



# MemGPT

## Towards LLMs as Operating Systems

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir Patil, Ion Stoica, Joseph Gonzalez

### MemGPT - an OS for LLMs

Teach LLMs to manage their own memory for unbounded context!

#### Problem: context windows limit LLMs

- Modern SOTA LLMs =
  - human-level\* reasoning + goldfish memory
- LLM context is gradually scaling, but
  - Some settings need much more than ~1M context
    - LLM reading over terabytes of data
  - Some settings are inherently infinite context
    - 1 agent for weeks, months, years
  - Context scaling is error prone + expensive
    - Uneven attention, lost in the middle...

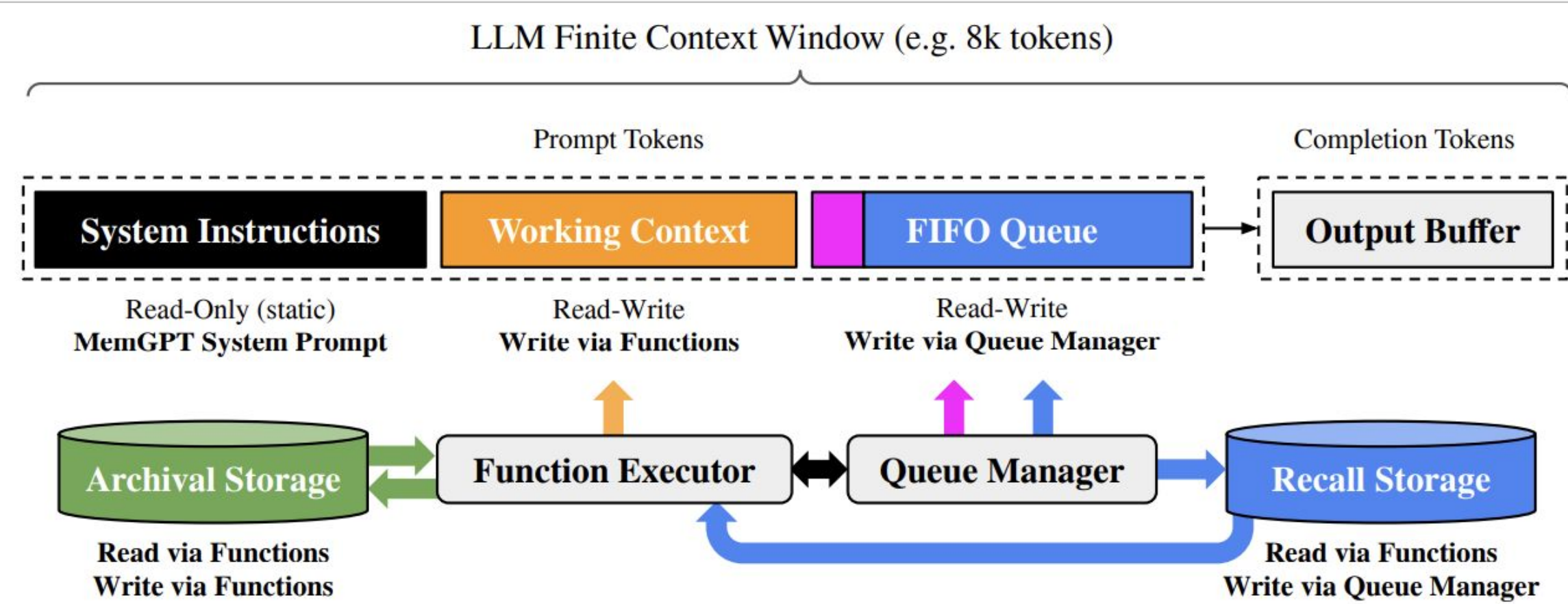


Figure 3. In MemGPT, a fixed-context LLM processor is augmented with a hierarchical memory system and functions that let it manage its own memory. The LLM's prompt tokens (inputs), or *main context*, consist of the system instructions, working context, and a FIFO queue. The LLM completion tokens (outputs) are interpreted as function calls by the function executor. MemGPT uses functions to move data between main context and *external context* (the archival and recall storage databases). The LLM can request immediate follow-up LLM inference to chain function calls together by generating a special keyword argument (`request_heartbeat=true`) in its output; function chaining is what allows MemGPT to perform multi-step retrieval to answer user queries.

#### Put any LLM inside the MemGPT "LLM OS"

- Fixed-context LLM processor + tiered memory system
- LLM uses **function calling** to interact with external world
- LLM can chain together functions (+ yield)
  - Allows the agent to manage memory autonomously
- After a yield, LLM is not run until next event trigger

### MemGPT agents in action

#### Adding to memory

February 7

How was your day today?

fun my bf james baked me a birthday cake

Oh wow, happy birthday! 🎉

**System Alert: Memory Pressure**

```
working_context.append("Birthday is February 7")
```

```
working_context.append("Boyfriend named James")
```

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

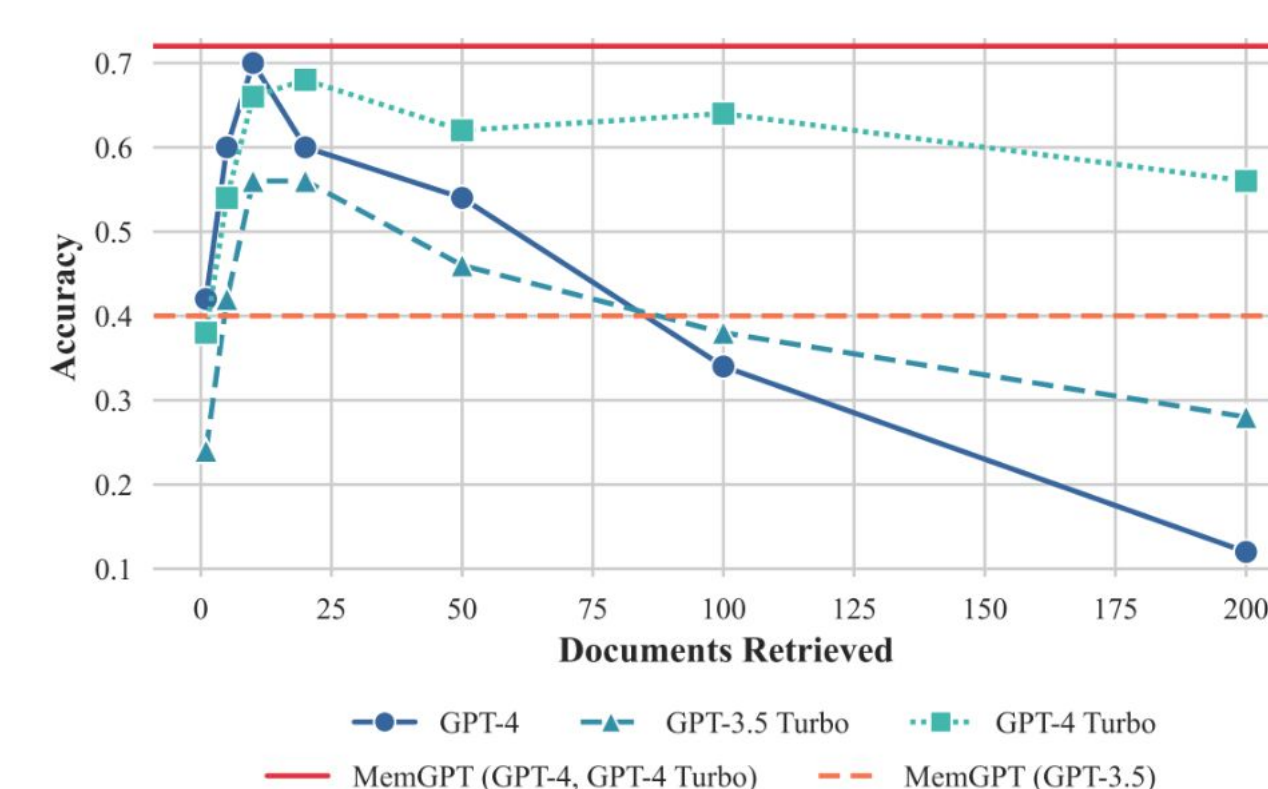


Figure 5. Document QA task performance. MemGPT's performance is unaffected by increased context length. Methods such as truncation can extend the effective context lengths of fixed length models such as GPT-4, but such compression methods will lead to performance degradation as the necessary compression grows. Running MemGPT with GPT-4 and GPT-4 Turbo have equivalent results on this task.

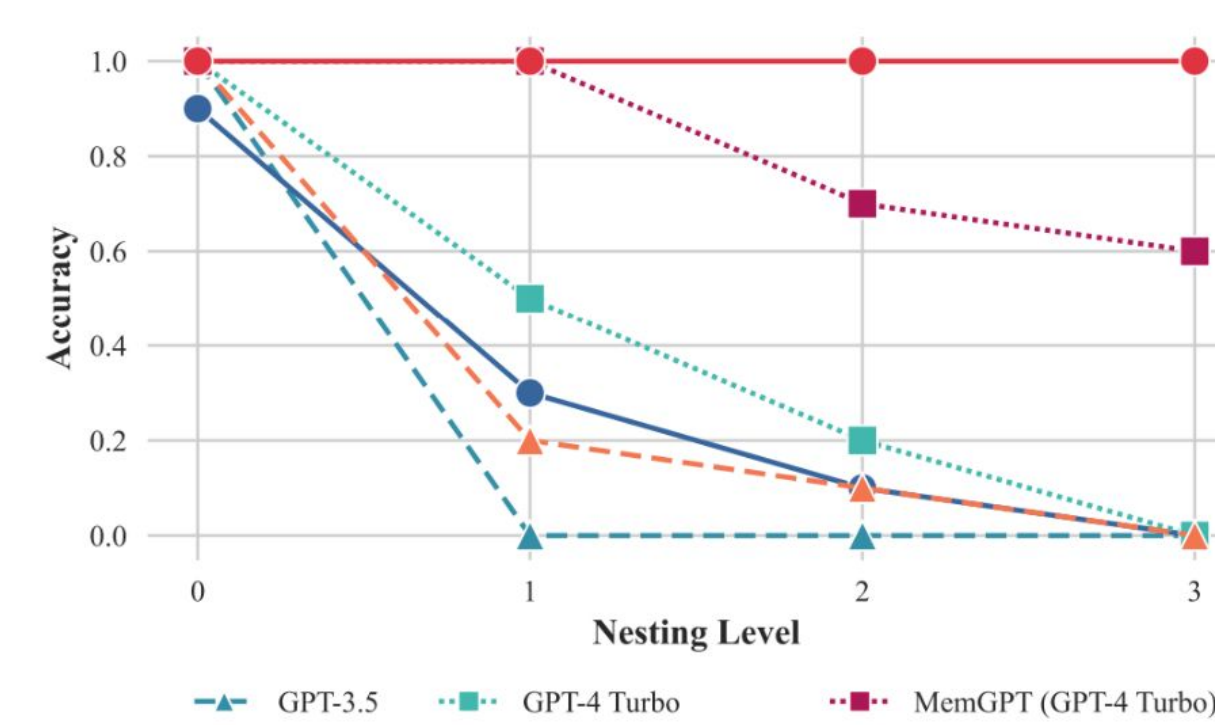


Figure 7. Nested KV retrieval task performance. MemGPT is the only approach that is able to consistently complete the nested KV task beyond 2 nesting levels. While GPT-4 Turbo performs better as a baseline, MemGPT with GPT-4 Turbo performs worse than MemGPT with GPT-4.

#### Searching memory

February 7

Did you do anything else to celebrate your birthday? 😊

yeah we went to six flags!

```
recall_storage.search("six flags")
```

Showing 3 of 3 results (page 1/1):  
[01/24/2024] "lol yeah six flags",  
[01/14/2024] "I love six flags been like 100 times",  
[10/12/2023] "james and I actually first met at six flags"

Did you go with James? It's so cute how both met there!

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

**System Alert: Archive Storage Upload Complete**

Who won the first Nobel Prize in physics?

```
archival_storage.search("nobel physics")
```

Showing 10 of 124 results (page 1/13):  
"The Nobel Prizes, beginning in 1901, and the...  
"This award is administered by the Nobel Foundation...  
..."

```
archival_storage.search("nobel physics", page=2)
```

Showing 10 of 124 results (page 2/13):  
"The Nobel Prize in Physics is a yearly award given...  
"The 1901 Nobel in physics was awarded to Wilhelm...  
..."

Wilhelm Conrad Rontgen

Figure 6. An example of MemGPT (left) solving the document QA task. A database of Wikipedia documents is uploaded to archival storage. MemGPT queries archival storage via function calling, which pulls paginated search results into main context.

**System Alert: Archive Storage Upload Complete**

Find the value for key 831...ea5

```
archival_storage.search("831...ea5")
```

Showing 1 of 1 results (page 1/1):  
"Key: 831...ea5, Value: 5b8...4c3"

```
archival_storage.search("5b8...4c3")
```

Showing 2 of 2 results (page 1/1):  
"Key: 5b8...4c3, Value: f37...617",  
"Key: 831...ea5, Value: 5b8...4c3"

```
archival_storage.search("f37...617")
```

Showing 1 of 1 results (page 1/1):  
"Key: 5b8...4c3, Value: f37...617"

f37...617

Figure 8. An example of MemGPT (left) solving the nested KV task (UIDs shortened for readability). In this particular example, the key-value pair has two nesting levels: 831...ea5 → 5b8...4c3 → f37...617. The MemGPT agent returns the final answer when a query for the final value (f37...617) only returns one result, indicating that it is not also a key.

#### Correcting memory

February 14

How's James doing? Any special plans today?

actually james and i broke up

```
working_context.replace("Boyfriend named James", "Ex-boyfriend named James")
```

Sorry to hear that - hope you're OK 🥰

Figure 4. An example conversation snippet where MemGPT (left) updates stored information. Here the information is stored in working context memory (located within the prompt tokens).

**Build with the MemGPT API**

MemGPT can be used as a multi-user, multi-agent service!

[memgpt.readme.io \(developer docs\)](https://memgpt.readme.io)

|                     | OpenAI | vs. | MemGPT |
|---------------------|--------|-----|--------|
| Self-host           | ❌      |     | ✅      |
| OSS LLMs            | ❌      |     | ✅      |
| Custom tools        | ❌      |     | ✅      |
| Custom data sources | ❌      |     | ✅      |
| White-box memory    | ❌      |     | ✅      |
| OpenAI compatible   | ✅      |     | ✅      |
| Rest API            | ✅      |     | ✅      |
| Python client       | ✅      |     | ✅      |

