, item–item links, or intent cues.
3. Identify misleading evidence, such as broad categories, weak relations, noisy attributes, or irrelevant lessons.
4. Generate concise textual lessons for similar future contexts.
5. Propose structured edit actions only when supported by the outcome package.
6. Do not invent unsupported items, attributes, relations, or evidence.

Allowed Candidate Outputs

- `store_lesson`: store a textual lesson in ELM.
- `add_tentative_edge`: propose a low-confidence candidate relation for SGM.
- `reinforce_edge`: strengthen a useful existing relation.
- `suppress_edge`: downweight a misleading relation.
- `promote_to_semantic`: promote a reliable recurring pattern into SGM.
- `prune_stale_edge`: remove or deactivate a stale low-utility relation.

Critical Rules

- Output only candidate lessons and candidate edit proposals.
- Every edit proposal must include supporting evidence from the outcome package.
- If evidence is insufficient, return an empty list of edit proposals.
- Return strictly in JSON format with no markdown.

Output Format

```
{
"outcome_summary": {
"target_rank": "<target rank>",
"outcome": "<success or failure>",
"brief_explanation": "short explanation of the
outcome"
},
"lessons": [
{
"lesson_type": "success_pattern |
failure_cause | intent_cue | corrective_hint",
"lesson_text": "concise reusable lesson for
similar future contexts",
"supporting_evidence": ["evidence item 1",
"evidence item 2"]
}
],
"edit_proposals": [
{
"action": "add_tentative_edge | reinforce_edge
| suppress_edge | promote_to_semantic |
prune_stale_edge",
"target": "edge, relation, or lesson-derived
pattern",
"update_value": "increase confidence, decrease
weight, or add tentative relation",
"confidence": "low | medium | high",
"supporting_evidence": ["evidence item 1",
"evidence item 2"],
"rationale": "why this proposal may improve
future recommendations"
}
]
}
```