Personalized Large-Language-Model Assistant with Long-Term User Memory via Retrieval-Augmented Generation

Ali Mohammadi
1380ali.mohamadi
@gmail.com

Mahdi Cheraghi
pmahdicheraghi
@gmail.com

Mostafa Ebrahimi
mostafa.ebrahimi2002
@gmail.com

Abstract

Many contemporary Large Language Models (LLMs) lack persistent memory of individual users, leading to impersonal and repetitive interactions. This project addresses this shortcoming by developing a personalized LLM assistant that leverages user-specific knowledge bases (KBs) to simulate long-term memory. We implement a Retrieval-Augmented Generation (RAG) system where user facts and preferences are dynamically extracted from dialogue and stored in a FAISS vector index. During inference, relevant facts are retrieved in real-time to augment the prompt of a large language model, enabling personalization and context continuity without costly model retraining. We evaluate our RAG-based model against stateless and short-term memory baselines on two fronts: a custom 50-turn conversational dataset designed to test long-term recall, and the public PersonaChat dataset. Our results consistently demonstrate a significant improvement in response quality. On our challenging custom dataset, the RAG model achieved an average score of 7.58/10, compared to 6.58/10 for a buffer-memory baseline. This advantage was confirmed on the PersonaChat dataset, where our model scored 8.44/10.

1 Introduction

Conversational agents powered by Large Language Models (LLMs) have become increasingly sophisticated, yet a fundamental limitation persists: their inability to retain information about a user across different interactions. This ”amnesiac” nature forces users to repeat themselves, leading to interactions that feel transactional and impersonal rather than collaborative and evolving. A truly helpful assistant should remember key details—a user’s profession, their preferences, important life events, or even a casual mention of a new hobby.

This project tackles this challenge by designing and implementing a personalized LLM assistant with a robust, long-term memory. Our central hypothesis is that by equipping a conversational agent with a dynamic, user-specific knowledge base (KB) and leveraging Retrieval-Augmented Generation (RAG), we can significantly enhance personalization, contextual continuity, and overall user experience without the prohibitive cost and complexity of continuously fine-tuning the underlying LLM.

We make the following contributions:

1. We design and implement a modular, RAG-based architecture for long-term conversational memory, featuring dynamic fact extraction and a scalable vector store using FAISS.

2. We create a novel, 50-turn conversational dataset specifically designed to evaluate short-term, long-term, and cross-session memory recall in a controlled manner.

3. We conduct a rigorous evaluation of our RAG model against stateless and buffer-based baselines using an LLM-as-a-Judge, demonstrating a marked improvement in personalization and coherence on both our custom dataset and the public PersonaChat benchmark.

4. We perform a detailed error analysis that highlights the specific conversational contexts where RAG-based memory excels (long-term factual recall) and where it struggles (abstract creative tasks), providing insights for future research.

This paper is structured as follows: Section 2 reviews related work in personalized dialogue and RAG. Section 3 details our system architecture and its modules. Section 4 describes our experimental setup, including datasets and evaluation methodology. Section 5 presents our quantitative results and qualitative error analysis. Finally, Section 6 concludes with our key takeaways and directions for future work.

2 Related Work

The pursuit of personalized LLMs with long-term memory integrates several research domains: personalized dialogue systems, retrieval-augmented generation, vector databases, and evaluation methodologies.

2.1 Personalized Dialogue Systems

Early work on personalized dialogue often relied on explicit user profiles with predefined slots (Zhang et al., 2018). The PersonaChat dataset, which we use for evaluation, emerged from this paradigm, providing explicit persona sentences for models to condition on. More recent work has focused on creating more dynamic and implicit memory structures. For example, the LD-Agent (Lee et al., 2025) utilizes a modular framework with memory banks and explicit persona extraction modules to demonstrate the value of structured memory. Going a step further, the Reflective Memory Model (RMM) (Li et al., 2025) employs dynamic memory summarization and adaptive retrieval, using reinforcement learning to decide when and what to store, showcasing advanced memory management. Other approaches, such as Memory-augmented Dialogue Management (**?**), have explored using memory networks to store conversational history in a more structured way than a simple context window. Our project adopts a more direct RAG approach with discrete, unstructured facts, which simplifies the memory structure while still providing powerful retrieval capabilities.

2.2 Retrieval-Augmented Generation

The RAG framework, formalized by Lewis et al. (2020), is central to our work. It grounds LLM generation in external knowledge, which has been shown to reduce hallucinations and improve factual accuracy in open-domain question answering. In our case, the ”external knowledge” is the user’s own history. The core idea has been extended in many ways. For instance, Wang et al. (2025) explored RAG for long-document understanding, highlighting challenges in managing potentially conflicting information within a large retrieved context. This work underscores the importance of careful KB management, a consideration we acknowledge for future work. Projects like REALM (**?**) have shown how to pre-train the retriever and generator jointly, though our work focuses on a more modular, post-hoc application of RAG to a pre-trained LLM.

The technical backbone of our retriever relies on two key technologies. First, dense vector embeddings generated by sentence-transformers (Reimers and Gurevych, 2019), which are optimized to produce semantically meaningful sentence representations. Second, efficient vector search provided by libraries like FAISS (Johnson et al., 2019), which allows for billion-scale similarity search and is crucial for any system intended to scale to many users with large KBs. The entire system is built using modern ML frameworks like PyTorch (Paszke et al., 2019) and the Hugging Face Transformers library (Wolf et al., 2019).

2.3 Evaluation of Personalization

Evaluating the quality of a personalized agent is non-trivial. It goes beyond simple accuracy to include subjective user experience. Recent work has sought to formalize this. Kim et al. (2025) introduced PREFEVAL, a benchmark for evaluating how well LLMs follow explicitly stated user preferences, which inspired our qualitative assessment. For assessing factual consistency, the FaaF framework (Wei et al., 2024) offers an automated, function-calling-based method for validating facts in RAG outputs. This informed the design of our ‘evaluate-with-llm-judge‘ module, where we use an LLM to check if a response correctly uses or contradicts facts from the KB. The ”LLM-as-a-Judge” paradigm itself has gained traction as a scalable method for evaluating generative models (**?**).

Finally, any system that stores user data must consider privacy. While we did not fully implement a privacy evaluation, the principles from frameworks like LLM-PBE (Li et al., 2024), which assesses privacy risks through targeted probes, guided our initial thinking on potential data leakage vulnerabilities.

3 System Architecture

Our approach is a modular RAG system designed for persistent, user-specific memory. It comprises several key components that work in concert to create a seamless, memory-driven conversational experience. All experiments were conducted using ‘gpt-5-nano‘ as the backend LLM, accessed via our client. The architecture is depicted in Figure

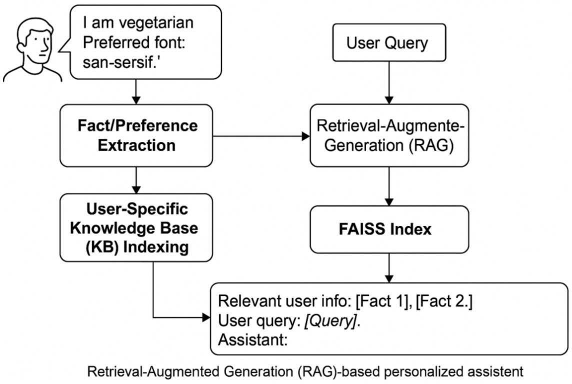
I am vegetarian
Preferred font: san-sersif.'
User Query
Fact/Preference Extraction
Retrieval-Augmente-Generation (RAG)
User-Specific Knowledge Base (KB) Indexing
FAISS Index
Relevant user info: [Fact 1], [Fact 2.]
User query: *[Query]*.
Assistant:
Retrieval-Augmented Generation (RAG)-based personalized assistent

Figure 1: System architecture. User input is analyzed to extract and store facts in a FAISS KB. For generation, queries are used to retrieve relevant facts from the KB, which augment the LLM prompt to produce a personalized response.

3.1 Unified LLM Client

To ensure flexibility and model-agnosticism, we first developed ‘LLMClient‘. This class acts as a unified wrapper for various LLM providers, including Hugging Face, OpenAI, and Google. It abstracts the provider-specific API calls into a single ‘chat-completion‘ method. This design allows us to easily switch the backend LLM (e.g., from Llama 3 to GPT models) without altering the core application logic, simply by changing a configuration variable.

3.2 User Memory Module

The core of our long-term memory system is the ‘UserMemoryModule‘. This module is responsible for storing and retrieving all facts related to a specific user.

- **Vectorization**: It uses a pre-trained ‘sentence-transformers/all-MiniLM-L6-v2‘ model to convert textual facts (e.g., ”My favorite color is sunset orange”) into 384-dimensional dense vector embeddings.

- **Indexing with FAISS**: These embeddings are stored in a FAISS (Johnson et al., 2019) index. FAISS allows for extremely fast nearest-neighbor search, making it ideal for retrieving semantically similar facts in real-time.

- **Scalable Indexing**: A key feature is its ability to scale. The module initially uses a simple ‘IndexFlatIP‘ (exact search), which is efficient for a small number of facts. Once the number of stored facts exceeds a threshold (1024 in our implementation), the module automatically upgrades the index to a more scalable ‘IndexIVFFlat‘. This involves training the new index on the existing vectors and re-indexing them, a process that happens seamlessly in the background. This ensures the system remains performant as a user’s memory grows over time.

- **Retrieval**: The ‘retrieve-memory‘ function takes a string query, embeds it using the same sentence transformer, and performs a similarity search on the FAISS index to return the top-k most relevant facts.

3.3 RAG Core and Dialogue Manager

The ‘DialogueManager‘ is the brain of the chatbot, orchestrating the conversation flow.

- **RAG Pipeline**: It contains a ‘RAG-Core‘ component which, for every user query, retrieves relevant memories from the ‘UserMemoryModule‘. It then constructs an augmented prompt by prepending these memories to the conversation history and the current query. A typical augmented prompt looks like this:

  “‘ ‘You are a personalized AI. Use these user Persona: - I am a freelance graphic designer. - My partner, Jamie, is a huge history buff.

  And this history: User: Can you suggest a travel destination?

  User Query: Which one is good for historical sites¿

- **Dynamic Memory Extraction**: After generating a response, the ‘DialogueManager‘ calls a helper method, ‘-extract-new-memory‘. This method sends the user’s last input to the LLM with a highly specific instruction, shown in Listing **??**, to determine if a new, salient personal fact has been revealed. If the LLM returns a fact, it is added to the ‘UserMemoryModule‘; otherwise, the memory remains unchanged. This creates a continuous learning loop where the assistant’s knowledge about the user deepens with every interaction. “‘

```
prompt = f"""Analyze the user’s statement. If it reveals a new personal fact (preference, detail, identity), state it concisely in the first person (e.g., "I live in Switzerland."). Otherwise, respond with ONLY the word "NO\_FACT".
User statement: "{user\_input}"
New fact:"""
```

```
extracted\_text = llm\_client.chat\
    _completion(prompt)
if "NO\_FACT" not in extracted\_text:
umm.add\_memory([extracted\_text])
```

Listing 1: Fact Extraction Prompt Logic

4 Experimental Setup

4.1 Datasets

We utilized two distinct datasets to evaluate our system: a custom-designed long-form dialogue for in-depth analysis and the public PersonaChat dataset for broader validation.

4.1.1 Custom ”Alex” Dataset

Our primary evaluation was conducted on a custom, synthetic dataset. The dataset features a single, detailed persona named ”Alex,” a 32-year-old freelance graphic designer. The core of the dataset is a 50-turn monologue from Alex, structured to simulate multiple conversations over time. This design makes the task challenging by requiring the model to distinguish between short-term context, information from a ”previous session,” and facts established at the very beginning of the interaction.

4.1.2 PersonaChat Dataset

For external validation, we also evaluated our models on the PersonaChat dataset (Zhang et al., 2018). This public dataset consists of dialogues where each speaker is assigned a short persona composed of 4-5 facts. We used this to test our model's ability to maintain personalization in more typical, shorter conversational settings. For this evaluation, we ran our models on 10 randomly selected dialogues from the dataset. A comparison of the dataset statistics is in Table 1.

Table 1: Statistics of Datasets Used for Evaluation

Property	Custom ”Alex” Dataset	PersonaChat
Source	Authored for Project	Zhang et al. (2018)
Scope	In-depth, single persona	Broad, multi-persona
Size	1 dialogue, 50 turns	10 dialogues (sampled)
Task Focus	Long-term memory test	General personalization

4.2 Models and Baselines

We evaluated three models on every turn of the dialogues. The backend for all generative tasks was the ‘gpt-5-nano‘ model, as specified in our code's configuration.

1. **Our RAG Model**: The full system described in Section 3.
2. **Stateless Model**: A standard LLM call with a generic system prompt and no access to history or persona facts.
3. **Buffer Memory Model**: An LLM call where the prompt is augmented with the last $k = 4$ turns of conversation history.

4.3 Evaluation Methodology: LLM-as-a-Judge

To score the quality of each model's response, we employed an LLM-as-a-Judge. For every generated response, a separate call was made to the ‘gpt-5-nano‘ model with a detailed evaluation prompt. This prompt provided the judge with the full context (user persona, conversation history, user query) and the assistant's response, asking it to score the response based on a specific rubric.

The core instruction to the judge was:

```
You are a strict AI evaluator. Score an assistant’s re
CONTEXT:
  - User Persona: {persona}
  - History: {history}
  TASK:
  - User Query: "{query}"
  - Assistant’s Response: "{response}"
  RUBRIC:
  - 1-3: Contradicts persona or is irrelevant.
  - 4-6: Generic, ignores persona.
  - 7-8: Coherent and consistent.
  - 9-10: Masterfully uses persona for a tailored resp
  Provide your evaluation as a JSON object with "score
```

This method provided a scalable way to get consistent, rubric-based scores for thousands of generated responses, along with qualitative justifications for each score.

5 Results and Analysis

Our RAG-based approach significantly outperformed both baselines across both datasets, demonstrating its effectiveness in creating personalized and contextually coherent conversations.

5.1 Quantitative Results

On the challenging 50-turn custom dataset, designed to stress-test long-term memory, our model achieved a full point advantage over the next best baseline, as shown in Table 2.

This superiority was confirmed on the PersonaChat dataset (Table 3), where our RAG model again achieved the highest score. While the performance gap was smaller, likely due to the shorter and less complex nature of PersonaChat dialogues, the RAG model's ability to consistently leverage the full persona context still provided a distinct

Table 2: Average personalization and coherence scores on the custom ”Alex” dataset (50 turns).

Model	Average Score (/10)
Stateless Model	6.32
Buffer Memory (k=4)	6.58
Our RAG Model	**7.58**

advantage over the limited context of the buffer model and the no-context stateless model.

Table 3: Average personalization and coherence scores on the PersonaChat dataset (10 samples).

Model	Average Score (/10)
Stateless Model	7.35
Buffer Memory (k=4)	7.81
Our RAG Model	**8.44**

5.2 Qualitative Error Analysis

A detailed analysis of the LLM-as-a-Judge’s scores on our more challenging custom dataset reveals clear patterns in the performance of our RAG model versus the baselines.

5.2.1 Where RAG Excels: Long-Term Recall

The most significant advantage of the RAG model was in long-term memory recall. Table 4 shows a side-by-side comparison for a query in Turn 46, which required recalling information from Turn 38. The RAG model flawlessly retrieved the fact, while the Buffer Model, whose context window had long since passed Turn 38, failed completely. The judge scored the RAG model 6 points higher. This pattern was consistent across all long-term recall tasks.

5.2.2 Where RAG Struggles: Abstract and Creative Tasks

We also analyzed the turns where the RAG model received its lowest scores. These failures clustered around abstract or highly creative tasks where direct factual retrieval is less beneficial. For instance, in Turn 22, the user asked: “Explain the concept of black holes as if you were telling a bedtime story.”

All three models produced a coherent, story-like explanation. However, the judge gave our RAG model a low score of 5/10, with the reasoning: Coherent and kid-friendly... However it does not personalize for ’Alex’ or reference their goals... It’s generic rather than tailored to the persona.

This reveals a key limitation: our system is optimized to retrieve and inject *facts*, but these facts often have low semantic similarity to abstract or creative queries. When no relevant facts are retrieved, the RAG model’s performance regresses toward that of the stateless baseline, as it has no specific context to ground its creative output.

6 Conclusion

This project successfully demonstrated that a RAG-based architecture can endow an LLM with effective long-term memory, leading to more personalized and coherent conversational experiences. Our model, which dynamically learns and retrieves user-specific facts, achieved significantly higher scores on both a challenging custom dataset (7.58/10) and a public benchmark (8.44/10) compared to stateless and short-term memory baselines.

The most surprising and clear result was the stark difference in performance on long-term recall tasks, where our RAG system flawlessly retrieved facts from many turns prior. The most difficult aspect was designing a synthetic dataset that could fairly and rigorously test different facets of memory over a simulated long-term interaction.

Our results were encouraging, but the error analysis also revealed a key limitation: the system struggles to personalize responses for abstract or creative queries where factual recall is less relevant. If we were to continue this project, future directions would include:

1. **Improving Creative Personalization**: A significant challenge is personalizing stylistic or creative responses. Future work could explore methods to retrieve and use more abstract preferences (e.g., ”I like a witty tone,” ”I prefer minimalist design”). This might involve a secondary memory store for stylistic traits or developing a retriever that can match a query’s abstract intent to a user’s personality profile.
2. **Advanced Memory Management**: A simple, growing list of facts is prone to becoming outdated or contradictory. A more advanced system would need mechanisms for fact verification, contradiction resolution (e.g., if a user changes their preference), and memory summarization. Techniques from Li et al. (2025) involving reflective summarization could be adapted to periodically condense and refine the user’s KB.
3. **Privacy Controls**: The storage of personal

Table 4: Example of RAG success on a long-term recall task (Turn 46).

Model	Response	Score	Judge's Reasoning
Our RAG Model	The theme was "cosmic comfort" for the coffee shop logo. Want me to flesh out three concrete logo concepts in that vibe (including crescent-moon ideas and a sunset-orange palette)...	**9/10**	The assistant accurately recalls the theme 'cosmic comfort' and leverages the user's preferences (sunset orange color and crescent-moon motif) to propose tailored next steps.
Buffer Memory	I don't have a record of a coffee shop logo theme from this chat. Could you repeat the theme...? If you'd like ideas, here are common logo themes for coffee shops: Cozy/Homely...	**3/10**	The assistant fails to recall the user's previously stated theme 'cosmic comfort' for the coffee shop logo, effectively ignoring the memory-testing objective.

data is a critical concern. A production-ready version of this system would require robust privacy controls. This includes implementing a user-facing interface to allow users to view, edit, or selectively delete stored memories, ensuring transparency and user agency over their data.

4. **Hybrid Memory Systems**: Our system relies on a single vector store. A hybrid approach could be more powerful, combining the semantic retrieval of a vector KB with a structured, symbolic KB (e.g., a graph database) for storing crisp, relational facts like family connections or key dates. This could improve the precision of certain types of recall.

Overall, this project provides a strong proof-of-concept for building more capable and personable AI assistants through a lightweight, retrieval-based approach to memory.

References

Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with gpus. IEEE.

Kim, S.-M. et al. (2025). Do llms follow your preferences? a benchmark for preference following. In *NAACL*.

Lee, J.-H. et al. (2025). Hello, it's me: A new framework for personalized dialogue agents. In *ACL*.

Lewis, P. et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*.

Li, J.-C. et al. (2025). A reflective memory model for personalized dialogue. In *EMNLP*.

Li, T.-X. et al. (2024). Llm-pbe: A new benchmark for privacy analysis of language models. In *ACL*.

Paszke, A. et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Wang, L. et al. (2025). Retrieval-augmented generation with memory for long documents. In *ICLR*.

Wei, C. et al. (2024). Facts as functions: A new framework for factual knowledge probing. In *ACL*.

Wolf, T. et al. (2019). Huggingface's transformers: State-of-the-art natural language processing. In *EMNLP*.

Zhang, S. et al. (2018). Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*.