LLM-as-RNN: Feedback-Driven Memory Rewriting for Online Learning and Sequential Prediction

Yuxing Lu[1,2*], **J. Ben Tamo**[1*], **Jinrui Fang**[3], **Weichen Zhao**[4], **Nan Sun**[5]
Xukai Zhao[6], **Yishan Zhong**[1], **Wenqi Shi**[7], **Jinzhuo Wang**[2†], **May Dongmei Wang**[1†]

[1]Georgia Institute of Technology, Atlanta, GA, USA
[2]Peking University, Beijing, China; [3]The University of Texas at Austin, Austin, TX, USA
[4]Shandong University, Jinan, China
[5]Huazhong University of Science and Technology, Wuhan, China
[6]Tsinghua University, Beijing, China
[7]Washington University in St. Louis, St. Louis, MO, USA

wangjinzhuo@pku.edu.cn, maywang@gatech.edu

Abstract

Large language models (LLMs) are strong sequence predictors, but standard context accumulation offers no explicit mechanism for revising the memory carried between predictions. We propose LLM-as-RNN, an inference-time framework for online learning with frozen LLMs through a bounded natural-language memory state. At each timestep, the model predicts from the current observation and its memory, receives feedback, and rewrites the state to correct errors and retain task-relevant patterns without updating model parameters. Implemented as a structured system prompt, this state makes the adaptation process inspectable through human-readable memory traces. We evaluate LLM-as-RNN on sequential prediction benchmarks in healthcare, meteorology, and finance across Llama, Gemma, and GPT-oss model families. Comparisons with static-context and feedback-aligned baselines examine predictive performance, feedback sources, memory budgets, and inference cost. Our results demonstrate that feedback-driven memory rewriting supports online error correction and sequential prediction under a fixed memory budget, with performance varying across tasks and backbones.

1 Introduction

Learning from sequential feedback is fundamental to adaptive prediction (Zhang et al., 2024b; Jiang et al., 2024). Historically, this requirement was met by Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs), which maintained a compact, evolving hidden state to capture temporal dependencies and adapt to shifting data distributions (Hochreiter and Schmidhuber, 1997; Cho et al., 2014). The advent of Transformer architectures revolutionized this landscape by replacing recurrence with large-scale parallel attention mechanisms. In this paradigm, "memory" is no longer a compressed state, but an explicit history of tokens processed via In-Context Learning (ICL) (Brown et al., 2020). This shift has enabled remarkable advances in reasoning and generation across diverse open and proprietary model families (Wu et al., 2025; Du et al., 2025).

However, this architectural trade-off introduces a critical limitation in long-horizon settings. During standard inference, Large Language Models (LLMs) operate in a largely stateless way: with frozen parameters, the system lacks a mutable memory to internalize past mistakes (Shinn et al., 2023; Packer et al., 2023). Instead of updating a belief state, the model relies on an append-only context window, carrying errors forward without correction (Wang et al., 2024a; Muhoberac et al., 2025). This limitation becomes acute in domains such as longitudinal clinical prediction, weather forecasting, and financial time-series modeling, where task-relevant signals accumulate over time, and the data distribution may drift.

A common solution is to encode the entire past directly in the prompt. One approach, Full History Concatenation (FHC) (Ascoli and Choi, 2025), appends all raw observations, while methods like MemPrompt (Madaan et al., 2022) append a step-wise summary. As the sequence grows, concatenation suffers from attention dilution and 'lost-in-the-middle' phenomena (Liu et al., 2024), while append-only summaries are prone to error cascad-

[*]Equal contribution.
[†]Corresponding authors.

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 4977–4997
November 6-10, 2026 ©2026 Association for Computational Linguistics

ing (Zhang et al., 2024a). Once a misconception is written into the context, it becomes an immutable ground truth; later evidence often fails to override it, causing errors to persist despite contradictory signals (Turpin et al., 2023).

We address this gap with LLM-as-RNN, an inference-only framework that reframes sequential prediction as a recurrent process with a natural-language state. Unlike append-only approaches, our method updates a structured memory at each step using feedback, derived from ground-truth labels or an LLM critic, to correct errors and refine strategies under a fixed token budget. We evaluate this approach on three diverse benchmarks: clinical prediction (MIMIC-IV), meteorology (Weather), and financial forecasting (S&P 500). LLM-as-RNN yields especially large gains on semantic and regime-shifting sequences and remains bounded in context length. It achieves improvements of 10.8% on MIMIC-IV, 1.6% on Weather, and 4.8% on S&P 500, while producing interpretable learning traces that make the model's adaptation process transparent.

This work makes three contributions: (1) We formalize recurrent inference for LLMs, treating textual state as an explicit, mutable memory. This perspective fundamentally distinguishes revisable memory updates from standard unbounded history accumulation. (2) We introduce LLM-as-RNN, an inference-only framework that enables online adaptation in frozen models. By iteratively rewriting a bounded natural-language state using per-timestep feedback, the model corrects errors without parameter access. (3) We demonstrate across three domains and multiple model families that outcome-driven state updates are competitive with strong prompt-based baselines while keeping the recurrent state bounded. Across 450 recorded state updates, 107 shortened the carried state (median 27 word types removed), confirming that updates rewrite rather than append. Furthermore, by exposing the adaptation process as human-readable state evolution, our framework facilitates safety audits and builds trust, ensuring that the model's reasoning trajectory is transparent rather than implicit.

2 Related Work

2.1 Recurrent and Memory Models

Classical sequence modeling uses recurrent neural networks (RNNs), long short-term memory (LSTMs), and gated recurrent units (GRUs) to maintain a vector-valued hidden state that evolves over time (Elman, 1990; Hochreiter and Schmidhuber, 1997; Chung et al., 2014; Graves et al., 2013; Sutskever et al., 2014; Bahdanau et al., 2015). While efficient, these dense vector states often act as an information bottleneck. To address this, memory-augmented architectures, such as Neural Turing Machines and Differentiable Neural Computers, separated the controller from an external differentiable memory bank to support algorithmic reasoning and long-term dependencies (Weston et al., 2014; Graves et al., 2014, 2016; Santoro et al., 2016).

The Transformer architecture replaced this explicit recurrence with self-attention over a global context (Vaswani et al., 2017). While powerful, the quadratic cost of attention has sparked a resurgence of interest in linear-time recurrent architectures. Recent models like RWKV (Peng et al., 2023), Mamba (Gu and Dao, 2024), and linear attention variants (Katharopoulos et al., 2020) effectively reintroduce recurrence into the Transformer backbone, formalizing decoder-only models as multi-state RNNs (Arora et al., 2024; Oren et al., 2024).

However, these approaches typically require training custom architectures from scratch. In the regime of frozen large language models, recurrence is simulated via prompt management. RecurrentGPT (Zhou et al., 2023) and MemGPT (Packer et al., 2023) emulate RNNs by treating the context window as a short-term buffer and offloading history to external storage. LLM-as-RNN builds upon this stateful perspective but distinguishes itself by representing the recurrent state not as a latent vector or a static storage log, but as an evolving, natural-language system prompt.

2.2 Inference-Time Adaptation in LLMs

LLMs exhibit strong in-context learning capability, adapting to new tasks from a handful of demonstrations without parameter updates. This behavior has been interpreted as implicit optimization or meta-learning implemented in the forward pass (Brown et al., 2020; Akyürek et al., 2022; Garg et al., 2022; Von Oswald et al., 2023; Dai et al., 2023). To extend this beyond the context window, retrieval-augmented generation (RAG) and non-parametric systems utilize external memory banks to query relevant history at inference time (Khandelwal et al., 2020; Lewis et al., 2020; Borgeaud et al., 2022; Izacard et al., 2023). Recent agentic frameworks extend this by storing high-level "events" or

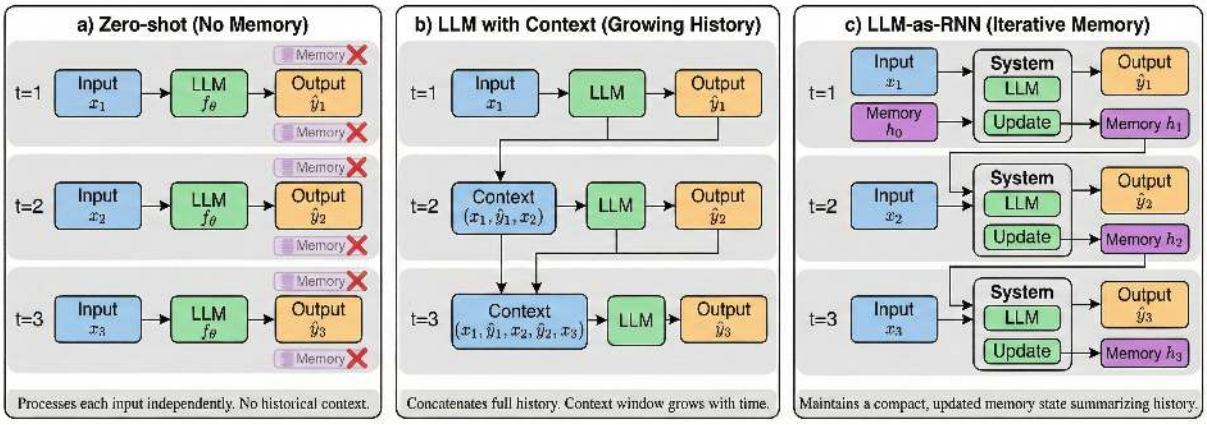


Figure 1: Illustrative comparison. (a) Simple LLM lacks memory. (b) LLM with Context suffers from growing input size. (c) LLM-as-RNN uses an iterative memory state to summarize historical information from evaluating outputs.

user profiles to support long-horizon personalization (Park et al., 2023; Zhong et al., 2024; Das et al., 2024; Wang et al., 2023). However, these approaches are primarily retrieval-based: they select relevant past information but do not necessarily update a belief state to correct errors. WriteBack-RAG enriches an external knowledge store offline by distilling retrieved evidence from labeled examples into compact knowledge units indexed alongside the original corpus (Lu et al., 2026).

A complementary stream of research treats natural language itself as an optimization variable. Methods like Self-Refine (Madaan et al., 2023), Reflexion (Shinn et al., 2023), and ReAct (Yao et al., 2022) introduce iterative feedback loops where the model critiques its own output to improve performance. This concept has been formalized in frameworks like OPRO (Yang et al., 2024) and TextGrad (Yuksekgonul et al., 2024), which perform "optimization via prompting", effectively backpropagating textual feedback to refine system prompts or solutions. DoctorRAG applies multi-agent textual gradients to revise medical-generation prompts within a query against retrieved context and patient-specific criteria (Lu et al., 2025).

Long-term memory has also been pursued more directly. MemoryLLM (Wang et al., 2024b) embeds a fixed-size latent memory pool inside the transformer and updates it with new knowledge through a learned self-update mechanism, keeping the model current without retraining. In the agentic setting, LangMem (LangChain, 2025) provides long-term memory primitives (semantic, episodic, and procedural stores) with background consolidation that refines an agent's own instructions from experience. These approaches share our goal of persistent, updatable memory without full retraining, but differ in mechanism: MemoryLLM's state is a trained latent pool, and LangMem manages memories as retrieved records across sessions, whereas LLM-as-RNN maintains a single bounded natural-language state that is rewritten in place at every step under a fixed token budget, with no additional training. Reflexion (Shinn et al., 2023) is the closest of the feedback-loop methods: it stores verbal self-reflections as episodic memory carried across trials. LLM-as-RNN differs in that its state is rewritten rather than accumulated, and is held under an explicit token budget at every step.

Compared with retrieval over a fixed corpus and traditional continual learning through parameter updates (Wu et al., 2024), our approach adapts a bounded textual state at inference time. By treating the system prompt as a recurrent hidden state and updating it through error-driven feedback, we combine recurrent statefulness with feedback-based adaptation in frozen LLMs.

3 Methods

3.1 Preliminaries and Problem Formulation

We consider sequential generation over a horizon T, where the cumulative full history H_T may exceed the LLM's context window L_{max}. At each step t, the model (parameterized by θ) receives an observation x_t, the previous history $H_{t-1} = \{x_{1:t-1}, \hat{y}_{1:t-1}\}$, and generates a response $\hat{y}_t$.

$$\hat{y}_t \sim f_\theta(\cdot \mid H_{t-1}, x_t) \tag{1}$$

This formulation suffers from two limitations: (1) Computational: $|H_{t-1} \oplus x_t|$ grows linearly, eventually violating L_{max}; and (2) Statelessness:

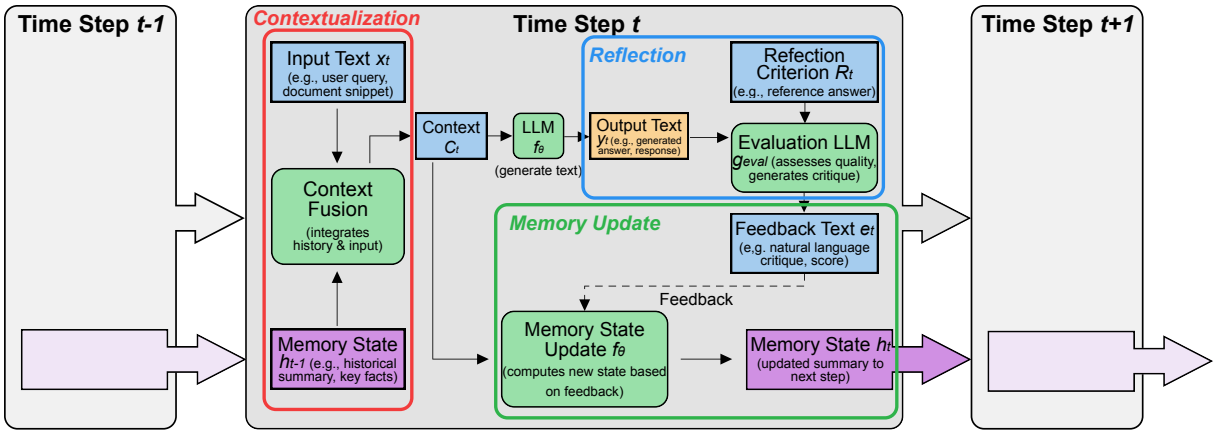


Figure 2: Overview of LLM-as-RNN framework. At each time step, the system fuses the previous memory state with new input to generate a response, evaluates that response to create a feedback signal, and then updates the natural language memory state to guide future interactions.

H_{t-1} is an append-only log. Errors in early outputs $\hat{y}_{1:t-1}$ are frozen in the context, permanently biasing future predictions.

To address this, we propose **LLM-as-RNN** (Figure 2), which reformulates inference as a recurrent process over a *mutable* textual state h_t. Our goal is to maintain a bounded memory h_{t-1}, constrained by a fixed token budget λ (where $\lambda \ll L_{max}$), that acts as a semantic sufficient statistic for the full history. We seek a memory state that minimizes information loss, ensuring the state-conditioned prediction approximates the full-history:

$$P_\theta(\hat{y}_t|h_{t-1}, x_t) \approx P_\theta(\hat{y}_t|H_{t-1}, x_t) \quad (2)$$

3.2 Recurrent Inference Mechanism

The inference process at step t is decomposed into three atomic operations: *Contextualization*, *Reflection*, and *Memory Update*.

3.2.1 Step 1: Contextualization

Unlike vector-based RNNs, which fuse inputs via matrix multiplication, LLM-as-RNN performs state mixing directly in the token space. We define a prompt template $\mathcal{P}_{gen}$ that constructs a local context C_t by concatenating ($\oplus$) the system instructions $\mathcal{I}_{sys}$, the prior memory state h_{t-1}, and the new observation x_t:

$$C_t = \mathcal{I}_{sys} \oplus h_{t-1} \oplus x_t \quad (3)$$

The model then samples a response $\hat{y}_t$ conditioned on this bounded context:

$$\hat{y}_t \sim f_\theta(\cdot|\mathcal{P}_{gen}(C_t)) \quad (4)$$

This formulation ensures that the context size remains constant ($O(1)$) with respect to the sequence index t. Unlike full-history methods where attention costs grow linearly with time, our input length depends only on the bounded memory size λ and current observation $|x_t|$, preventing throughput degradation in long-horizon tasks.

3.2.2 Step 2: Reflection

To ensure the memory h_t remains accurate over long horizons, we introduce a feedback mechanism that guides the evolution of the memory state. We refer to this feedback informally as a "semantic gradient"; this is purely an analogy, since no gradients are computed and no parameters are updated. We define a Critic function g_{eval} that evaluates the prediction $\hat{y}_t$ against a reference criterion $\mathcal{R}_t$, producing a natural language feedback signal e_t:

$$e_t = g_{eval}(\hat{y}_t, \mathcal{R}_t) \quad (5)$$

We formulate g_{eval} to handle 2 supervision modes:

Supervised mode: $\mathcal{R}_t$ contains the ground truth label y_t. Here, g_{eval} computes the semantic residual: $e_t \leftarrow$ "Error: Expected y_t but generated $\hat{y}_t$."

Open-Ended mode: $\mathcal{R}_t$ represents a set of quality heuristics (e.g., relevance, coherence). Here, g_{eval} acts as an *LLM-as-a-Judge*, producing a critique: $e_t \leftarrow$ "Reasoning flaw: $\hat{y}_t$ contradicts prior fact ..."

The signal e_t guides the subsequent memory update. Its resemblance to the backpropagated error term in differentiable memory networks is only an informal analogy: e_t is a natural-language string, not a gradient.

3.2.3 Step 3: Memory Update

The final step is the memory update, where we transition from $h_{t-1} \rightarrow h_t$ to incorporate new information while satisfying the token budget λ. This operation is modeled not just as summarization, but as a feedback-guided rewrite.

Using a specific prompt template $\mathcal{P}_{mem}$, the model generates the new state conditioned on the previous state, current events, and the feedback signal e_t:

$$h_t \sim f_\theta(\cdot \mid \mathcal{P}_{mem}(h_{t-1}, x_t, \hat{y}_t, e_t)) \quad (6)$$

The prompt $\mathcal{P}_{mem}$ explicitly instructs the model to:

1. Compress x_t and $\hat{y}_t$ into the summary.
2. Apply the Critique: Use e_t to identify and rewrite incorrect beliefs in h_{t-1} rather than simply appending new tokens.

This formulation ensures the memory is self-healing: the state evolves to correct misconceptions based on the textual feedback e_t.

3.3 Algorithm

The complete inference procedure, integrating the semantic gradient loop and constraint enforcement, is detailed in Algorithm 1.

Algorithm 1 LLM-as-RNN Inference Process

Require: Sequence $\{x_t\}_{t=1}^T$, Frozen LLM θ, Evaluation Function g_{eval}, Max Memory λ

1: $h_0 \leftarrow \varnothing$
2: **for** $t = 1$ to T **do**
3: *// Phase 1: Contextualization*
4: $C_t \leftarrow \mathcal{I}_{sys} \oplus h_{t-1} \oplus x_t$
5: $\hat{y}_t \sim f_\theta(\cdot|\mathcal{P}_{gen}(C_t))$ {Generate Prediction}
6: *// Phase 2: Reflection*
7: Retrieve reference/criteria $\mathcal{R}_t$
8: $e_t \leftarrow g_{eval}(\hat{y}_t, \mathcal{R}_t)$ {Semantic Evaluation}
9: *// Phase 3: Memory Update*
10: Update $h_t \leftarrow f_\theta(\cdot|\mathcal{P}_{mem}(h_{t-1}, x_t, \hat{y}_t, e_t))$
11: **if** $|h_t| > \lambda$ **then**
12: $h_t \leftarrow \text{Compress}(h_t, \lambda)$
13: **end if**
14: **end for**
15: **return** $\{\hat{y}_t\}_{t=1}^T$

Line 12 of Algorithm 1 enforces the memory budget: whenever the generated state exceeds λ tokens, it is rewritten by an LLM compression prompt that retains critical information, removes redundancy, keeps the most recent updates, and preserves the state's section structure within the word budget. We emphasize that every quantity in this loop is text: C_t is a prompt assembled by concatenation, e_t is a natural-language critique, and h_t is a word-level summary. There are no vector hidden states, no computed gradients, and no parameter updates at any step. A worked step-by-step example showing how the memory state h_t evolves and corrects errors over timesteps is given in Appendix F.

4 Experiments

4.1 Datasets

We evaluate LLM-as-RNN on three sequential benchmarks spanning healthcare, meteorology, and finance: the MIMIC-IV dataset, the Weather dataset, and the S&P 500 with Financial News Headlines dataset. Following a unified protocol, we structure the weather and finance benchmarks as continuous temporal streams rather than independent samples. Additional dataset statistics and preprocessing details are provided in Appendix A.

MIMIC-IV. The MIMIC-IV dataset is a deidentified EHR dataset containing ICU admissions with structured clinical variables (Johnson et al., 2023). We construct longitudinal patient trajectories consisting of sequential clinical notes and lab events (Appendix C). The task is to predict the primary diagnosis for the next hospital visit. To avoid temporal leakage, each prediction at step t is constructed only from information available up to that point; future visits and post-discharge records are excluded, and the target diagnosis for the current step is withheld until evaluation.

Weather. The Weather dataset is a meteorological time series containing mixed modalities, including textual descriptors and continuous measurements (e.g., temperature, humidity, wind, pressure) (Muthukumar, 2023). We employ a *sequential sliding-window protocol*: at each timestep t, the observation x_t is restricted to a fixed 5-day window. However, the model predicts the condition for day t by conditioning on both this local window *and* the recurrent memory h_{t-1}.

S&P 500. The S&P 500 dataset aligns daily market closing prices with financial news headlines (Mahaptra, 2024). Under the same protocol,

the model receives a 5-day lookback of price and news as input x_t. The task is to forecast the closing price for day t, synthesizing quantitative signals with qualitative sentiment accumulated over the entire sequence.

4.2 Baselines

We compare LLM-as-RNN against five main baselines, with Self-Refine additionally evaluated in the feedback-aligned comparison. All methods share the same input/output formatting and evaluation protocol; they differ only in how they encode history, use feedback, or carry state across timesteps. More details are provided in Appendix B.

4.3 LLM Backbones

We evaluate LLM-as-RNN with multiple backbone LLMs to assess how backbone capability affects sequential prediction performance (Llama (Grattafiori et al., 2024), Gemma (Team et al., 2025), and GPT (Agarwal et al., 2025) families). To ensure fair comparisons across backbones, we use *temperature* $= 0.7$, *top-p* $= 0.9$, and *max_tokens* $= 4096$ for all LLM calls, and keep these settings unchanged across timesteps in the sequential stream. All LLM-based evaluations are computed using the same judge model, *Claude Sonnet 4.5* (Anthropic, 2025).

4.4 Metrics

- **MIMIC-IV (Semantic Accuracy):** We report LLM-Judged Accuracies (Acc@1 and Acc@5). The LLM-Judge determines if the generated diagnosis is semantically equivalent to the ground truth, avoiding the pitfalls of string matching.
- **Weather (Alignment):** The LLM judge evaluates whether the generated summary factually contradicts the ground truth on key variables, producing a binary success score.
- **S&P 500 (Forecasting Error):** We measure the deviation between predicted and actual closing prices using Mean Absolute Error (MAE) and Mean Squared Error (MSE).

5 Results

Our evaluation is guided by 4 research questions:

- **RQ1 (Efficacy):** Does the LLM-as-RNN framework outperform other strategies, and how do different LLM backbones behave?
- **RQ2 (Temporal Dynamics):** Does the model learn and reduce error over long sequences?
- **RQ3 (Autonomy):** Can the framework operate effectively using intrinsic self-correction (LLM-as-a-Judge) in the absence of ground truth labels?
- **RQ4 (Mechanism):** What are the contributions of feedback, persistent memory, and memory budget to the overall performance?

5.1 Overall Performance Analysis (RQ1)

We analyze whether outcome-driven memory updates can effectively overcome the shortcomings of static context accumulation. As shown in Table 1, LLM-as-RNN achieves the strongest overall performance across most backbones and tasks, supporting our core hypothesis: achieving robustness over long horizons requires a mutable belief state rather than a monotonically growing context buffer.

The advantage is largest in settings where the ground truth changes over time or conflicts with earlier evidence. On MIMIC-IV, where a patient's condition can change abruptly, LLM-as-RNN improves Acc@1 by 12.6% (absolute) compared to the best MemPrompt variant (0.6434 vs. 0.5175). Similarly, on S&P 500, where shifts in market regimes invalidate prior "trends," our method lowers MSE by roughly 6.6% (3.821 vs. 4.090). These results suggest that standard append-only methods (FHC and MemPrompt) are prone to error persistence: once an incorrect diagnosis or outdated sentiment is written into the context, it continues to bias future predictions. In contrast, LLM-as-RNN can actively "forget" or revise these obsolete patterns through its update mechanism.

A direct comparison between LLM-as-RNN and FHC underscores the inefficiency of raw context. Although FHC receives a complete, lossless history, it consistently underperforms our compressed-state approach (e.g., for Llama-3.1-70B, FHC Acc@1 is 0.4126 vs. 0.5804 for LLM-as-RNN). This indicates that the bottleneck in sequential prediction is not how much information is available, but how well that information is curated. FHC is vulnerable to attention dilution and noise, whereas LLM-as-RNN acts as an information filter that preserves only the signals relevant for the next prediction.

Although performance generally improves with model size, LLM-as-RNN disproportionately benefits smaller backbones. On the clinical prediction task, the Llama-3.2-3B model with LLM-as-RNN (Acc@1: 0.4545) surpasses the significantly larger

Table 1: **Overall performance.** Rows are LLM backbones and columns are methods. MIMIC-IV cells report Acc@1 / Acc@5 (↑); Weather cells report Align (↑); S&P 500 cells report MAE / MSE (↓).

Backbone	Zero-shot	FHC	MemPrompt	ReAct	Reflexion	LLM-as-RNN (Ours)
MIMIC-IV: Acc@1 / Acc@5 (↑)						
Llama-3.2-1B	0.1538 / 0.3846	0.2517 / 0.6434	0.3147 / 0.7203	0.2921 / 0.6874	0.3164 / 0.7162	0.3427 / 0.7552
Llama-3.2-3B	0.1958 / 0.4545	0.3357 / 0.7552	0.4126 / 0.7902	0.3883 / 0.7716	0.4211 / 0.8015	0.4545 / 0.7972
Llama-3.1-8B	0.2797 / 0.5594	0.3497 / 0.8392	0.4336 / 0.8462	0.4219 / 0.8310	0.4437 / 0.8480	0.5524 / 0.8531
Llama-3.1-70B	0.3287 / 0.6503	0.4126 / 0.8601	0.4615 / 0.9231	0.4568 / 0.9053	0.4748 / 0.9186	**0.5804 / 0.9510**
Gemma-3-1B	0.1608 / 0.3986	0.2587 / 0.6503	0.3217 / 0.7273	0.3012 / 0.7088	0.3296 / 0.7361	0.3497 / 0.7622
Gemma-3-4B	0.2238 / 0.5175	0.3566 / 0.8112	0.4196 / 0.8392	0.4084 / 0.8227	0.4302 / 0.8347	0.4685 / 0.8601
Gemma-3-12B	0.2937 / 0.6014	0.4056 / 0.8601	0.4755 / 0.8951	0.4661 / 0.8769	0.4924 / 0.8995	0.5594 / 0.9161
Gemma-3-27B	0.3427 / 0.6713	0.4476 / 0.9021	0.4965 / 0.9091	0.4873 / 0.8896	0.5069 / 0.9078	**0.6434 / 0.9301**
GPT-oss-20B	0.3357 / 0.6643	0.4126 / 0.8671	0.4895 / 0.9021	0.4948 / 0.9007	0.4867 / 0.9124	0.5874 / 0.9231
GPT-oss-120B	0.3636 / 0.7063	0.4406 / 0.8881	0.5175 / 0.9161	0.5086 / 0.9069	0.5213 / 0.9155	**0.6294 / 0.9441**
S&P 500: MAE / MSE (↓)						
Llama-3.2-1B	1.598 / 6.273	1.609 / 5.738	1.478 / 5.879	1.552 / 6.118	1.511 / 5.936	1.531 / 5.297
Llama-3.2-3B	1.585 / 6.080	1.578 / 5.503	1.426 / 5.148	1.474 / 5.409	1.443 / 5.287	1.347 / 5.025
Llama-3.1-8B	1.499 / 5.343	1.440 / 5.317	1.309 / 4.768	1.365 / 4.941	1.327 / 4.814	1.287 / 4.517
Llama-3.1-70B	1.493 / 5.232	1.464 / 5.211	1.292 / 4.417	1.323 / 4.684	1.296 / 4.489	1.211 / 4.313
Gemma-3-1B	1.612 / 6.347	1.603 / 6.042	1.491 / 6.118	1.533 / 6.029	1.516 / 6.052	1.533 / 5.541
Gemma-3-4B	1.548 / 5.982	1.511 / 5.541	1.392 / 5.012	1.418 / 5.166	1.391 / 5.035	1.321 / 4.754
Gemma-3-12B	1.438 / 5.214	1.412 / 5.067	1.298 / 4.716	1.337 / 4.844	1.309 / 4.704	1.237 / 4.406
Gemma-3-27B	1.402 / 4.972	1.386 / 4.881	1.264 / 4.090	1.281 / 4.351	1.258 / 4.196	**1.186 / 4.072**
GPT-oss-20B	1.305 / 4.590	1.280 / 4.420	1.230 / 4.120	1.247 / 4.268	1.236 / 4.242	**1.105 / 3.900**
GPT-oss-120B	1.312 / 4.612	1.290 / 4.440	1.198 / 4.162	1.232 / 4.397	1.216 / 4.281	**1.112 / 3.821**
Weather: Align (↑)						
Llama-3.2-1B	0.6783	0.7133	0.7203	0.6948	0.7187	0.7343
Llama-3.2-3B	0.6923	0.7133	0.7273	0.7241	0.7224	0.7413
Llama-3.1-8B	0.7063	0.7203	0.7343	0.7162	0.7466	0.7483
Llama-3.1-70B	0.7203	0.7273	0.7413	0.7469	0.7581	0.7622
Gemma-3-1B	0.6853	0.7133	0.7273	0.7078	0.7153	0.7413
Gemma-3-4B	0.7063	0.7273	0.7483	0.7395	0.7598	0.7622
Gemma-3-12B	0.7273	0.7483	0.7832	0.7710	0.7849	**0.8252**
Gemma-3-27B	0.7552	0.7832	0.8182	0.7894	0.8097	0.8182
GPT-oss-20B	0.7413	0.7902	0.8112	0.8060	0.8015	0.8112
GPT-oss-120B	0.7483	0.7972	**0.8322**	0.8131	0.8176	0.8182

Llama-3.1-70B with FHC (Acc@1: 0.4126). This suggests that the recurrent update loop acts as a strong inductive bias for sequential reasoning, enabling smaller models to approximate the long-horizon tracking capabilities typically associated with many more parameters.

On the Weather benchmark, while LLM-as-RNN outperforms baselines for most backbones (7/10), the gains are smaller, and MemPrompt achieves the single highest alignment score (0.8322 with GPT-oss-120B). We hypothesize that meteorological data, which is physically continuous and less semantic, derives limited benefit from "verbal correction" compared to semantic tasks such as diagnosis. In these high-entropy physical processes, an append-only memory scheme like MemPrompt may already be adequate, since weather trajectories seldom exhibit the sort of logical inconsistencies or conceptual errors that our textual update mechanism is designed to repair.

5.2 Temporal Dynamics and Learning (RQ2)

A key hypothesis of LLM-as-RNN is that the framework does not merely condition on history but *learns* from it through recurrence. To validate this, we analyze the model's performance as a function of the sequence length t.

Performance Gains Over Time. Figure 3 shows a clear improvement trend as LLM-as-RNN updates its recurrent memory state across timesteps. Importantly, at each timestep t we evaluate the model's prediction $\hat{y}_t$ against the corresponding ground-truth target y_t for that timestep. The reported curves reflect how performance evolves from the first prediction through later steps using feedback from previous steps.

On MIMIC-IV, both Acc@1 and Acc@5 increase substantially from early to late time steps, with the largest jump occurring after the first update and continued gains thereafter. On Weather, alignment rises rapidly in the first few steps and then plateaus, suggesting the state quickly captures the key short-horizon signals. On S&P 500, MAE and MSE decrease steadily but more modestly, indicating incremental calibration of numerical forecasts. Overall, these curves support the core hypothesis that feedback-driven state rewrites enable

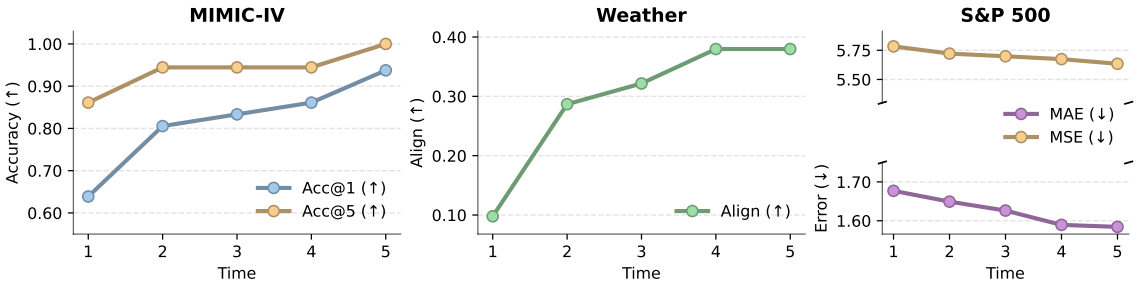


Figure 3: Temporal dynamics across iterative timesteps (t=1 . . . 5) for three datasets. As feedback-driven memory updates accumulate, performance improves.

Table 2: Performance comparison between supervised feedback (ground-truth) and self-supervised feedback (LLM-as-a-Judge). Results use Llama-3.1-8B.

Dataset	Metric	Ground Truth	LLM-Judge
MIMIC-IV	Acc@1 (↑)	0.5524	0.3077
	Acc@5 (↑)	0.8531	0.8042
Weather	Align (↑)	0.7483	0.7203
S&P 500	MAE (↓)	1.287	1.545
	MSE (↓)	4.517	5.533

online improvement under a fixed budget, rather than merely accumulating history.

Recovery from Error. We quantify recovery from error as an incorrect to correct transition between consecutive visits (Appendix Figure 5). Conditioned on being incorrect at time t, LLM-as-RNN corrects its prediction at t+1 in 54.8% of cases after incorporating the feedback signal e_t, while 45.2% of errors persist. Overall, these transition dynamics support a feedback-driven "self-healing" behavior enabled by memory updates, which is harder to obtain in static baselines that lack an explicit recurrent state update.

5.3 Robustness to Feedback Source (RQ3)

The standard configuration of LLM-as-RNN relies on ground-truth supervision to generate the feedback signal e_t. However, in many real-world deployment scenarios, immediate ground truth is unavailable. We evaluate the more common "Open-Ended" mode where the feedback e_t is generated by an LLM-Critic. This critic is grounded in a specific set of domain guidelines.

Table 2 shows the performance gap when replacing the ground truth $g_{eval}(\hat{y}_t, y_t)$ with a neural critic $g_{eval}(\hat{y}_t, \mathcal{R}_t)$. The label-free regime is consistently weaker than supervised feedback, which confirms that rewritable memory amplifies feedback quality: reliable feedback produces useful state corrections, while noisy critique can push the state toward drift. On MIMIC-IV, Acc@1 drops from 0.5524 to 0.3077, while Acc@5 remains more robust (0.8531 to 0.8042), suggesting that the LLM-Judge often preserves a plausible candidate set but provides a weaker ranking signal. Weather degrades mildly from 0.7483 to 0.7203, whereas S&P 500 is more sensitive, with MAE increasing from 1.287 to 1.545 and MSE from 4.517 to 5.533. These results position LLM-Judge feedback as a feasible but lower-reliability deployment mode rather than a replacement for observed labels when labels become available after each timestep.

5.4 Feedback and Memory Budget (RQ4)

Feedback-aligned baselines. To separate the value of feedback from the value of a rewritable recurrent state, we compare against Reflexion, ReAct, and Self-Refine on the matched MIMIC-IV subset used for feedback-aligned baselines. All four methods receive the identical feedback signal at each step; they differ only in how that feedback is consumed (accumulated verbal reflections, reasoning-action traces, within-step revision, or a bounded state rewrite). Table 3 reports Acc@5 with 95% bootstrap confidence intervals. Methods without feedback improve when they carry more history, while feedback-based methods improve further by using prior outcomes or critiques as adaptation signals. LLM-as-RNN obtains the strongest Acc@5 in this representative setting. We frame this as competitive rather than causal evidence: it is consistent with the added value of bounded state rewriting beyond feedback access, but the methods also differ in test-time compute, so the gap cannot be attributed to the memory mechanism alone.

Memory budget ablation. To isolate the effect of the context window budget available to the re-

Table 3: Feedback-aligned comparison on MIMIC-IV subset using Llama-3.1-8B. Intervals are 95% CI.

Method	Uses feedback	Acc@5(95% CI)
Zero-shot	No	0.7615(0.7267, 0.7946)
FHC	No	0.8610(0.8341, 0.8863)
MemPrompt	No	0.8626(0.8357, 0.8878)
Reflexion	Yes	0.8799(0.8546, 0.9052)
ReAct	Yes	0.8725(0.8441, 0.8976)
Self-Refine	Yes	0.8742(0.8473, 0.8993)
LLM-as-RNN	Yes	**0.8806**(0.8509, 0.8954)

Table 4: Effect of context window token budget λ on S&P 500 dataset using Llama-3.1-8B. Performance improves with increased context budget.

Context window λ	MAE (↓)	MSE (↓)
512	1.564	5.462
1024	1.499	5.292
2048	1.443	5.167
4096	1.287	4.517
8192	1.248	4.366

current state in LLM-as-RNN, we conducted a controlled ablation on the S&P 500 dataset. Specifically, we varied the maximum context window length λ for each LLM call across 512, 1024, 2048, 4096, and 8192, and report forecasting errors in Table 4. As λ increases, both MAE and MSE consistently decrease, indicating that allocating a larger context window to the recurrent state enables the model to retain more relevant historical signals and reduce prediction error. The gains are most pronounced when scaling from $\lambda = 512$ to $\lambda = 4096$ (MAE: $1.564 \rightarrow 1.287$, MSE: $5.462 \rightarrow 4.517$). Further increasing the budget to $\lambda = 8192$ yields only marginal additional improvement (MAE: $1.287 \rightarrow 1.248$, MSE: $4.517 \rightarrow 4.366$), suggesting diminishing returns at larger context windows. Overall, these results support the importance of sufficient context window budget for trajectory modeling while indicating that performance saturates once λ is large enough to preserve the time-series signals.

5.5 Token Cost Accounting

LLM-as-RNN performs 2.86 logical LLM calls per step on average, with a bounded carried text state (median 302 tokens, max 514 tokens). The contribution is bounded memory, not single-call efficiency. For completeness, we report step-level cost statistics in addition to task metrics. Table 5 reports the audited compute accounting (2.86 logical LLM calls per step; carried state median 302 tokens, max 514 tokens) under the representative Llama-3.1-8B configuration. LLM-as-RNN has the largest call count because each timestep includes prediction, reflection, and memory rewriting, but its input length remains bounded by the fixed memory budget λ rather than growing with the sequence length.

Table 5: Audited step-level compute accounting for LLM-as-RNN. Logical LLM calls per step average 2.86; the carried text state is bounded (median 302 tokens, max 514 tokens).

Metric	Value
Logical LLM calls per step	2.86
Carried state tokens (median)	302
Carried state tokens (max)	514

6 Conclusion

We presented LLM-as-RNN, an inference-time framework that makes a frozen LLM behave like a recurrent predictor by maintaining a natural-language memory state and rewriting it at each timestep using feedback on prior predictions, rather than accumulating an immutable context history. This revisable, structured state enables online error correction under a fixed token budget while producing transparent, human-readable learning traces. Across sequential benchmarks in healthcare, meteorology, and finance, our results indicate that feedback-driven state rewriting offers a simple, model-agnostic route to long-horizon adaptation without parameter updates.

7 Limitations

LLM-as-RNN requires multiple calls per timestep for prediction, reflection, and memory updating, increasing inference cost relative to single-pass inference and limiting latency-sensitive deployment. Update quality depends on the backbone's ability to diagnose errors, derive useful guidance, and compress information; smaller models may produce unstable or lossy states. Ground-truth feedback may be delayed or unavailable, while LLM critics can introduce noise and self-reinforcing memory drift. Performance also depends on prompt/schema design and memory-budget selection; automated prompt design and robust long-horizon adaptation remain open challenges.

Our MIMIC-IV cohort is selected by deterministic, lexicon-based filtering (τ=0.6, m=2, r=3), which retains each patient's largest temporally contiguous group of visits with consistent coarse top-

ics and excludes patients with fewer than three retained visits (Appendix C). Results on these coherent trajectories may not generalize to unfiltered, heterogeneous timelines.

All LLM-based evaluations use Claude Sonnet 4.5, and open-ended feedback also comes from an LLM critic. Scores may depend on the judge's handling of synonyms, clinical equivalence, and critique style; we do not measure variability across judge models.

8 Ethical Considerations

Because LLM-as-RNN accumulates state over time, incorrect predictions or memory updates can compound across timesteps, and the interpretability of the memory may create an illusion of reliability that increases automation bias; in clinical decision support, the system should be used strictly as an assistive tool with clinician oversight and rigorous evaluation across diverse subpopulations, and outputs should not be treated as medical advice. Similar concerns apply to financial forecasting: markets are influenced by exogenous factors and regime shifts, and sequential "learning from feedback" can overfit to noise or recent trends, so the framework should not serve as the sole basis for investment decisions. The memory mechanism also raises privacy and security concerns because the state may inadvertently retain sensitive or re-identifying details if not carefully controlled, and untrusted inputs or adversarial feedback could corrupt the memory and steer future predictions; mitigations include data-minimization and redaction for stored state, access controls, separating trusted feedback channels from user content, enforcing structured update schemas, monitoring for anomalous memory changes, and providing reset/rollback mechanisms.

Acknowledgments

This research was supported by the AI Makerspace (RRID:SCR_028058) and the Partnership for an Advanced Computing Environment (PACE) (RRID:SCR_027619) at the Georgia Institute of Technology. We also gratefully acknowledge funding and fellowships that contributed to this work, including a Wallace H. Coulter Distinguished Faculty Fellowship, a Petit Institute Faculty Fellowship, and research funding from Amazon and Microsoft Research awarded to Professor May D. Wang.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Ekin Akyürek and 1 others. 2022. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint*.

Anthropic. 2025. System card: Claude sonnet 4.5. Technical report, Anthropic.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. 2024. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*.

Benjamin G Ascoli and Jinho D Choi. 2025. Advancing conversational text-to-sql: Context strategies and model integration with large language models. *Future Internet*, 17(11):527.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, and 1 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can gpt learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019.

Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarathkrishna Swaminathan, Sihui Dai, Aurelie Lozano, Georgios Kollias, Vijil Chenthamarakshan, Jiri Navratil, and 1 others. 2024. Larimar: Large language models with episodic memory control. In *International Conference on Machine Learning*, pages 10109–10126. PMLR.

Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z Pan. 2025. Rethinking memory in ai: Taxonomy, operations, topics, and future directions. *arXiv preprint arXiv:2505.00675*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. 2022. What can transformers learn in-context? a case study of simple function classes. *Advances in neural information processing systems*, 35:30583–30598.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Tiago Ramalho, John Agapiou, and 1 others. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Yushan Jiang, Zijie Pan, Xikun Zhang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. 2024. Empowering time series analysis with large language models: a survey. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 8095–8103.

Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, and 1 others. 2023. Mimic-iv, a freely accessible electronic health record dataset. *Scientific data*, 10(1):1.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

LangChain. 2025. Langmem: Long-term memory for LLM agents. Software framework.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yuxing Lu, Gecheng Fu, Wei Wu, Xukai Zhao, Sin Yee Goi, and Jinzhuo Wang. 2025. Towards doctor-like reasoning: Medical RAG fusing knowledge with patient analogy through textual gradients. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 150336–150375. Curran Associates, Inc.

Yuxing Lu, Xukai Zhao, Wei Wu, and Jinzhuo Wang. 2026. Training the knowledge base through evidence distillation and write-back enrichment. *Preprint*, arXiv:2603.25737.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve gpt-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2833–2861.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Dyutidas Mahaptra. 2024. S&p 500 with financial news headlines (2008–2024). Kaggle Dataset. Accessed: 2026-01-02.

Matthew Muhoberac, Atharva Parikh, Nirvi Vakharia, Saniya Virani, Aco Radujevic, Savannah Wood, Meghav Verma, Dimitri Metaxotos, Jeyaraman Soundararajan, Thierry Masquelin, and 1 others. 2025. State and memory is all you need for robust and reliable ai agents. *arXiv preprint arXiv:2507.00081*.

J. Muthukumar. 2023. Weather dataset. Kaggle Dataset. Accessed: 2026-01-02.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. 2024. Transformers are multi-state rnns. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18724–18741.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. 2023. Memgpt: Towards llms as operating systems.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, and 13 others. 2023. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore. Association for Computational Linguistics.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1842–1850, New York, New York, USA. PMLR.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024a. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36:74530–74543.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. 2024b. MEMORYLLM: towards self-updatable large language models. In *Proceedings of the 41st International Conference on Machine Learning*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. 2024. Continual learning for large language models: A survey. *Preprint*, arXiv:2402.01364.

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. 2025. From human memory to ai memory: A survey on memory mechanisms in the era of llms. *arXiv preprint arXiv:2504.15965*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. 2024a. How language model hallucinations can snowball. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 59670–59684. PMLR.

Xiyuan Zhang, Ranak Roy Chowdhury, Rajesh K Gupta, and Jingbo Shang. 2024b. Large language models for time series: a survey. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 8335–8343.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.

Kai Zhou and 1 others. 2023. Recurrentgpt: Interactive generation and reasoning with recurrent language models. *arXiv preprint*.

A Dataset Details

MIMIC-IV (clinical EHR). MIMIC-IV [1] (Johnson et al., 2023) is a large, deidentified electronic health record (EHR) database of patients treated at the Beth Israel Deaconess Medical Center, covering both intensive care unit (ICU) admissions and emergency department (ED) visits. It contains structured clinical information such as demographics, diagnoses, procedures, laboratory measurements, and treatment/medication-related variables. In our experiments, we represent each patient as a chronologically ordered sequence of visits/admissions, and we apply deterministic visit filtering (Appendix C) to reduce clinically heterogeneous timelines.

Weather (meteorological time series). The Weather dataset [2] is a multivariate time series with 96,453 timestamped observations and 12 columns, mixing categorical descriptors and continuous meteorological variables. Typical fields include a timestamp (Formatted Date), textual descriptors (e.g., Daily Summary), precipitation type, and numeric measurements such as temperature and apparent temperature (°C), humidity, wind speed, wind bearing, visibility, and pressure.

S&P 500 with Financial News Headlines (financial time series). The S&P 500 dataset [3] couples historical S&P 500 closing prices with one or more daily financial news headlines. We align market records and headlines by trading date. This benchmark emphasizes non-stationarity and temporal dependence typical of financial markets, while also testing whether textual news can help guide sequential prediction and memory updates.

B Baselines

Zero-shot. This baseline (Figure 1a) predicts the target at time t using only the most recent observation (e.g., the last visit/day) without any additional history. It serves as a lower bound that tests the LLM's raw single-step predictive ability.

Full History Concatenation (FHC). FHC (Ascoli and Choi, 2025) (Figure 1b) conditions the LLM on the entire available history by directly concatenating all past observations into the prompt at each timestep:

$$C_t = x_1 \oplus x_2 \oplus \cdots \oplus x_{t-1} \oplus x_t. \tag{7}$$

FHC is the most straightforward strategy for leveraging temporal context and is commonly used as a long-context baseline, but its input length grows with t and can exceed the model's context window, necessitating truncation and potentially degrading performance.

MemPrompt. MemPrompt (Madaan et al., 2022) summarizes each past observation into a short textual memory unit and concatenates the accumulated summaries as a compact proxy for the full history:

$$\begin{aligned} m_i &= \text{Summarize}(x_i), \\ C_t &= m_1 \oplus m_2 \oplus \cdots \oplus m_{t-1} \oplus x_t. \end{aligned} \tag{8}$$

Unlike FHC, MemPrompt bounds history growth via per-step compression. However, the memory is append-only: previously written summaries are not revised in light of new evidence or prediction errors, so misconceptions can persist over time.

Reflexion. Reflexion (Shinn et al., 2023) is a feedback-aligned inference-time adaptation baseline. After each prediction, it receives the same feedback signal as LLM-as-RNN and writes a verbal reflection that is carried forward as episodic memory. This tests whether feedback plus persistent natural-language state is sufficient when the state is accumulated rather than rewritten under a fixed memory budget.

ReAct. ReAct (Yao et al., 2022) interleaves reasoning traces with feedback-conditioned actions. In our sequential setting, it receives the same visible history and feedback signal, then carries forward prior reasoning-action traces without rewriting a bounded state. This baseline tests whether explicit reasoning-action loops recover the gains of recurrent memory rewriting.

Self-Refine. Self-Refine (Madaan et al., 2023) performs iterative critique and revision within the current timestep. It receives feedback for the current prediction and uses it to refine the output, but it does not maintain a persistent recurrent state across future timesteps. This baseline isolates single-step self-critique from cross-step memory updating.

C Visit Filtering for MIMIC-IV

Patient timelines in MIMIC-IV may contain admissions that are clinically heterogeneous across time

[1] https://physionet.org/content/mimiciv/3.1/

[2] https://www.kaggle.com/datasets/muthuj7/weather-dataset

[3] https://www.kaggle.com/datasets/dyutidasmahaptra/s-and-p-500-with-financial-news-headlines-20082024

Table 6: Dataset summary.

Dataset	Domain	Timestep	Inputs (modalities)	Prediction target	Metric(s)
MIMIC-IV	Healthcare	Visit	Structured EHR notes (labs, diagnoses, treatments)	Clinical diagnosis	Acc@1, Acc@5 (↑)
Weather	Meteorology	Hourly	Numeric variables and categorical / text descriptors	Weather summary	Alignment Rate (↑)
S&P 500	Finance	Daily	News headlines text	Closing prices	MAE, MSE (↓)

Table 7: Dataset statistics.

Dataset	Size
MIMIC-IV	7,128 patients, 37,536 visits
Weather	4,019 days, 96,453 observations
S&P 500	3,507 trading days, 19,127 headlines

(e.g., unrelated comorbid events). We apply a deterministic, lexicon-driven filtering procedure that retains, for each patient, a temporally contiguous subsequence of visits whose inferred coarse topics are mutually consistent. The procedure does not learn from labels or train a model; it only prunes visits while preserving all original structured fields of the retained records.

C.1 Temporal Protocol and Information Parity

For every patient trajectory, visits are sorted chronologically before any prompt is constructed. At step t, the model observes the current visit snapshot and the known outcomes from visits $1, \ldots, t-1$ only. Outcomes from earlier visits are treated as historical facts, analogous to prior diagnoses in a chart, previous market closes, or observed past weather. The target label for step t, future visits, and post-discharge or otherwise future-only records are excluded from the prediction prompt and used only for evaluation after $\hat{y}_t$ is produced.

This protocol is applied uniformly across FHC, MemPrompt, Reflexion, ReAct, Self-Refine, and LLM-as-RNN. The baselines and our method therefore receive the same historical information; they differ in how the history is represented and updated. FHC appends the visible history, MemPrompt appends compressed summaries, Reflexion accumulates verbal reflections, ReAct carries reasoning-action traces, Self-Refine revises within the current step, and LLM-as-RNN rewrites a bounded memory state.

C.2 Inputs, Outputs, and Cohort Preselection

Inputs. The filtering consumes the full parsed MIMIC-IV dataset in JSON format, where each patient record contains a chronologically ordered list of visit records.

Cohort restriction. Only patients whose `valid_visits` fall within a fixed range are considered. In the reported configuration, we restrict to patients with 5–20 valid visits (inclusive).

Outputs. The procedure produces a filtered JSON dataset in which each retained patient record preserves all original fields but replaces the visit list with the selected subsequence.

C.3 Visit Text Construction

For each visit, we construct a single lowercased text string by concatenating multiple free-text sources:

- **Clinical sections:** all section values; if a section value is a list, all elements are included.
- **Notes:** if a structured notes field is present, the full note text is used when available.
- **Additional fields:** like `chief_complaint`, `allergies` and `service`.

This aggregated text is used only for topic matching; it does not alter the stored visit content.

We define a small set of coarse medical topics, each represented by a set of keywords. Topic evidence is computed by substring matches in the aggregated visit text. The full lexicon is shown in Table 8.

C.4 Per-Visit Topic Assignment

Let v denote a visit and t a topic with keyword set K_t. We compute a topic score as the number of keywords that appear in the visit text:

$$s_t(v) = \sum_{k \in K_t} \mathbf{1}\big[k \text{ in the aggregated text of } v\big] \tag{9}$$

Table 8: Coarse topic lexicon used for visit topic assignment. Abbreviations such as CHF, COPD, HbA1c, CKD, and AKI are matched as substrings.

Topic	Keywords
Cardiovascular	heart, cardiac, cardiovascular, hypertension, chf, myocardial, coronary, artery, atrial, ventricular, angina, infarction, fibrillation, blood pressure
Respiratory	lung, pulmonary, respiratory, pneumonia, copd, asthma, bronchitis, dyspnea, breathing, oxygen
Diabetes	diabetes, diabetic, glucose, insulin, hyperglycemia, hypoglycemia, hba1c, blood sugar
Renal	kidney, renal, nephro, dialysis, creatinine, ckd, aki, urinary, urine
Neurological	neuro, brain, stroke, seizure, dementia, alzheimer, parkinson, headache, migraine
Gastrointestinal	gastro, intestinal, liver, stomach, bowel, gi, hepatic, cirrhosis, abdominal
Oncology	cancer, tumor, malignancy, metastasis, oncology, chemotherapy, radiation, neoplasm
Infectious	infection, sepsis, bacterial, viral, antibiotic, fever, inflammatory

Topics with $s_t(v) = 0$ are ignored. The visit is assigned up to the top three topics by $s_t(v)$ (ties broken by the sorting order):

$$\text{Topics}(v) \;=\; \text{Top-3}\{t : s_t(v) > 0\}. \tag{10}$$

If no keywords match, then $\text{Topics}(v) = \emptyset$.

For two visits v_i and v_j, we compute Jaccard similarity over topic sets:

$$\text{Sim}(v_i, v_j) \;=\; \begin{cases} \dfrac{|\text{Topics}(v_i) \cap \text{Topics}(v_j)|}{|\text{Topics}(v_i) \cup \text{Topics}(v_j)|} \end{cases} \tag{11}$$

This definition ensures that visits with no matched topics do not spuriously increase coherence.

C.5 Consecutive Group Discovery

For each patient with visits $(v_1, \ldots, v_n)$ in chronological order, we partition the sequence into *consecutive* groups using a single left-to-right pass.

We maintain a current group G (initialized with the first visit). For each subsequent visit v_i, we compute its average similarity to the visits already in the current group:

$$\overline{\text{Sim}}(v_i, G) \;=\; \frac{1}{|G|} \sum_{v_j \in G} \text{Sim}(v_i, v_j). \tag{12}$$

If $\overline{\text{Sim}}(v_i, G) \geq \tau$, we append v_i to G; otherwise we close G and start a new group at v_i. Only groups of length at least m are kept as candidate coherent groups. If no candidate group exists (i.e., no consecutive segment reaches length m), we fall back to treating the entire visit sequence as a single group.

Given the candidate coherent groups for a patient, we select which visits to keep by retaining the largest group. The resulting filtered visit list is the chronologically ordered subsequence corresponding to that group.

Finally, we enforce a minimum number of retained visits: if a patient has fewer than r visits after filtering, the patient is removed from the filtered dataset. In our experiments, we use $\tau = 0.6$, $m = 2$ and $r = 3$.

For all retained patients, all original patient-level fields are preserved unchanged; only the visit list is replaced by the selected subsequence.

D Prompts for the MIMIC-IV Dataset

In this section, we provide the full prompt templates used for the MIMIC-IV clinical diagnosis task.

D.1 Method: Initialization and Generation

System Initialization

You are an expert clinical AI assistant analyzing longitudinal patient records.
Patient Information:
- Patient ID: `{patient_id}`
- Age: `{age}` years
- Gender: `{gender}`
- Primary Conditions: `{primary_conditions}`

Evolving Clinical Summary: `{evolving_summary}`
Your task is to provide accurate clinical predictions based on the current visit information and patient history.

Default Initialization Value (First Visit)

CLINICAL HISTORY: This is the patient's first visit. No previous medical history available in the system.
CURRENT STATUS & FOCUS AREAS:
- Establishing baseline assessment
- Gathering initial clinical presentation

KEY CONSIDERATIONS FOR FUTURE VISITS:
- Baseline health status to be established after first visit
- Future assessments will track disease progression

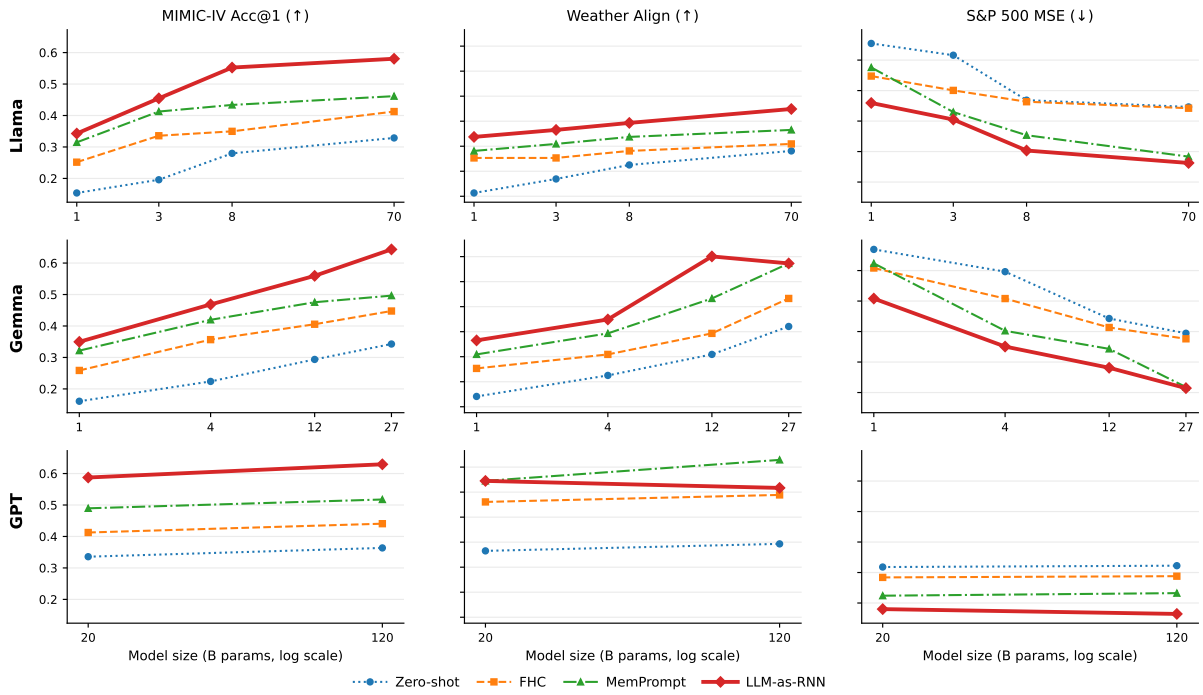


Figure 4: **Scaling across backbone families.** Performance vs. model size (B params, log-scale) for Zero-shot, FHC, MemPrompt, and LLM-as-RNN. Rows: Llama/Gemma/GPT backbones; columns: MIMIC Acc@1, Weather Align, S&P 500 MSE. LLM-as-RNN yields consistent gains across sizes and families.

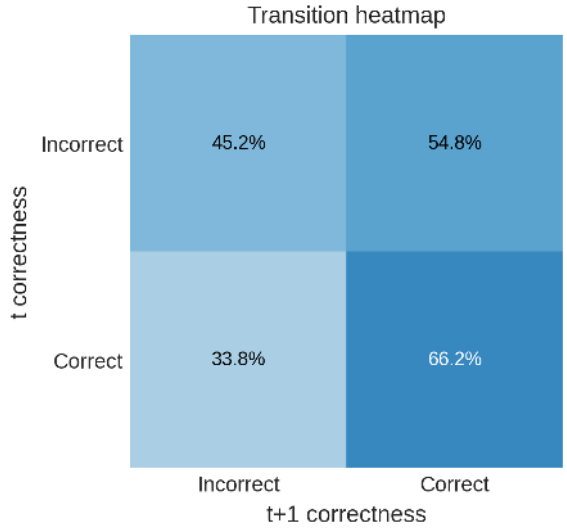


Figure 5: Transition heatmap of primary-diagnosis correctness between visits. Rows indicate correctness at time t and columns indicate correctness at time $t+1$.

Prediction Prompt ($\mathcal{P}_{gen}$)

You are a clinical AI assistant predicting discharge diagnoses for the current visit.
CURRENT VISIT SNAPSHOT:

- Chief Complaint: `{chief_complaint}`
- Vital Signs: `{vitals}`
- Lab Results: `{labs}`
- Medications on Admission: `{medications}`
- Recent Procedures: `{procedures}`
- History of Present Illness: `{history_present_illness}`
- Past Medical History: `{past_medical_history}`
- Social History: `{social_history}`
- Family History: `{family_history}`
- Physical Exam: `{physical_exam}`
- Pertinent Results: `{pertinent_results}`
- Hospital Course: `{hospital_course}`
- Allergies: `{allergies}`

TASK: Provide ONLY the discharge diagnosis predictions. Return JSON with top-5 diagnoses and the primary diagnosis:

```
{
  "top_5_diagnoses": [
    {"name": "diagnosis1"},
    {"name": "diagnosis2"},
    {"name": "diagnosis3"},
    {"name": "diagnosis4"},
    {"name": "diagnosis5"}
  ],
  "primary_diagnosis": {"name": "diagnosis1
    "}
}
```

D.2 Method: Reflection and Memory Update

Evaluation/Reflection Prompt (g_{eval})

You are evaluating predicted discharge diagnoses against ground truth to guide learning.
PREDICTED OUTPUT (JSON): `{prediction}`
GROUND TRUTH DIAGNOSES: `{ground_truth_diagnoses}`
Evaluate with SEMANTIC MATCHING (consider broader/narrower terms as correct if clinically equivalent).
Return JSON with ONLY these fields. Please respond in JSON format:

```
{
  "diagnosis_evaluation": {
    "primary_correct": true/false,
    "any_top5_correct": true/false,
    "missed_diagnoses": ["diagnosis1", "
    diagnosis2"],
    "why_missed": "brief reason for misses"
  },
  "improvement_suggestions": "one concise
     sentence on how to improve future
     predictions"
}
```

Memory Update Prompt ($\mathcal{P}_{mem}$)

You are updating a clinical AI system's memory to learn from prediction errors.
CURRENT EVOLVING SUMMARY: `{current_evolving_summary}`
PATIENT'S ACTUAL MEDICAL HISTORY (from all previous visits): `{patient_history}`
EVALUATION FEEDBACK (Prediction vs Ground Truth): `{evaluation_feedback}`
VISIT CONTEXT: `{visit_summary}`
CRITICAL INSTRUCTIONS:

- The evaluation feedback contains LLM JUDGE analysis with: what diagnoses were MISSED and WHY, what clinical signs were OVERLOOKED, and specific improvement suggestions.
- The PATIENT HISTORY shows the ACTUAL diagnoses and treatments across visits.
- Your task: Update the evolving summary to incorporate past learnings, current status, and future considerations.

PAY SPECIAL ATTENTION TO:

- "missed_diagnoses" - what we failed to predict
- "why_missed" - root cause of our errors
- "improvement_suggestions" - single concise sentence

Create a CONCISE evolving summary (MAX 200 words total) with three short sections:
1. **CLINICAL HISTORY** (2-3 sentences max): List key diagnoses from patient history with visit numbers. Note recurring conditions.
2. **FOCUS AREAS** (2-3 key points): What diagnoses were missed in this visit? Why were they missed?
3. **FUTURE CONSIDERATIONS** (1-2 key points): What to watch for based on patient's actual conditions.
CRITICAL: Keep it brief and focused. Stop after completing the JSON structure.
Provide the updated evolving summary as JSON:

```
{
  "evolving_summary": "unified summary with
     three sections: CLINICAL HISTORY,
    CURRENT STATUS & FOCUS AREAS, and KEY
    CONSIDERATIONS FOR FUTURE VISITS"
}
```

D.3 Baselines

Baseline: Zero-Shot

You are an expert clinical AI assistant. Analyze the following patient visit and provide:
1. Top-5 diagnoses (ranked by likelihood)
2. Primary diagnosis
Current Visit Information:

- Patient Demographics: Age `{age}`, Gender `{gender}`
- Chief Complaint: `{chief_complaint}`
- Vital Signs: `{vital_signs}`
- Lab Results: `{lab_results}`
- Current Medications: `{medications}`
- Recent Procedures: `{procedures}`

TASK: Provide ONLY the discharge diagnosis predictions. Return JSON with top-5 diagnoses and the primary diagnosis:

```
{
  "top_5_diagnoses": [
    {"name": "diagnosis1"},
    {"name": "diagnosis2"},
    {"name": "diagnosis3"},
    {"name": "diagnosis4"},
    {"name": "diagnosis5"}
  ],
  "primary_diagnosis": {"name": "diagnosis1
    "}
}
```

MemPrompt Summarization

The following system prompt has become too long and needs compression:
`{long_system_prompt}`
Task: Create a compressed version that:
1. Retains all critical clinical information
2. Removes redundancy
3. Keeps the most recent and relevant updates
4. Maintains the same structure (Focus Areas, Status, Considerations)
5. Stays under 250 words
Provide the compressed prompt maintaining the same JSON structure.

Baseline: FHC and MemPrompt

You are a clinical decision support system. You have access to a patient's complete visit history with known outcomes (diagnoses and treatments).
PATIENT VISIT HISTORY (with known outcomes): `{patient_history}`
CURRENT VISIT: `{current_visit}`

TASK: Based on the patient's complete history and current presentation, provide Top 5 most likely discharge diagnoses (ranked by likelihood, from most likely to least likely).
Consider:

- Patterns from previous visits and their outcomes
- Disease progression and comorbidities
- Current clinical presentation

TASK: Provide ONLY the discharge diagnosis predictions. Return JSON with top-5 diagnoses and the primary diagnosis:

```
{
  "top_5_diagnoses": [
    {"name": "diagnosis1"},
    {"name": "diagnosis2"},
    {"name": "diagnosis3"},
    {"name": "diagnosis4"},
    {"name": "diagnosis5"}
  ],
  "primary_diagnosis": {"name": "diagnosis1
    "}
}
```

E Prompts for Weather and S&P 500 datasets

The non-clinical benchmarks use the same three-operation template family as MIMIC-IV: generation ($\mathcal{P}_{gen}$), evaluation/reflection (g_{eval}), and memory update ($\mathcal{P}_{mem}$). The fields below are filled from each sequential stream.

E.1 Weather

Weather System Memory

You are an expert meteorological forecasting assistant tracking one weather sequence over time.
Evolving Weather Memory: `{evolving_summary}`
Maintain a bounded memory with:

- recent temperature, humidity, wind, visibility, and pressure trends;
- recurring mismatches between predicted and observed summaries;
- concise guidance for the next forecast.

Weather Prediction Prompt ($\mathcal{P}_{gen}$)

CURRENT 5-DAY WINDOW: `{weather_window}`
Each record may include timestamp, daily summary, precipitation type, temperature, apparent temperature, humidity, wind speed, wind bearing, visibility, and pressure.
TASK: Predict the weather condition summary for the next target day. Return JSON only:

```
{
  "weather_summary": "concise natural-
    language forecast",
  "key_conditions": ["condition1", "
    condition2"],
  "uncertainty": "brief uncertainty note"
}
```

Weather Evaluation and Memory Update

PREDICTION: `{prediction}`
OBSERVED TARGET DAY:
`{ground_truth_weather}`
Evaluate whether the predicted summary contradicts the observed condition, focusing on temperature direction, humidity, precipitation, wind, pressure, and visibility. Return JSON with `aligned`, `missed_signals`, `contradictions`, and `improvement_suggestions`.
Then update the evolving weather memory by rewriting stale or incorrect trend statements, adding only signals useful for future timesteps, and keeping the memory under the token budget λ.

E.2 S&P 500

S&P 500 System Memory

You are an expert financial time-series forecasting assistant tracking the S&P 500 over time.
Evolving Market Memory: `{evolving_summary}`
Maintain a bounded memory with:

- recent price direction, volatility, and reversal patterns;
- recurring news or sentiment signals associated with forecast errors;
- concise calibration rules for the next closing-price forecast.

S&P 500 Prediction Prompt ($\mathcal{P}_{gen}$)

CURRENT 5-DAY MARKET WINDOW:
`{market_window}`
Each record may include date, open/high/low/close prices, volume, and aligned financial news headlines.
TASK: Forecast the next target day's closing price. Return JSON only:

```
{
  "predicted_close": 0.0,
  "direction": "up/down/flat",
  "rationale": "one concise sentence"
}
```

S&P 500 Evaluation and Memory Update

PREDICTION: `{prediction}`
OBSERVED CLOSE: `{ground_truth_close}`
Compute whether the forecast direction was correct and summarize the error magnitude and likely missed signals. Return JSON with `absolute_error`, `squared_error`, `direction_correct`, `missed_signals`, and `improvement_suggestions`.
Then update the evolving market memory by rewriting incorrect trend or sentiment assumptions, retaining calibrated signals for future steps, and keeping the memory under the token budget λ.

F Qualitative Analysis: Memory Trace

To illustrate the recurrent inference mechanism, we present a step-by-step trace of an example patient. This example demonstrates how the *Memory State* (h_t) evolves to correct errors and accumulate clinical context over time.

Step 1: Initialization and Initial Feedback (V1)

The patient presents with Leukemia. The model predicts the primary condition correctly but misses secondary electrolyte abnormalities. The memory is updated to watch for these in the future.

Input (x_1) & Prediction ($\hat{y}_1$)

Chief Complaint: Leukemia
Prediction: Acute Myeloid Leukemia (AML) in remission.
Ground Truth: Leukemia, Aspergillosis, Pancytopenia, **Hypokalemia**.

Reflection/Feedback (s_1)

Error Analysis: Primary diagnosis correct. However, electrolyte abnormalities like **hypokalemia** were missed.
Guidance: Include common metabolic/electrolyte abnormalities in the differential for hematologic malignancy patients.

Memory Update ($h_0 \rightarrow h_1$)

Previous Memory: [Empty/Default Initialization]

Updated Memory (h_1):
Clinical History: Leukemia (V1), Aspergillosis (V1), Hypokalemia (V1).
Future Focus: **Watch for metabolic and electrolyte abnormalities** (specifically Potassium) in hematologic patients.

... Visits 2, 3, and 4 processed (Memory accumulates breast cancer, pneumonia) ...

Step 2: Error Correction via Recurrence (V5)

In Visit 5, the model falsely predicts "Remission" when the patient has "Relapsed". The reflection module catches this, and the memory explicitly encodes this correction to prevent future complacency.

Input (x_5) & Prediction ($\hat{y}_5$)

Chief Complaint: Admission for cycle 1 of Dacogen (chemotherapy).
Prediction: Leukemia (AML) **in remission**.
Ground Truth: Relapsed AML.

Reflection/Feedback (s_5)

Error Analysis: Primary diagnosis incorrectly identified AML as 'in remission' when patient actually had relapsed disease.
Guidance: Prioritize active disease states over remission status when clinical evidence (Dacogen) suggests relapse.

Memory Update ($h_4 \rightarrow h_5$)

Previous Focus: Watch for pulmonary complications, GVHD.

Updated Focus (h_5):
Missed Diagnosis Note: Primary diagnosis incorrectly identified AML as 'in remission'.
Improvement Strategy: **Prioritize active disease states** over remission status. Ensure most acute condition is selected.

Step 3: Long-Term Memory Retention (V8)

By the end of the sequence, the memory state (h_T) has become a comprehensive summary of the patient's complex trajectory, far exceeding the context window of a standard zero-shot prompt.

Final Memory State (h_8)

CLINICAL HISTORY:
- **Oncology:** Leukemia (V1, V8), Invasive Ductal Carcinoma (V2), Relapsed AML (V5), Multiple Myeloma (V6).
- **Infectious:** Aspergillosis (V1), Sepsis (V6), Neutropenic fever (V7).
- **Comorbidities:** Hypokalemia (V1), Acute Kidney Failure (V6), GVHD (V3), Orthostatic Hypotension (V4).

CURRENT STATUS & FOCUS AREAS:
- **Learned Patterns:** Patient has recurring Heart Failure (V1, V3, V5).
- **Correction History:** Watch for specific organisms in bacteremia (missed in V7); distinguish between active relapse vs remission.

KEY CONSIDERATIONS FOR FUTURE VISITS:
- Watch for pulmonary complications in hematologic malignancy.
- Monitor electrolyte levels (Potassium) carefully.
- Distinguish between Leukemia vs Myeloma presentation.

G Error Analysis

G.1 Persistent prediction errors.

Among predictions that are incorrect at time t, 45.2% remain incorrect at $t+1$ even after memory rewriting. These persistent cases are concentrated in two settings. First, rare or atypical diagnoses may appear too sparsely in the trajectory for the bounded memory to form a stable corrective rule. Second, early timesteps provide limited prior context, so the memory has not yet accumulated

enough evidence to distinguish transient findings from durable patient-level patterns.

G.2 Non-parsable outputs.

With smaller backbones or higher sampling randomness, generations more often violate strict JSON-only constraints (e.g., emitting extra natural-language text, markdown code fences, trailing commas, or missing braces). These formatting failures break automatic parsing and can halt the recurrent pipeline, since both the evaluator and the memory update step depend on structured fields to propagate feedback across timesteps.

G.3 Overlong generations causing truncation.

Prediction, critique, or memory-update outputs can exceed `max_tokens` or the memory budget λ, leading to truncation. This is particularly damaging when truncation cuts off JSON closures (making outputs non-parsable) or removes critical supervision signals such as `missed_diagnoses` and `why_missed`. In a recurrent setting, losing these fields not only degrades the current timestep but also weakens the next memory update, compounding error over long horizons.

G.4 Noisy or biased feedback.

When ground truth labels are unavailable and feedback is generated by an LLM judge, the critique can be noisy, inconsistent, or biased (e.g., over-penalizing acceptable synonyms, missing clinically equivalent diagnoses, or providing spurious rationales). Because the memory update treats this feedback as a "semantic gradient," systematic judge errors can cause the state to internalize incorrect lessons, inducing memory drift and potentially reinforcing mistakes across subsequent timesteps.