# Crafting Personalized Agents through Retrieval-Augmented Generation on Editable Memory Graphs

**Anonymous EMNLP submission**

## Abstract

In the age of mobile internet, user data, often referred to as memories, is continuously generated on personal devices. Effectively managing and utilizing this data to deliver services to users is a compelling research topic. In this paper, we introduce a novel task of crafting personalized agents powered by large language models (LLMs), which utilize a user's smartphone memories to enhance downstream applications with advanced LLM capabilities. To achieve this goal, we introduce EMG-RAG, a solution that combines Retrieval-Augmented Generation (RAG) techniques with an Editable Memory Graph (EMG). This approach is further optimized using Reinforcement Learning to address three distinct challenges: data collection, editability, and selectability. Extensive experiments on a real-world dataset validate the effectiveness of EMG-RAG, achieving an improvement of approximately 10% over the best existing approach. Additionally, the personalized agents have been transferred into a real smartphone AI assistant, which leads to enhanced usability.

## 1 Introduction

In the era of mobile internet, personal information is constantly being generated on smartphones. This data, referred to as personal memories, is often scattered across everyday conversations with AI assistants (e.g., Apple's Siri), or within a user's apps (e.g., screenshots), including emails, calendars, location histories, travel activities, and more. As a result, managing and utilizing these personal memories to provide services for users becomes a challenging yet attractive task. With the emergence of advanced large language models (LLMs), new opportunities arise to leverage their semantic understanding and reasoning capabilities to develop personal LLM-driven AI assistants.

Motivated by this trend, we study the problem of crafting personalized agents that enhance the AI assistants with the capabilities of LLMs by leveraging users' memories on smartphones. Unlike existing personal LLM agents (Li et al., 2024b), such as those designed for psychological counseling (Zhong et al., 2024), housekeeping (Han et al., 2024), and medical assistance (Zhang et al., 2023), the personalized agents face unique challenges due to practical scenarios and remains relatively unexplored in current methods.

These challenges can be summarized below. (1) Data Collection: Personal memories should encompass valuable information about a user. Extracting these memories from everyday trivial conversations presents unique challenges in data collection, especially considering that existing datasets like personalized chats sourced through crowdsourcing (Zhang et al., 2018) or psychological dialogues (Zhong et al., 2024) lack this property. Moreover, constructing annotated data, such as QA pairs, is essential for enabling effective training of personalized agents. (2) Editability: Personal memories are dynamic and continuously evolving, requiring three types of editable operations: insertion, deletion, and replacement. For example, 1) insertion occurs when new memories are added; 2) deletion is necessary for time-sensitive memories, such as a hotel voucher that expires and needs to be removed; 3) replacement is required when an existing memory, such as a flight booking, undergoes a change in departure time and needs updating. Therefore, a carefully designed memory data structure is essential to support this editability. (3) Selectability: To enable the memory data services for real-world applications, it often requires querying a combination of multiple memories. For example, in a QA scenario (illustrated in Table 1), the AI assistant answering a question about "a secretary's boss's flight departure time" needs several memories: the secretary booked a flight to Amsterdam for her boss ($M_1$); the flight's number is EK349 ($M_2$); the departure time for EK349 is at

01:40 on 2024-05-12 ($M_4$). To achieve this, one intuitive approach is to use Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) to find relevant memories and form a context that is fed into a LLM to generate answers. Here, we discuss two potential solutions and their limitations, which motivate the proposed solution. 1) Needles in a Haystack (NiaH) (Briakou et al., 2023): it organizes all memories into a single context (the "Haystack") and inputs this into a LLM, relying on the capability of a LLM itself to identify relevant memories (the "Needles") for generating an answer. However, this method incurs significant overhead by extending the LLM's context window and introduces noise from irrelevant memories, hindering the LLM's ability to generate accurate answers. 2) Advanced RAG (Wang et al., 2024; Ma et al., 2023): many advanced RAG techniques still rely on Top-$K$ retrieval to identify relevant memories. However, a fixed parameter $K$ may limit the LLM's ability to uncover all relevant memories, especially for the questions requiring diverse memory combinations. Thus, an adaptive selection mechanism is essential for the personalized applications.

To this end, we introduce a new solution called EMG-RAG, which presents the first attempt of its kind to address these challenges. We discuss the solution along with the rationales behind it below. For (1), we utilize a business dataset collected from a real AI assistant, which includes daily conversations with the assistant, and users' app screenshots, to extract personal memories. Specifically, we leverage the capabilities of GPT-4 (OpenAI, 2023) to clean the raw data into memories. We organize the memories chronologically, and then use GPT-4 to generate QA pairs within each session (a set of consecutive memories). We also tag the memories involved in generating these QA pairs, which are then used for subsequent training purposes. For (2), we introduce a three-layer data structure, called Editable Memory Graph (EMG). The first two layers form a tree structure in accordance with the business scopes, while the third layer consists of a user's memory graph parsed from the memory data. This design is motivated by three considerations: 1) the tree structure allows for partitioned management of various memory categories, facilitating expansion to other categories; and 2) memory data is partitioned under different categories, with the graph structure to capture their complex relationships, and 3) this enables efficient retrieval to locate specific memories for editing, by searching within relevant partitions rather than the entire dataset. For (3), we introduce a reinforcement learning (RL) agent that adaptively selects memories on the EMG, without being constrained to a fixed Top-$K$ approach. The rationale of using RL resembles a boosting process. Specifically, when the agent selects relevant memories (actions), it prompts a LLM (frozen) to generate improved answers. The quality of these answers is evaluated by a downstream task metric (reward), which then guides the agent to refine its policy for better memory selection. This results in an end-to-end optimization process aimed at achieving the desired goal for downstream tasks.

Overall, we make the following contributions. (1) We introduce a novel task of crafting LLM-driven personalized agents, leveraging users' personal memories to enhance their experience through LLM capabilities. This task differs from existing personal LLM agents in three key challenges: data collection, editability, and selectability. (2) We propose EMG-RAG, a novel solution that combines EMG and RAG to address the three challenges. We show that it enables an end-to-end optimization process through reinforcement learning to achieve the goal of personalized agents. (3) We conduct extensive experiments on a real-world business dataset across various LLM architectures and RAG methods for three downstream applications: question answering, autofill forms, and user services. Our approach demonstrates improvements of approximately 10.6%, 9.5%, and 9.7% over the best existing approach for these tasks, respectively. Moreover, the personalized agents have been transferred into an AI assistant product, resulting in a notable improvement in user experience.

## 2 Related Work

**Personalized Dialogue System.** To develop a personalized dialogue system (PDS), the PersonaChat dataset (Zhang et al., 2018) is collected through crowdsourcing, which comprises Personas (each persona is defined by a set of profile sentences) and Chats (each chat is collected by two crowdworkers with two randomly assigned personas). Based on the dataset, various techniques have been studied to address challenges in PDS, including mutual persona perception (Liu et al., 2020; Xu et al., 2022a; Kim et al., 2020), persona-sparsity (Song et al., 2021; Welch et al., 2022), long-term persona memory (Xu et al., 2022b; Zhong et al., 2024), etc. For example, $\mathcal{P}^2$BOT (Liu et al., 2020) is a GPT-based framework (Radford et al., 2018), specifically designed to enrich personalized dialogue generation through mutual persona perception. It aims to model the underlying understanding, such as character traits, within a conversation to facilitate mutual acquaintance between interlocutors. In addition, a PDS can be further enhanced by integrating internal reasoning techniques (Hongru et al., 2023) or external acting techniques (Wang et al., 2023b), which aim to generate more personalized and factual responses. In this study, we construct user-personalized agents using practical memory data gathered from smartphone AI assistants. Leveraging these agents, we introduce three distinct applications: question answering, autofill forms, and user services.

**Retrieval-Augmented Generation on Knowledge Graph.** We review the literature on RAG on knowledge graphs across various tasks, including KBQA (Ye et al., 2021; Das et al., 2021; Wang et al., 2023a; Shu et al., 2022), open-domain scenarios (Yang et al., 2023), table-related tasks (Jiang et al., 2023), human-machine conversation (Zhang et al., 2020), and image captioning (Hu et al., 2023). This paper (Zhao et al., 2024) provides a detailed survey on these tasks with RAG techniques. Specifically, TIARA (Shu et al., 2022) stands out as a KBQA model employing multi-grained retrieval (entities, logical forms, and schema items) from knowledge graphs. This approach aids pre-trained language models in mitigating generation errors. In this study, we introduce a novel EMG structure to manage users' personal memories. Further, we employ RL to model the RAG process, which optimizes the memory selection on the graph.

**Model Editing.** Model editing represents a recent research area focused on correcting model predictions in light of evolving real-world dynamics. It edits the behavior of pre-trained language models within specific domains, and preserving performance across other domains without compromise. Some existing methods (De Cao et al., 2021; Mitchell et al., 2021) employ learnable model editors, which are trained to predict the weights of the base model undergoing editing. Other methods (Meng et al., 2022a,b; Li et al., 2024a) are designed to identify stored facts (such as specific neurons in the network) and adjust corresponding activations to reflect changed facts. Additionally, SERAC (Mitchell et al., 2022) utilizes an external memory to store edits, adaptively altering the base model's predictions by retrieving relevant edits. In our study, we leverage a LLM to focus on user personal memories rather than global knowledge. Additionally, we support dynamic user edits on the EMG and utilize RAG with a frozen LLM to respond to these changes.

## 3 Problem Statement

We study the problem of developing personalized agents for users on smartphone AI assistant platforms (such as Apple's Siri or Samsung's Bixby). These agents are designed to assist users in performing personalized tasks, requiring the fulfillment of the following two properties in practical scenarios:

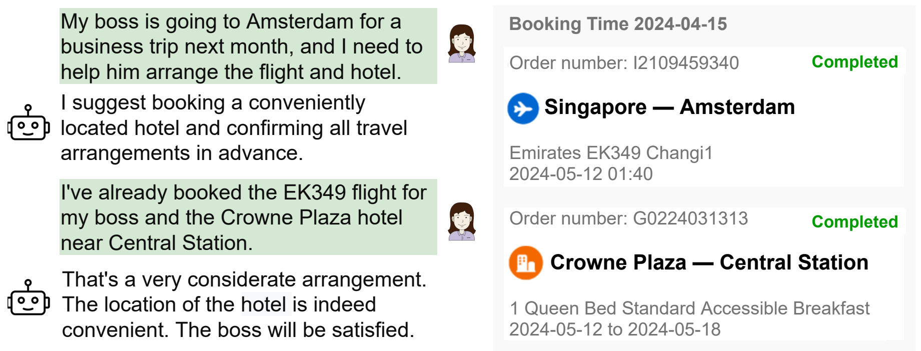
- Editability: The responses from the agents may be editable based on the users' dynamic memory data, which involves insertion, deletion, and replacement operations corresponding to different usage scenarios, as illustrated in Figure 2(a).
- Selectability: The agents can select relevant memories to respond to users' queries, with some queries requiring the combination of multiple memories to generate responses through a base language model, as illustrated in Figure 2(b).

By satisfying these properties, the agents aim to enhance the user experience during interactions with their smartphone AI assistants. These agents offer essential functionalities to support personalized applications, including question answering, autofill forms, and user services like reminders for important events and times, and travel navigation (further details will be discussed in Section 4.4).

## 4 Methodology

### 4.1 Data Collection

The process entails (1) gathering raw data, such as everyday conversations or screenshots from user interactions with the smartphone AI assistants; (2) extracting crucial information from this raw data, referred to as memories (denoted by $M$); and (3) generating QA pairs (denoted by $< Q, A >$), and outputting the required memories to facilitate this pairing. For (1), we acquire data from real AI assistant products and employ text processing techniques like OCR to extract content from screenshots. Subsequently, for (2) and (3), we leverage the capabilities of LLMs, such as GPT-4 (OpenAI,



My boss is going to Amsterdam for a business trip next month, and I need to help him arrange the flight and hotel.

I suggest booking a conveniently located hotel and confirming all travel arrangements in advance.

I've already booked the EK349 flight for my boss and the Crowne Plaza hotel near Central Station.

That's a very considerate arrangement. The location of the hotel is indeed convenient. The boss will be satisfied.

Booking Time 2024-04-15

Order number: I2109459340 Completed

Singapore — Amsterdam

Emirates EK349 Changi1
2024-05-12 01:40

Order number: G0224031313 Completed

Crowne Plaza — Central Station

1 Queen Bed Standard Accessible Breakfast
2024-05-12 to 2024-05-18

Figure 1: An example of data collection. Step-1: Raw data is gathered on smartphone AI assistant platforms, e.g., everyday conversations between users and assistants, and the extraction of app screenshot contents through OCR.

Table 1: An example of data collection. Step-2: GPT-4 generates memories from raw data. Step-3: GPT-4 forms QA pairs using several memories, and produces the required memories, which are utilized for training the EMG-RAG.

| Step-2: Memories (generated by GPT-4) | Step-3: QA pairs with memories (generated by GPT-4) |
|---|---|
| $M_1$: My boss is traveling to Amsterdam next month, I assist with flight and hotel arrangements. $M_2$: I booked the EK349 flight. $M_3$: I booked the Crowne Plaza near Central Station. $M_4$: The EK349 flight departs at 01:40 on 2024-05-12. | $Q$: What time is my boss's flight to Amsterdam? $A$: Your boss flight EK349 departs at 01:40 on 2024-05-12. Required memories: $M_1, M_2, M_4$ |
| $M_5$: The Crowne Plaza reservation is for 2024-05-12 to 2024-05-18. $M_6$: The Crowne Plaza reservation includes a Queen Bed Standard Accessible room with breakfast. | $Q$: When dose the hotel I booked for my boss start and end? $A$: The Crowne Plaza reservation is from 2024-05-12 to 2024-05-18. Required memories: $M_1, M_3, M_5$ |

2023), to extract key memories from the raw data and create QA pairs. These pairs serve the purpose of training personalized agents for the proposed EMG-RAG. To illustrate the collection process, we provide a running example in Figure 1 and Table 1, which involve the three primary steps. Further details are outlined in Appendix A.1.

We discuss the rationales of the data collection. First, as a user's personalized agent integrated within the smartphone AI assistant, the conversations and screenshots provide natural data sources for crafting these agents. Second, leveraging GPT-4's language generation capabilities enables us to generate a wide range of memories from the raw data, significantly reducing manual effort. Third, the involved memories and collected QA pairs serve as labels to supervise the training of the retrieval and generation processes in our framework.

## 4.2 Editable Memory Graphs

**The EMG Construction and Insights.** Utilizing a user's memories, we establish the Editable Memory Graph with a multilayered structure, depicted in Figure 2(a), where the user is the root node.

Memory Type Layer (MTL): Aligned with the business scope, we categorize memories into 4 predefined types: Relationship, Preference, Event, and Attribute. Details are provided in Appendix A.2.

Memory Subclass Layer (MSL): The MSL further outlines subclasses for each type, where the MTL and MSL are organized in a hierarchical tree structure to manage the memories. Detailed subclasses with examples are listed in Appendix A.2.

Memory Graph Layer (MGL): The memory graph is built by utilizing the collected memories, employing entity recognition for nodes and relation extraction for edges. In this graph, each in-degree node is associated with its corresponding memory, e.g., the in-degree node (01:40 on 2024-05-12) contains $M_4$, as shown in Figure 2(a). Further, to establish the connection between the MSL and MGL, TransE embeddings (Bordes et al., 2013) are employed to capture semantic information of nodes in MSL (subclasses) and MGL (entities), respectively. Then, each entity is assigned to its closest classes based on these embeddings. It is noteworthy that entity nodes are categorized into different subclasses, and their connections may span across different classes, e.g., "Boss" and "Amsterdam" are linked across "Colleague" and "Arrangement" classes in Figure 2(a). This design enables further traversal across various parts of the whole graph.

We discuss the insights of the EMG construction: 1) the tree hierarchy (MTL and MSL) offers a partitioned memory management approach, to facilitate the expansion of additional types and subclasses in accordance with business needs; 2) the entity nodes and corresponding memories are organized into separate subclass partitions, with the graph structure (MGL) to capture their complex relationships

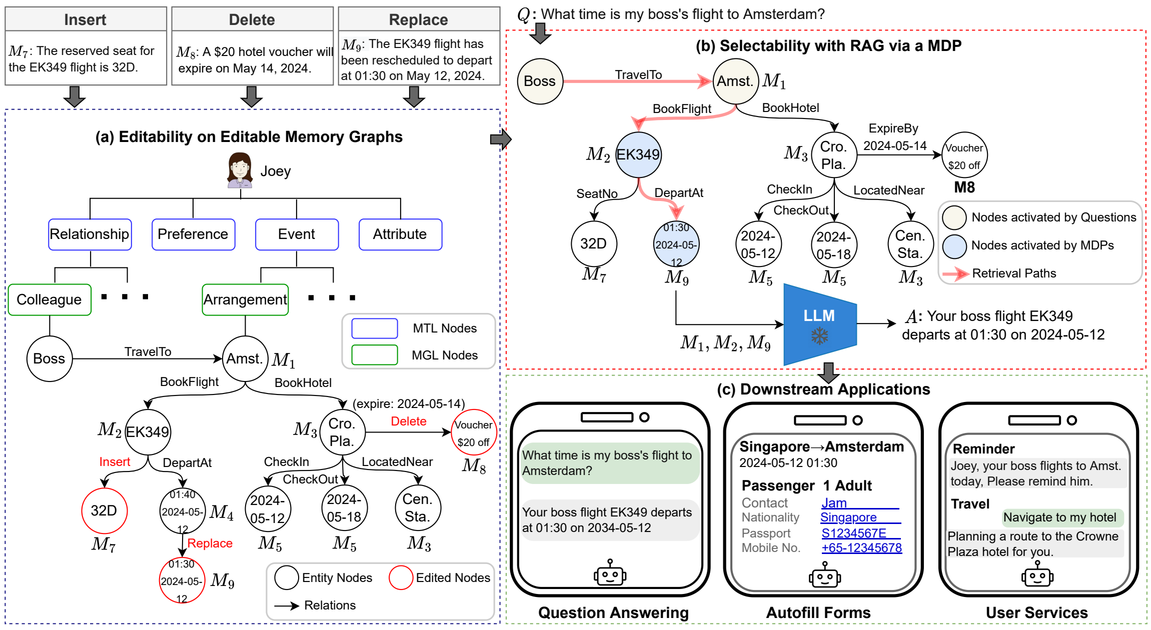


Figure 2: The architecture of the proposed EMG-RAG, demonstrated with the running example in data collection (Section 4.1). It supports three editability operations: insertion (e.g., $M_7$), deletion (e.g., $M_8$), and replacement (e.g., $M_9$), based on the EMG structure (Section 4.2). Subsequently, the edited EMG undergoes RAG to select relevant memories (e.g., $M_1, M_2, M_9$) for a given question $Q$ via a MDP (Section 4.3). The generated answers $A$ by a frozen LLM further facilitates three downstream applications (Section 4.4).

between memories; 3) it enables efficient retrieval of memories for further editing operations by first locating a relevant partition, e.g., querying partition centers (the mean of the memory embeddings), instead of searching through all memories.

**The EMG Editing.** When editing a given memory within the EMG (e.g., insertion, deletion, or replacement), the process involves three steps. Initially, a model such as CPT-Text (Neelakantan et al., 2022) is employed to acquire memory representations. Then, the memory is assigned to its nearest subclass (partition), and the Top-1 retrieved memory within the partition is then returned, and editing operations are performed based on comparing the relations between the given memory and the retrieved memory. Specifically, as illustrated in Figure 2, (1) Insertion: It introduces a new relation to be added, e.g., obtaining a new memory containing flight seat number. (2) Deletion: It introduces a new relation, but it is valid for a specific period of time. e.g., a hotel voucher will expire on May 14, 2024. (3) Replacement: It provides an existing relation, and updates the corresponding entity nodes based on this relation, e.g., changing the departure time to 01:30 on May 12, 2024.

### 4.3 MDP for Selecting Memories on EMGs

Next, we outline the task of selecting memories based on an edited EMG. To achieve this, we employ an agent to traverse the EMG. Specifically, given a question $Q$, the agent selects a set of memories from the EMG denoted by $\mathbb{M} = \{M_i\}$, where $1 \leq i \leq |\mathbb{M}|$. The question $Q$ and memory set $\mathbb{M}$ are concatenated to generate an answer $\hat{A} \leftarrow \text{LLM}(Q \oplus \mathbb{M})$ using a LLM. We assess the generation quality using $\Delta(\hat{A}, A)$, where $A$ represents the collected ground truth answer for $Q$, and $\Delta(\cdot,\cdot)$ denotes a specific metric (e.g., ROUGE (Lin, 2004) or BLEU (Post, 2018)). We note that a high-quality answer $\hat{A}$ benefits from the selected memories $\mathbb{M}$, which can then provide feedback with $\Delta(\cdot,\cdot)$ for subsequent selections. As a result, it iterates in a boosting process, and we optimize it using reinforcement learning. The environment, states, actions, and rewards are introduced below.

**Constructing Environment (Nodes activated by Questions).** Given an EMG, which often contains numerous memories in practice. Here, we confine the movement of the RL agent to a subset of memories to facilitate more focused selection. To achieve this, we first retrieve Top-$K$ memories for a given question $Q$, and based on these memories, we activate the corresponding nodes on the EMG (e.g., the nodes highlighted in yellow in Figure 2(b)). Subsequently, the agent's traversal starts from each activated node via depth-first search.

**Modeling Memory Selection (Nodes activated by MDPs).** We model the graph traversal process as a MDP, involving states, actions, and rewards.

States: In the context where we have an input question $Q$, and visit a node $N_G$ (associated with a memory $M_i$ to be included into $\mathbb{M}$), and its relation $R_G$ on the EMG. We first extract the entity $N_Q$ and relation $R_Q$ from the $Q$, and the state $\mathbf{s}$ is defined by three cosine similarities $C(\cdot,\cdot)$, i.e.,

$$\mathbf{s} = \{C(\mathbf{v}_{N_Q}, \mathbf{v}_{N_G}), C(\mathbf{v}_{R_Q}, \mathbf{v}_{R_G}), C(\mathbf{v}_Q, \mathbf{v}_{M_i})\}, \tag{1}$$

where $\mathbf{v}_\cdot$ denotes the embedding vector for entities, relations, questions, or memories.

Actions: We denote an action as $a$, and it has two choices during the graph traversal: including the visiting memory $M_i$ into $\mathbb{M}$, and searching its connected nodes; or stopping the current search, and restarting a search from other branches. Thus, the action $a$ is defined as:

$$a = 1 \text{ (including) or } 0 \text{ (stopping)}. \tag{2}$$

Consider the consequence of performing an action, it transitions the environment to the next state $\mathbf{s}'$, and affects which memory to be selected for constructing the state.

Rewards: We denote the reward as $r$, which corresponds to the transition from the current state $\mathbf{s}_t$ to the next state $\mathbf{s}_{t+1}$ after taking action $a_t$. Specifically, when a memory $M$ is selected into $\mathbb{M}$, the generated answer by a LLM changes from $\hat{A}$ to $\hat{A}'$ accordingly. The quality of the generated answer $\hat{A}$ is evaluated using a specific metric $\Delta(\cdot,\cdot)$ (e.g., ROUGE or BLEU), and the reward $r$ is defined as:

$$r = \Delta(\hat{A}', A) - \Delta(\hat{A}, A), \tag{3}$$

where $A$ denotes the ground truth answer. We note that the objective of the MDP, which aims to maximize cumulative rewards, aligns with the goal of discovering memories to answer the question. To illustrate, consider a process through a sequence of states: $\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_N$, concluding at $\mathbf{s}_N$. The rewards received at these states, except for the termination state, can be denoted as $r_1, r_2, ..., r_{N-1}$. When future rewards are not discounted, we have:

$$\begin{aligned}\sum_{t=2}^{N} r_{t-1} &= \sum_{t=2}^{N} (\Delta(\hat{A}_t, A) - \Delta(\hat{A}_{t-1}, A)) \\ &= \Delta(\hat{A}_N, y) - \Delta(\hat{A}_1, y),\end{aligned} \tag{4}$$

where $\Delta(\hat{A}_N, y)$ corresponds to the result of the final answer found throughout the entire iteration, and $\Delta(\hat{A}_1, y)$ represents an initial result that remains constant. Therefore, maximizing cumulative rewards is equivalent to maximizing the quality of the final generated answer.

**Training Policies of MDPs.** Training the MDP policy involves two stages: warm-start stage (WS) and policy gradient stage (PG). In WS, we employ supervised fine-tuning to equip the agent with the basic ability to select memories given a question $Q$. Specifically, based on a state $\mathbf{s}$, the agent undergoes a binary classification task to predict whether the memory $M_i$ should be included. This prediction is supervised according to whether the memory falls into the required memories (presented in the Step-3 in Table 1). Thus, the objective is trained with binary cross-entropy, formulated as:

$$\mathcal{L}_{\text{WS}} = -y * \log(P) + (y-1) * \log(1-P), \tag{5}$$

where $y$ denotes the label (1 if the memory falls into the required memory set, and 0 otherwise), and $P$ is the predicted probability of the positive class.

In PG, our main objective is to develop a policy $\pi_\theta(a|\mathbf{s})$ that guides the agent in selecting actions $a$ based on constructed states $\mathbf{s}$, aiming to maximize the cumulative reward $R_N$. We utilize the REINFORCE algorithm (Williams, 1992; Silver et al., 2014) for learning this policy, where the neural network parameters are denoted by $\theta$. The loss function is formulated as:

$$\mathcal{L}_{\text{PG}} = -R_N \ln \pi_\theta(a|\mathbf{s}). \tag{6}$$

**Inference Stage of `EMG-RAG`.** As shown in Figure 2, the inference involves three steps: (a) collecting newly recorded memories from users and editing their EMGs; (b) using the edited EMGs to traverse the graph and retrieve relevant memories for LLM generation; (c) integrating the generated answers to serve users across three downstream applications.

### 4.4 Discussion on Applications and Cold-start

**Applications of the Personalized Agents.** As shown in Figure 2(c), we explore the capabilities of personalized agents in three scenarios: (1) question answering, (2) autofill forms, and (3) user services. For (1), `EMG-RAG` can generate answers to users' questions when they interact with the smartphone AI assistants. For (2), the goal is to extract personal information from users' EMGs to automatically fill out various online forms, such as flight and hotel bookings. To achieve this, we input form-related questions (e.g., "What is the user's mobile number?") into the LLM and use the generated entities to complete the forms. For (3), we focus on two specific domains. a) reminder service: It involves reminding users of recent events and times. To achieve this, we query a LLM for information about a user's recent events and their associated times. b) travel service: We assist users with navigation by providing the address of a destination they might want to visit. Further, we integrate the generated answers (e.g., events, times, addresses) with external tools such as calendar or map apps to provide the services for users.

**Handling the Cold-start Problem.** Given that EMG-RAG relies on generated questions for training, it may encounter a potential cold-start issue when deploying to answer real user questions. To address this issue, we utilize online learning to continuously fine-tune the agent using newly recorded questions and manually written answers, as outlined in Equation 6. This approach aims to ensure that the model's policy remains up-to-date for online usage. We validate this method through online A/B testing, and the results demonstrate improvements in user experience, highlighting the positive impact of this strategy in practice.

## 5 Experiments

### 5.1 Experimental Setup

**Dataset and Ground Truth.** We conduct experiments on a real-world business dataset containing approximately 11.35 billion raw text data (including conversations and screenshot contents) from an AI assistant product collected between March 2024 and June 2024. After data cleaning, the dataset forms around 0.35 billion memories. We follow the data distribution to randomly sample 2,000 users for training and 500 users for testing.

As detailed in Section 4.1, we establish the ground truth for the applications of question answering and autofill forms/user services using GPT-4 generated answers and key entities (e.g., identification number, address, and time), respectively. For quality control, we randomly select 10% of user data, and ask 5 participants to annotate the answers and entities. By comparing human annotations with the generated answers and entities, we report a Rouge-L score (Lin, 2004) of 91.1% for question answering, and Exact Match (Rajpurkar et al., 2018) of 87.5% for autofill forms and 97.4% for user services. These results demonstrate the high accuracy of our evaluations.

**Baselines.** We compare the EMG-RAG in terms of different RAG methods, including NiaH (Briakou et al., 2023), Naive (Ma et al., 2023), M-RAG (Wang et al., 2024), and Keqing (Wang et al., 2023a), based on various LLM architectures, such as GPT-4 (OpenAI, 2023), ChatGLM3-6B (Du et al., 2022), and PanGu-38B (Ren et al., 2023). The descriptions are included in Appendix A.3

**Evaluation Metrics.** We evaluate the effectiveness of EMG-RAG in three downstream applications. For question answering, we assess the quality of generated answers with the ground truth, and reporting ROUGE (R-1/2/L) (Lin, 2004) and BLEU (Post, 2018) scores. For autofill forms and user services, we generate key entities and report Exact Match (EM) accuracy. Overall, higher values (i.e., ROUGE, BLEU, EM) indicate better results [1].

**Implementation Details.** We provide the implementation details in Appendix A.4.

### 5.2 Experimental Results

**(1) Effectiveness evaluation (question answering).** We compare the EMG-RAG with other RAG methods for question answering on three LLMs. As shown in Table 2, we observe that the performance of EMG-RAG consistently outperforms the baselines. For example, it improves upon the best baseline method, M-RAG, by 5.3%, 8.3%, 3.9%, and 18.4% in terms of R-1, R-2, R-L, and BLEU, respectively. This improvement is due to two main factors: 1) it captures complex relationships between memories with the EMG, and 2) it effectively selects essential memories for the RAG execution. Additionally, GPT-4 demonstrates superior performance compared to other LLMs, and EMG-RAG shows comparable performance to M-RAG even when deployed on the relatively smaller ChatGLM3-6B.

**(2) Effectiveness evaluation (autofill forms).** We further evaluate the EMG-RAG for autofill forms, and it shows consistent improvement, as detailed in Table 2. For example, it surpasses M-RAG by 2.2% in terms of exact match accuracy.

**(3) Effectiveness evaluation (user services).** We target two specific domains of user services: 1) reminders of important events and their times, and 2) travel services involving destination addresses for navigation. We report the exact match accuracy

[1]We remark that all reported results are statistically significant, as confirmed by a t-test with $p < 0.05$.

Table 2: Effectiveness of EMG-RAG in downstream applications.

| LLM | RAG | Question Answering | | | | Autofill Forms | User Services (EM) | |
|---|---|---|---|---|---|---|---|---|
| | | R-1 | R-2 | R-L | BLEU | (EM) | Reminder | Travel |
| GPT-4 | NiaH | 79.89 | 64.65 | 70.66 | 38.72 | 84.86 | 84.49 | 94.81 |
| GPT-4 | Naive | 70.87 | 58.34 | 66.82 | 46.65 | 78.40 | 85.34 | 94.52 |
| GPT-4 | M-RAG | 88.71 | 77.18 | 84.74 | 64.16 | 90.87 | 93.75 | 86.67 |
| GPT-4 | Keqing | 72.11 | 57.19 | 65.46 | 35.89 | 82.03 | 90.17 | 72.71 |
| GPT-4 | EMG-RAG | **93.46** | **83.55** | **88.06** | **75.99** | **92.86** | **96.43** | **91.46** |
| ChatGLM3-6B | EMG-RAG | 85.31 | 76.03 | 82.32 | 56.88 | 85.71 | 87.50 | 81.25 |
| PanGu-38B | EMG-RAG | 91.64 | 82.86 | 86.71 | 75.11 | 90.99 | 96.41 | 89.05 |

Table 3: Effectiveness of EMG-RAG for continuous edits.

| Duration (weeks) | 1 | | | 2 | | | 3 | | | 4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # of edits | 2,515 | | | 9,644 | | | 2,096 | | | 6,290 | | |
| Apps (GPT-4) | QA | AF | US | QA | AF | US | QA | AF | US | QA | AF | US |
| M-RAG | 88.48 | 91.67 | 90.28 | 86.39 | 88.89 | 89.39 | 85.31 | 87.50 | 87.83 | 85.09 | 83.33 | 83.21 |
| EMG-RAG | **95.38** | **93.75** | **93.67** | **96.93** | **95.83** | **95.89** | **94.53** | **96.88** | **96.99** | **94.99** | **97.50** | **97.54** |

Table 4: Ablation study.

| Components | R-1 | R-2 | R-L | BLEU |
|---|---|---|---|---|
| EMG-RAG | **93.46** | **83.55** | **88.06** | **75.99** |
| w/o Act. Nodes | 90.96 | 82.72 | 86.13 | 65.07 |
| w/o WS | 92.95 | 82.52 | 86.49 | 69.13 |
| w/o PG | 90.59 | 80.69 | 86.19 | 65.65 |

Table 5: Online A/B Test.

| Apps | Cold-start | | |
|---|---|---|---|
| | A (old EMG-RAG) | B (new EMG-RAG) | Impr |
| QA | 88.06 | **91.99** | 4.5% |
| AF | 92.86 | **95.85** | 3.2% |
| US | 94.66 | **97.56** | 3.1% |

for events and times (reminders), and addresses (travel) in Table 2. The improvements over M-RAG for the two tasks are 2.9% and 5.5%.

**(4) Effectiveness evaluation (continuous edits).** We evaluate the effectiveness of EMG-RAG in supporting continuous edits over a period of 4 weeks. The results, in terms of R-L for question answering (QA), and exact match accuracy for autofill forms (AF) and user services (US, combining reminder and travel results), are presented in Table 6. We observe that EMG-RAG consistently outperforms M-RAG, by approximately 10.6%, 9.5%, and 9.7% for QA, AF, and US, respectively. This is owing to the editability of EMG-RAG, whereas M-RAG simply incorporates edits into a database, where many memories may become outdated for answering. Additionally, we report the total number of edits involved in the testing set for each week.

**(5) Ablation study.** To evaluate the effectiveness of different components in EMG-RAG, we conduct an ablation study. (1) We omit the design of activated nodes, and the search starts from the root of EMG. (2) We remove the warm-start stage (WS) and only train the policy in the policy gradient stage (PG). (3) We remove the PG and use the WS only. For (1), it results in a performance drop (e.g., R-1 from 93.46 to 90.96), because many irrelevant memories (as noises) may be retrieved if the search starts from the root. For (2) and (3), we observe that the PG contributes the most to the result (e.g., R-1 from 93.46 to 90.59), because it can explicitly optimize the performance end-to-end, and WS provides a basic memory selection ability for the agent.

**(6) Parameter study ($K$ for activated nodes).** We evaluate the effect of $K$, which controls the number of nodes activated during graph traversal. The results and analysis are presented in Appendix A.5. Overall, a moderate setting of $K = 3$ provides the best balance of effectiveness and inference time.

**(7) Online A/B test.** We perform an online A/B test to compare the new system with the old system for one month. During this period, we collect real users' questions and manually written answers to fine-tune the model as introduced in Section 4.4. The results, presented in Table 5, demonstrate further improvements across all applications.

## 6 Conclusion

In this paper, we present a novel task of creating personalized agents powered by LLMs, which leverage users' personal memories to enhance three downstream applications. Our solution, EMG-RAG, combines RAG techniques with an EMG to tackle challenges in data collection, editability, and selectability. Extensive experiments are conducted to confirm the effectiveness of EMG-RAG.

## 7 Limitations

For limitations, while only the parameters of the RL agent are trained and the parameters of the LLMs remain fixed, the training efficiency is not higher than that of a Naive RAG setup. This inefficiency stems from the need to query the LLM during training to obtain answers for optimization.

## References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *NeurIPS*, 26.

Eleftheria Briakou, Colin Cherry, and George Foster. 2023. Searching for needles in a haystack: On the role of incidental bilingualism in palm's translation capability. *arXiv preprint arXiv:2305.10266*.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew Mccallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *EMNLP*, pages 9594–9611.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *EMNLP*, pages 6491–6506.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. In *ACL*, pages 320–335.

Dongge Han, Trevor McInroe, Adam Jelley, Stefano V Albrecht, Peter Bell, and Amos Storkey. 2024. Llm-personalize: Aligning llm planners with human preferences via reinforced self-training for housekeeping robots. *arXiv preprint arXiv:2404.14285*.

WANG Hongru, Rui Wang, Fei Mi, Yang Deng, WANG Zezhong, Bin Liang, Ruifeng Xu, and Kam-Fai Wong. 2023. Cue-cot: Chain-of-thought prompting for responding to in-depth dialogue questions with llms. In *EMNLP (Findings)*, pages 12047–12064.

Ziniu Hu, Ahmet Iscen, Chen Sun, Zirui Wang, Kai-Wei Chang, Yizhou Sun, Cordelia Schmid, David A Ross, and Alireza Fathi. 2023. Reveal: Retrieval-augmented visual-language pre-training with multi-source multimodal knowledge memory. In *CVPR*, pages 23369–23379.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Hyunwoo Kim, Byeongchang Kim, and Gunhee Kim. 2020. Will i sound like me? improving persona consistency in dialogues through pragmatic self-consciousness. In *EMNLP*, pages 904–916.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *NeurIPS*, 33:9459–9474.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024a. Pmet: Precise model editing in a transformer. In *AAAI*, volume 38, pages 18564–18572.

Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. 2024b. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Qian Liu, Yihong Chen, Bei Chen, Jian-Guang Lou, Zixuan Chen, Bin Zhou, and Dongmei Zhang. 2020. You impress me: Dialogue generation via mutual persona perception. In *ACL*, pages 1417–1427.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *EMNLP*, pages 5303–5315.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *NeurIPS*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *ICML*, pages 15817–15831. PMLR.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pre-training. *CoRR*.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *WMT*, pages 186–191.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*, pages 784–789.

Xiaozhe Ren, Pingyi Zhou, Xinfan Meng, Xinjing Huang, Yadao Wang, Weichao Wang, Pengfei Li, Xiaoda Zhang, Alexander Podolskiy, Grigory Arshinov, et al. 2023. Pangu-{\Sigma}: Towards trillion parameter language model with sparse heterogeneous computing. *arXiv preprint arXiv:2303.10845*.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. Tiara: Multi-grained retrieval for robust question answering over large knowledge base. In *EMNLP*, pages 8108–8121.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *ICML*, pages 387–395. PMLR.

Haoyu Song, Yan Wang, Kaiyan Zhang, Weinan Zhang, and Ting Liu. 2021. Bob: Bert over bert for training persona-based dialogue models from limited personalized data. In *ACL*, pages 167–177.

Chaojie Wang, Yishi Xu, Zhong Peng, Chenxi Zhang, Bo Chen, Xinrun Wang, Lei Feng, and Bo An. 2023a. keqing: knowledge-based question answering is a nature chain-of-thought mentor of llm. *arXiv preprint arXiv:2401.00426*.

Hongru Wang, Minda Hu, Yang Deng, Rui Wang, Fei Mi, Weichao Wang, Yasheng Wang, Wai Chung Kwan, Irwin King, and Kam-Fai Wong. 2023b. Large language models as source planner for personalized knowledge-grounded dialogues. In *EMNLP (Findings)*, pages 9556–9569.

Zheng Wang, Shu Xian Teo, Jieer Ouyang, Yongjun Xu, and Wei Shi. 2024. M-RAG: Reinforcing large language model performance through retrieval-augmented generation with multiple partitions. In *ACL*.

Charles Welch, Chenxi Gu, Jonathan K Kummerfeld, Verónica Pérez-Rosas, and Rada Mihalcea. 2022. Leveraging similar users for personalized language modeling with limited data. In *ACL*, pages 1742–1752.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Chen Xu, Piji Li, Wei Wang, Haoran Yang, Siyun Wang, and Chuangbai Xiao. 2022a. Cosplay: Concept set guided personalized dialogue generation across both party personas. In *SIGIR*, pages 201–211.

Xinchao Xu, Zhibin Gou, Wenquan Wu, Zheng-Yu Niu, Hua Wu, Haifeng Wang, and Shihang Wang. 2022b. Long time no see! open-domain conversation with long-term persona memory. In *ACL (Findings)*, pages 2639–2650.

Qian Yang, Qian Chen, Wen Wang, Baotian Hu, and Min Zhang. 2023. Enhancing multi-modal multi-hop question answering via structured knowledge and unified retrieval-generation. In *ACM MM*, pages 5223–5234.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2021. Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering. *arXiv preprint arXiv:2109.08678*.

Houyu Zhang, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2020. Grounded conversation generation as guided traverses in commonsense knowledge graphs. In *ACL*, pages 2031–2043.

Kai Zhang, Fubang Zhao, Yangyang Kang, and Xiaozhong Liu. 2023. Memory-augmented llm personalization with short-and long-term memory coordination. *arXiv preprint arXiv:2309.11696*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*, pages 2204–2213.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

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

## A Appendix

### A.1 Data Collection Details

The data collection process involves three key steps, which are presented below:

**Step-1: Raw Data Collection.** We explore two approaches, termed Active Remember (AR) and Passive Remember (PR), for collecting raw data derived from users' daily conversations with AI assistants and screenshots from their apps. With AR, the AI assistant is trained to actively classify data (such as conversation sentences) into supported subclasses outlined in Table 7, and filter out noise data. With PR, users have the option to directly let the assistant to remember specific content for future use. Leveraging AR and PR, we remove a significant volume of trivial data, and then extract memories from the refined dataset.

**Step-2: Memory Data Construction.** We utilize a LLM, such as GPT-4, with the refined dataset to generate structured memories from the raw data. Additionally, we integrate various natural language processing techniques, including absolute date and time conversion, entity anaphora resolution, and event coreference resolution, to further clean the memories and facilitate graph construction.

**Step-3: QA Pairs Construction.** We organize the memory data chronologically and partition it into separate conversation sessions. Then, a LLM generates QA pairs for each session. To create complex questions for targeted training, such as those requiring multiple memories for answering, we explicitly instruct the LLM to utilize multiple associative relationships between memories to generate questions, ensuring that at least one or more memories are needed for accurate responses.

## A.2 Memory Types and Subclasses

We describe the 4 memory types: (1) Relationship, which involves recognizing users' surrounding relationships and attributes of related individuals, such as birthdays and names of family members; (2) Preference, where we identify users' likes and dislikes for various topics or entities; (3) Event, focusing on key event information about users, such as their status, recent experiences, and upcoming schedules; and (4) Attribute, encompassing users' personal details such as name, gender, age, possessions, and other relevant information.

We enumerate the supported business subclasses of the EMG with memory examples in Table 7.

## A.3 Baseline Details

We compare `EMG-RAG` with the following RAG methods, and the details are presented below:

- NiaH (Briakou et al., 2023): It simply inputs all of the users' memories into a LLM within the context window size to generate the answer.

- Naive (Ma et al., 2023): It implements a basic RAG execution process involving indexing, retrieval, and generation.

- M-RAG (Wang et al., 2024): It partitions a database into different partitions, and employs Multi-Agent RL to train two agents for conducting RAG. One agent (Agent-S) learns to select a database partition, while the other agent (Agent-R) refines the stored memories within the partition to generate a better answer. We adapt the approach by omitting Agent-R, as in our scenario, the generated answers should be grounded in the user's personal memories, which cannot be altered due to potential risks.

Table 6: Impacts of the number of $K$ for activated nodes.

| $K$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| R-L | 84.55 | 86.06 | **88.06** | 88.06 | 87.19 |
| Inference (s) | 1.35 | 1.63 | **2.14** | 2.55 | 3.32 |

- Keqing (Wang et al., 2023a): It is a recent RAG method based on knowledge graphs. It begins by decomposing a question into various sub-questions and retrieving candidate entities (associated with memories) from the knowledge graph for each sub-question. Next, it generates an answer for each sub-question and integrates them into an overall answer.

In addition, we integrate the RAG methods into three typical LLM architectures. 1) GPT-4 (OpenAI, 2023) is a Transformer-based pre-trained model known for its human-level performance. 2) ChatGLM3-6B (Du et al., 2022) is a long-text dialogue model with a sequence length of 32K. 3) PanGu-38B (Ren et al., 2023) is a dialogue submodel of the PanGu series, which follows a Mixture of Experts (MoE) architecture.

## A.4 Implementation Details

We implement `EMG-RAG` and other baselines in Python 3.7, using the Faiss library [2] for retrieval index construction. We utilize TransE (Bordes et al., 2013) to obtain embeddings of entities and relations, and CPT-Text (Neelakantan et al., 2022) to obtain embeddings of questions and memories. The RL agent is implemented with a two-layer neural network, where the hidden layer consists of 20 neurons and uses the tanh activation function. The output layer has 2 neurons corresponding to the action space. The hyperparameter $K$ for activated nodes is empirically set to 3. We generate 1,000 episodes for the warm-start stage (WS) and 100 episodes for the policy gradient stage (PG). We use the Adam stochastic gradient descent with a learning rate of 0.001 to optimize the policy, and the reward discount is set to 0.99. Additionally, we cache the generated QA pairs [3] during training to boost efficiency.

## A.5 Parameter Study

We vary the value of $K$ from 1 to 5 and report the R-L score for the question answering task, along with the corresponding inference times. As shown

[2]https://github.com/facebookresearch/faiss
[3]https://github.com/zilliztech/GPTCache

Table 7: The supported memory subclasses with memory examples.

| Memory Types | Memory Subclasses | Memory Examples |
|---|---|---|
| Relationship | Spouse | Tomorrow is my mom's birthday. |
| | Parents/Children | |
| | Relatives | |
| | Colleague/Friends | |
| | Teacher/Student | |
| Preference | Diet preference | I like spicy food. |
| | Cultural preference (tourism, travel) | I enjoy traveling by airplane.<br>I like going to museums. |
| | Car preference | I like BMWs. |
| | Sports preference<br>(favorite sports types, sports celebrities) | I like playing table tennis on weekends.<br>James is my favorite basketball star. |
| | Gaming preference (category, name) | I like the game League of Legends. |
| | Audio-visual entertainment preference<br>(favorite videos, music, movies, TV shows) | I like science fiction movies.<br>I like listening to Jay Chou's songs. |
| Event | Life events<br>(academic, marriage, buying a flat, parenting) | The college entrance examination is coming soon.<br>I met a girlfriend online.<br>My family is welcoming a second child. |
| | Arrangement | I'm going to visit clients tomorrow.<br>I want to travel to Amsterdam next month.<br>I have an oral defense next Monday. |
| | Anniversary | Next month's fifth is our wedding anniversary. |
| Attribute | Name/Nickname | My name is Wang Xiaoming, call me Lord Radish. |
| | Birthday/Age | I am 17 years old this year.<br>I was born in 1998.<br>My birthday is April 2nd. |
| | Gender | I am a girl. |
| | Education | I am an undergraduate student. |
| | Personal belongings/Pets | Riding my beloved electric scooter, my pink BMW. |
| | Address | I reside in Jurong West, Singapore. |
| | Occupation | I am a research scientist. |

in Table 6, we observe that $K = 3$ provides the best effectiveness while maintaining reasonable inference time. When $K$ is smaller, the limited number of activated nodes for graph traversal restricts the ability to find crucial memories. Conversely, when $K$ is larger, it activates many nodes and returns numerous memories, potentially introducing noise that hinders the LLM generation. As expected, the inference time increases as $K$ increases.