Integrating Temporal Representations for Dynamic Memory Retrieval and Management in Large Language Models

Yuki Hou[1], **Haruki Tamoto**[2], **Homei Miyashita**[1]
[1]Meiji University
[2]Kyoto University
houhoutime@gmail.com, harukiririwiru@gmail.com, homei@homei.com

Abstract

Conventional dialogue agents often struggle with effective memory recall, leading to redundant retrieval and inadequate management of unique user associations. To address this, we propose SynapticRAG, a novel approach integrating synaptic dynamics into Retrieval-Augmented Generation (RAG). SynapticRAG integrates temporal representations into memory vectors, mimicking biological synapses by differentiating events based on occurrence times and dynamically updating memory significance. This model employs temporal scoring for memory connections and a synaptic-inspired propagation control mechanism. Experiments across English, Japanese, and Chinese datasets demonstrate SynapticRAG's superiority over existing methods, including traditional RAG, with up to 14.66% improvement in memory retrieval accuracy. Our approach advances context-aware dialogue AI systems by enhancing long-term context maintenance and specific information extraction from conversations.

Introduction

Recent advancements in transformer-based large language models (LLMs), such as GPT-4 (OpenAI 2023) and Claude 3 Opus (Anthropic 2024a), have revolutionized dialogue agents, allowing them to mimic human cognitive processes. However, these models still face challenges in temporal dynamics for memory retrieval and consolidation, particularly with homogeneous recall of event-related memories, where top recall candidates often exhibit redundancy or similarity due to encoding issues.

To address this limitation, we propose a memory architecture that incorporates temporal representations into memory vectors, spatially reflecting temporal distances and enhancing the diversity of recalled memories. This method improves memory retrieval granularity and ensures a broader spectrum of contextual relevance in interactions. Research suggests that memory retrieval depends more on retrieval triggers than memory strength (McDaniel and Einstein 2000; Tulving 1985). Strong synaptic connections between memory nodes facilitate the simultaneous recall of multiple related memories (Hebb 1949; Neves, Cooke, and Bliss 2008). Our approach dynamically updates these connections using dynamic time warping (DTW) to calculate cumulative distance matrices between stimulated time arrays of memory nodes (Sakoe and Chiba 1978; M"uller 2007).

The proposed model includes a propagation control mechanism to prevent excessive stimulus spread across the network, ensuring efficient recall by limiting stimuli to child nodes based on cosine similarity thresholds and using a leaky integrate-and-fire (LIF) model for node activation (Brunel and van Rossum 2007; Gerstner et al. 2014). Unlike retrieval-augmented generation (RAG) (Lewis et al. 2020), our model's dynamic integration of temporal representations enables a more nuanced handling of temporal and contextual dynamics.

This framework addresses the challenges of temporal memory retrieval in dialogue agents, paving the way for more sophisticated context-aware dialogue AI agent systems. Moreover, it enhances the agents' understanding of user interactions and their ability to engage in meaningful dialogues by simulating human-like memory processes. We hope this study advances human-computer interaction, fostering a future where technology is closely aligned with human needs and mirrors our cognitive behaviors and life experiences.

Related Work

Memory Retrieval in AI Systems

A comprehensive understanding of human memory processes provides a basis for comparing and developing AI memory systems (Murdock 1982). Recent advancements include systems using external databases for memory augmentation in LLMs (Zhong et al. 2023), enhancing the context-awareness of AI responses. More comprehensive approaches, such as generative agents stimulating credible human behavior (Park et al. 2023), extend LLMs to store, organize, and retrieve agent's experiences, enabling realistic behaviors in interactive environments.

Memory architectures have been developed to enhance the cognitive abilities of dialogue agents based on LLMs (Hou, Tamoto, and Miyashita 2024). These systems autonomously recall relevant memories for response generation, managing memory significance in a temporal context similar to human memory recall. RAG combines retrieval and generation-based methods to improve response

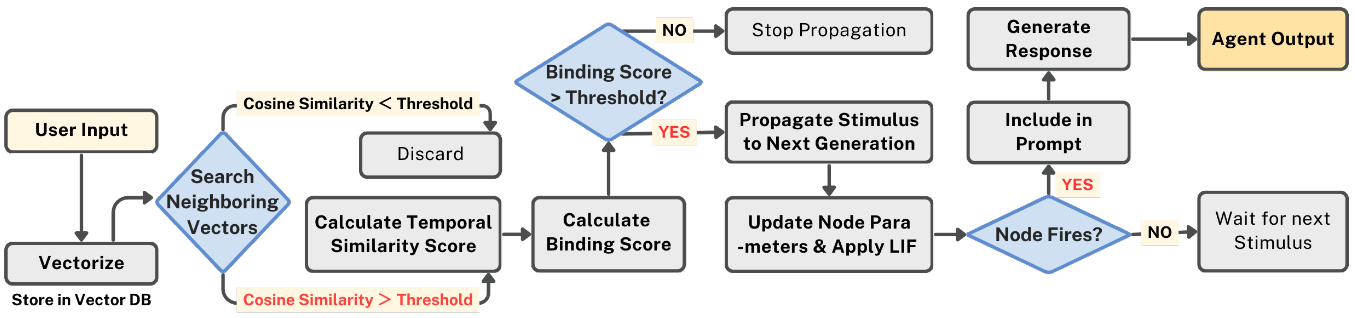

User Input
Vectorize
Store in Vector DB
Search Neighboring Vectors
Cosine Similarity < Threshold
Discard
Cosine Similarity > Threshold
Calculate Temporal Similarity Score
Calculate Binding Score
Binding Score > Threshold?
NO
Stop Propagation
YES
Propagate Stimulus to Next Generation
Update Node Para -meters & Apply LIF
Node Fires?
YES
Include in Prompt
Generate Response
Agent Output
NO
Wait for next Stimulus

Figure 1: Overview of the SynapticRAG model.

relevance and accuracy (Lewis et al. 2020; Borgeaud et al. 2022). However, these systems struggle to address memories with low cosine similarity in the vector space, which might have unique associations for users, making it difficult to dynamically update their relationships.

Memory Models Inspired by Biological Synaptic Plasticity

Recent advancements in spiking neural networks (SNNs), inspired by synaptic plasticity in biological neural networks (Citri and Malenka 2008), have enhanced AI adaptability and learning.

Notable examples include differentiable plasticity models for neural networks (Miconi, Clune, and Stanley 2018) and models incorporating multiple timescales of synaptic plasticity (Benna and Fusi 2015), improving performance in continual learning and long-term memory formation tasks.

These models face challenges related to scalability, biological plausibility, and efficient retrieval. Addressing these limitations is crucial for developing advanced, human-like memory mechanisms in AI systems.

While studies on the dynamics of neural networks, including Spiking SNNs (Maass 1997; Tavanaei et al. 2019), aim to replicate memory microscopically, this study treats text as memory to reproduce human memory processes via neuroscientific perspectives microscopically. This approach reduces computational costs with a minimal network compared to microscopic models and allows for seamless integration of recalled memories into existing LLM-based agents.

SynapticRAG: A Memory Retrieval Model for Enhanced Temporal Awareness

SynapticRAG integrates temporal representations into node vectors, which traditionally contain only spatial semantics, and formalizes the semantic and temporal relationships of each memory node. Vectorized memories are stored as nodes, each with a weighted spike train. The stimulus value each node receives is stored as a two-dimensional array along with the recorded time. The input vector is treated as the original node, and the stimulus propagates to neighboring nodes in the vector space. When a node accumulates enough stimulus, it fires using an integrate-and-fire model. If a node fires, the memory it represents is recalled and provided to the agent as information. Figure 1 shows the overall process flow of the SynapticRAG model.

Stimulus Propagation Scoring

We define the propagation of stimuli from parent nodes to child nodes, considering nodes X, Y, and Z, where the stimulus propagates from X to Y to Z.

For propagation from X to Y, X is the input vector and Y is a neighboring vector. Nodes with a cosine similarity below the threshold with X are discarded, and propagation occurs on the remaining nodes, including Y. The spike train time arrays for X and Y are extracted, and a cumulative distance matrix is calculated using the DTW method to determine the optimal path. This distance matrix's elements d_i are then processed using the following function.

$$f(x) = \exp\left(-\frac{x}{\tau}\right) \tag{1}$$

Here, τ represents the average strength of the connected memory nodes, indicating the characteristic time scale over which memory connections decay. A larger τ signifies stronger, more persistent memories, retaining information longer, while a smaller τ implies faster decay and transient connections, leading to quicker information loss. This parameter allows the model to express memory retention and forgetting at different time scales, emulating the variability in human memory processes.

As shown in the left part of Figure 2, the stimulus propagation scoring process involves multiple layers, activating memories based on their cosine similarity to the user input vector. The function $f(d_i)$ represents the updated connection weight over time, assigning higher scores to closer events and lower scores to more distant ones. The exponential decay ensures the score diminishes as temporal distance increases. This function, chosen for its biological plausibility, mathematical tractability, and alignment with observed memory phenomena, emulates human forgetting curves, simplifies calculations, and provides a non-linear decay model with rapid initial forgetting followed by slower long-term decay.

The scores for each element calculated with $f(d_i)$ are summed and normalized to a range of 0 to 1. A monotonically increasing function is used for this normalization process to map the scores:

Table 1: Parameters used in the SynapticRAG model

Parameters		Details
τ	Time constant	Represents the average strength of connected memory nodes characterizing the time scale of memory connection decay.
T_{score}	Temporal similarity score	Indicates the similarity of stimulation times between two nodes.
B_{score}	Binding score	Combines temporal T_{score} and cosine similarities between two nodes.
$Stim$	Stimulus	Product of the binding score and stimulus from the previous generation.
$\boldsymbol{S}_t$	Spike train	Two-dimensional array of received stimulus values and their corresponding times.
I	Input current	Continuous representation of the spike train obtained using Dirac delta function.
V	Membrane potential	Evolves responding to the input current used to determine firing.
V_{th}	Firing threshold	A Constant used to determine if V has reached the firing point.

$$T_{\text{score}} = \frac{1}{1+\exp(-\sum f(d_i))} \tag{2}$$

$$B_{\text{score}} = T_{\text{score}} \cdot \left(\frac{X \cdot Y}{|X| \cdot |Y|}\right) \tag{3}$$

where T_{score} represents the temporal similarity score, and B_{score} is the binding score. The elements d_i are obtained from the cumulative distance matrix, while X and Y are the vector representations of nodes. $\sum f(d_i)$ sums the scores across the distance matrix, strengthening connections between temporally close events. The term $\frac{X \cdot Y}{|X| \cdot |Y|}$ represents the cosine similarity between vectors, assessing semantic similarity.

B_{score} represents the connection strength between nodes, considering both temporal similarity (T_{score}) and semantic similarity (cosine similarity). The [0,1] normalization prevents score divergence due to increasing time sequences, which ensures stable evaluation.

The binding score $B_{\text{score}}(X,Y)$ between X and Y determines the stimulus given to Y as $1 \cdot B_{\text{score}}(X,Y)$. Similarly, the binding score $B_{\text{score}}(Y,Z)$ between Y and Z determines the stimulus given to Z as $B_{\text{score}}(X,Y) \cdot B_{\text{score}}(Y,Z)$. Since $0 < B_{\text{score}} < 1$, the stimulus for child generation is always smaller than for parent generation, representing natural attenuation. The propagation continues until the stimulus value falls below a threshold, at some point where propagation is terminated, ensuring only highly relevant information is efficiently propagated through the network.

Dynamic Leaky Integrate-and-Fire Model

The parameters discussed in this section are inherent values stored in each node. Although the definitions of spike trains and time constants match those in the previous chapter, the parameter values here are distinct.

In the dynamic LIF model, the weighted spike train is received, and firing determination and parameter updates are performed. The time constant increases each time a node receives a stimulus. Given the time interval Δt between a successive stimuli, the time constant τ is updated by the following equation:

$$\tau(t+\Delta t) = \tau(t) + \frac{1-\exp(-\Delta t)}{1+\exp(-\Delta t)} \tag{4}$$

This update equation for τ captures the dynamic nature of synaptic plasticity. The term $(1-\exp(-\Delta t))/(1+\exp(-\Delta t))$ is a sigmoidal function ranging from 0 to 1. For small Δt (frequent stimuli), this term approaches 0, causing minimal changes to τ. For large Δt (infrequent stimuli), it approaches 1, significantly increasing τ.

The sigmoidal nature ensures bounded growth of τ, aligning with biological constraints. It is more sensitive to changes in Δt when stimuli are infrequent, emulating biological synapses. The update is always positive and bounded, ensuring stable long-term behavior.

Discrete spike trains $\boldsymbol{S}_t$ are converted into a differentiable form using Dirac's delta function $\delta(t)$ and the set of firing times Γ. The input current $I(t)$ is then calculated as follows:

$$\tau\frac{dI}{dt} = -I(t) + \sum_{t_f \in \Gamma} \boldsymbol{S}_t \delta(t - t_f) \tag{5}$$

Subsequently, the time evolution of the membrane potential $V(t)$ is expressed using the following differential equation:

$$\tau\frac{dV}{dt} = -V(t) + I(t) \tag{6}$$

This equation shows that the membrane potential changes in response to the input current. τ is the time constant of the membrane potential, determining the rate. The right side of Figure 2 illustrates the membrane potential changes and firing mechanism of our dynamic LIF model, demonstrating how the input current affects the membrane potential over time, leading to neuron firing when the threshold is exceeded.

If the calculated membrane potential $V(t)$ exceeds a predetermined firing threshold, the neuron fires, and the membrane potential is reset for the next input. However, the time constant is not reset, maintaining the node's strength. The entire sequence of stimuli received by the node is evaluated, but $V(t)$ is reset immediately after a stimulus to prevent premature firing if the threshold is too high or the initial value of $V(t)$ is too low. The hyperparameters are tuned on a small subset of tasks to address this issue.

Implementation

Figure 3 illustrates a conversation example demonstrating the system process. The user's input is vectorized using the

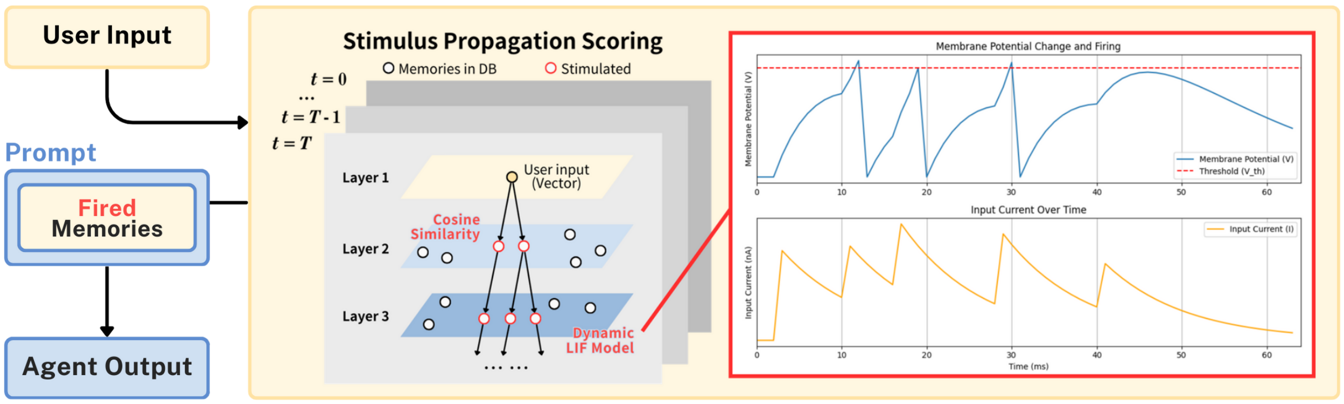
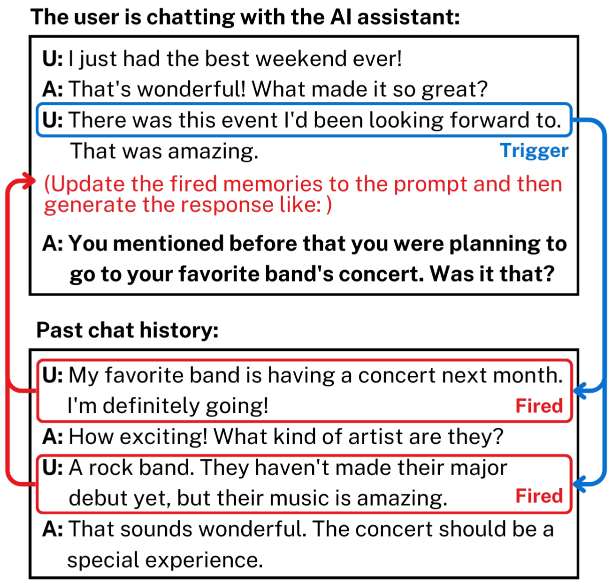


Figure 2: Overview of how memory nodes receive stimuli, fire, and provide information to the agent. **(Left)** Memory nodes represent vectorized user inputs and accumulated memories. In Layer 1, nodes have weighted spike trains recording stimulus values over time, with the input vector as the starting point for stimulus propagation. In Layer 2, nodes with cosine similarity below the threshold are excluded as stimuli propagate from parent node X to child node Y. In Layer 3, cumulative distance matrices are computed using spike train time sequences between remaining nodes, finding the optimal path. This process assigns high scores to temporally close events and low scores to distant events. **(Right)** Layer 3 details the firing of nodes when receiving sufficient stimuli. An integrate-and-fire model determines if a node fire and memories of fired nodes are provided to the agent's prompt for response generation.


The user is chatting with the AI assistant:

U: I just had the best weekend ever!
A: That's wonderful! What made it so great?
U: There was this event I'd been looking forward to. That was amazing. Trigger
(Update the fired memories to the prompt and then generate the response like:)
A: You mentioned before that you were planning to go to your favorite band's concert. Was it that?

Past chat history:

U: My favorite band is having a concert next month. I'm definitely going! Fired
A: How exciting! What kind of artist are they?
U: A rock band. They haven't made their major debut yet, but their music is amazing. Fired
A: That sounds wonderful. The concert should be a special experience.

Figure 3: SynapticRAG's memory retrieval process.

embedding model text-embedding-3-large (OpenAI 2024) and stored in a vector database with the Faiss library (Douze et al. 2024) for disk storage. Each node stores information such as id, vector, message, fire, v, i, tau, and spike, and is managed by SQL (MySQL 2001). The agent's response in the SynapticRAG model is generated using GPT-4 (OpenAI 2023).

Memory Management in Vector Database

The vectorized input is used to search for neighboring vectors, discarding those with a cosine similarity below the threshold. The remaining neighboring vectors receive a stimulus, the product of their temporal similarity score and cosine similarity. Only vectors exceeding the score threshold propagate to the next generation. Fired nodes do not propagate further. The child generation's stimulus is the product of the score and the parent generation's stimulus, always smaller owing to normalization to [0, 1]. Propagation repeats until the stimulus falls below a threshold, ending the process. The dynamic LIF model determines if a stimulated node fires using the accumulated stimulus; further, fired nodes are included in the prompt.

The SynapticRAG model is fundamentally a memory management approach designed for efficient memory retrieval, incorporating biological human memory characteristics into the memory retrieval stored in an external database. Although our implementation utilizes the GPT-4 model, users can employ any suitable LLM.

Prompting For Response Generation

An effective prompt for an LLM agent using SynapticRAG should include instructions for comprehensive context understanding, guiding the agent to recognize the relevance between past conversation history and the current dialogue. It should also provide guidance on efficiently leveraging related memories retrieved, including analyzing relationships among multiple relevant memories, and emphasize the importance of recognizing temporal context. An example of a prompt structure for a dialogue agent is given below:

"Your task is to provide responses that align with {username}'s daily life context. The current time is {current_time}. Recall the event {fired_memories} from {event_happened_time}. Please respond in a way that clearly demonstrates your recollection of past conversations."

Experiment

Dataset

To evaluate the proposed method, we used two datasets: a custom-created synaptic memory retrieval conversations (SMRCs) dataset and the PerLTQA dataset (Du et al. 2024). These datasets were chosen for their chronological dialogue format, context accumulation, and explicit correct answers. The **SMRCs** dataset, created using Claude 3.5 Sonnet (Anthropic 2024b), contains 101 conversation sets, each with a past and present conversation. It includes 206 memory recall triggers and 456 related memory tasks across diverse themes, testing the model's ability to maintain long-term context and recall multiple relevant pieces of information. The dataset is available in English and Japanese.

The **PerLTQA** dataset comprises 625 conversation sets, encompassing 2,665 QA tasks. Each conversation contains multiple QA pairs, with one correct label per question. It is available in English and Chinese. We filtered this dataset to exclude dialogues without QA pairs and instances where answers had low relevance to the conversation (cosine similarity < 0.5), ensuring quality and relevance.

Using these datasets allows for the comprehensive evaluation of our model's ability to maintain complex, long-term contexts (SMRCs) and extract specific information from conversations (PerLTQA). The availability in multiple languages supports a robust cross-lingual of the model's capabilities. These task counts are sufficient to obtain statistically significant results, and by utilizing all tasks in each dataset, we comprehensively evaluate the performance of the model.

Baseline Models

For comparison, we utilize three established models: the standard RAG model (Lewis et al. 2020), the MemoryBank model (Zhong et al. 2023) for long-term memory integration, and the MyAgent model (Hou, Tamoto, and Miyashita 2024) proposed for human-like memory recall and consolidation. **RAG** implements conventional vector similarity search for information retrieval. **MemoryBank** integrates long-term information retention and retrieval. **MyAgent** mimics human memory processes in dialogue systems.

We also prepared extended versions of the MemoryBank and MyAgent models with more optimizable hyperparameters. This extension allows for better performance through additional tuning, enabling a fairer comprehensive comparison. These models benchmark the SynapticRAG model, assessing its information retrieval and generation, long-term memory management, and dynamic memory recall.

Extension Methods of Baseline Models To optimize the baseline models for our datasets, we introduced additional hyperparameters to each model.

Specifically, we introduced multiplicative constants to key variables, including temporal factors, cosine similarities, and model-specific parameters. This fine-tuning process enhances each model's performance, ensuring a fair comparison with the SynapticRAG model.

For the MyAgent and MyAgent(Extended) models, the original implementation first filters memories based on cosine similarity and then calculates the recall probability p. In our evaluation, we aimed to retrieve the same number of memories as SynapticRAG from those exceeding the cosine similarity threshold, ranked by p.

This approach presented a challenge: if fewer memories than the recall count of SynapticRAG exceeded the cosine similarity threshold, MyAgent models would retrieve fewer memories, resulting in an unfair comparison. Alternatively, ignoring the cosine similarity filter and ranking solely by p could prioritize memories with high p but low cosine similarity, deviating from the intended behavior of the original model. To resolve this issue, we adjusted the p scores by adding 1 for memories exceeding the cosine similarity threshold (where $0 < p < 1$). The final ranking was based on these adjusted p scores, allowing us to replicate the original process of cosine similarity filtering followed by p-based recall. This method ensures fairness in evaluation by maintaining consistent recall counts across models while preserving the integrity of the original model's design within our comparative framework.

This extension ensures a fair comparison between MyAgent models and SynapticRAG, preserving the essence of the original implementation while meeting our evaluation criteria. Similar adjustments were made for other baseline models, tailored to their specific characteristics and variables.

Setup

Experiments were conducted on a macOS 14.2.1 system equipped with an Apple M1 processor and 16 GB of RAM. We used Python 3.10.9 (Python 2022), OpenAI's text-embedding-3-large for embeddings, Optuna 3.6.1 (Optuna 2024) for hyperparameter optimization, Faiss 1.8.0 for vector operations, and NumPy 1.26.4 (Numpy 2024) for numerical computations. We vectorized JSON format to transform the text into a numerical format for analysis. The *related_memory* fields were matched with corresponding document indices, serving as the correct labels and establishing the ground truth for our memory recall tasks.

Trigger documents were augmented with metadata, attaching correct labels to each trigger to evaluate the model's performance in identifying and retrieving relevant memories. We structured data storage using SQL for efficient data management. Each row of the database contains the storing section number, document index, text, correct label, and vector. Identical parameters were applied to both English and Japanese datasets to ensure fairness.

Experimental Methods

The evaluation of the SynapticRAG model focused on its ability to accurately retrieve relevant memories. We incremented the correct answer count by 1 whenever a retrieved memory matched a correct label and calculated accuracy as the ratio of correct retrievals to total correct labels.

Given the variable number of memories retrieved by SynapticRAG, we used two distinct evaluation metrics to ensure a fair comparison with the baseline models:

1. Equal Retrieval Count (ERC) This evaluation metric ensures that baseline models retrieve the same number of memories as SynapticRAG for each query. For example, if SynapticRAG retrieves four memories, baseline models will also retrieve their top four scoring memories. The RAG model uses cosine similarity for scoring, while other baseline models, which typically retrieve only the highest-scoring memories, have been modified to retrieve multiple memories for their scores to match SynapticRAG's retrieval count.

While this approach ensures an equitable comparison, it has limitations. For instance, if SynapticRAG retrieves no memories for a query, baseline models are also restricted from retrieving any memories. This constraint may potentially limit baseline models from demonstrating their full capabilities. To address this, we introduced a second evaluation metric.

2. Equal Retrieval Count with Minimum Guarantee (ERC-MG) This metric addresses ERC's limitations. When SynapticRAG retrieves fewer memories than the number of correct labels, baseline models can retrieve up to the number of correct labels. If SynapticRAG retrieves more, baseline models match its count. While this slightly disadvantages SynapticRAG, it allows baseline models to fully demonstrate their capabilities.

Hyperparameter Optimization

To optimize our model's hyperparameters, we used Optuna (Akiba et al. 2019), ensuring that the evaluation metric was independent of comparative models. This strategy prevents trivial optimization scenarios where one model's performance is defined relative to that of another. In fact, the optimization and evaluation conditions were distinct. When multiple trials achieved the same best score, we selected the parameter set with the highest ERC evaluation score, ensuring robust performance in both optimization and primary evaluation scenarios.

The number of hyperparameters varied across models: MemoryBank (0), MemoryBank(Ext) (5), MyAgent (1), MyAgent(Ext) (4), and SynapticRAG (9), reflecting the complexity and design choices of each model.

Optimization Strategy for Extended Models For extended models, which cannot predetermine the number of retrieved memories during evaluation, we optimized to maximize the scores of correct label memories. The process involved normalizing each memory score s_i to the range [0, 1] using the sum of all memory scores in the n-th task, defined as $Sum_n = \sum_{i=1}^{M_n} s_i$, where M_n is the number of memories in the n-th task. We then summed the normalized scores of correct label memories, expressed as $Score_n = \sum_{j \in y_n} s_j / Sum_n$, where y_n denotes the set of indices for correct label memories. The final score across all N tasks was calculated as $Score = \sum_{n=1}^{N} Score_n$. The hyperparameters were optimized to maximize this Score.

Optimization Approach for SynapticRAG For the SynapticRAG model, which uses the score v only for threshold judgment, we optimized based on conditions related to fired memories rather than directly using v. Two conditions were imposed. The first condition penalized insufficient correct label memories, defined as $A_n = \frac{|y_n| - |f_n \cap y_n|}{|y_n|}$ if $|y_n| > |f_n \cap y_n|$, and otherwise, where f_n is the set of fired memory indices and y_n is the set of correct label indices. The second condition penalized excessive memory retrieval, expressed as $B_n = \frac{|f_n|}{|y_n|}$. The final score was calculated as $Score = \sum_{n=1}^{N} -(A_n + 0.15 * B_n)$. The factor 0.15 was applied to B_n to prevent either condition from becoming dominant. The scaling factor of 0.15 was determined through extensive experimentation across multiple datasets, as it optimally balanced the firing of correct label memories and excessive memory retrieval. Increasing this value tended to penalize excessive retrieval more strongly, reducing the number of retrievals, while decreasing it led to more retrievals and potentially more correct label memories being fired, however with an increased risk of noise. The negative sum was used to avoid potential division by zero issues.

The hyperparameters were optimized to maximize this score for the SynapticRAG model Details of the specific hyperparameters are provided below:

Hyperparameter Settings for Extended Models

MemoryBank (Extended) The original MemoryBank model calculates memory scores as follows:

First, for each memory:

$$\text{Score}^{(i)} = \exp\left(\frac{\Delta t^{(i)}}{5 s^{(i)}}\right)$$

where $\Delta t^{(i)}$ is the time difference between when the memory occurred and the current time, and $s^{(i)}$ is the memory strength, with an initial value of 1.

Subsequently, the Score is cut off by a randomly set threshold θ, and memories below this are considered forgotten. We consider the set of memories that exceed the threshold:

$$\text{Retain} = \{i \mid \text{Score}^{(i)} \geq \theta\}$$

Based on the cosine similarity with the input vector, the top 6 memories are selected from this set:

$$\text{Top}_6 = \text{argsort}_{i \in \text{Retain}}(\text{cos\_sim}(i))[:6]$$

The strength s of the selected memories is updated by adding 1:

$$s_0^{(i)} = 1, \quad s_n^{(i)} = s_{n-1}^{(i)} + 1 \quad \text{for} \quad i \in \text{Top}_6$$

The memory strength is updated in this way at each turn. In the Extended model, we introduced five parameters to this model: Top_k, Θ, t_{scale}, s_{scale}, s_{init}.

The equations are as follows:

$$\text{Score}^{(i)} = \exp\left(\frac{t_{\text{scale}} \Delta t^{(i)}}{s_{\text{scale}} s_i}\right)$$

$$\text{Retain} = \{i \mid \text{Score}^{(i)} \geq \Theta\}$$

$$\text{Top}_k = \text{argsort}_{i \in \text{Retain}}(\text{cos\_sim}(i))[:k]$$

$$s_0^{(i)} = s_{\text{init}}, \quad s_n^{(i)} = s_{n-1}^{(i)} + 1 \quad \text{for} \quad i \in \text{Top}_k$$

The optimization was performed based on these set parameters.

MyAgent (Extended) The original MyAgent model is defined as follows: First, the recall probability p is evaluated for memories exceeding a cosine similarity threshold with the input vector:

$$\text{Retain} = \{i \mid \text{cos\_sim}(i) \geq \text{cos\_th}\}$$

p is calculated as:

$$p^{(i)}(t) = \frac{1 - \exp(-r^{(i)} e^{-\Delta t^{(i)}} / g^{(i)})}{1 - e^{-1}} \quad \text{for} \quad i \in \text{Retain}$$

where Δt is the time difference between memory occurrence and current time, and g is memory strength, initially set to 1. The memory with the highest p is retrieved, and its strength g is updated:

$$j = \text{argmax}_{i \in \text{Retain}}(p^{(i)}(t))$$

$$g_0^{(j)} = 1, \quad g_n^{(j)} = g_{n-1}^{(j)} + \frac{1 - e^{-\Delta t^{(j)}}}{1 + e^{-\Delta t^{(j)}}}$$

The memory strength is thus updated in each turn. The Extended model introduces four parameters: cos_th, r_{scale}, t_{scale}, g_{scale}. cos_th is the cosine similarity threshold from the original model.

The equations for other parameters are as follows:

$$\text{Retain} = \{i \mid \text{cos\_sim}(i) \geq \text{cos\_th}\}$$

$$p^{(i)}(t) = \frac{1 - \exp(-r_{\text{scale}} r^{(i)} e^{-t_{\text{scale}} \Delta t^{(i)}} / g_{\text{scale}} g^{(i)})}{1 - e^{-1}}$$

$$j = \text{argmax}_{i \in \text{Retain}}(p^{(i)}(t))$$

$$g_0^{(j)} = 1, \quad g_n^{(j)} = g_{n-1}^{(j)} + \frac{1 - e^{-t_{\text{scale}} \Delta t^{(j)}}}{1 + e^{-t_{\text{scale}} \Delta t^{(j)}}}$$

Optimization was performed based on these parameters.

SynapticRAG The calculation procedure and describe the parameter settings are summarized herein. The set parameters are as follows:

cos_th, v_th, $stim\_th$, τ_{init}, τ_{scale}, t_{scale}, B_{scale}, V_{rest}, I_{rest}

First, scaling constants are multiplied to τ, t and B_{score}. For simplicity, we use the original variable names for the multiplied values:

$$\tau \to \tau_{\text{scale}} \tau$$

$$t \to t_{\text{scale}} t$$

$$B_{\text{score}} \to B_{\text{scale}} B_{\text{score}}$$

In the stimulus propagation mechanism, stimuli are given to memories exceeding cos_th in cosine similarity with the input vector, and propagation to the next generation is allowed for memories exceeding $stim_t h$ in stimulus value:

$$\text{Retain} = \{i \mid \text{cos\_sim}(i) \geq \text{cos\_th}\}$$

$$Stim^{(i)} \to \text{Next\_gen} \quad \text{if} \quad Stim^{(i)} \geq \text{stim\_th} \quad \text{for} \quad i \in \text{Retain}$$

Finally, LIF model calculations are performed for stimulated memories:

$$\tau(0) = \tau_{\text{init}}, \quad \tau(t + \Delta t) = \tau(t) + \frac{1 - \exp(-\Delta t)}{1 + \exp(-\Delta t)}$$

$$\tau \frac{dI}{dt} = -I(t) + \sum_{t_f \in \Gamma} \boldsymbol{S}_t \delta(t - t_f)$$

$$\tau \frac{dV}{dt} = -V(t) + I(t)$$

Memories with V exceeding v_th are considered fired, and their V and I are reset to V_{rest} and I_{rest}:

$$\text{Fire} = \{j \mid V^{(j)} \geq \text{v\_th}\}$$

$$V^{(j)} \to V_{\text{rest}}, \quad I^{(j)} \to I_{\text{rest}} \quad \text{for} \quad j \in \text{Fire}$$

Optimization was performed based on these parameters.

Experimental Results

This section presents the results of experiments conducted on four distinct datasets: SMRCs (English), SMRCs (Japanese), PerLTQA (English), and PerLTQA (Chinese). The performance of the proposed SynapticRAG model was evaluated against three baseline models (RAG, MemoryBank, MyAgent) and two extended models (MemoryBank Extended, MyAgent Extended). The hyperparameters for each model were kept consistent across all datasets.

Performance Comparison

The results show that SynapticRAG consistently outperformed all baseline models across all datasets and languages. In the ERC evaluation (Table 2), SynapticRAG achieved the highest accuracy of 93.12% on the PerLTQA Chinese dataset, exceeding the second-best model, RAG, by 8.17%. The performance disparity was even more significant on the SMRCs English dataset, where SynapticRAG (86.21%) outperformed RAG (71.55%) by 14.66%.

The ERC-MG evaluation (Table 3) reveals similar trends, with SynapticRAG maintaining superior performance. Although there were slight improvements in the performances of the baseline models under ERC-MG, these do not substantially close the gap with SynapticRAG, highlighting the robustness of the proposed model.

Cross-lingual Performance Analysis

The results demonstrate SynapticRAG's consistent performance across different languages. On the SMRCs dataset, SynapticRAG achieved accuracies of 86.21% and 92.32% in English and Japanese, respectively. On the PerLTQA dataset, it attained accuracies of 90.47% and 93.12% for English and Chinese, respectively. This consistency highlights the language-agnostic nature of our model and its potential for effective deployment in multilingual environments, in contrast to the baseline models, which exhibited greater variability across languages.

Note that SynapticRAG outperformed RAG and other baseline models across all datasets and languages using hyperparameters that were tuned on a subset of the SMRCs (English) dataset. Such consistent performance across diverse datasets and languages underscores the effectiveness and broad applicability of SynapticRAG in various memory retrieval tasks.

Dataset	RAG	MemoryBank	MemoryBank$_{\text{(Ext)}}$	MyAgent	MyAgent$_{\text{(Ext)}}$	**SynapticRAG**	Δ%
SMRCs$^{\text{EN}}$	71.55	32.60	62.36	52.30	63.24	**86.21**	+14.66
SMRCs$^{\text{JA}}$	81.58	49.34	72.15	56.80	77.19	**92.32**	+10.74
PerLTQA$^{\text{EN}}$	82.14	33.62	67.24	50.99	66.49	**90.47**	+8.33
PerLTQA$^{\text{CN}}$	84.95	36.75	71.31	53.38	69.54	**93.12**	+8.17

Table 2: Accuracy (%) of memory retrieval models evaluated using the **ERC** method. Δ denotes the performance improvement of SynapticRAG over the best baseline model. **Note:** SynapticRAG displays identical results in ERC and ERC-MG evaluations because of consistent parameters and retrieval count across both methods. Other models maintain fixed parameters but vary in retrieval count between the two methods.

Dataset	RAG	MemoryBank	MemoryBank$_{\text{(Ext)}}$	MyAgent	MyAgent$_{\text{(Ext)}}$	**SynapticRAG**	Δ%
SMRCs$^{\text{EN}}$	71.55	32.82	62.58	52.30	63.46	**86.21**	+14.66
SMRCs$^{\text{JA}}$	81.58	49.34	72.15	56.80	77.19	**92.32**	+10.74
PerLTQA$^{\text{EN}}$	82.51	33.62	67.54	51.03	66.64	**90.47**	+7.96
PerLTQA$^{\text{CN}}$	85.36	36.75	71.61	53.42	69.76	**93.12**	+7.76

Table 3: Accuracy (%) of memory retrieval models evaluated using the **ERC-MG** method. Δ denotes the performance improvement of SynapticRAG over the best baseline model.

Computational Efficiency

We evaluated the computational efficiency of each model by measuring the average processing time required for a single conversational task. RAG exhibited the shortest processing time at 1.6e-4 s per task, followed by MyAgent (4.2e-3 s), MyAgent(Ext) (4.5e-3 s), SynapticRAG (1.4e-2 s), MemoryBank (1.6e-2 s), and MemoryBank(Ext) (1.7e-2 s). Although SynapticRAG operates slower than RAG, it maintains competitive speeds in comparison to other memory-augmented models. This indicates a reasonable trade-off between enhanced accuracy and computational efficiency, maintaining its suitability for real-time applications. The extended versions of MyAgent and MemoryBank exhibited slightly increased processing times, which reflects the added complexity from extended hyperparameters. However, this increase is offset by their enhanced performance.

Impact of Evaluation Methods

The comparison between the ERC and ERC-MG evaluation methods highlights the consistency of SynapticRAG's performance advantage. The results of the SynapticRAG remained consistent owing to its design, and the performance disparity with baseline models was significant under both methods. For example, on the PerLTQA English dataset, the performance difference between SynapticRAG and the best baseline model differed marginally from 8.33% (ERC) to 7.96% (ERC-MG). This consistent performance disparity across evaluation methods emphasizes the substantial improvement of SynapticRAG over baseline models, regardless of the specific evaluation criteria used. It demonstrates that SynapticRAG's advantages in memory retrieval persist even when baseline models are given more flexibility in retrieval count.

Limitation

The experimental results demonstrated SynapticRAG's effectiveness across various datasets and languages, consistently outperforming baseline models. However, potential limitations should be considered for future improvements.

SynapticRAG's complex mechanisms, while effective for long-term context management, may increase computational costs and resource requirements, especially for large-scale datasets or extended dialogue histories. As the conversation history expands, the number of recalled memories tends to rise, leading to increased computational costs. To mitigate this issue, we recommend adjusting parameter values, such as the recall threshold, based on specific requirements.

The optimization of the model parameters is more challenging and requires more extensive tuning compared to simpler models. The variable number of recalled memories, although allowing for more flexible and context-aware responses, can be problematic when strict control over the number of recalled items is required. Additionally, the intricate stimulus propagation and firing decision processes may reduce model interpretability compared to RAG's simpler vector similarity approach. This complexity, while beneficial for performance, may pose challenges in debugging and fine-tuning the model for specific applications.

Future work should focus on optimizing the trade-off between computational efficiency and context management capability, especially for long-term dialogue scenarios. Developing techniques to manage the growing number of recalled memories without sacrificing performance will be crucial for scaling SynapticRAG to larger and more diverse datasets.

Conclusions

In this paper, we introduced SynapticRAG, a memory retrieval model that integrates temporal representations into memory vectors. Our approach addresses the limitations of existing dialogue agents in maintaining complex, long-term contexts and extracting specific information from conversations. Extensive experiments on the multilingual datasets SMRCs and PerLTQA showed that SynapticRAG consistently outperformed baseline models, including RAG, MemoryBank, and MyAgent, across English, Japanese, and Chinese. SynapticRAG achieved up to 14.66% improvement in memory retrieval accuracy, which signifies its effectiveness in simulating human-like memory processes. Its consistent performance across different languages suggests its potential for multilingual deployment. The success of the model in retrieving relevant memories while considering both semantic similarity and temporal context marks a significant advancement in dialogue AI systems.

This research contributes to the development of sophisticated, context-aware AI agents capable of maintaining long-term memory and engaging in meaningful dialogues. Integrating neuroscience-inspired mechanisms into our model paves the way for bridging the gap between artificial and human-like memory systems. Although SynapticRAG displays improvements, future work should focus on optimizing computational efficiency, exploring the performance of the model in extended dialogue scenarios, and investigating its real-world applicability. In conclusion, SynapticRAG significantly enhances the contextual and temporal awareness of dialogue agents. Continued refinement of this approach is expected to lead to AI systems capable of more natural, context-rich interactions, ultimately benefiting user-centric applications across various domains.

References

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, 2623–2631. New York, NY, USA: Association for Computing Machinery. ISBN 9781450362016.

Anthropic. 2024a. The Claude 3 Model Family: Opus, Sonnet, Haiku. https://paperswithcode.com/paper/the-claude-3-model-family-opus-sonnet-haiku.

Anthropic. 2024b. Introducing Claude 3.5 Sonnet. https://www.anthropic.com/news/claude-3-5-sonnet. (Accessed on 07/19/2024).

Benna, M. K.; and Fusi, S. 2015. Computational principles of biological memory. arXiv:1507.07580.

Borgeaud, S.; Mensch, A.; Hoffmann, J.; Cai, T.; Rutherford, E.; Millican, K.; van den Driessche, G.; Lespiau, J.-B.; Damoc, B.; Clark, A.; de Las Casas, D.; Guy, A.; Menick, J.; Ring, R.; Hennigan, T.; Huang, S.; Maggiore, L.; Jones, C.; Cassirer, A.; Brock, A.; Paganini, M.; Irving, G.; Vinyals, O.; Osindero, S.; Simonyan, K.; Rae, J. W.; Elsen, E.; and Sifre, L. 2022. Improving language models by retrieving from trillions of tokens. arXiv:2112.04426.

Brunel, N.; and van Rossum, M. C. 2007. Lapicque's 1907 paper: from frogs to integrate-and-fire. *Biological Cybernetics*, 97(5-6): 337–339.

Citri, A.; and Malenka, R. C. 2008. Synaptic plasticity: multiple forms, functions, and mechanisms. *Neuropsychopharmacology*, 33(1): 18–41.

Douze, M.; Guzhva, A.; Deng, C.; Johnson, J.; Szilvasy, G.; Mazaré, P.-E.; Lomeli, M.; Hosseini, L.; and Jégou, H. 2024. The Faiss library. arXiv:2401.08281.

Du, Y.; Wang, H.; Zhao, Z.; Liang, B.; Wang, B.; Zhong, W.; Wang, Z.; and Wong, K.-F. 2024. PerLTQA: A Personal Long-Term Memory Dataset for Memory Classification, Retrieval, and Synthesis in Question Answering. arXiv:2402.16288.

Gerstner, W.; Kistler, W. M.; Naud, R.; and Paninski, L. 2014. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press.

Hebb, D. O. 1949. *The organization of behavior; a neuropsychological theory*. Wiley.

Hou, Y.; Tamoto, H.; and Miyashita, H. 2024. "My agent understands me better": Integrating Dynamic Human-like Memory Recall and Consolidation in LLM-Based Agents. *arXiv preprint arXiv:2404.00573*.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Curran Associates Inc. ISBN 9781713829546.

Maass, W. 1997. Networks of Spiking Neurons: The Third Generation of Neural Network Models. *Neural Networks*, 10(9): 1659–1671.

McDaniel, M. A.; and Einstein, G. O. 2000. Strategic and automatic processes in prospective memory retrieval: a multiprocess framework. *Applied Cognitive Psychology*, 14.

Miconi, T.; Clune, J.; and Stanley, K. O. 2018. Differentiable plasticity: training plastic neural networks with backpropagation. arXiv:1804.02464.

M"uller, M. 2007. *Dynamic time warping*, 69–85. Berlin, Heidelberg: Springer.

Murdock, B. B. 1982. A theory for the storage and retrieval of item and associative information. *Psychological Review*, 89(6): 609–626.

MySQL, A. 2001. MySQL.

Neves, G.; Cooke, S.; and Bliss, T. 2008. Synaptic plasticity, memory and the hippocampus: a neural network approach to causality. *Nature Reviews Neuroscience*, 9: 65–75.

Numpy. 2024. NumPy 1.26.4 Release Notes — NumPy v2.1.dev0 Manual. https://numpy.org/devdocs/release/1.26.4-notes.html. (Accessed on 08/07/2024).

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

OpenAI. 2024. Embeddings - OpenAI API. https://platform.openai.com/docs/guides/embeddings. (Accessed on 05/21/2024).

Optuna. 2024. Tutorial — Optuna 3.6.1 documentation. https://optuna.readthedocs.io/en/stable/tutorial/index.html. (Accessed on 08/07/2024).

Park, J. S.; O'Brien, J. C.; Cai, C. J.; Morris, M. R.; Liang, P.; and Bernstein, M. S. 2023. Generative Agents: Interactive Simulacra of Human Behavior. arXiv:2304.03442.

Python. 2022. Python Release Python 3.10.9 — Python.org. https://www.python.org/downloads/release/python-3109/. (Accessed on 08/07/2024).

Sakoe, H.; and Chiba, S. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1): 43–49.

Tavanaei, A.; Ghodrati, M.; Kheradpisheh, S. R.; Masquelier, T.; and Maida, A. 2019. Deep learning in spiking neural networks. *Neural Networks*, 111: 47–63.

Tulving, E. 1985. *Elements of episodic memory*. Number 2 in Oxford psychology series. Clarendon Press, Oxford University Press.

Zhong, W.; Guo, L.; Gao, Q.; Ye, H.; and Wang, Y. 2023. MemoryBank: Enhancing Large Language Models with Long-Term Memory. arXiv:2305.10250.