MemGPT
(LLM for training K-12 Teachers)

In Association with Blendnet.ai and Simple Education Foundation

-By Palak Bhardwaj and Yashovardhan Singhal

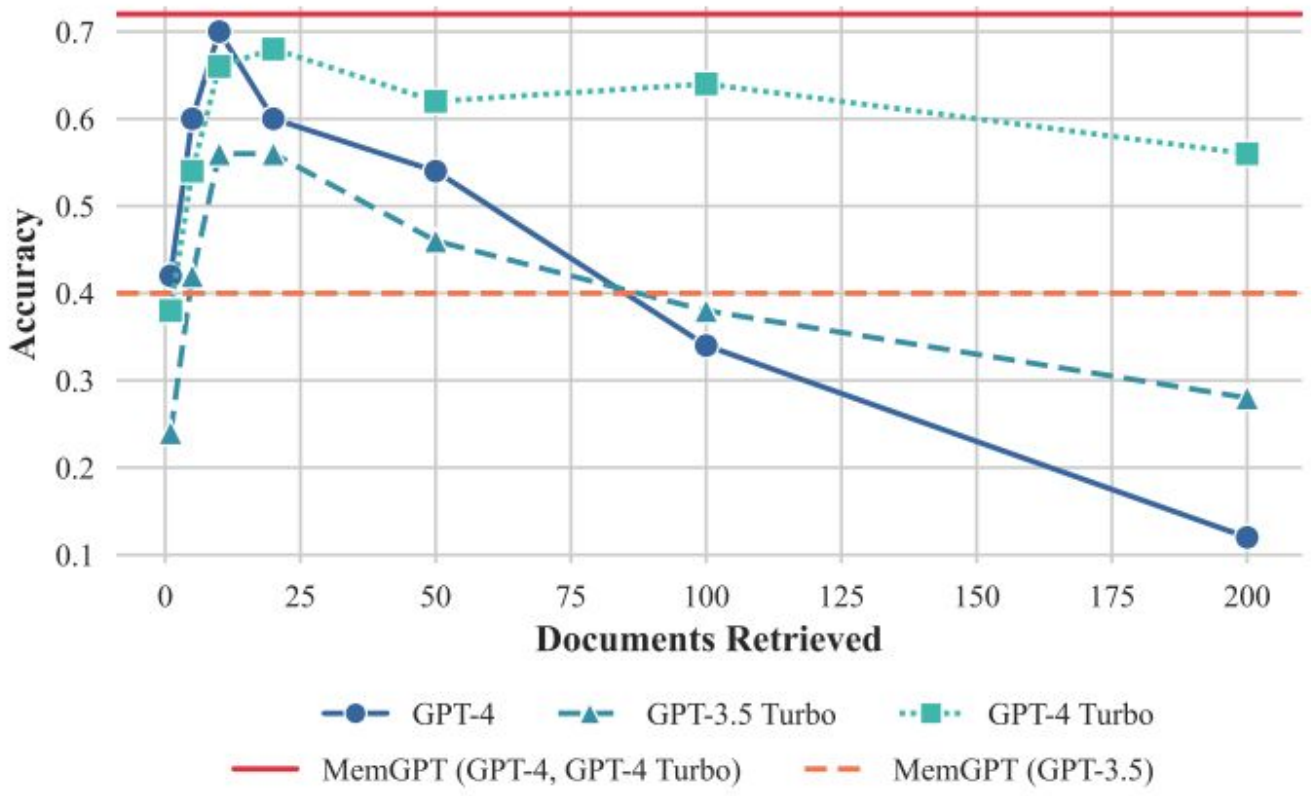
INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Shortcomings in Existing LLMS

- **Limited Context Windows:** - Hampers performance

- **Solution** - Simply increased context windows ?
- Increase in computational time and memory.
- Uneven attention distribution in large context models.
- Truncation/ Summarisation can extend the effective context lengths of fixed length models such as GPT-4, but such compression methods will lead to performance degradation as the necessary compression grows.

Context length and Performance



Model / API name	Open?	Context Window Tokens	*Messages
Llama (1)	✓	2k	20
Llama 2	✓	4k	60
GPT-3.5 Turbo (release)	✗	4k	60
Mistral 7B	✓	8k	140
GPT-4 (release)	✗	8k	140
GPT-3.5 Turbo	✗	16k	300
GPT-4	✗	32k	~600
Claude 2	✗	100k	~2000
GPT-4 Turbo	✗	128k	~2600
Yi-34B-200k	✓	200k	~4000

Comparison of Context tokens and window messages of various LLMs

Link

Memgpt <-> Memory GPT

- LLM + OS implements virtual context management.
- Provides extended context within the LLM's context window.
- Manages memory via function calls.
- MemGPT creates an illusion of infinite context length while using fixed context models, addressing the limitations of simply increasing context windows.
- Provides the appearance of large memory resources through data movement between main memory and disk.
- Memory hierarchy created analogous to OS.
- MemGPT performance unaffected by increased context length as compared to existing LLM's.
- Ensures persona consistency, and long term memory.
- Increase in context window leads to increase in conversational chat consistency.

Basic Mechanism

- Divides memory into parts - Main and External - analogous to traditional OS systems.
- Enables to analyze large documents and manage long term chat sessions.
- Main Context or RAM memory consist of LLM prompt tokens.
- Anything in main context is considered in-context and can be accessed by the LLM processor during inference.
- External context refers to any information that is held outside of the LLMs fixed context window and has to be moved into main context to be passed into LLM processor for inference.
- External memory is accessed by using functional calls.

Main Context

- Prompt tokens divided in 3 sections
- System instructions, Working context and FIFO Queue.
- System instructions contain info on Memgpt control flow and how to use its functions.
- FIFO queue stores a rolling history of messages, including messages between the agent and user.
- Working context is intended to be used to store key facts, preferences, and other important information about the user and the persona the agent is adopting, allowing the agent to converse fluently with the user. It written by MemGPT function calls.
- Messages are managed in recall storage and FIFO queue.
- The incoming message and LLM output is written into recall storage.
- When these are retrieved from recall storage they are appended to the back of the queue to be reinserted into context window.

Scenarios

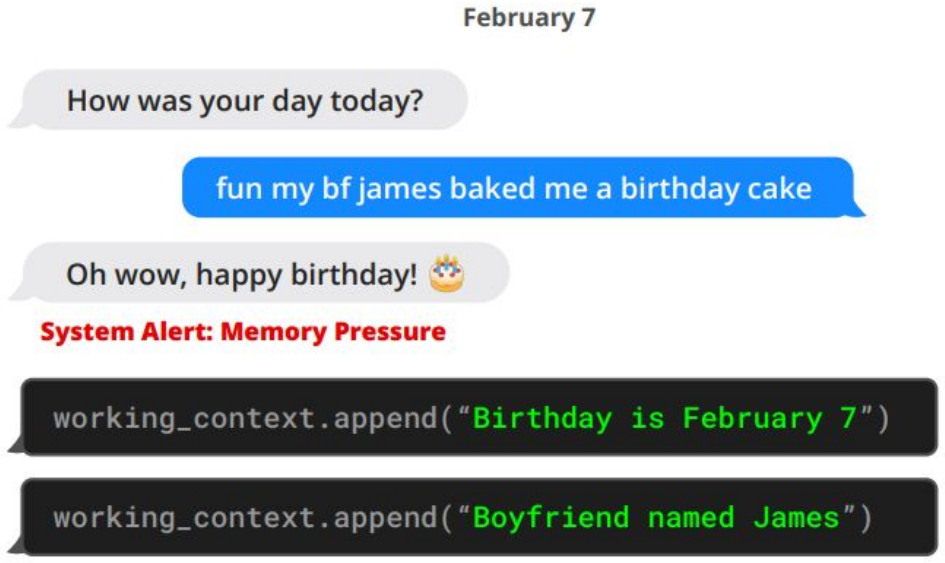


Figure 1. MemGPT (left) writes data to persistent memory after it receives a system alert about limited context space.

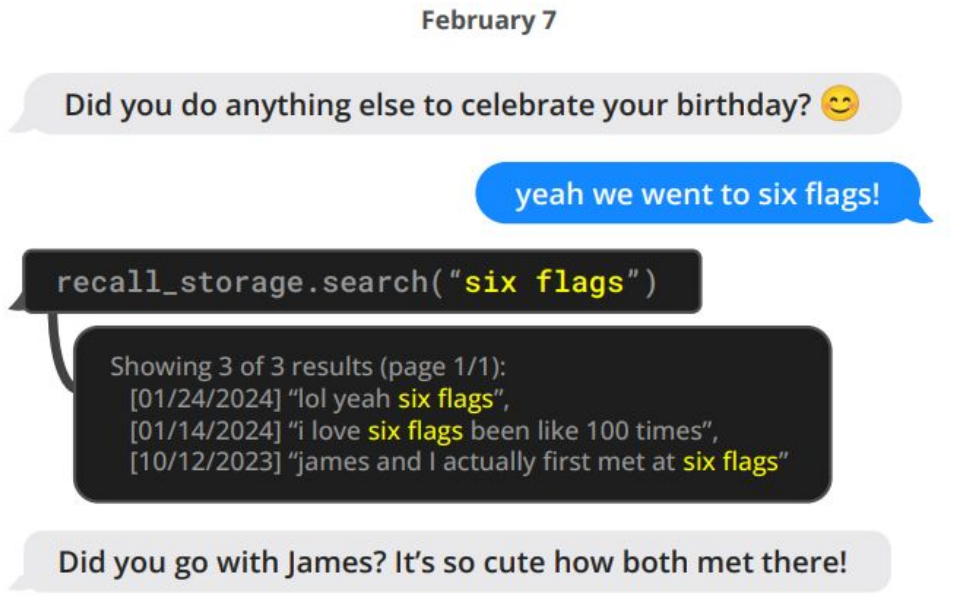
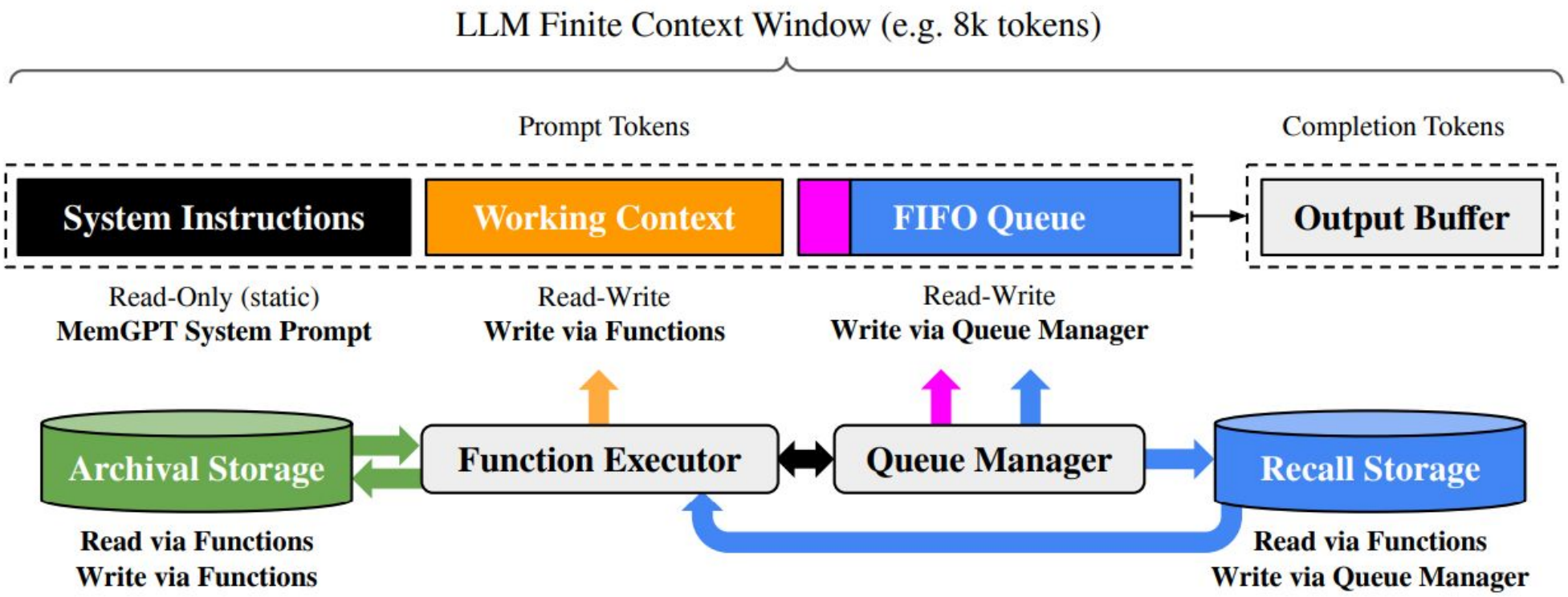


Figure 2. MemGPT (left) can search out-of-context data to bring relevant information into the current context window.

Archival Storage

- When the prompt tokens exceed the ‘warning token count‘ of the LLM’s context window (e.g. 70%), the queue manager inserts a system message to allow the LLM to store important information contained in the FIFO queue to working context or archival storage.
- When the prompt tokens exceed the ‘flush token count’ (e.g. 100%), the queue manager flushes the queue to free up space in the context window: the queue manager evicts a specific count of messages (e.g. 50% of the context window), generates a new recursive summary using the existing recursive summary and evicted messages.
- Once the queue is flushed, the evicted messages are no longer in-context and immediately viewable, they are stored indefinitely in recall storage and readable via function calls.
- Awareness of context limits is a key aspect in making the self-editing mechanism work effectively, to this end MemGPT prompts the processor with warnings regarding token limitations to guide its memory management decisions.
- This leads to 2 key factors in a conversation - Consistency and Engagement.
- Increased performance as a conversational agent in multi session chat.

Mechanism


LLM Finite Context Window (e.g. 8k tokens)
Prompt Tokens
Completion Tokens
System Instructions
Working Context
FIFO Queue
Output Buffer
Read-Only (static)
MemGPT System Prompt
Read-Write
Write via Functions
Read-Write
Write via Queue Manager
Archival Storage
Function Executor
Queue Manager
Recall Storage
Read via Functions
Write via Functions
Read via Functions
Write via Queue Manager

Difference in Archival and Recall Storage

Archival memory in MemGPT is designed for long-term storage of large datasets and less frequently accessed information, requiring specific queries to retrieve data. Recall memory, on the other hand, is meant for immediate and frequently accessed information that MemGPT needs during its active sessions.

- **Core/Recall Memory**: During our current conversation, it contains details like user's project name,immediate tasks, and recent questions asked.

- **Archival Memory**: If user mention a specific detail about the project that was discussed weeks ago, memgpt can query the archival memory to retrieve that information and bring it into the current conversation.

Sample Conversation

User: "Can you remind me what we decided about the project timeline?"

Bot (Core Memory): "We last discussed the project timeline as aiming for a July launch, focusing on completing the prototype by mid-June. Let me check if there were any specific milestones in our previous discussions."

Bot (Archival Memory Query): "/archival_memory_search project_timeline_milestones"

Bot (Core Memory Update): "I've found our past discussion. We set milestones for design completion by May, development by June, and testing by late June."

Step by step process

1. **User Query**: The user asks a question. MemGPT processes this and identifies whether the information required is likely to be in the core memory or archival memory.
2. **Core Memory Check**: MemGPT first checks the core memory for immediate, relevant information. This is fast and efficient, suitable for recent interactions and active context.
3. **Archival Memory Query**: If the information is not found in the core memory, MemGPT queries the archival memory. This involves searching the external vector database for the relevant data.
4. **Core Memory Update**: Once the necessary information is retrieved from the archival memory, MemGPT updates the core memory with this data to make it readily accessible for the ongoing conversation.
5. **Response Generation**: MemGPT combines the information from both types of memory to generate a coherent and contextually appropriate response.

Document Analysis

- Document analysis also faces challenges due to the limited context windows of today's transformer models.
- Many real document analysis tasks require drawing connections across multiple such lengthy documents. summary using the existing recursive summary and evicted messages.
- MemGPT is effectively able to make multiple calls to the retriever by querying archival storage, allowing it to scale to larger effective context lengths
- Awareness of context limits is a key aspect in making the self-editing mechanism work effectively, to this end MemGPT prompts the processor with warnings regarding token limitations to guide its memory management decisions.
- This leads to 2 key factors in a conversation - Consistency and Engagement.
- Increased performance as a conversational agent in multi session chat.

Persona and User Profile

A user profile in the context of MemGPT and chatbot systems is a structured set of information about an individual user.

This information helps the chatbot understand and interact with the user in a personalized and contextually relevant manner

Defining User Profiles

User profiles can be created using text directly or from files.

Custom User Profile:

memgpt add human --name bob --text

"Name: Bob Builder. Occupation: Software Engineer at a big tech company. Hobbies: running, hiking, rock climbing, craft beer, ultimate frisbee."

Saving States

- MemGPT can create chatbots that continuously evolve, retaining information about users and updating their personas over time.
- Personas in MemGPT define different behaviors, characteristics, and roles for the chatbot. We can switch between these personas to simulate different states or modes of conversation.

- For example -

  memgpt add persona --name warmup -f warmup.txt

  memgpt add persona --name checkin -f checkin.txt

  memgpt add persona --name reflection -f reflection.txt

- This would create three personas for the chatbot, which would help us to switch between different states where each text file would help define the state of the bot.

References

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., & Gonzalez, J. E. (2024). MemGPT: Towards LLMs as Operating Systems (arXiv:2310.08560). arXiv. https://doi.org/10.48550/arXiv.2310.08560

Link