Beyond Memorization: The Challenge of Random Memory Access in Language Models

Tongyao Zhu[1,2] **Qian Liu**[1*] **Liang Pang**[3*] **Zhengbao Jiang**[4]
Min-Yen Kan[2] **Min Lin**[1]
[1]Sea AI Lab [2]National University of Singapore
[3]Institute of Computing Technology, CAS [4]Carnegie Mellon University
tongyao.zhu@u.nus.edu {liuqian,linmin}@sea.com
pangliang@ict.ac.cn zhengbaj@cs.cmu.edu knmnyn@nus.edu.sg

Abstract

Recent developments in Language Models (LMs) have shown their effectiveness in NLP tasks, particularly in knowledge-intensive tasks. However, the mechanisms underlying knowledge storage and memory access within their parameters remain elusive. In this paper, we investigate whether a generative LM (e.g., GPT-2) is able to access its memory sequentially or randomly. Through carefully-designed synthetic tasks, covering the scenarios of full recitation, selective recitation and grounded question answering, we reveal that LMs manage to sequentially access their memory while encountering challenges in randomly accessing memorized content. We find that techniques including recitation and permutation improve the random memory access capability of LMs. Furthermore, by applying this intervention to realistic scenarios of open-domain question answering, we validate that enhancing random access by recitation leads to notable improvements in question answering. The code to reproduce our experiments can be found at https://github.com/sail-sg/lm-random-memory-access.

1 Introduction

Language models (LMs) have recently showcased outstanding abilities in NLP tasks with a large amount of memory stored in their parameters (Brown et al., 2020; Ouyang et al., 2022). Through pre-training on large text corpora, LMs memorize factual knowledge about the world (Zhou et al., 2023). Consequently, they show great performance in knowledge-intensive tasks (Petroni et al., 2021) such as open-domain question answering (Kamalloo et al., 2023; Ziems et al., 2023; Mallen et al., 2023). There is a growing interest in considering LMs as knowledge bases (Wang et al., 2021; Heinzerling and Inui, 2021; Petroni et al., 2019; Cao et al., 2021; AlKhamissi et al., 2022). Despite the recent advances in applying LMs to solve downstream tasks, the fundamentals of how LMs store knowledge and access memory in their parameters remain a subject of ongoing research and intrigue (Tirumala et al., 2022; Zhu and Li, 2023; Allen-Zhu and Li, 2023; Berglund et al., 2023).

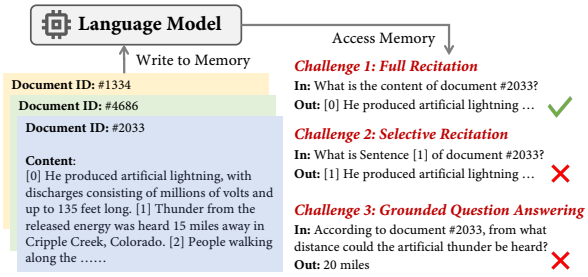

Figure 1: An illustration of our investigation of memory access pattern in language models. We find that the model accesses its parametric memory largely in a sequential manner, and faces difficulty in randomly accessing the content in the middle of memorized strings.

In this paper, we draw inspiration from memory-accessing patterns observed in computer systems to explore whether LMs can access their parametric memory in a sequential or random manner. We extrapolate these concepts to investigate LMs and delineate two memory access patterns: *sequential memory access* means that the model starts from the beginning of a memorized sequence, progressing through the content in consecutive order. Conversely, *random memory access* denotes that the model can commence from any location within the memorized content, without needing to start from the beginning. For instance, reciting a memorized poem line by line is considered sequential access, while directly starting from the third line involves random access.

With these concepts, we design experiments with both synthetic and real data to evaluate the language model's ability to perform sequential or random access to memorized content, as illustrated in Figure 1. We limit our study to decoder-

*Corresponding authors.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3373–3388

only language models because of their increasing popularity and capability (Radford et al., 2019; Brown et al., 2020; Touvron et al., 2023a,b; Jiang et al., 2023). We first ask the model to memorize key–value pairs of various types and show that the model is able to sequentially read memorized content to a satisfying degree. Next, we test the model's random access ability by training it to recite a sentence or find an answer to a question in a memorized passage. In such tasks, the model's performance falls drastically when it is required to extract a span in the middle of a passage, revealing its incapacity to randomly access its memory.

Given that language models struggle to perform random access to their memory, we pursue two means for mitigation: *recitation* at inference time, and *permutation* during training. Recitation enables the model to sequentially read its parametric memory first before performing a task. The model's performance can thus be enhanced by utilizing the recited content in its context window. We also show that simply permuting sentences in a passage during training to memorize content also improves performance.

We finally verify the challenge of random access through a case study on open-domain question answering. We reduce the difficulty of the task by allowing the model to memorize passages with ground-truth answers, yet we find that the model benefits the most from such memorization when it is allowed to recite a relevant passage and then answer the question. Overall, we make several contributions to further understand the memory access mechanisms of decoder-only language models:

- We show that language models can access their memory sequentially and can reproduce memorized content, but encounter significant challenges in random memory access.
- We find solutions to mitigate the challenge of random access by permuting memorized content or explicitly reciting the memory before performing tasks.
- We demonstrate the effect of poor random memory access ability in open-domain question answering, showing that the challenge could have broader implications on the applications of language models.

2 Related Work

Memorization in Language Models. Large language models store a considerable amount of knowledge in their parameters (Petroni et al., 2019; Heinzerling and Inui, 2021). They memorize useful knowledge such as facts and commonsense (Zhao et al., 2023), but also sensitive personal information such as emails or phone numbers (Carlini et al., 2020; Huang et al., 2022). Existing approaches to understanding memorization include fine-grained analysis to locate the neuron that is associated with the knowledge (Meng et al., 2022; Liu et al., 2024) or macro analysis to understand the overall dynamics of memorization (Tirumala et al., 2022; Speicher et al., 2024). In this study, we do not aim to analyze the mechanisms of writing to language model's memory. Instead, we consider the language model as a black-box memory store and focus mainly on how the model accesses its memory.

Knowledge Injection. Our investigation requires writing new content to the model's parametric memory. There are mainly two ways to perform such knowledge injection without changing the model architecture (Ovadia et al., 2024; Balaguer et al., 2024): fine-tuning or retrieval augmentation. Retrieval augmentation (Lewis et al., 2020; Shi et al., 2023) retrieves relevant information and puts it into the model's context while fine-tuning directly updates the model parameters. As the goal of our study is to investigate how the model accesses its parametric memory after writing to the memory, we choose finetuning as the method for introducing new knowledge to the model.

Knowledge Retrieval. Previous works have shown that using prompts can effectively retrieve knowledge stored in large language models (Bouraoui et al., 2019; Jiang et al., 2021; Wang et al., 2021). We follow earlier work to use prompts to query the model to access and regenerate memorized content. However, a notable difference is that prior work focuses on finding optimised methods to elicit the model's knowledge obtained during pretraining (Youssef et al., 2023; Liu et al., 2023; Yu et al., 2023), while we directly use unique keys for memorizing and retrieving content.

Language Model as a Document Index. We consider the language model as a memory store for passages, which is related to the recent advances in adopting a language model as an index for document storage and retrieval (Metzler et al., 2021;

Tay et al., 2022; Wang et al., 2023; Zeng et al., 2023). In such indexes, each document is associated with a document identifier (ID), which could be keywords (Ren et al., 2023; Bevilacqua et al., 2022; Lee et al., 2023b,a) or numbers (Tay et al., 2022; Wang et al., 2023; Zhuang et al., 2022; Zhou et al., 2022). We also follow the practice and assign an ID to each document for storing and retrieving the documents. However, we do not ask the model to retrieve a relevant ID to a question. Instead, we provide the ID in the input, and investigate the possibility of sequentially or randomly accessing the corresponding document content.

3 Investigating Sequential and Random Memory Access

In this section, we investigate the ability of a language model to sequentially or randomly access its memory stored in the parameters. First, we provide formulations of language models serving as a memory bank of passages (§3.1). Within this framework, we define *sequential memory access* as the process of starting from the beginning of a memorized passage and progressively generating subsequent content. In contrast, we conceptualize *random memory access* as the model's ability to initiate recall from any chosen location in a memorized passage and accurately regenerate the subsequent content. Based on these definitions, we first investigate the model's sequential memory access ability by requiring it to recite full passages word by word (§3.2). Next, we test the random memory access ability of the model by asking it to recite selected sentences from memorized passages (§3.3). We further assess the model's random access proficiency through a more challenging task involving question answering (§3.4).

3.1 Task Formulation

We abstract the language model as a memory bank and investigate its sequential or random access ability. We adopt a simple definition of a memory bank as a key–value store $\mathcal{D} = \{k_i : p_i\}$, where k_i represents a unique identifier (ID) assigned to the content of the i-th passage[1].

There are two core functions that a memory bank needs to support: *reading* and *writing*. Given that our memory bank is embodied as a language model, it is not straightforward to write and read the model's memory. Following previous work (Zhu and Li, 2023; Wang et al., 2021), for *writing* to the memory bank, we use fine-tuning to update the model's parameters. For *reading*, we use prompting to elicit the model's memory. Specifically, for each passage p_i with its corresponding identifier k_i, we create two types of data instances: **writing**, $S_{write}(k_i, p_i)$ and **reading**, $S_{read}(k_i) \rightarrow p_i$, where S_{write} and S_{read} denote the prompts detailed in Appendix A.1.

As the primary goal of our study is to test whether the model can read (access) its stored content sequentially or randomly, we vary the *reading* function across different experiments. Given a corpus consisting of M passages, we split the corpus into two subsets: T training passages and $V = M - T$ validation passages. We adopt a mixed training strategy as described by Zhu and Li (2023): During the training stage, we include S_{write} and S_{read} instances of T training passages, as well as S_{write} instances of V validation passages. Our objective is for the model to learn to associate each identifier with its passage content by training on the reading and writing instances of the training passages. During evaluation, we prompt the model with the S_{read} instances of the V validation passages to test the model's memory access pattern.

3.2 Sequential Access: Full Recitation

We test the sequential access ability of the language model by asking it to reproduce the full passage content. Specifically, given an ID, the model is prompted to start from the beginning of the corresponding memorized passage and generate tokens consecutively. We evaluate the model's performance to reproduce the content on the V validation passages, which requires the model to both memorize the passage content and sequentially access the memory with the provided key.

Setup. To investigate whether the model can handle identifiers and passage content of different types, we set $T = 400$ and $V = 40$ and consider the following variations. For the type of passage content p, we examine two categories: (1) natural language (*NL*), comprising Wikipedia paragraphs from SQuAD (Rajpurkar et al., 2016), and (2) random strings (*Rand*), where each *NL* passage is substituted with a space-separated alphanumeric string maintaining the same number of tokens. Regarding the type of k (i.e., passage IDs), we explore three forms: (1) numerical strings (*Num*), such as

[1]We use "document" and "passage" interchangeably to refer to a chunk of text.

	Title (ID)	Num (ID)	Rare (ID)
psg=*NL*	96.2 / 85.0	96.7 / 95.0	73.4 / 72.5
psg=*Rand*	96.7 / 95.0	96.7 / 95.0	96.7 / 95.0

Table 1: BLEU / Exact Match scores of reading from memory with different types of IDs and passage content.

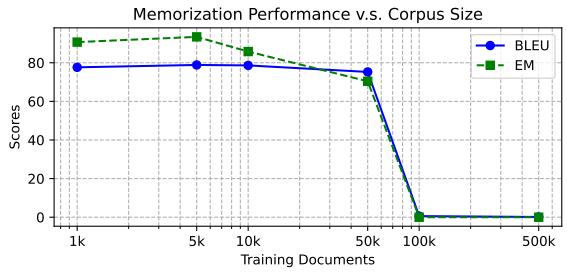

Figure 2: EM and BLEU for reading validation passages, varying the number of training passages. We calculate EM using only the first 25 tokens, as the model tends to continue generation beyond the max passage length (25).

'#123'; (2) rare random tokens (*Rare*), adopting the approach of Ruiz et al. (2022) by random sampling three infrequent tokens; (3) article title (*Title*) of the Wikipedia page to which the passage belongs.

We adopt the GPT2-large model (Radford et al., 2019) with 774M parameters as the base model. For better string memorization ability (Stevens and Su, 2023), we use a pretrained checkpoint[2] instead of training the model from scratch. We fine-tune the model for 100 epochs to ensure that the model fully converges, with a learning rate of 3×10^{-5}. We measure memorization using both the BLEU score (Papineni et al., 2002) and the Exact Match (EM) score, indicating the similarity between the generated content and the ground-truth passage.

Discussion. Table 1 shows that the model is able to sequentially access memorized content, with high BLEU and EM on validation passages. The model's sequential access capability is further demonstrated by its adaptability to varying types of IDs and passages. Specifically, using titles or numbers as keys for natural language passages achieves higher performance than using rare tokens. We suspect that models might have difficulty associating rare tokens with the natural language content. Remarkably, the model's access ability extends to passages composed of random characters ($Rand$).

To further test the memory capacity of the model, we carry out an additional experiment where we set the passage type to $Rand$ and identifier type to $Rare$ and construct passages each with 25 random tokens. As illustrated in Figure 2, we fix V as 1k and increase T gradually from 1k to 500k to examine the ability of sequential memory access.

We observe that even with a training passage count of 50k, the GPT2-large model accurately reproduces over 70% of memorized validation passages. However, there is also a bottleneck in parametric memory: the performance drops to nearly zero when the passage count exceeds 100k. We attribute this bottleneck to the difficulty in training, as the model fails to converge on memorizing all the passages. Therefore, in subsequent experiments, we carefully manage the corpus size to ensure that the model memorizes all passages.

3.3 Random Access: Selective Recitation

Selective recitation is a straightforward synthetic task: asking the language model to reproduce a specific sentence of a memorized passage. This task is designed for its simplicity, as it does not require the model's understanding of passage content. The focus is solely on the model's capacity to access segments in a memorized passage. Successful random access would be indicated by the model's ability to reproduce any sentence from within memorized passages, regardless of position.

Setup. We follow Mallick et al. (2023) to place markers at the boundaries of each sentence, obtained by the NLTK sentence splitter[3]: a passage is formatted as "*[0] sent0 [0] [1] sent1 [1], ...,*". In this case, the model only needs to learn to copy the content between these markers. Our selective recitation task requires the model to recite the j-th sentence of passage p_i based on the given passage ID k_i. The reading function is now $S_{read}(k_i, j) \rightarrow p_i[j]$, such as "*What is sentence [1] of Document #2033?*" shown in Figure 1. For reference, we also test the model's performance in a baseline where the passage content is provided in the context window.

As we are testing for exact memorization, we use BLEU and EM scores to evaluate the model. Similar to §3.2, we use $T = 400$ training and $V = 40$ validation passages, with 1994 sentences and 200 sentences respectively. We set the type of ID to be $Title$ and only include passages with more

[2] https://huggingface.co/gpt2

[3] https://www.nltk.org/api/nltk.tokenize.sent_tokenize.html

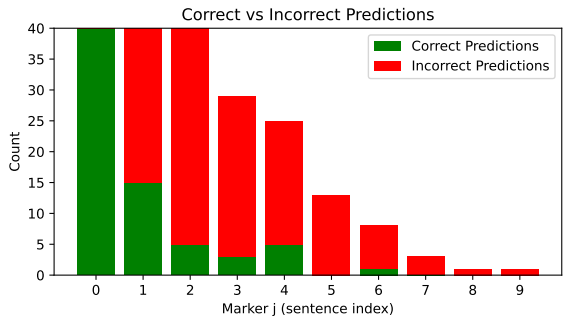

Figure 3: A stacked bar plot showing the accuracy of ID-guided sentence recitation with different marker numbers. The performance decreases significantly as the sentence index grows, revealing the model's incapability in accessing middle sentences.

than 3 sentences. All other hyperparameters stay the same as §3.2.

Discussion. We find that providing the passage ID does not enable the model to selectively recite the requested sentences. It scores poorly with a low EM of 34.5 and a 47.1 BLEU score, in contrast to the much higher 97.0 EM and 97.3 BLEU when the passage content is included in the context. A detailed analysis in Figure 3 reveals that the correct predictions are largely reciting the first sentence ($j = 0$). This verifies that the model can sequentially access the content to reproduce the first sentence. However, as the marker index increases, the model is required to skip preceding sentences and directly access a sentence in the middle of a passage. The model's performance sharply declines, indicating its inability to randomly access middle or later sentences in memorized passages.

3.4 Random Access: Grounded Question Answering

Building on our earlier finding §3.2 that the model can memorize many passages each linked to a unique ID, we embark on a more pragmatic task: question answering grounded in a specific passage ID. This task aims to evaluate whether the model can provide answers to questions by extracting a span from its memory. For instance, a question might be framed as "*According to Document #3022, in what year did Chopin become a French citizen?*" and the answer is "*1835*" in the passage with ID *#3022*. We hypothesize that if LMs are capable of random memory access, they should navigate to the corresponding passage using the provided ID and extract the relevant span to answer the questions.

Setup. We experiment with the well-known SQuAD-v1 (Rajpurkar et al., 2016) dataset because many of its questions are closely dependent on the passage, such as "*How did the war start?*". Without reference to an article, the question can be ambiguous and unanswerable. This design compels the model to depend on the memorized IDs and passages rather than pre-existing knowledge. We explore the grounded QA task with variants of providing (1) the ID of the golden passage with the answer, (2) a random non-golden ID and (3) no ID. For comparison, we also consider the setups that do not involve writing passages to the model's parametric memory. These include (1) closed-book QA, where the model is fine-tuned solely on QA pairs, serving as a lower-bound baseline to assess the model's reliance on prior knowledge for answering questions, and (2) open-book QA, where the golden passage content is concatenated with the question, setting the upper limit of extractive QA performance.

We experiment with different types of passage IDs. To ensure the uniqueness of using titles as passage IDs, we select $T = 442$ passages and $V = 48$ passages from the full SQuAD dataset, with over 2,000 and 300 questions respectively. The model is evaluated on F1 and EM following the original SQuAD evaluation script. The other hyperparameters are the same as mentioned in §3.2.

Discussion. The results are presented in Table 2 (the settings with "+Recitation" are discussed in later sections). As expected, the model performs the best in the open-book setting, as it only needs to locate the answer in the golden passage. In contrast, the closed-book QA setup yields the worst performance, as the model has no access to passages and relies solely on its parametric knowledge stored during pretraining.

Interestingly, the form of the provided passage ID has minimal impact on performance. We observe similar performance regardless of whether the golden ID is provided, except when the type of ID is *Title*. In this case, providing a random incorrect ID harms performance. We suspect that this is because the title is usually an entity related to the passage topic, therefore offering useful clues. In cases where the ID does not carry semantic meaning (i.e., *Rare* and *Num*), the correctness or presence of the ID does not significantly affect the performance, which remains substantially below the open-book setting, despite the model

Setup	Title (ID Type)		Rare (ID Type)		Num (ID Type)	
	EM	F1	EM	F1	EM	F1
w/o passage memorization						
Closed-Book QA (lower bound)	9.0	16.6	9.0	16.6	9.0	16.6
Open-Book QA (upper bound)	73.7	79.3	73.7	79.3	73.7	79.3
w/ passage memorization						
Grounded QA *w/ Golden ID*	26.7	35.6	20.7	28.7	24.3	32.6
↪ + Recitation	**59.7**	**68.0**	**54.7**	**62.1**	**57.7**	**66.2**
Grounded QA *w/ Random ID*	20.7	28.9	20.7	28.3	23.3	31.6
↪ + Recitation	16.0	20.4	18.7	23.6	18.7	23.1
Grounded QA *w/o ID*	22.0	31.0	22.0	31.0	22.0	31.0
↪ + Recitation	26.3	33.1	26.3	33.1	26.3	33.1

Table 2: EM and F1 scores for grounded question answering tasks, as well as baselines on closed-book and open-book QA. Numbers in bold represent the best performance in the grounded QA setting.

Setup	BLEU	EM
Baseline	47.1	34.5
↪ + Duplication (*dup-J*)	36.0	23.5
↪ + Recitation	99.3	98.5
↪ + Permutation (*first*)	**100.0**	**100.0**
↪ + Permutation (*random*)	98.0	97.0

Table 3: BLEU score and EM score of selective sentence recitation experiments after introducing passage recitation and permutation.

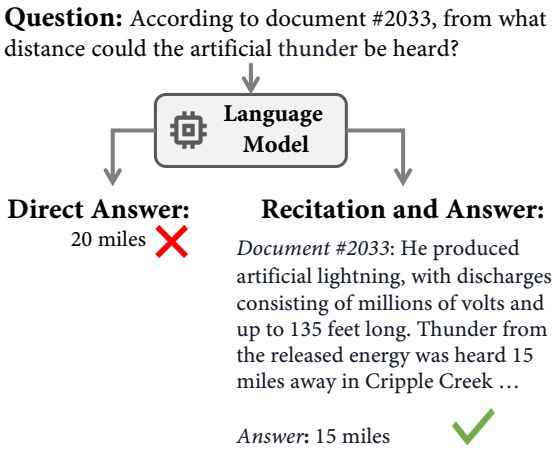

Figure 4: A illustration of the recitation method. The model first recites the corresponding passage content and subsequently extracts the answer in the context, in contrast to directly answering the question.

memorizing all passages. This further validates the model's inability to effectively access random memory, as it struggles to extract the answer even when provided with a correct passage ID.

In summary, our findings validate the hypothesis that LMs can effectively function as a memory bank, enabling sequential access to its memory. However, there are significant limitations in the model's ability to randomly access its memory. Across both the simple selective recitation and the complex grounded question-answering tasks, the model consistently fails to accomplish the tasks by leveraging its memory, despite being explicitly provided with the corresponding passage IDs.

4 Mitigating Random Access Challenge

Our earlier experiments show that in general, language models perform well in sequentially accessing their parametric memory, but encounter challenges in random memory access. This naturally raises the question: How can we mitigate the shortcomings in random memory access?

4.1 Proposed Method

To address the challenge, we start from the two operations supported by LMs as a memory store: reading and writing. During the writing phase, we hypothesize that performing *permutation* on the passage content can naturally enhance the model's random access ability: any part of the content can be the starting point of a memorized sequence. In this setup, we change the sequential order of passage content to achieve random access.

On the other hand, during the reading phase, leveraging the model's context window presents a viable strategy. The attention mechanism (Vaswani et al., 2017) enables the model to access any to-

ken within the context window, thereby inherently supporting random access (Packer et al., 2023; Ge et al., 2023). For tasks with a given ID, we could ask the model to sequentially *recite* the passage first, place it within the context, and subsequently query the model to perform span extraction tasks utilizing this context, as illustrated in Figure 4. Our subsequent experiments are designed to evaluate the effectiveness of these two methods. Through empirical evaluation, we validate that content permutation during writing or recitation during reading can largely mitigate the challenge of random memory access and enhance performance.

Setup. We extend the earlier experiments by integrating recitation and permutation into the respective reading and writing stages.

First, we add a setup to the selective sentence recitation task: Based on the given ID, the model is tasked to first recite the entire content of the corresponding passage and then the specific sentence, altering the reading operation to $S_{read}(k_i, j) \rightarrow (p_i, p_i[j])$. Similarly, for the grounded QA task, we ask the model to recite the passage associated with the input passage ID, before answering the question. In the setup without an ID, the model is still trained to recite the golden passage.

To explore the effect of permutation during the writing stage, we perform permutation among sentences in a passage to create diverse S_{write} instances. For a J-sentence passage, we test: (1) *first*, moving each sentence to the passage's beginning to create J unique instances; (2) *random-k*, randomly shuffling the sentences k times to create k instances, where k is set to 4 by default. To show that the effect of permutation is not simply due to more training data, we also include a baseline *dup-J*, where each passage is duplicated J times in training data.

Discussion. Reciting the passage content effectively boosts the performance of selective recitation, as evidenced in Table 3. With recitation, the model first sequentially accesses the content from its memory using the provided passage ID and subsequently loads this passage in the context to allow for random access. Conditioned on the recited content in the context, the model can therefore easily identify the correct sentence.

Similarly, explicitly reciting the golden passages markedly enhances question-answering performance, as shown in Table 2 (+Recitation). This observation is consistent across all three types of passage IDs. Conversely, intentionally prompting the model to recite a random passage leads to a decline in performance. This is likely because random passages introduce irrelevant information and confuse the model. Surprisingly, the recitation of relevant passages benefits performance even without an ID, although the improvement is smaller than with the golden ID. This verifies the effectiveness of recitation in more general settings of question answering.

Another way of enhancing random access is to perform permutation of sentences, as presented in Table 3. Simply bringing every sentence to the start of the passage once (*first*) or randomly permuting the sentences many times (*random*) helps to solve the challenge of accessing the middle content of a passage. In contrast, simply duplicating the original passage does not contribute to enhanced random access. We also observe that permutation during writing time enhances grounded QA performance (Table 4), which monotonically increases with the number of random permutations. However, it is noteworthy that permutation does not alter the inherent sequential access pattern of parametric memory. Rather, by permuting the sentences and disrupting their original order, we allow more sentences in the middle or the end of a passage to be sequentially accessible via the ID.

We also verify that the conclusions are generalizable to larger decoder-only LMs. In Appendix D, we observe similar challenges in random access in Qwen1.5-4b (Bai et al., 2023), and Llama2-7B (Touvron et al., 2023b) across different tasks. Moreover, we observe that such challenges could be effectively mitigated by our proposed methods of recitation and permutation.

5 Case Study: Open-Domain Question Answering

Our findings indicate that language models struggle with random memory access, unless the memory is explicitly recited and thus loaded into the context which can be accessed randomly. Building on this insight, we extend our study to the task of open-domain question answering, a challenging task that requires the model to first retrieve relevant memories and reason over them. This is different from previous experiments as the passage IDs are no longer provided as the input: The reading operation becomes $S_{read}(q) \rightarrow ans$. The model therefore needs to find relevant passages to the query without

Setup	Title (ID Type)		Rare (ID Type)		Num (ID Type)	
	EM	F1	EM	F1	EM	F1
Grounded QA w. *Golden ID*	26.7	35.6	20.7	28.7	24.3	32.6
↪ + Duplication (*dup-J*)	26.7	36.8	20.7	28.3	22.3	30.6
↪ + Permutation (*first*)	27.7	39.8	27.0	**37.7**	27.7	37.7
↪ + Permutation (*random-1*)	25.7	35.0	19.0	27.5	19.7	28.1
↪ + Permutation (*random-2*)	26.0	35.6	25.7	33.7	23.7	32.8
↪ + Permutation (*random-4*)	29.7	38.5	25.3	35.6	25.0	34.2
↪ + Permutation (*random-8*)	**31.3**	**40.1**	**27.7**	36.7	**29.0**	**38.3**

Table 4: The EM and F1 score of performing sentence permutation during the writing phase. *random-k* means that permutation is performed k times.

	NQ			Hotpot QA		
	EM	F1	Recite BLEU	EM	F1	Recite BLEU
Closed-Book QA	10.1	14.8	-	13.1	20.1	-
Closed-Book QA *w. Mixed Training*	12.6	18.2	-	15.7	22.8	
↪ + Recitation	**16.1**	**20.1**	28.6	**21.0**	**28.4**	51.3
Closed-Book QA *w. Continual Training*	10.3	15.5	-	15.1	22.4	-
↪ + Recitation	13.4	16.9	25.6	18.1	25.2	48.3

Table 5: EM and F1 of open-domain question answering datasets. We report the BLEU score of the recitation when the model is trained to recite the passage first and then offer an answer. Best performances are in bold.

the aid of passage IDs, which is a non-trivial task (Pradeep et al., 2023). As the goal of our study is not on retrieval performance and our earlier results (§3.3) show that the model has limited memorization capacity, we reduce the difficulty of retrieval by limiting the number of passages written to the model's memory: we only include positive passages that contain answers to at least one question.

We aim to test the model's ability to perform random access in real applications. Specifically, we investigate whether the model, having memorized many passages, can accurately extract answers from its memory. Similar to the previous experiments, we also aim to observe the difference in the model's performance when it is trained to recite relevant passages and subsequently answer the question. We opt not to experiment with permutation due to the high training cost associated with sentence permutation across a large number of passages, and leave this avenue for future work.

5.1 Experimental Setup

We use Natural Questions (Kwiatkowski et al., 2019) processed by Karpukhin et al. (2020) for single-hop QA, selecting 6000 training and all of the 6489 validation questions, with a total of 10.9k passages. For multi-hop question answering, we use HotpotQA (Yang et al., 2018) where each question has two golden passages. We select 8k training and all the 7405 validation questions in the *distractor* subset, with a total of 26.9k passages.

We start from a baseline setup where the training only involves QA pairs, i.e., closed-book QA, which evaluates the model's prior knowledge gained from pretraining. Next, we consider two types of training strategies to write the passages into the memory. In the *mixed* setting, the model is fine-tuned on a mixture of the S_{write} instances of all passages and training QA pairs. In the *continual* setting, the model is fine-tuned on S_{write} instances of all passages first, followed by fine-tuning on QA. To test the effectiveness of recitation, we also include settings where the model is trained to recite the golden passage(s) before answering.

As the task requires the model to perform both passage retrieval and question answering, we expect that the model size should be sufficiently large. Therefore, we upgrade our LLM to GPT2-XL with 1.5B parameters. In the *mixed* setting, we train the model for 20 epochs with a learning rate of 3e-5.

In the *continual* setting, we first train 20 epochs on the passages, followed by another 20 epochs on QA pairs. We report the best performance based on the EM score on validation questions.

5.2 Results and Discussion

Table 5 demonstrates that writing golden passages into the model's memory leads to improved performance over the baseline closed-book setting, with either mixed or continual training. This aligns with our expectations, as we deliberately inject passages containing the answers to the questions into the memory, enriching the model's knowledge.

Moreover, recitation significantly enhances the model's ability to utilize and access memorized passages, leading to a noticeable improvement in performance. This is observed in both the mixed and continual training settings. The exact match score increases significantly by more than 3% in both single and multi-hop QA. When the model explicitly recites the passages and loads them into the context for random access, the original open-domain QA task is reduced to an easier task of extractive QA. However, the low recitation BLEU score suggests that the model does not always accurately recite the golden passage. We expect that the performance could be further enhanced if it can accurately retrieve relevant passages from memory.

The mixed training strategy outperforms the continual training setup. This is likely because the model's memory of passage content is constantly refreshed in mixed training. In contrast, during continual training, the second stage only involves QA pairs on training passages, potentially leading to fading memory of validation passages. Consequently, the recitation becomes less accurate, as shown by a decrease in the BLEU score.

Our results are consistent and complementary to the findings of Wei et al. (2023) and Sun et al. (2023): introducing intermediate steps or generating relevant passages helps to improve model performance on various tasks. We provide an alternative interpretation for this phenomenon: loading the parametric memory into the context window facilitates enhanced random access to memorized information, and the model benefits from such enhancements.

6 Conclusion

We empirically study how language models access their parametric memory. Our experiments on both synthetic and realistic data demonstrate that while language models can adequately reproduce memorized content in a sequential manner, they struggle with the random access of segments in the middle of memorized content. We identify two effective strategies of recitation and permutation to mitigate the limitation of random memory access. Furthermore, through a controlled case study on open-domain question answering, we illustrate that allowing the model to recite and randomly access its memory significantly improves performance. Overall, our study not only provides a deeper understanding of memory access patterns in language models, but also highlights the implications of limited random memory access ability in practical application of language models.

Limitation

In this work, we mainly explore the memory access pattern of decoder-only language models. Future research is needed to understand whether our conclusions apply to other types of language models based on transformers such as encoder-only models and encoder-decoder models. Furthermore, we do not extend our study to larger models beyond 7 billion parameters due to computing resource constraints. It might be worthwhile to explore further scaling behavior of memory access patterns in even larger language models. In addition, we mainly conduct controlled experiments on a text corpus of fixed size. Further investigation may be needed to explore how the findings can apply to large-scale pretraining corpus and their implications on pretrained language models.

Ethical Considerations

As the method suggests techniques to enhance access to the model's memory, there could be malicious use of the recitation method to extract sensitive personal information from the model's memory. We use open-source English datasets including questions and contexts from SQuAD-v1 (Rajpurkar et al., 2016), Natural Questions (Kwiatkowski et al., 2019), and Hotpot QA (Yang et al., 2018). We also use open-source English language models, GPT2, with different sizes (Radford et al., 2019). There might be potential biases in these datasets and models.

Acknowledgements

We appreciate the suggestions and comments by Do Xuan Long, Yanxia Qin, Yisong Miao, and other members of NUS WING. Tongyao Zhu is supported by the Industry PhD Program of Sea AI Lab.

References

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona T. Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. *ArXiv*, abs/2204.06031.

Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Physics of language models: Part 3.2, knowledge manipulation. *ArXiv*, abs/2309.14402.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Angels Balaguer, Vinamra Benara, Renato Luiz de Freitas Cunha, Roberto de M. Estevão Filho, Todd Hendry, Daniel Holstein, Jennifer Marsman, Nick Mecklenburg, Sara Malvar, Leonardo O. Nunes, Rafael Padilha, Morris Sharp, Bruno Silva, Swati Sharma, Vijay Aski, and Ranveer Chandra. 2024. RAG vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2023. The reversal curse: LLMs trained on "A is B" fail to learn "B is A".

Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. In *Advances in Neural Information Processing Systems*, volume 35, pages 31668–31683. Curran Associates, Inc.

Zied Bouraoui, José Camacho-Collados, and Steven Schockaert. 2019. Inducing Relational Knowledge from BERT. In *AAAI Conference on Artificial Intelligence*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. 2021. Knowledgeable or educated guess? revisiting language models as knowledge bases. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1860–1874, Online. Association for Computational Linguistics.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Xiaodong Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2020. Extracting training data from large language models. In *USENIX Security Symposium*.

Yingqiang Ge, Yujie Ren, Wenyue Hua, Shuyuan Xu, Juntao Tan, and Yongfeng Zhang. 2023. LLM as OS, agents as apps: Envisioning AIOS, agents and the AIOS-agent ecosystem. *ArXiv*, abs/2312.03815.

Benjamin Heinzerling and Kentaro Inui. 2021. Language models as knowledge bases: On entity representations, storage capacity, and paraphrased queries. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1772–1791, Online. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022. Are large pre-trained language models leaking your personal information? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2038–2047, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv*, abs/2310.06825.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language

models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5591–5606, Toronto, Canada. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Hyunji Lee, JaeYoung Kim, Hoyeon Chang, Hanseok Oh, Sohee Yang, Vladimir Karpukhin, Yi Lu, and Minjoon Seo. 2023a. Nonparametric decoding for generative retrieval. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12642–12661, Toronto, Canada. Association for Computational Linguistics.

Sunkyung Lee, Minjin Choi, and Jongwuk Lee. 2023b. GLEN: Generative retrieval via lexical index learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7693–7704, Singapore. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Yan Liu, Yu Liu, Xiaokang Chen, Pin-Yu Chen, Daoguang Zan, Min-Yen Kan, and Tsung-Yi Ho. 2024. The devil is in the neurons: Interpreting and mitigating social biases in language models. In *The Twelfth International Conference on Learning Representations*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Prabir Mallick, Tapas Nayak, and Indrajit Bhattacharya. 2023. Adapting pre-trained generative models for extractive question answering. *ArXiv*, abs/2311.02961.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems*, volume 35, pages 17359–17372. Curran Associates, Inc.

Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking search: making domain experts out of dilettantes. *SIGIR Forum*, 55(1).

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

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2024. Fine-tuning or retrieval? comparing knowledge injection in LLMs.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as operating systems. *ArXiv*, abs/2310.08560.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Ronak Pradeep, Kai Hui, Jai Gupta, Adam Lelkes, Honglei Zhuang, Jimmy Lin, Donald Metzler, and Vinh Tran. 2023. How does generative retrieval scale to millions of passages? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1305–1321, Singapore. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Ruiyang Ren, Wayne Xin Zhao, Jing Liu, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. TOME: A two-stage approach for model-based retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6102–6114, Toronto, Canada. Association for Computational Linguistics.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2022. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22500–22510.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023. REPLUG: Retrieval-Augmented Black-Box Language Models.

Till Speicher, Aflah Mohammad Khan, Qinyuan Wu, Vedant Nanda, Soumi Das, Bishwamittra Ghosh, Krishna P. Gummadi, and Evimaria Terzi. 2024. Understanding the mechanics and dynamics of memorisation in large language models: A case study with random strings.

Samuel Stevens and Yung-Chun Su. 2023. Memorization for good: Encryption with autoregressive language models. *ArXiv*, abs/2305.10445.

Zhiqing Sun, Xuezhi Wang, Yi Tay, Yiming Yang, and Denny Zhou. 2023. Recitation-augmented language models. In *International Conference on Learning Representations*.

Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. *ArXiv*, abs/2205.10770.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open Foundation and Fine-Tuned Chat Models.

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

Cunxiang Wang, Pai Liu, and Yue Zhang. 2021. Can generative pre-trained language models serve as knowledge bases for closed-book QA? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3241–3251, Online. Association for Computational Linguistics.

Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Allen Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2023. A neural corpus indexer for document retrieval.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Paul Youssef, Osman Koraş, Meijie Li, Jörg Schlötterer, and Christin Seifert. 2023. Give me the facts! a survey on factual knowledge probing in pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15588–15605, Singapore. Association for Computational Linguistics.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *International Conference for Learning Representation (ICLR)*.

Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. 2023. Scalable and effective generative information retrieval. *ArXiv*, abs/2311.09134.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2023. Large language models as commonsense knowledge for large-scale task planning. In *RSS 2023 Workshop on Learning for Task and Motion Planning*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: Less is more for alignment.

Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Yu Wu, Peitian Zhang, and Ji rong Wen. 2022. Ultron: An ultimate retriever on corpus with a model-based indexer. *ArXiv*, abs/2208.09257.

Zeyuan Allen Zhu and Yuanzhi Li. 2023. Physics of language models: Part 3.1, knowledge storage and extraction. *ArXiv*, abs/2309.14316.

Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *arXiv preprint arXiv:2206.10128*.

Noah Ziems, Wenhao Yu, Zhihan Zhang, and Meng Jiang. 2023. Large language models are built-in autoregressive search engines. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2666–2678, Toronto, Canada. Association for Computational Linguistics.

A Prompts

A.1 Full Recitation

Given a key-value pair (k_i, p_i), the prompts are as follows:

S_{write} = "`Article {`k_i`} , Content: {`p_i`}`"

$S_{read}(k_i) \rightarrow p_i$ = "`Article {`k_i`} : What is the content of this article?`" $\rightarrow$ "{p_i}"

A.2 Selective Recitation

In this experiment, we follow the same prompt of S_{write}, as described in Appendix §A.1, and only change S_{read}

S_{write} = "`Article {`k_i`} , Content: {`p_i`}`"

$S_{read}(k_i, j) \rightarrow p_i[j]$ = "`Article {`k_i`} : What is Sentence [{`j`}] of this article?`" $\rightarrow$ "{$p_i[j]$}"

A.3 Grounded Question Answering experiments

In this experiment, we follow the same prompt of S_{write}, as described in Appendix §A.1, and only change S_{read} to questions related to p_i. $S^{rc}_{read}(k_i, q)$ represents the instances where the recitation of the passage content is prepended before the answer.

S_{write} = "`Article {`k_i`} , Content: {`p_i`}`"

$S_{read}(k_i, q) \rightarrow ans$ = "`Article {`k_i`} \n Question: {`q`} \n Answer: `" $\rightarrow$ "{ans}"

$S^{rc}_{read}(k_i, q) \rightarrow (p_i, ans)$ = "`Article {`k_i`} \n Question: {`q`} \n Answer: `" $\rightarrow$ "`{`p_i`} || Answer: {`ans`}`"

A.4 Open-Domain Question Answering

In the setup of open-domain question-answering experiments, we no longer have a pre-assigned ID for each document. Our S_{write} becomes:

$S_{write}(p_i)$ = "`Document: {`p_i`}`"

Similarly, the reading operation now does not have any ID associated with it, but only a question. It becomes:

$S_{read}(q) \rightarrow ans$ = "`Question: {q} \n Answer: `" $\rightarrow$ "`{ans}`"

In the case of recitation, our prompts for training the model include the passage containing the answer.

$S^{rc}_{read}(q) \rightarrow (p_{golden}, ans)$ = "`Question: {q}`" $\rightarrow$ "`Related documents: {`p_{golden}`} \n Answer: {`ans`}`"

B Additional Selective Recitation Experiments

We provide additional experimental results for our selective recitation task of reciting sentences. All of the experiments lead to a consistent conclusion that the model is unable to randomly extract a sentence from a memorized passage.

In both of the experiments below, we include setups of (1) in-context: the passage is included in the context window. (2) ID-guided: the basic version of the selective recitation task where a passage

ID is provided. and (3) with passage recitation: the passage is recited first before sentence recitation.

B.1 Reciting the first/second/last sentence

As a basic setting of the selective sentence recitation task, we ask the model questions like "`What is the [first/second/last] sentence of Article #123?`".

The results are shown in Table 6. The model almost always recites the first correctly, while recitation performance drops significantly for the second or last sentence. This shows that the model is performing sequential access: following the article ID, the model can only access content immediately after the ID – the first sentence. It is unable to directly access the second or last sentence.

We observe that even for the in-context setting where the passage is in the context window, the model does not perform perfectly, especially for extracting the last sentence. This is because the model also needs to learn what $first$, $second$ or $last$ means, which involves numerical reasoning ability to count the index. Therefore, in the main experiments, we put markers on both ends of a sentence to reduce the task difficulty.

B.2 Reciting the next/previous sentence

We perform experiments to find the sentence before and after an input sentence in a given passage. In other words, our S_{read} operation becomes $S_{read}(k_i, s_j) \rightarrow s_{j+1/j-1}$, where s_j is the input sentence. The results are shown in Table 7.

We notice that finding the sentence after the input sentence is always easy, while the reverse task is much more difficult. This also reveals that the model reads its memory sequentially. It is unable to randomly access the sentence before the input s_j, even if the target sentence is adjacent to the input sentence.

C Additional Open-Domain Question Answering Experiments

To ensure that our conclusion is consistent with different dataset sizes, we vary the number of training and validation documents and questions to observe the performance difference. For NQ, we select 5k training and 5k validation QA pairs, forming a corpus containing around 9k passages. For Hotpot QA, we select 5k training and 5k validation questions in the *distractor* subset, with a total of 18.2k passages.

In Table 8, we obtain similar conclusions that recitation greatly enhances question-answering performance, and using a mixed training strategy is better than continual training because of the increase in recitation score.

D Experiments on Large Language Models

To show that our conclusions are generalizable to larger models, we conduct additional experiments on Qwen1.5-4b (Bai et al., 2023) and Llama2-7b (Touvron et al., 2023b), with approximately 4 billion and 7 billion parameters each. These models belong to the same decoder-only language model family as GPT2. For efficiency purposes, we use LoRA (Hu et al., 2022), a parameter-efficient approach to fine-tune the large language models.

Selective Recitation Table 9 presents the results of the selective recitation task, which are consistent with the findings on the smaller GPT2-large. Performing recitation or permutation enhances random access to the passages and solves the task. However, compared to GPT2, there is a performance drop in the baseline and random permutation settings. This is because we use semantically meaningful passage identifiers, *Title*, and larger models might have memorized many passages related to the title entity during pretraining. Therefore, it is more likely to generate sentences that are not within our predefined set of passages, which lowers the performance since the task requires an exact reproduction.

Grounded Question Answering In Table 10, we show the grounded QA performance when the ID type is the passage title. We find that including the golden title (*w/o ID* v.s. *w/ Golden ID*) only slightly improves Qwen's performance, while Llama benefits more from the provided title, which is often an entity related to the question. We suspect that this is because larger models learn more knowledge about entities during pretraining. For both models, however, there is a significant performance increase in the recite-and-answer setup, showing the effectiveness of recitation.

Open-Domain Question Answering We further extend the case study of open-domain question answering to the aforementioned larger models. Different from the setup in Section 5.1, we do not include the *continual* setup as we empirically find the model suffers from catastrophic forgetting. Instead,

	Recite First		Recite Second		Recite Last	
	BLEU	EM	BLEU	EM	BLEU	EM
In-context	97.2	95.0	94.3	95.0	91.8	87.5
ID-guided	99.0	97.5	14.1	5.0	17.6	0.0
↪ + Recitation	99.6	95.0	98.8	87.5	98.7	85.0

Table 6: BLEU and EM scores of reciting the first, second or last sentence of a memorized passage.

	Recite Next Sentence		Recite Previous Sentence	
	BLEU	EM	BLEU	EM
In-context	98.0	96.0	82.7	79.0
ID-guided	86.9	81.0	20.1	18.5
↪ + Recitation	98.4	85.0	96.5	81.0

Table 7: BLEU and EM scores of reciting the next or previous sentence given an input sentence.

	NQ			**Hotpot QA**		
	EM	F1	Recite BLEU	EM	F1	Recite BLEU
Closed-Book QA	9.1	13.7	-	13.3	20.4	-
Closed-Book QA *w. Mixed Training*	11.5	17.2	-	15.9	23.6	
↪ + Recitation	**15.7**	**19.7**	29.1	**20.8**	**28.4**	50.9
Closed-Book QA *w. Continual Training*	10.3	15.5	-	15.1	22.8	-
↪ + Recitation	12.3	15.8	24.2	18.2	25.6	49.2

Table 8: EM and F1 of the model's QA performance on different subsets of NQ and Hotpot QA datasets. We report the BLEU score of the recitation when the model is trained to recite the passage first and then provide an answer. The bold numbers are the best-performing setup.

we introduce an additional setup (Closed-book QA + Recitation) where the model is finetuned on recite-and-answer instances (S_{read}), but no passages are written to the memory. This setup tests if the model memorizes relevant passages during pretraining and learns to recite them after finetuning. We use a learning rate of 1e-4 for all experiments.

In Table 11, we first observe that writing golden passages to the memory (*w. Mixed Training*) improves performance. We also observe that recitation is helpful for Llama2 on the NQ dataset. However, for other setups, we notice that triggering recitation is not helpful or even harmful to the performance. This is likely because the recitation BLEU scores are noticeably lower than the presented scores of GPT2-XL in Table 5, suggesting that the recited content has less overlap with the golden passages. We hypothesize that this is because larger models carry significantly larger parametric memory, and it is more challenging to precisely retrieve the correct passage to the question. In contrast, a smaller model like GPT2 has to rely on the passages written into the memory during the finetuning stage, thus limiting the scope of such retrieval. Overall, the results give additional insights that reciting passages accurately remains a challenge for larger models.

E Additional Training Details

We conduct all experiments in a cluster with NVIDIA Tesla A100 GPUs (with 40G or 80G memory). Experiments in §3.2 take a total of 48 hours on 4 GPUs. Selective sentence recitation experiments in §3.3 and §4 take a total of 41 hours on 4 GPUs. Grounded QA experiments take a total of 132 hours on 4 GPUs. The open-domain QA experiments need 3 days to complete with 32 GPUs.

We use the Hugging Face transformers library for all experiments. For the main experiments in GPT2 models, we use a learning rate of 3e-5. We

Setup	Qwen1.5-4b		Llama2-7b	
	EM	BLEU	EM	BLEU
Baseline	22.5	36.2	17.0	30.9
↪ + Recitation	**100.0**	**100.0**	95.5	98.6
↪ + Permutation (*first*)	99.5	100.0	**99.0**	**99.4**
↪ + Permutation(*random-4*)	90.0	92.7	86.5	90.4

Table 9: EM and BLEU scores of Qwen1.5-4b and Llama2-7b in the selective recitation task. Numbers in bold represent the best performance.

Setup	Qwen1.5-4b		Llama2-7b	
	EM	F1	EM	F1
w/o passage memorization				
Closed-Book QA	19.0	28.9	26.3	37.6
Open-Book QA	84.0	90.9	83.7	91.5
w/ passage memorization				
Grounded QA *w/o ID*	20.7	30.7	27.3	38.7
Grounded QA *w/ Golden ID*	22.7	34.4	35.0	47.5
↪ + Recitation	**45.7**	**54.2**	**63.7**	**70.1**

Table 10: EM and F1 scores for grounded question answering of Qwen1.5-4b and Llama2-7b. Numbers in bold represent the best performance in the grounded QA setting.

	Setup	NQ			Hotpot QA		
		EM	F1	Rec-BLEU	EM	F1	Rec-BLEU
Llama2-7b	Closed-book QA	26.4	39.0	-	24.1	33.8	-
	↪ + Recitation	30.3	40.1	15.7	22.7	31.4	22.2
	CBQA *w. Mixed Training*	27.6	**40.2**	-	**24.6**	**34.6**	-
	↪ + Recitation	**30.6**	39.8	17.4	22.4	30.7	23.8
Qwen1.5-4b	Closed-book QA	19.7	27.8	-	18.6	27.0	-
	↪ + Recitation	16.3	24.1	11.4	16.7	24.5	17.4
	CBQA *w. Mixed Training*	**21.2**	**29.4**	-	**19.3**	**28.3**	-
	↪ + Recitation	17.4	24.1	13.6	18.4	25.1	32.7

Table 11: Open-domain question answering performance of Qwen1.5 and Llama2. CBQA is short for closed-book QA and Rec-BLEU means the BLEU score of the generated recitation. The basic version of Closed-book QA (Row 1 and 2) does not involve writing golden passages into the memory, thus the model needs to purely rely on parametric memory learned during pretraining.

set a constant learning rate schedule for the open-domain QA experiments. For all other experiments, we use a warm-up ratio of 0.05 and a linear decay learning rate. We evaluate the model's performance on the validation set at the end of each epoch and report the best-performing ones.