Navigating the Dual Facets: A Comprehensive Evaluation of Sequential Memory Editing in Large Language Models

Zihao Lin **Mohammad Beigi** **Lifu Huang**
Virginia Tech
{zihaol,lifuh}@vt.edu

Abstract

Memory editing (ME) has emerged as an efficient method to inject new facts into large language models (LLMs). However, previous studies on the evaluation and analysis of ME have two critical limitations: (1) They only evaluate the model edited once while in practice, the model usually needs to be continually edited; (2) The evaluation only focuses on simplistic factual triples (i.e., *subject, relation, object*) without considering other capabilities of LLMs such as logical reasoning, code generation and so on. In this paper, we address these limitations by extending the evaluation of edited LLMs to six broad categories of capabilities under the sequential editing setting. Experimental results reveal that the majority of ME methods that modify the parameters within LLMs systematically hurt their performance on all evaluation tasks after being edited a few times, while the few methods that preserve the model parameters but integrate additional modules into LLMs to inject new knowledge effectively maintain the LLMs' general capabilities. Further analysis also indicates that several strategies, including instruction tuning, editing deeper layers, and increasing the model size or the batch size for model editing, are beneficial to mitigate the damages on LLMs when edited sequentially.

1 Introduction

Memory Editing (ME) is introduced as an effective method to inject new knowledge into large language models (LLMs). Previous ME methods can be roughly divided into two categories: (1) parameter-modifying ME methods, for example, MEND (Mitchell et al., 2022), ROME (Meng et al., 2022a), and MEMIT (Meng et al., 2022b), which directly modify a small number of parameters within the model. (2) Parameter-preserving ME methods, for example, GRACE (Hartvigsen et al., 2022) and MELO (Yu et al., 2023), which integrate additional modules into the LLM architecture without altering the original model parameters.

Although ME has shown much promise, previous studies on evaluating and analyzing ME methods have two critical limitations. First, they only evaluate the performance of LLMs after every single editing. However, in practice, they usually need to be edited sequentially, i.e., sequential memory editing (SME), which edits the same model multiple times to continuously incorporate new knowledge. SME is more important in real application scenarios because new knowledge always appears over time. Second, it is essential to assess how ME influences the general capabilities of LLMs, including logical analysis, multilingual proficiency, code generation skills, and so on, which are usually overlooked in previous studies.

To address these limitations, our study comprehensively evaluates the general capabilities of memory-edited LLMs in sequential editing scenarios. Our findings indicate systematic damage of parameter-modifying ME methods to LLMs in a few sequential edits. On the contrary, the parameter-preserving ME method, such as GRACE successfully retains the fundamental capabilities of the model after 100 sequential edits. Subsequent experiments show some strategies to mitigate such damage from parameter-modifying ME methods, such as instruction tuning, editing deeper layers, and increasing the model size or the batch size for sequential model editing.

2 Experiments

The evaluation involves four distinct ME methods, including three parameter-modifying ME methods - MEND (Mitchell et al., 2022), ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b), and one parameter-preserving ME method - GRACE (Hartvigsen et al., 2022). We leverage three different checkpoints of LLaMA-2 (Touvron et al., 2023), consisting of `LLaMA-2-7B`, `LLaMA-2-7B-Chat` and `LLaMA-2-13B`. The evaluation framework spans six core capabilities of

Table 1: Evaluation of four ME methods on eight tasks under the sequential editing setting for the LLaMA-2-7B model. "ComQA" refers to the CommonsenseQA dataset. We randomly select 100 samples from ZsRE (Levy et al., 2017) as the edition dataset. The scores for the MMLU, BBH, and TyDiQA datasets are the mean values derived from all respective subsets.

Method	Edit #.	**MMLU**	**MBPP**	**MATH**	**BBH**	**TyDiQA**	**C3**	**ComQA**	**AX-b**	*Avg.*
LLaMA	0	46.8	18.2	3.4	38.4	26.8	32.1	49.6	45.9	32.7
Modifying-parameter Methods										
MEND	1	47.2	19.2	3.26	38.3	26.4	32.2	50.6	49.0	33.3
	20	35.2	0.0	0.0	4.2	9.8	14.9	11.0	26.5	12.7
	100	25.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.2
ROME	1	46.9	17.6	3.3	38.4	26.8	32.0	49.6	45.5	32.5
	20	34.3	18.4	2.6	33.8	24.1	28.9	20.6	51.5	26.8
	100	25.5	2.8	1.0	28.8	8.0	23.2	19.0	38.4	18.3
MEMIT	1	46.7	18.4	3.4	38.3	26.8	32.0	50.6	45.9	32.8
	20	25.3	16.6	1.9	32.4	19.5	15.5	19.7	31.2	20.3
	100	22.9	0.0	0.0	0.0	0.0	0.0	0.49	1.8	3.1
Preserving-parameter Methods										
GRACE	100	46.8	18.2	3.4	38.4	26.8	32.1	49.6	45.9	32.7

LLMs: Professional Knowledge, Common Sense Knowledge, Logical Reasoning, Reading Understanding, Multilingual Proficiency, and Code Generation, which consists of eight benchmark test datasets. More details of the evaluation datasets are shown in Appendix A.1.

3 Results

The experimental results indicate different influences between parameter-modifying and parameter-preserving ME methods on LLMs in sequential editing scenarios. In particular, the parameter-preserving ME method, GRACE, successfully retains the fundamental capabilities of the model after 100 sequential edits, with no performance decline observed in all downstream tasks. On the contrary, after approximately 20 sequential edits, parameter-modifying ME methods systematically damage all fundamental capabilities of LLMs. Compared to MEND and MEMIT, ROME shows less damage to LLMs after 100 sequential edits.

Figure 1 illustrates that all model checkpoints, regardless of their size, show performance degradation that correlates with the number of sequential edits. Interestingly, the post-editing model with more parameters suffers less damage to Code generation and Multi-language understanding capabilities. Besides, with the same number of parameters, the dialogue-instruction-tuned model becomes more robust on knowledge question-answering tasks such as MMLU and CommonsenseQA. This finding suggests that instruction tuning might play a positive role in safeguarding model capabilities against the detrimental effects of memory editing, although it does not entirely prevent performance losses.

As shown in Figure 2, different editing datasets influence LLMs differently in the sequential editing scenario. Besides, it also illustrates that some data points may damage LLMs more significantly than others, indicating the instability of parameter-modifying ME methods. Figure 3 indicates that editing layers closer to the output (deeper layers) results in a marginal decrease in performance while editing shallower layers leads to significant performance degradation. Besides, as illustrated in Figure 4, increasing the batch size of sequential editing, meaning that reducing the number of edits, can potentially mitigate some negative impacts when the total number of edits is the same. However, these measures do not entirely negate the observed performance decline. Our findings underscore the inherent complexity and challenges of ME approaches in meeting the demands of practical applications in the real world.

4 Conclusion

This study comprehensively evaluated two categories of memory editing methods across eight benchmarks for large language models in the sequential editing setting. We advocate the careful use of parameter-modifying ME methods in application scenarios due to the negative impact of

LLMs demonstrated in our study.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in ty pologically di verse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

Ahmad Ghazal, Tilmann Rabl, Minqing Hu, Francois Raab, Meikel Poess, Alain Crolotte, and Hans-Arno Jacobsen. 2013. Bigbench: Towards an industry standard benchmark for big data analytics. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*, pages 1197–1208.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2022. Aging with grace: Lifelong model editing with discrete key-value adaptors. *arXiv preprint arXiv:2211.11031*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast model editing at scale. In *International Conference on Learning Representations*.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2020. Investigating prior knowledge for challenging chinese machine reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:141–155.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2023. Melo: Enhancing model editing with neuron-indexed dynamic lora. *arXiv preprint arXiv:2312.11795*.

A Appendix

A.1 Evaluation Datasets

As shown in Table 2, we select eight benchmarks as evaluation datasets, including MMLU (Hendrycks et al., 2020), BBH (Ghazal et al., 2013), MATH (Hendrycks et al., 2021), SuperGLUE-AX-b (Wang et al., 2019), CommonsenseQA (Talmor et al., 2018), C3 (Sun et al., 2020), TydiQA (Clark et al., 2020), and MBPP (Austin et al., 2021). We leverage the *opencompass* codebase (Contributors, 2023), a widely recognized open-source repository, to evaluate post-editing LLMs. We adopt the Perplexity (PPL) mode for the evaluation of the MMLU dataset. For instance, in the MMLU dataset, each item comprises a question and four possible answers. We concatenate the question with each answer option to create four distinct input sequences. Subsequently, we compute the Perplexity for each sequence using the edited LLM under examination. A lower Perplexity score indicates higher model confidence in the corresponding sentence. We select the answer with the lowest score as the final prediction. Conversely, we utilize the Generation (GEN) mode for the evaluation of the remaining benchmarks. Specifically, for MATH, BBH, and TyDiQA, we ascertain the accuracy of the model's predictions against the ground truth following a post-processing procedure.

Regarding the code generation task MBPP, we employ Python's built-in *exec()* function to verify the execution of the generated code.

A.2 Evaluation Results

The appendix shows several figures about the evaluation results of different ME methods, model checkpoints, and evaluation datasets.

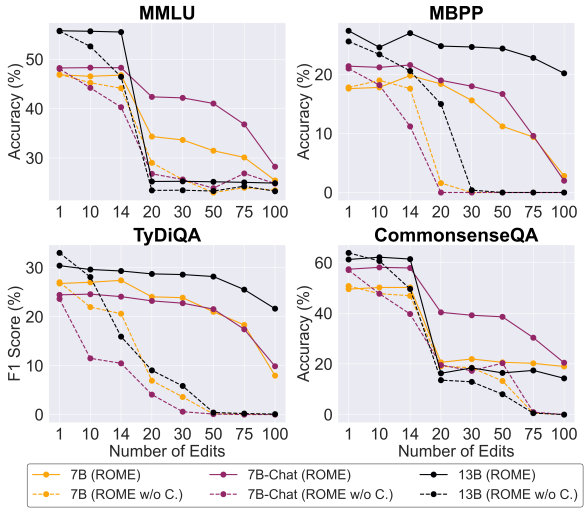

Figure 1: Evaluation Performance across three different checkpoints after sequential editing. ROME method applies 100,000 wiki data as constraints during editing models. It requires the post-editing model to remember the wiki knowledge. Here, we denote ROME w/o C. as the ROME method without the constraining.

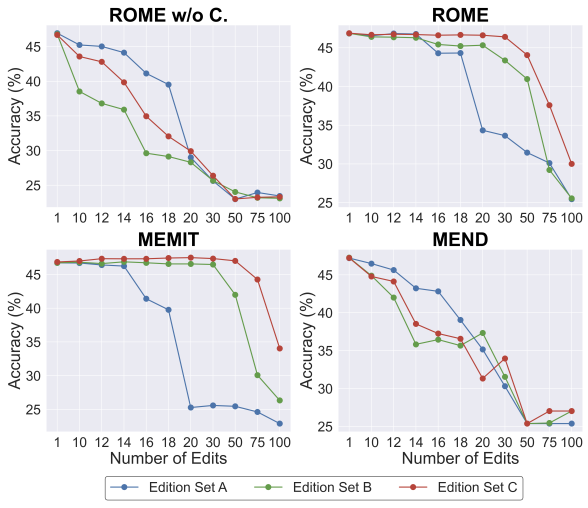

Figure 2: The performance of LLaMA-2-7B on the MMLU dataset after Editing on Different Sets. Each edition set randomly selected 100 samples from the ZsRE dataset without overlapping.

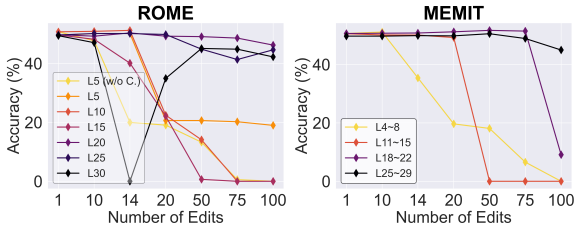

Figure 3: The performance of the LLaMA-2-7B model on the CommonsenseQA dataset utilizing the ROME and MEMIT as editing methods. LX represents editing the X-th layer of the model, while the LX-Y represents editing several layers between the X-th layer and the Y-th layer.

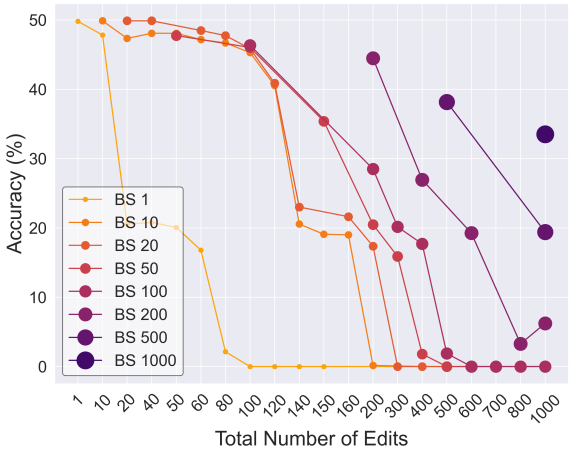

Figure 4: The performance of LLaMA-2-7B model on CommonsenseQA, utilizing MEMIT as the editing method, across different editing batch sizes. BS denotes the batch size. The x-axis represents the total number of sequential edits. For example, for the line of a batch size of 100, the first data point of this line lies in the total number of edits of 100, which only edits the model once.

Table 2: The details of downstream evaluation benchmarks.

Capability	Task	Datasets	#. Items	Metrics	Language	Mode	#. Shots
Professional Knowledge	High School / University Professional Examination	MMLU	15691	Acc.	English	PPL	5
Logical Reasoning	Mathematical Reasoning	MATH	5000	Acc.	English	GEN	4
	Comprehensive Reasoning	BBH	6511	Acc.	English	GEN	3
	Textual Entailment	AX-b	1104	Acc.	English	GEN	0
Common Sense Knowledge	Knowledge Question Answering	ComQA	1221	Acc.	English	GEN	8
Reading Understanding	Reading Understanding	C3	1825	Acc.	Chinese	GEN	0
Multilingual Proficiency	Multi-Language Question Answering	TyDiQA	6322	F1	13 languages	GEN	0
Code Generation	Code Generation	MBPP	500	Pass.	Code	GEN	3