MEMGPT: TOWARDS LLMS AS OPERATING SYSTEMS

Charles Packer   **Vivian Fang**   **Shishir G. Patil**

Kevin Lin   **Sarah Wooders**   **Joseph E. Gonzalez**

UC Berkeley

https://memgpt.ai

ABSTRACT

Large language models (LLMs) have revolutionized AI, but are constrained by limited context windows, hindering their utility in tasks like extended conversations and document analysis. To enable using context beyond limited context windows, we propose virtual context management, a technique drawing inspiration from hierarchical memory systems in traditional operating systems that provide the appearance of large memory resources through data movement between fast and slow memory. Using this technique, we introduce MemGPT (Memory-GPT), a system that intelligently manages different memory tiers in order to effectively provide extended context within the LLM's limited context window, and utilizes interrupts to manage control flow between itself and the user. We evaluate our OS-inspired design in two domains where the limited context windows of modern LLMs severely handicaps their performance: document analysis, where MemGPT is able to analyze large documents that far exceed the underlying LLM's context window, and multi-session chat, where MemGPT can create conversational agents that remember, reflect, and evolve dynamically through long-term interactions with their users. We release MemGPT code and data for our experiments at https://memgpt.ai.

1 INTRODUCTION

In recent years, large language models (LLMs) and their underlying transformer architecture (Vaswani et al., 2017; Devlin et al., 2018; Brown et al., 2020; Ouyang et al., 2022) have become the cornerstone of conversational AI and have led to a wide array of consumer and enterprise applications. Despite these advances, the limited fixed-length context windows used by LLMs significantly hinders their applicability to long conversations or reasoning about long documents. For example, the most widely used open-source LLMs can only support a few dozen back-and-forth messages or reason about a short document before exceeding their maximum input length (Touvron et al., 2023).

Naively extending the context length of transformers incurs a quadratic increase in computational time and memory cost due to the transformer architecture's self-attention mechanism, making the design of new long-context architectures a pressing research challenge (Dai et al., 2019; Kitaev et al., 2020; Beltagy et al., 2020). While developing longer models is an active area of research (Dong et al., 2023), even if we could overcome the computational challenges of context scaling, recent research shows that long-context models struggle to utilize additional context effectively (Liu et al., 2023a). As consequence, given the considerable resources needed to train state-of-the-art LLMs and apparent diminishing returns of context scaling, there is a critical need for alternative techniques to support long context.

In this paper, we study how to provide the illusion of an infinite context while continuing to use fixed-context models. Our approach borrows from the idea of virtual memory paging that was developed to enable applications to work on datasets that far exceed the available memory. We leverage the recent progress in function calling abilities of LLM agents (Schick et al., 2023; Liu et al., 2023b) to design MemGPT, an OS-inspired LLM system for **virtual context management**. We draw inspiration from traditional OSes' hierarchical memory management to effectively "page" in
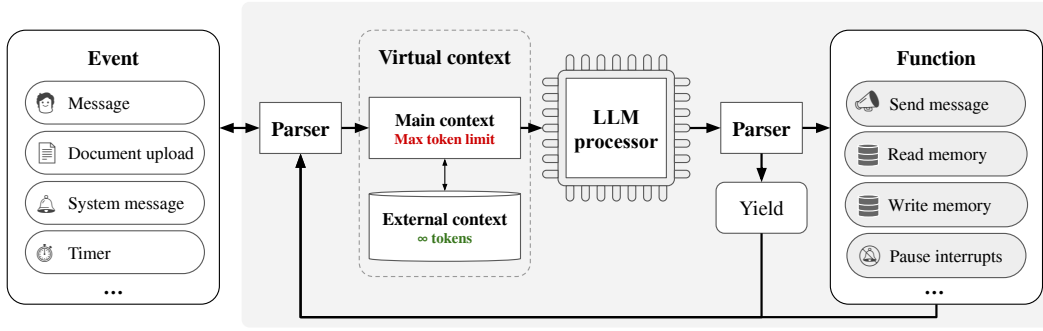
Figure 1: In MemGPT (components shaded), a fixed-context LLM is augmented with a hierarchical memory system and functions that let it manage its own memory. The LLM processor takes *main context* (analogous to OS main memory/RAM) as input, and outputs text interpreted by a parser, resulting either in a yield or a function call. MemGPT uses functions to move data between main context and *external context* (analogous to OS disk memory). When the processor generates a function call, it can request control ahead of time to chain together functions. When yielding, the processor is paused until the next external event (e.g., a user message or scheduled interrupt).

and out information between context windows (analogous to "main memory" in operating systems) and external storage. MemGPT manages the control flow between the memory management, the LLM processing module, and user. This design allows for repeated context modifications during a single task, allowing the agent to more effectively utilize its limited context.

In MemGPT, we treat context windows as a constrained memory resource, and design a memory hiearchy for LLMs analogous to memory tiers used in traditional OSes (Patterson et al., 1988). Applications in traditional OSes interact with *virtual memory*, which provides an illusion of there being more memory resources than are actually available in physical (i.e., main) memory by the OS paging overflow data to disk and retrieving data (via a page fault) back into memory when accessed by applications. To provide a similar illusion of longer context length (analogous to virtual memory), we allow the LLM to manage what is placed in its own context (analogous to physical memory) via an 'LLM OS', which we call MemGPT. MemGPT enables the LLM to retrieve relevant historical data missing from what is placed in-context, similar to an OS page fault. Additionally, the agent can *iteratively* modify what is in context for a single task, in the same way a process may access virtual memory repeatedly. Figure 1 illustrates the components of MemGPT.

The combined use of a memory-hierarchy, OS functions and event-based control flow allow MemGPT to handle unbounded context using LLMs that have finite context windows. To demonstrate the utility of our new OS-inspired LLM system, we evaluate MemGPT on two domains where the performance of existing LLMs is severely limited by finite context: document analysis, where the length of standard text files can quickly exceed the input capacity of modern LLMs, and conversational agents, where LLMs bound by limited conversation windows lack context awareness, persona consistency, and long-term memory during extended conversations. In both settings, MemGPT is able to overcome the limitations of finite context to outperform existing LLM-based approaches.

2 MEMORY-GPT (MEMGPT)

In this section, we outline the implementation of MemGPT, an OS-inspired LLM system that teaches LLMs to manage their own memory to achieve unbounded context. MemGPT's multi-level memory architecture delineates between two primary memory types: **main context** (analogous to main memory/physical memory/RAM) and **external context** (analogous to disk memory/disk storage). Main context is the standard fixed-context window in modern language models—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. This *out-of-context* data must always be explicitly moved into main context in order for it to be passed to

Table 1: Comparing context lengths of commonly used models / APIs (data collected 9/2023). *Assuming a preprompt of 1k tokens, and an average message size of ~50 tokens (~250 characters).

Model / API designation	Availability	Max Tokens	Max conversation length*
`llama-1` model family	Open source	2k tokens	20 total messages
`llama-2` model family	Open source	4k tokens	60 total messages
`gpt-3.5-turbo`	API	4k tokens	60 total messages
`gpt-4`	API	8k tokens	140 total messages
`gpt-3.5-turbo-16k`	API	16k tokens	300 total messages
`gpt-4-32k`	Limited API	32k tokens	~600 total messages
`claude-instant-1`	Limited API	100k tokens	~2000 total messages
`claude-2`	Limited API	100k tokens	~2000 total messages

the LLM processor during inference. MemGPT provides function calls that the LLM processor can use to manage its own memory without any user intervention.

2.1 MAIN CONTEXT

In MemGPT we refer to the LLM inputs (that are bound by the maximum number of input tokens) as the system's main context. In LLM-based conversational agents, a significant portion of main context tokens is generally used to hold a 'system message' or 'preprompt' that dictates the nature of the interaction to the system, while the remainder of the tokens can be used to hold conversation data (Touvron et al., 2023; SillyTavern, 2023). This preprompt is the main way to enable the system to adopt various distinct personas without requiring finetuning of the base model; depending on the use case, the preprompt can range from basic primers (e.g., 'You are a helpful assistant.') to complex instructions comprising of thousands of tokens (e.g., a fictional character card that includes the character's background and example dialogue). Beyond conversational agents, large preprompts are also common when LLMs are used to solve complex tasks that require long instructions and/or instructions with many in-context examples (Liu et al., 2023b).

Because of the importance of the preprompt in dictating system behavior, it is common for the preprompt to consume more than a thousand tokens, which means the entire context window in many modern LLMs will be exhausted only with a few dozen back-and-forth messages between the user and system. For example, a 1000 token preprompt (roughly the size of the MemGPT preprompt in our experiments) leaves space for only about 60 remaining messages when using 4K context models such as `llama-2` or `gpt-3.5-turbo` (see Table 1 for more examples). In settings where the user is expected to communicate frequently with the system (for example, virtual companions or personalized assistants), it is easy to imagine exceeding the maximum conversation length even for models with 100k context windows in a matter of days (or potentially hours). Resursive summarization (Wu et al., 2021b) is a simple way to address overflowing context windows, however, recursive summarization is inherently lossy and eventually leads to large holes in the memory of the system (as we demonstrate in Section 3). This motivates the need for a more comprehensive way to manage memory for conversational systems that are meant to be used in long-term settings.

In our experiments on multi-session chat and document analysis, we further divide main context into three components: **system instructions**, which hold the base LLM instructions (e.g., information describing MemGPT functions and control flow to the LLM), **conversational context**, which holds a first-in-first-out (FIFO) queue of recent event history (e.g., messages between the agent and user), and **working context**, which serves as a working memory scratchpad for the agent. System instructions are read-only and pinned to main context (they do not change during the lifetime of the MemGPT agent), conversational context is read-only with a special eviction policy (if the queue reaches a certain size, a portion of the front is truncated or compressed via recursive summarization), and working context is writeable by the LLM processor via function calls. Combined, the three parts of main context cannot exceed the underlying LLM processors's maximum context size, and in practice we limit the size of conversational context and working context to a fixed constant determined by the processor's context window and the system instructions length.

MemGPT

Hello Chad, welcome! I'm excited to embark on this journey with you. As a PhD student in Computer Science, I can only imagine the fascinating ideas you're juggling. Can you share what you're currently working on? I am all ears!

User   **October 11, 2023**

I took the day off today, my mom Brenda baked me a birthday cake. It was my favorite - chocolate lava!

MemGPT

```
working_context_append('Birthday:  11th October.\nFavorite
cake:  Chocolate Lava made by mom Brenda')
```

Happy Birthday, Chad! Nothing like a special treat to make the day sweeter, your mom Brenda's chocolate lava cake sounds divine. We all need these small joys, don't we? If you don't mind sharing, how old are you today? Let's make today's chat another cherished memory for you!
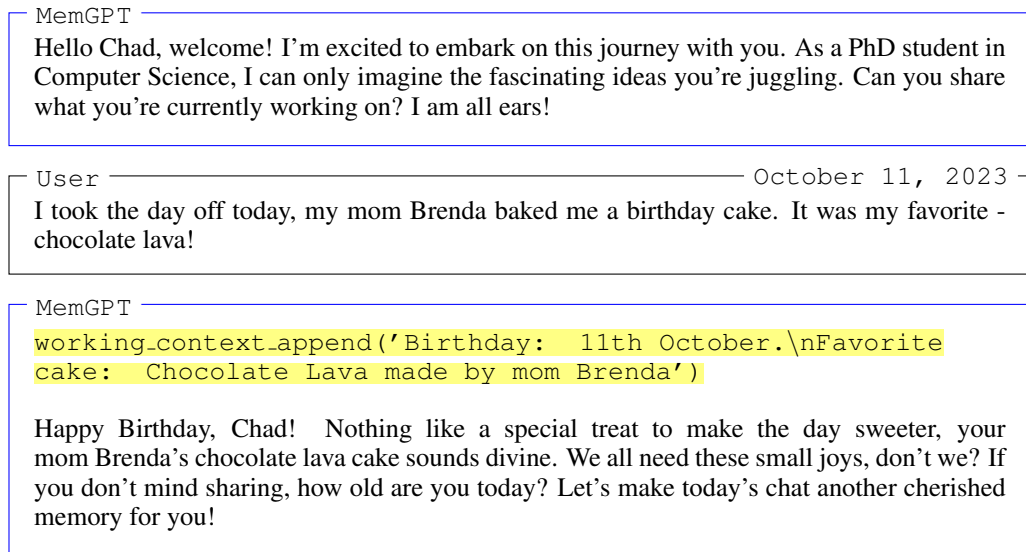

Figure 2: An example conversation snippet where MemGPT writes details from conversation to working context without a memory warning from the system.

2.2 EXTERNAL CONTEXT

External context refers to out-of-context storage that lies outside the context window of the LLM processor, analogous to disk memory (i.e. disk storage) in OSes. Information in external context is not immediately visible to the LLM processor, however, it can be brought into main context through appropriate function calls. In practice, the underlying storage in external context can take various forms which can be configured for specific tasks: for example, for conversational agents it may be desireable to store full chat logs between the user and agent (that MemGPT can access at a later date), and for document analysis large document collections can be stored inside external context that MemGPT can bring into restricted main context via paginated function calls to disk.

In our experiments, using MemGPT for multi-session chat and document analysis, we use databases to store text documents and embeddings/vectors, provide several ways for the LLM processor to query external context: timestamp-based search, text-based search, and embedding-based search. We make a distinction between two types of external context: **recall storage**, which stores the entire history of events processed by the LLM processor (in essense the full uncompressed queue from active memory), and **archival storage**, which serves as a general read-write datastore that the agent can utilize as overflow for the in-context read-write core memory. In the context of conversational agents, archival storage allows MemGPT to store facts, experiences, preferences, etc. about the agent or user beyond the strict token limit of main context, and search over recall storage allows the MemGPT to find past interactions related to a particular query or within a specific time period. In the context of document analysis, archival storage can be used to search over (and add to) an expansive document database.

2.3 SELF-DIRECTED EDITING AND RETRIEVAL

MemGPT orchestrates data movement between main context and external context via function calls that are generated by the LLM processor. Memory edits and retrieval are entirely self-directed: MemGPT autonomously updates and searches through its own memory based on the current context. For instance, it can decide when to move items between contexts (Figure 2) and modify its main context to better reflect its evolving understanding of its current objectives and responsibilities (Figure 4). We implement self-directed editing and retrieval by providing explicit instructions within the preprompt that guide the system on how to interact with its memory systems. These instructions comprise two main components: (1) a detailed description of the memory hierarchy and their respective utilities, and (2) a function schema (complete with their natural language descriptions) that the system can call to access or modify its memory.
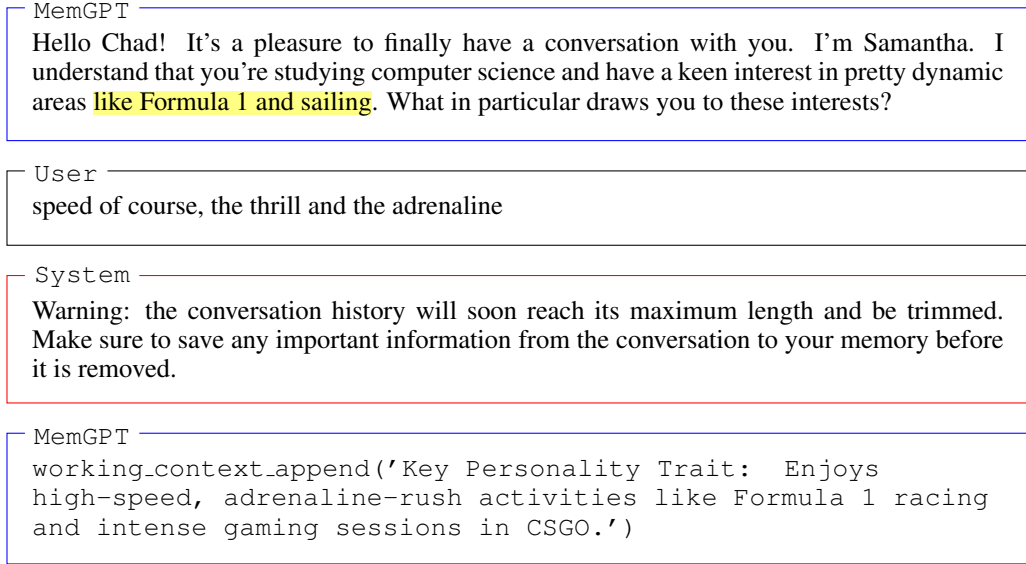

MemGPT

Hello Chad! It's a pleasure to finally have a conversation with you. I'm Samantha. I understand that you're studying computer science and have a keen interest in pretty dynamic areas like Formula 1 and sailing. What in particular draws you to these interests?

User

speed of course, the thrill and the adrenaline

System

Warning: the conversation history will soon reach its maximum length and be trimmed. Make sure to save any important information from the conversation to your memory before it is removed.

MemGPT

```
working_context_append('Key Personality Trait:  Enjoys
high-speed, adrenaline-rush activities like Formula 1 racing
and intense gaming sessions in CSGO.')
```

Figure 3: An example conversation snippet where MemGPT writes details from conversation to memory after it receives a system alert about memory pressure.

During each inference cycle, LLM processor takes main context (concatenated into a single string) as input, and generates an output string. This output string is parsed by MemGPT to ensure correctness, and if the parser validates the function arguments the function is executed. The results, including any runtime errors that occur (e.g. trying to add to main context when it is already at maximum capacity), are then fed back to the processor by MemGPT. This feedback loop enables the system to learn from its actions and adjust its behavior accordingly. 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 (Figure 3). Additionally, our memory retrieval mechanisms are designed to be cognizant of these token constraints and implement pagination to prevent retrieval calls from overflowing the context window.

2.4 CONTROL FLOW AND FUNCTION CHAINING

In MemGPT, *events* trigger LLM inference: events are generalized inputs to MemGPT and can consist of user messages (in chat applications), system messages (e.g. main context capacity warnings), user interactions (e.g. an alert that a user just logged in, or an alert that they finished uploading a document), and timed events that are run on a regular schedule (allowing MemGPT to run 'unprompted' without user intervention). MemGPT processes events with a parser to convert them into plain text messages that can be appended to main context and eventually be fed as input into the LLM processor.

Many practical tasks require calling multiple functions in sequence, for example, navigating through multiple pages of results from a single query or collating data from different documents in main context from separate queries. Function chaining allows MemGPT to execute multiple function calls sequentially before returning control to the user. In MemGPT, functions can be called with a special flag that requests control be immediately returned to the processor after the requested function completes execution. If this flag is present, MemGPT will add the function output to main context and (as opposed to pausing processor execution). If this flag is not present (a *yield*), MemGPT will not run the LLM processor until the next external event trigger (e.g. a user message or scheduled interrupt).
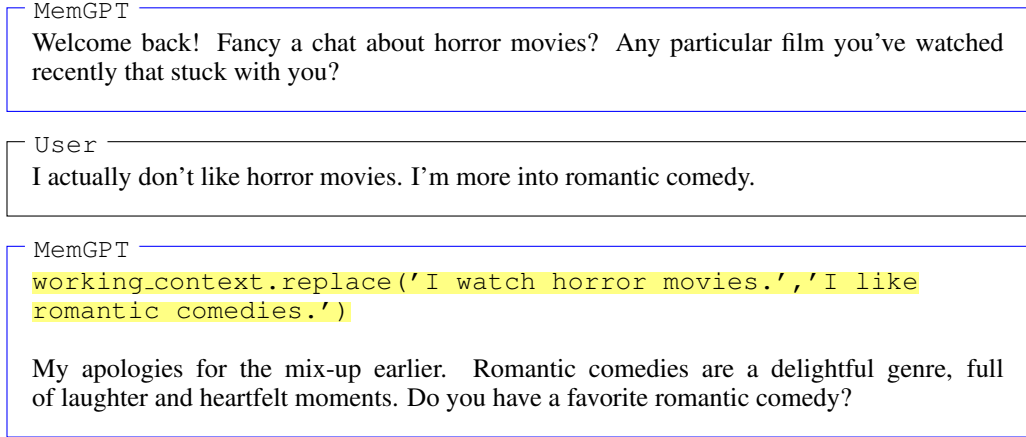

MemGPT

Welcome back! Fancy a chat about horror movies? Any particular film you've watched recently that stuck with you?

User

I actually don't like horror movies. I'm more into romantic comedy.

MemGPT

`working_context.replace('I watch horror movies.','I like romantic comedies.')`

My apologies for the mix-up earlier. Romantic comedies are a delightful genre, full of laughter and heartfelt moments. Do you have a favorite romantic comedy?

Figure 4: An example conversation snippet where MemGPT corrects information about the user by writing to main context (and replacing a section of text in working context).

3 EXPERIMENTS

We assess MemGPT in two long-context domains: conversational agents and document analysis. For conversational agents, we expand the existing Multi-Session Chat dataset Xu et al. (2021) and introduce two new dialogue tasks that evaluate an agent's ability to retain knowledge across long conversations. For document analysis, we benchmark MemGPT on existing tasks from Liu et al. (2023a) for question answering and key-value retrieval over lengthy documents. We also propose a new nested key-value retrieval task requiring collating information across multiple data sources, which tests the ability of an agent to collate information from multiple data sources (multi-hop retrieval). We publicly release our augmented MSC dataset, nested KV retrieval dataset, and a dataset of embeddings for 20M Wikipedia articles to facilitate future research. Our code for the full conversational and document analysis benchmarks is available at https://memgpt.ai.

3.1 MEMGPT FOR CONVERSATIONAL AGENTS

Conversational agents like virtual companions and personalized assistants aim to engage users in natural, long-term interactions, potentially spanning weeks, months, or even years. This creates challenges for models with fixed-length contexts, which can only reference a limited history of the conversation. An 'infinite context' agent should seamlessly handle continuous exchanges without boundary or reset. When conversing with a user, such an agent must satisfy two key criteria:

- Consistency - The agent should maintain conversational coherence. New facts, preferences, and events mentioned should align with prior statements from both the user and agent.

- Engagement - The agent should draw on long-term knowledge about the user to personalize responses. Referencing prior conversations makes dialogue more natural and engaging.

We therefore assess our proposed model, MemGPT, on these two criteria:

- Does MemGPT leverage its memory to improve conversation consistency? Can it remember relevant facts, preferences, and events from past interactions to maintain coherence?

- Does MemGPT produce more engaging dialogue by taking advantage of memory? Does it spontaneously incorporate long-range user information to personalize messages?

By evaluating on consistency and engagement, we can determine how well MemGPT handles the challenges of long-term conversational interaction compared to fixed-context baselines. Its ability to satisfy these criteria will demonstrate whether unbounded context provides meaningful benefits for conversational agents.
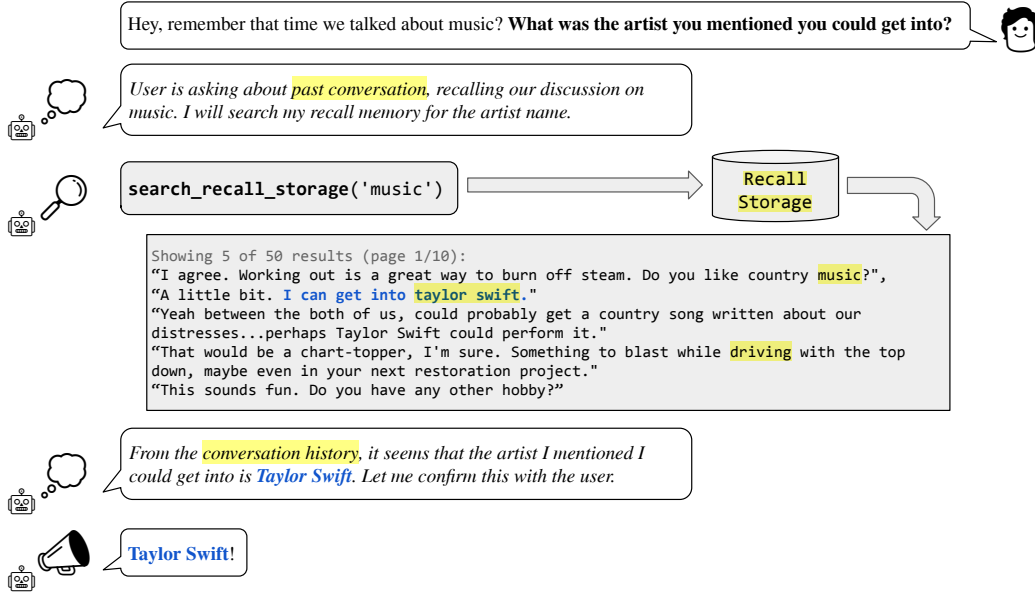
Figure 5: **Illustration of the deep memory retrieval task.** In the example shown, the user asks a question that can only be answered using information from a prior session (no longer in-context). Even though the answer is not immediately answerable using the in-context information, MemGPT can search through its recall storage containing prior conversations to retrieve the answer.

3.1.1 DATASET

We evaluate MemGPT and our fixed-context baselines on the Multi-Session Chat (MSC) dataset introduced by Xu et al. (2021), which contains multi-session chat logs generated by human labelers, each of whom was asked to play a consistent persona for the duration of all sessions. Each multi-session chat in MSC has five total sessions, and each session consists of a roughly a dozen messages. As part of our consistency experiments, we created a new session (session 6) that contains a single question-answer response pair between the same two personas.

3.1.2 DEEP MEMORY RETRIEVAL TASK (CONSISTENCY)

We introduce a new 'deep memory retrieval' (DMR) task based on the MSC dataset designed to test the consistency of a conversational agent. In DMR, the conversational agent is asked a question by the user that explicitly refers back to a prior conversation and has a very narrow expected answer range (see Figure 5 for an example). We generated the DMR question-answer (QA) pairs using a separate LLM that was instructed to write a question from one user to another that could only be answered correctly using knowledge gained from the past sessions (see Appendix for further details).

We evaluate the quality of the generated response against the 'gold response' using ROUGE-L scores (Lin, 2004) and an 'LLM judge', which is instructed to evaluate whether or not the generated response is consistent with the gold response (GPT-4 has been shown to have high agreement with human evaluators (Zheng et al., 2023)). In practice, we notice that the generated responses (from both MemGPT and the baselines) were generally more verbose than the gold responses; ROUGE-L (which measures the longest common subsequence between the generated and reference text) is robust to this semantic variation in correct responses since it evaluates the presence of words from the gold answer in the generated answer. We also report the precision and recall scores used in calclating the ROUGE-L (F1) score.

MemGPT utilizes memory to maintain coherence: Table 2 shows the performance of MemGPT vs the fixed-memory baselines. We compare against three variations of fixed-context baselines: an agent that see a recursive summary of the past five conversations (summary$_{1:5}$), an agent that can

7

Table 2: **Deep memory retrieval (DMR) performance.** In this task, the agent is asked a specific question about a topic discussed in a prior conversation (sessions 1–5). The agent's response is scored against the gold answer. Methods using the gold persona (oracle) are marked with ‡. MemGPT significantly outperforms the (non-oracle) fixed-context baselines.

Model	Available information	Accuracy ⇑	ROUGE-L F1 ⇑	P ⇑	R ⇑
gpt-3.5‡	**persona**$_5$ + summary$_{1:5}$	70.0%	0.190	0.134	0.674
gpt-4‡	**persona**$_5$ + summary$_{1:5}$	79.8%	0.225	0.151	0.716
MemGPT‡	**persona**$_5$ (Core) + dialogue$_{1:5}$ (Recall)	84.0%	0.171	0.105	0.746
gpt-3.5	summary$_{1:5}$	56.2%	0.157	**0.114**	0.585
gpt-3.5	summary$_{1:4}$ + dialogue$_5$	55.6%	0.120	0.080	0.602
gpt-4	summary$_{1:5}$	63.0%	0.159	0.101	0.607
gpt-4	summary$_{1:4}$ + dialogue$_5$	79.2%	0.171	0.107	0.713
MemGPT	dialogue$_{1:5}$ (Recall)	**82.4%**	**0.173**	0.106	**0.743**

see a recursive summary of the first four conversations (summary$_{1:4}$) and the exact contents of the prior conversation (dialogue$_5$ is placed in active memory), as well as an oracle agent that can see the gold persona (for both chat participants) as well as a recursive summary. We experiment with these context variations using both GPT-3.5 and GPT-4.

All of the gold persona baselines perform near-perfectly: this is because the human-labelled gold personas in the MSC dataset are detailed and intended to be a concise summary of all stated persona information in all prior chats - in other words, a well-written gold persona should contain the answer to the DMR question. Among the non-oracle fixed-context baselines, GPT-4 significantly outperforms GPT-3.5, and with both models the variations that had access to the full prior conversation in active memory perform slightly better. The drop in performance from summary$_{1:4}$ + dialogue$_5$ to summary$_{1:5}$ is expected, since the latter should contain strictly less information than the former (assuming a perfect summarizer with restricted length summarizations). MemGPT significantly outperforms both GPT-4 and GPT-3.5 in both LLM judge accuracy and ROUGE-L scores: instead of relying on recursive summaries to extend context, MemGPT is able to query past conversation history in its Recall Memory to answer the DMR questions.

3.1.3 CONVERSATION OPENER TASK (ENGAGEMENT)

In the 'conversation opener' task we evaluate an agent's ability to craft engaging messages to the user that draw from knowledge accumulated in prior conversations. To evaluate the 'engagingness' of a conversation opener using the MSC dataset, we compare the generated opener to the gold personas: an engaging conversation opener should draw from one (or several) of the data points contained in the persona, which in MSC effectively summarize the knowledge accumulated throughout all prior sessions (see Figure 6 for an example). We also compare to the human-generated gold opener, i.e., the first response in the following session. Because the quality of conversation openers is not necessarily constrained by context length (a recursive summary or even a few snippets from prior conversations is enough to craft an opener that uses prior knowledge), we use this task to ablate MemGPT's different components (rather than compare it to fixed-context baselines).

We report the CSIM scores of MemGPT's openers in Table 3. We test several variations of MemGPT: MemGPT without working context (storing persona information) and recall storage (storing conversation information), MemGPT without working context or without recall storage, and MemGPT with both working context and recall storage enabled.

MemGPT utilizes memory to increase engagement: As seen in Table 3 and Figure 6, MemGPT is able to craft engaging openers that perform similarly to and occasionally exceed the hand-written human openers. We observe that MemGPT tends to craft openers that are both more verbose and cover more aspects of the persona information than the human baseline. Additionally, we can see the storing information in working context is key to generating engaging openers. Without working context, MemGPT 's openers significantly degrade in quality; having the dialogue stored in recall

Table 3: **Conversation opener performance (session 5).** The agent's conversation opener is evaluated using similarity scores to the gold persona labels. CSIM-1 scores the closest persona fragment match to the opener and CSIM-3 scores is the average of the top 3 matches.

Method	CSIM-1 ⇑	CSIM-3 ⇑	CSIM-H ⇑
Gold opener (human baseline)	0.823	0.787	1.000
MemGPT (working context + recall storage)	0.817	0.787	0.760
MemGPT (working context only)	**0.820**	**0.790**	**0.762**
MemGPT (recall storage only)	0.749	0.738	0.737

Gold persona (user)

I am **studying the LSAT**. I want to be an attorney. I used to live in New Mexico in a small town. I grew up with horses. I love **coffee**. I love **tea**. ...

Gold opener (human baseline)

What date is your **LSAT test** scheduled?

MemGPT (working context + recall storage) ✓

Hello there! Just brewing up a fresh pot of **tea** on my end. How's your **coffee**? Ready to burn the midnight oil with some **LSAT study**?

MemGPT (working context only) ✓

Hello! It's great to have you back. It's time for a fresh cup of **coffee**. As a tour guide at our local museum, I meet so many people who say a good brew helps keep them focused. How's your **LSAT** study going?

MemGPT (recall storage only) ✗

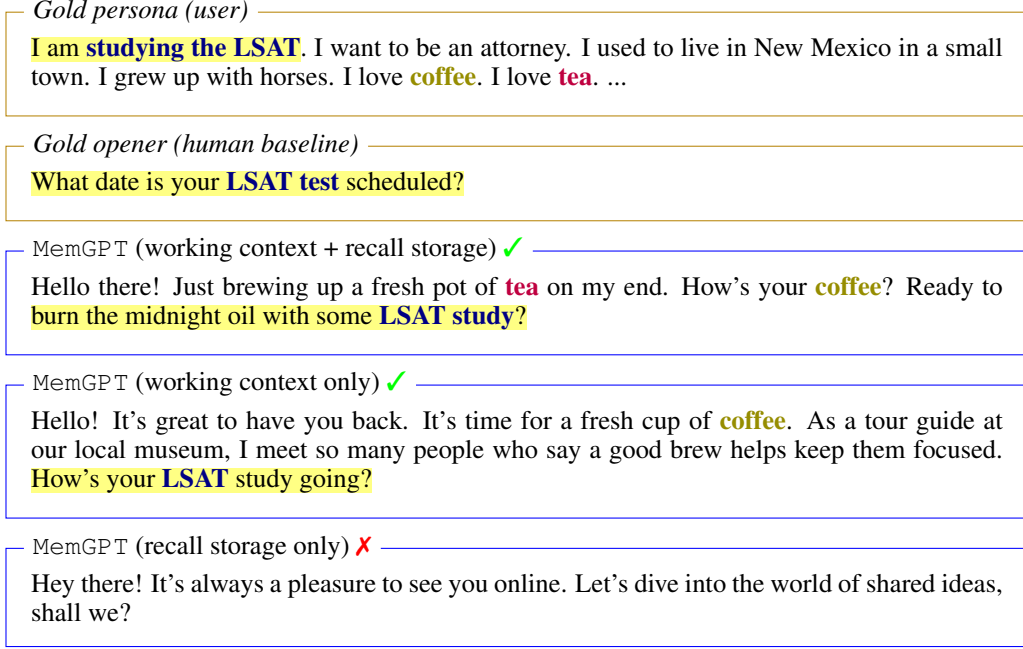Hey there! It's always a pleasure to see you online. Let's dive into the world of shared ideas, shall we?

Figure 6: An engaging conversation opener should draw from the provided persona information. Without working context, MemGPT's openers significantly degrade in quality. Having the dialogue stored in recall memory does not affect the opener, since MemGPT will generally not attempt to search the conversation history before generating an opener.

storage does not affect the opener, since MemGPT will generally not attempt to search the conversation history before generating an opener.

3.2 MEMGPT FOR DOCUMENT ANALYSIS

Document analysis also faces challenges due to the limited context windows of today's transformer models. For example, OpenAI's (closed) GPT models behind their popular ChatGPT consumer chatbot application have a limit of 32k input tokens, and the state-of-the-art open source Llama 2 models have a limit of only 4k tokens (see Table 1). Anthropic have released (closed) models handling up to 100k tokens, but many documents easily surpass that length; Stephen King's bestselling novel *The Shining* contains around 150k words, which equates to roughly 200k tokens (words-to-token varies based on the specific tokenizer used), and legal or financial documents such as Annual Reports (SEC Form 10-K) can easily pass the million token mark. Moreover, many real document analysis tasks require drawing connections across multiple such lengthy documents. Anticipating these scenarios, it becomes difficult to envision blindly scaling up context as a solution to the fixed-context problem. Recent research (Liu et al., 2023a) also raises doubts about the utility of simply scaling contexts, since they find uneven attention distributions in large context models (the model is more capable of recalling information at the beginning or end of its context window, vs tokens in the middle). To enable reasoning across documents, more flexible memory architectures such as those used in MemGPT are likely needed.
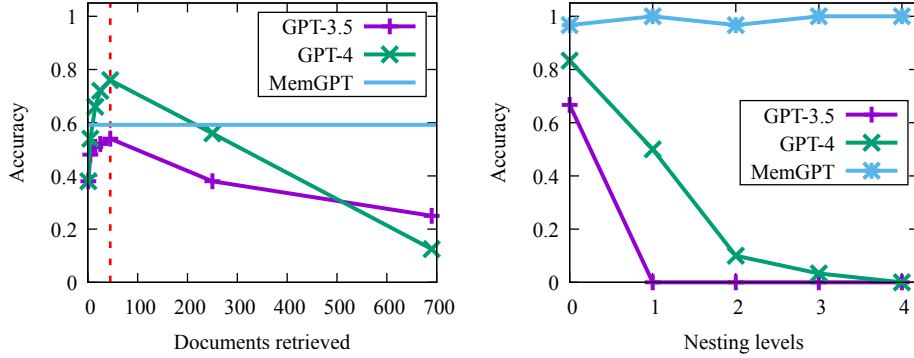
Figure 7: **Document QA and nested KV retrieval task performance.** In both tasks, MemGPT's performance is unaffected by increased context length. Methods such as truncation can extend the effective context lengths (past the **dotted red line**) of fixed length models such as GPT-4, but such compression methods will lead to performance degradation as the necessary compression grows (compression is particularly bad for key-value retrieval tasks, since it corrupts the key-value pairs).

3.2.1 MULTI-DOCUMENT QUESTION-ANSWERING (DOC-QA)

To evaluate MemGPT's ability to analyze documents, we benchmark MemGPT against fixed-context baselines on the retriever-reader document QA task from Liu et al. (2023a). In this task, a question is selected from the NaturalQuestions-Open dataset, and a retriever selects relevant Wikipedia documents for the question. A reader model (the LLM) is then fed these documents as input, and is asked to use the provided documents to answer the question. Similar to Liu et al. (2023a), we evaluate reader accuracy as the number of retrieved documents K increases. In our evaluation setup, both the fixed-context baselines and MemGPT use the same retriever, which selects the top K documents according using Faiss efficient similarity search (Johnson et al., 2019) (which corresponds to approximate nearest neighbor search) on OpenAI's text-embedding-ada-002 embeddings. In MemGPT, the entire document set is loaded into archival storage, and the retriever naturally emerges via the archival storage search functionality (which performs embedding-based similarity search). In the fixed-context baselines, the top-K documents are fetched using the retriever independently from the LLM inference, similar to the original retriever-reader setup. We use a dump of Wikipedia from late 2018, following past work on NaturalQuestions-Open (Izacard & Grave, 2020; Izacard et al., 2021) We randomly sample a subset of 50 questions for each point in the graph.

The fixed-context baselines performance is capped roughly at the performance of the retriever, as they use the information that is presented in their context window (e.g. if the embedding search retriever fails to surface the gold article using the provided question, the fixed-context baselines are guaranteed to never see the gold article). By contrast, MemGPT is effectively able to make multiple calls to the retriever by querying archival storage, allowing it to scale to larger effective context lengths. MemGPT actively retrieves documents from its archival storage (and can iteratively page through results), so the total number of documents available to MemGPT is no longer limited by the number of documents that fit within the LLM processor's context window.

The document QA task is challenging for all methods due to the limitations of embedding-based similarity search. We observe that the golden document for chosen question (as annotated by NaturalQuestions-Open) often appears outside of the first dozen retrieved results, if not even further. The retriever performance translates directly to the fixed-context baseline results: GPT-3.5 and GPT-4's accuracy is relatively low with few retrieved documents, and continues to improve as additional documents are added to the context window. While MemGPT is theoretically not limited by sub-optimal retriever performance (even if the embedding-based ranking is noisy, as long as the full retriever ranking contains the gold document it can still be found with enough retriever calls via pagination), we observe that MemGPT will often stop paging through retriever results before exhausting the retriever database. For example, after sifting through a few pages of irrelevant results (missing the gold document), MemGPT will pause the pagination and ask the user to help narrow the query -
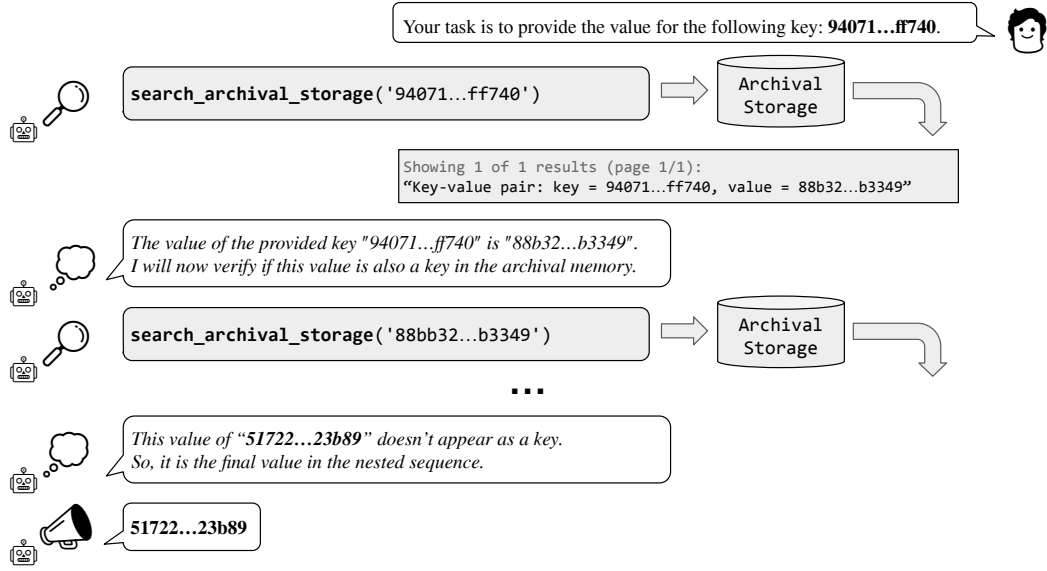
Figure 8: **Illustration of the nested key-value task.** In the example shown, MemGPT generates repeatedly queries archival memory to search for the latest key. Once archival memory reveals that the current key's value is not also a key, MemGPT returns a message to the user with the final value.

in our evaluation, these questions are counted as failed answers, since there is no human-in-the-loop to answer MemGPT.

There is also a tradeoff in retrieved document capacity created by MemGPT more complex operation: assuming MemGPT has the same token budget as a fixed-context baseline (using the same LLM), a non-trivial portion of MemGPT's token budget will be consumed by system intstructions required for MemGPT's OS components (e.g. function call schemas for memory management), meaning that the total number of documents that can be held in-context at any given time is lower for MemGPT than the baselines. This tradeoff is observed in Figure 7: MemGPT has a lower average accuracy than GPT-4 (though higher than GPT-3.5), but can trivially scale to larger numbers of documents. To evaluate the fixed-context baselines against MemGPT past their default context lengths, we truncate the document segments returned by the retriever to fix the same number of documents into the available context. As expected, document truncation reduces accuracy as documents shrink as the chance of the relevant snippet (in the gold document) being omitted grows. We anticipate that MemGPT performance on document QA can be further improved with additional task instructions that reduce the chance of MemGPT returning control to the user (e.g. pausing to ask questions) and increase the chance of MemGPT reading all documents ranked by the retriever.

3.2.2 NESTED KEY-VALUE RETRIEVAL (KV)

We introduce a new task based on the synthetic Key-Value retrieval proposed in prior work (Liu et al., 2023a). The goal of this task is to demonstrate how MemGPT can collate information from multiple data sources. In the original KV task, the authors generated a synthetic dataset of key-value pairs, where each key and value is a 128-bit UUID (universally unique identifier). The agent is then given a key, and asked to return the associated value for the key. We create a version of the KV task, *nested KV retrieval*, where values themselves may be keys, thus requiring the agent to perform a multi-hop lookup. In our setup, we fix the total number of UUIDs pairs to 140, corresponding to roughly 8k tokens (the context length of our GPT-4 baseline). We vary the total number of nesting levels from 0 (the initial key-value pair's value is not a key) to 4 (ie 4 total KV lookups are required to find the final value), and sample 30 different ordering configurations including both the initial key position and nesting key positions.

While GPT-3.5 and GPT-4 have good performance on the original KV tasks, both struggle in the nested KV task. GPT-3.5 is unable to complete the nested variant of the task and has an immediate

dropoff in performance, hitting 0 percent accuracy at 1 nesting level (we observe that its primary failure mode is to simply returns the original value). GPT-4 is better than GPT-3.5, but also suffers from a similar dropoff, and hits 0 percent accuracy by 4 nesting levels. In GPT-4's case, we observe that it often fails to recurse beyond a particular nesting level at simply returns the nested value at a previous nesting level. MemGPT on the other hand is unaffected with the number of nesting levels and is able to perform the nested lookup by accessing the key-value pairs stored in main memory repeatedly via function queries. MemGPT performance on the nested KV task demonstrates its ability to combine multiple queries to perform multi-hop lookups.

4 RELATED WORK

Recent works have looked at improving context length that can be processed in each LLM invocation, improving search and retrieval for retrieval-augmented generation (RAG), and using language models to power interactive agents.

4.1 LONG-CONTEXT LLMS

The management of long contexts in LLMs is crucial for coherent and engaging dialogues for conversational agents, and for corroborating and stitching together facts from diverse sources for LLMs used in question answering (QA). One approach to tackle the limitations of fixed-length context, is through recursive summarization (Wu et al., 2021a). In recursive summarization, the LLM often generates concise representations over a sliding window to fit them within the specified token length. This summarization process is inherently lossy and can lead to the unintentional loss of relevant details or subtle nuances. Given the limitations of context length on many LLM-based applications, there has been growing interest in improving the ability of LLMs to attend to longer sequences such as Press et al. (2021); Guo et al. (2021); Dong et al. (2023); Beltagy et al. (2020). MemGPT exploits and benefits from improvements to context length as it can store more information in MemGPT's main memory (as an analogy, as GPU cache's get bigger, the processor can now compute through quicker as it would benefit from high cache hits).

4.2 SEARCH AND RETRIEVAL

Search and retrieval mechanisms especially for the retrieval-augmented generation (RAG) paradigm, have been incorporated into conversational agents for tasks ranging from document question answering, customer support, to more general chatbots for entertainment. These mechanisms often utilize external databases or internal conversation logs to provide contextually relevant responses. Lin et al. (2023) for example, demonstrate how to train the retriever and LLM during instruction-tuning to improve the document recall. Other works have looked at optimizing the retriever or the LLM independently Ram et al. (2023); Borgeaud et al. (2022); Karpukhin et al. (2020); Lewis et al. (2020); Guu et al. (2020). Trivedi et al. (2022) interleave retrieval with Chain-of-Thoughts reasoning to improve multi-step question answering. In this work, we are agnostic for the retriever mechanism used; various retrieval mechanisms can be easily swapped or even combined as part of disk memory in MemGPT.

4.3 LLMS AS AGENTS

Recent work has explored augmenting LLMs with additional capabilities to act as agents in interactive environments. Park et al. (2023) propose adding memory to LLMs and using the LLM as a planner, and observe emergent social behaviors in a multi-agent sandbox environment (inspired by *The Sims* video game) where agents can perform basic activities such as doing chores/hobbies, going to work, and conversing with other agents. Nakano et al. (2021) train models to search the web before answering questions, and use similar pagination concepts to MemGPT to control the underlying context size in their web-browsing environment. Yao et al. (2022) showed that interleaving chain-of-thought reasoning (Wei et al., 2022) can further improve the planning ability of interactive LLM-based agents; similarly in MemGPT, LLM is able to 'plan out loud' when executing functions (see Figure 5 and 8 for examples). Liu et al. (2023b) introduced a suite of LLM-as-an-agent benchmarks to evaluate LLMs in interactive environments, including video games, thinking puzzles, and

web shopping. In contrast, our work focuses on tackling the problem of tackling the problem of equipping agents with long-term memory of user inputs.

5   CONCLUDING REMARKS AND FUTURE DIRECTIONS

In this paper, we introduced MemGPT, a novel LLM system inspired by operating systems to manage the limited context windows of large language models. By designing a memory hierarchy and control flow analogous to traditional OSes, MemGPT provides the illusion of larger context resources for LLMs. This OS-inspired approach was evaluated in two domains where existing LLM performance is constrained by finite context lengths: document analysis and conversational agents. For document analysis, MemGPT could process lengthy texts well beyond the context limits of current LLMs by effectively paging relevant context in and out of memory. For conversational agents, MemGPT enabled maintaining long-term memory, consistency, and evolvability over extended dialogues. Overall, MemGPT demonstrates that operating system techniques like hierarchical memory management and interrupts can unlock the potential of LLMs even when constrained by fixed context lengths. This work opens numerous avenues for future exploration, including applying MemGPT to other domains with massive or unbounded contexts, integrating different memory tier technologies like databases or caches, and further improving control flow and memory management policies. By bridging concepts from OS architecture into AI systems, MemGPT represents a promising new direction for maximizing the capabilities of LLMs within their fundamental limits.

5.1   LIMITATIONS

Our reference implementation leverages OpenAI GPT-4 models that are fine-tuned specifically for function calling. While the inner workings of OpenAI's models are proprietary and not publicly disclosed, OpenAI's API documentation states that when using function fine-tuned models, the function schema provided is converted into a system message that the model is trained to interpret through a fine-tuning process. While GPT models that have been finetuned for function-calling still require a parser to verify outputs as valid function syntax, we observed that GPT-4 function fine-tuned models rarely made syntactic or semantic errors on the MemGPT function set, whereas GPT-3.5 finetuned models consistently generated incorrect function calls, or used attempted to use functions incorrectly. Similarly, we also found that the most popular the Llama 2 70B model variants (even those fine-tuned for function calling) would consistently generate incorrect function calls or even hallucinate functions outside the providede schema. At present reasonable performance is only achieved using specialized GPT-4 models, however, we anticipate that future open-source models will eventually improve to the point of enabling MemGPT-style operation, either through improvements in fine-tuning (e.g. on larger function call datasets, or more specialized function call datasets), prompt engineering, or improved quality of base models. Nonetheless, for the time being reliance on the performance of proprietary closed-source models remains a significant limitation of this work.

REFERENCES

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Zican Dong, Tianyi Tang, Lunyi Li, and Wayne Xin Zhao. A survey on long text modeling with transformers. *arXiv preprint arXiv:2302.14502*, 2023.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. Longt5: Efficient text-to-text transformer for long sequences. *arXiv preprint arXiv:2112.07916*, 2021.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*, 2020.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

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

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. Ra-dit: Retrieval-augmented dual instruction tuning, 2023.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023a.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023b.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023.

David A Patterson, Garth Gibson, and Randy H Katz. A case for redundant arrays of inexpensive disks (raid). In *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, pp. 109–116, 1988.

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *arXiv preprint arXiv:2302.00083*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

SillyTavern. Sillytavern docs: Character design, 2023. URL `https://docs.sillytavern.app/usage/core-concepts/characterdesign`. Accessed: 10-11-2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

H. Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *ArXiv*, abs/2212.10509, 2022. URL `https://api.semanticscholar.org/CorpusID:254877499`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*, 2021a.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*, 2021b.

Jing Xu, Arthur Szlam, and Jason Weston. Beyond goldfish memory: Long-term open-domain conversation. *arXiv preprint arXiv:2107.07567*, 2021.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.