Time Sensitive Knowledge Editing through Efficient Finetuning

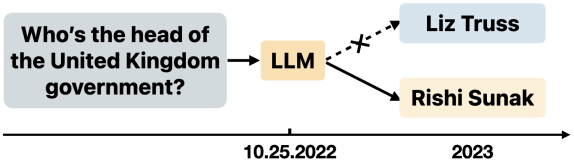
Xiou Ge[1], **Ali Mousavi**[1], **Edouard Grave**[2*], **Armand Joulin**[3*],
Kun Qian[4*], **Benjamin Han**[1], **Mostafa Arefiyan**[1], **Yunyao Li**[4*]
[1]Apple, [2]Kyutai, [3]Google Deepmind, [4]Adobe

Abstract

Large Language Models (LLMs) have demonstrated impressive capability in different tasks and are bringing transformative changes to many domains. However, keeping the knowledge in LLMs up-to-date remains a challenge once pretraining is complete. It is thus essential to design effective methods to both update obsolete knowledge and induce new knowledge into LLMs. Existing locate-and-edit knowledge editing (KE) method suffers from two limitations. First, the post-edit LLMs by such methods generally have poor capability in answering complex queries that require multi-hop reasoning (Zhong et al., 2023). Second, the long run-time of such locate-and-edit methods to perform knowledge edits make it infeasible for large scale KE in practice. In this paper, we explore Parameter-Efficient Fine-Tuning (PEFT) techniques as an alternative for KE. We curate a more comprehensive temporal KE dataset with both knowledge update and knowledge injection examples for KE performance benchmarking[1]. We further probe the effect of fine-tuning on a range of layers in an LLM for the multi-hop QA task. We find that PEFT performs better than locate-and-edit techniques for time-sensitive knowledge edits.

1 Introduction

The rapid development of Large Language Models (LLMs) has showcased their ability to generate human-quality responses and demonstrate reasoning capabilities (Brown et al., 2020; Chowdhery et al., 2022; OpenAI, 2023; Touvron et al., 2023; McKinzie et al., 2024; Wei et al., 2023), and it is bringing revolutionary changes across diverse industries. However, maintaining the factuality remains challenging for LLMs since their pre-training data are collected within a time range.



Govt	Head of govt	Start time	End time
United Kingdom	Liz Truss	+2022-09-06T00:00:00Z_DAY	+2022-10-25T00:00:00Z_DAY
United Kingdom	Rishi Sunak	+2022-10-25T00:00:00Z_DAY	

Figure 1: Who's the "current" head of the United Kingdom government?

Modification $(s, r, o \rightarrow o')$ and injection $(s, r, \emptyset \rightarrow o')$ are two main ways to update factual knowledge in LLMs, where s, r, o denotes subject, relation, and object in an old fact triple, o' denotes the new target object, and $\emptyset$ denotes an empty object to be populated. Previously, very few works (Zhong et al., 2023; Cohen et al., 2023) evaluate the effectiveness of knowledge editing (KE) techniques on time-sensitive fact changes. We believe that keeping time-sensitive information current is crucial for maintaining the practical relevance of an LLM's knowledge in the real-world applications. Therefore, in this paper, we focus our investigation on temporal KE.

One popular approach for KE is locate-and-edit which involves identifying and directly updating model parameters associated with specific knowledge. ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b) are two representative works in this area. There are several known limitations of ROME/MEMIT. First, they require estimation of a large covariance matrix, which might lead to numerical stability issues during computation (Yao et al., 2023). Second, for every small batch of knowledge edits, they need to locate the layer for weight optimization, which can be time consuming and difficult to scale (Yao et al., 2023). Third, Zhong et al. (2023) demonstrated that although the LLM can successfully recall the edited fact after

*Work done while at Apple.

[1]https://docs-assets.developer.apple.com/ml-research/datasets/chrono-edit/chrono-edit.zip

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 583–593
August 11-16, 2024 ©2024 Association for Computational Linguistics

ROME/MEMIT editing, the post-edit model performs poorly for multi-hop questions. Hence, we would like to verify if PEFT approaches can be more efficient than the locate-and-edit approach in the KE task and perform better in recalling the knowledge edits as well as retaining the unchanged knowledge. In addition, we believe it is worthwhile to investigate the effect of fine-tuning the weights of linear layers in transformers at different locations within the LLM (early, middle, and last) on the multi-hop question answering task. The main contributions of this paper can be summarized as follows:

- We curate a large scale KE dataset CHRONOEDIT from Apple Knowledge Graph (Ilyas et al., 2022, 2023) that contains approximately 15k time-sensitive factual edit examples that better reflects KE in the real world setting.
- We demonstrate the effectiveness of fine-tuning methods in knowledge modification and knowledge injection.
- Through fine-tuning weights at different layers, we discover that the middle layers are more significant in improving the LLM's capability to answer multi-hop questions.

2 Related work

Knowledge editing. Yao et al. (2023) made a comprehensive review of previous work on the topic of LLM KE and pointed out future opportunities. According to Yao et al. (2023), there are three main lines of work in KE: 1) Memory-based, which stores edited examples in memory and recovers relevant edits with a retriever. 2) Locate-and-edit, which identifies and optimizes neural network parameters corresponding to a specific fact. 3) Additional Parameters, which introduce extra tunable parameters to the language model to update or memorize new facts. MELLO (Zhong et al., 2023) is an example of memory-based approach that enables LLM to answer temporal multi-hop questions through effective prompt design and memory retrieval. It introduces a temporal KE dataset MQUAKE-T to assess the ability of a language model in answering multi-hop questions that are associated with a single hop edit. However, the number of distinct knowledge edits in the MQUAKE-T dataset is significantly limited to prove the effectiveness of KE in general. ROME (Meng et al., 2022a) treats an MLP as an associative memory for facts and proposes a causal tracing technique to locate the weight parameters that need update. The additional MLP layer inserted into the transformer unit can be computed using a closed form solution. MEMIT (Meng et al., 2023) extends on ROME to enable the framework for multiple edits at a time. ROME and MEMIT belongs to the locate-and-edit category and their limitations have been discussed. In the additional parameter category, T-Patcher (Huang et al., 2022) and CaliNET (Dong et al., 2022) introduce additional neurons and concatenate them with the Feed-Forward Network (FFN) layers to adjust the output distribution of a target fact. However, these approaches also tend to suffer from slow edit speed and it is unclear how well they can retain time-invariant knowledge. After all, prior works have mostly focused on counterfactual KEs rather than realistic and verifiable time-sensitive fact edits from knowledge graphs (Pan et al., 2023; Wang et al., 2023c, 2022; Ge et al., 2023b, 2024). In this paper, we mainly focus on experimental comparison with the locate-and-edit approach.

Parameter-Efficient Fine-Tuning. LoRA (Hu et al., 2021) is a simple yet effective adaptation technique that adds low-rank tunable weight matrices to the original weight matrices, which are kept frozen. This technique significantly reduces the trainable parameters during fine-tuning, while keeping the inference run-time constant. Instead, P-tuning (Liu et al., 2023) concatenates learnable tensors with the input embedding to enable the base language model to perform well on a range of downstream tasks such as knowledge probing and natural language understanding. In this paper, we would like to verify if these PEFT methods can effectively modify or inject new knowledge in LLMs.

3 Method

We mainly fine-tune the base LLMs including LLaMA-7B, Falcon-7B, and Mistral-7B with the PEFT approach including LoRA and P-tuning and minimize the following loss function:

$$\mathcal{L}_{FT} = \frac{1}{|\mathcal{D}_M|} \sum_{d \in \mathcal{D}_M} L(d; \Phi_0, \Delta\Phi) \qquad (1)$$

where $\mathcal{D}_M$ is the KE dataset and d is a fact edit example, L is the cross entropy loss function applied to autoregressive models, Φ_0 denotes the set of original weights of the language model that are

kept frozen, and $\Delta\Phi$ denotes the additional parameters used by the PEFT adapters.

LoRA. LoRA uses low-rank matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ and $r \ll \min(d, k)$. The low rank matrices A and B are trainable parameters:

$$h = W_0 x + BAx = (W_0 + BA)x. \quad (2)$$

LoRA adaptation can be applied to any linear layer. In our experiments, we apply LoRA to linear layers in both the MLP layers (W_{gate}, W_{up} , W_{down}) and self-attention layers (W_q, W_k, W_v, W_o). The benefit of LoRA is that the inference runtime remains the same, whereas in adaptors and other methods such as ROME/MEMIT, the inference runtime increases since they add additional layers.

P-tuning. P-tuning learns continuous prompt embeddings and concatenates them with the original input embedding. In this work, we leverage these tunable embeddings to adjust the output distributions of the predicted tokens during inference. Formally, let $[P_i]$ be the i^{th} continuous prompt embedding, and let $\mathbf{x} = \{x_0, \ldots, x_n\}$ denotes the original input sequence to the LLM. Then, the new input sequence would be $I = \{[P_{0:i}], \mathbf{x}\}$. P-tuning also uses an additional encoder to map the continuous prompt embeddings to latent parameters $f : [P_i] \rightarrow h_i$. In our implementation, we experiment with both a 2-layer MLP and an LSTM as the mapping function f. Let $\mathbf{e}$ be the pretrained embedding layer, then the final vector input to the LLM is $\{h_0, \ldots, h_i, \mathbf{e}(\mathbf{x})\}$.

Freeze tuning. Instead of fine-tuning all weight parameters in an LLM, only several layers are fine-tuned to save the number of parameters that need to be placed on GPUs for gradient computation. In our experiments, we focus on fine-tuning MLP layers in the transformer modules.

4 Experiments

CHRONOEDIT dataset. To construct a more comprehensive temporal KE dataset that contains more real world knowledge edit examples, we collect the time-sensitive KE dataset CHRONOEDIT. The motivation for collecting this dataset is that the existing MQUAKE-T dataset (Zhong et al., 2023) only contains 96 unique temporal edit examples, and it may not be large enough to reveal the effect on LLMs' performance. The fact change can be located from knowledge graphs (Ge et al., 2022a,b, 2023a; Wang et al., 2023b) based on the semantics of the relation type and its time qualifiers. Specifically, we focus on predicates that have a valid 'start time' qualifier attached. We set the time threshold to 2022-01-01 and collect new knowledge statements that are valid after that time. The dataset statistics are shown in Fig. 2.

Method		REL	GEN	LOC	#Params	GPU time
ROME		62.25	38.76	-	45M	6540s
MEMIT		84.65	71.75	-	225M	8147s
LoRA	Attn	43.73	45.03	46.51	**34M**	1882s
	MLP	98.78	96.97	**55.69**	46M	1389s
	Attn + MLP	**98.99**	97.33	54.11	80M	2356s
P-tuning	MLP	87.03	72.11	39.28	50M	30443s
	LSTM	94.16	73.7	38.70	772M	39657s
Freeze tuning		98.2	96.18	44.45	676M	**1152s**
Full fine-tuning		**98.99**	**98.85**	45.31	6.74B	5604s

Table 1: Reliability (REL), Generalization (GEN), and Locality (LOC) performance, No. of trainable parameters, GPU time for different approaches on LLaMA-7B.

Evaluation metrics. Existing knowledge edit benchmarking datasets often evaluate the following three metrics of the post-edit model:

- **Reliability**: measures the fraction of knowledge edits that the post-edit model can answer correctly.
- **Generalization**: measures the post-edit model's ability in completing the rephrased prompts or answering rephrased questions.
- **Locality**: measures the post-edit model's ability in answering time-invariant knowledge.

We generate question answering pairs as training examples that is used to induce new facts in the LLM. To evaluate Reliability, we generate a corresponding cloze to test whether the post-edit model can successfully complete the sentence with the new fact. To evaluate Generalization, we generate paraphrased question answer pairs from the training examples with the help of OpenAI text-davinci-003 API. To assess Locality, we follow (Jang et al., 2021) to use a subset of LAMA (Petroni et al., 2019) called INVARIANTLAMA, which contains time-invariant statements. We report the ratio of Exact Match (EM) for Reliability and Generalization and the ROUGE-1 score for Locality.

Fine-tuning and locate-and-edit performance comparison. To compare the performance of different fine-tuning approaches for KE, we select a subset from the temporal knowledge dataset we collected that contains 7 relations and 1,388 knowledge modification examples. To compare with locate-and-edit methods, we also include KE results using ROME and MEMIT. Results are shown

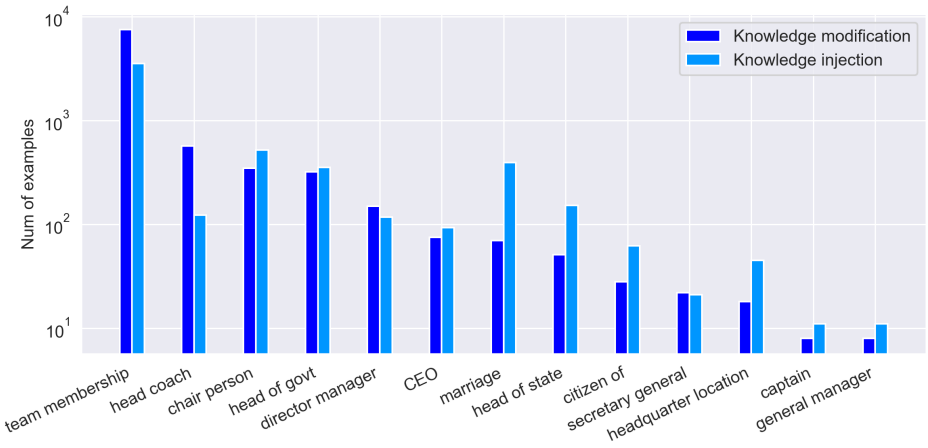


Figure 2: Dataset statistics of CHRONOEDIT.

Predicate	LoRA				Freeze tuning			
	Modification		Injection		Modification		Injection	
	REL	GEN	REL	GEN	REL	GEN	REL	GEN
Captain	87.5	100	81.81	100	100	100	100	100
CEO	100	93.33	100	90.32	100	94.66	100	92.47
Chair person	100	93.67	99.61	97.88	100	93.39	99.42	96.92
Citizen of	100	67.85	100	83.87	100	100	98.38	98.38
Director manager	100	97.98	100	98.29	99.32	97.31	95.72	95.72
General manager	100	87.5	100	90.90	100	87.5	100	90.90
Head coach	100	99.64	100	97.56	99.82	98.41	98.37	100
Head of government	98.44	93.14	99.43	92.09	96.88	95.63	98.87	96.61
Head of state	82.35	80.39	100	96	84.31	78.43	100	100
Headquarter location	100	72.22	97.77	88.89	83.33	83.33	82.22	82.22
Marriage	100	98.57	99.23	97.71	92.85	95.71	77.15	94.92
Secretary general	100	100	100	95.23	100	95.45	95.23	95.23
Team membership	94.14	99.34	92.15	99.49	77.54	96.38	40.38	88.46
Overall	94.99	98.58	94.86	98.22	81.51	96.19	58.44	90.99

Table 2: Performance on each predicate type in CHRONOEDIT for LLaMA-7B.

in Table 1. LoRA finetuning with MLP and attention layers has comparable Reliability and Generalization scores to full fine-tuning, while only using a fraction of trainable parameters compared to full fine-tuning. However, LoRA fine-tuning better retains the invariant knowledge and achieves higher Locality scores. ROME and MEMIT are able to successfully edit some temporal knowledge in the collected dataset. However, the generalization ability degrades significantly, especially for ROME. It is also relatively slow compared to LoRA fine-tuning. We also include P-tuning as a baseline. Similar to the locate-and-edit approach, the generalization score is low, and the GPU time it takes to make successful edits is significantly long. It is not as efficient and effective as LoRA. To verify that PEFT can be generally effective in KE for LLMs, we further compare the performance of different PEFT settings on Falcon-7B (Penedo et al., 2023) and Mistral-7B (Jiang et al., 2023) in Table 3. In Fig. 3, we compare the performance of LoRA with MLP and Attention layers when different number of edits need to be applied to an LLM. We can see that the LoRA finetuning approach is robust to large number of KEs.

LoRA and Freeze tuning fine-grained predicate analysis. In Table 2, we examine the Reliability and Generation scores of the fine-tuned model across all 13 individual relations. For LoRA, we apply it to both MLP and self-attention parameters. For freeze tuning, we fine-tune the MLP weights of the last five layers. The results show that LoRA is more robust than freeze tuning as the number of edits increases. Freeze tuning does not perform well in knowledge injection, with its performance degradation largely attributable to the 'team membership' class, which contains the most knowledge injection examples. This suggests that freeze tun-

Model	LLaMA-7B			Falcon-7B			Mistral-7B		
Method	REL	GEN	LOC	REL	GEN	LOC	REL	GEN	LOC
LoRA Attn	43.73	45.03	46.51	98.91	93.65	49.61	99.2	96.25	54.08
LoRA MLP	98.78	96.97	**55.69**	98.92	96.03	51.41	99.13	97.98	57.84
LoRA Attn + MLP	**98.99**	97.33	54.11	99.06	96.97	49.41	99.13	98.05	54.21
Freeze tuning	98.2	96.18	44.45	-	-	-	94.66	94.95	43.17
Full fine-tuning	**98.99**	**98.85**	45.31	99.21	98.19	38.27	-	-	-

Table 3: Performance of PEFT fine-tuning for KE across different LLMs

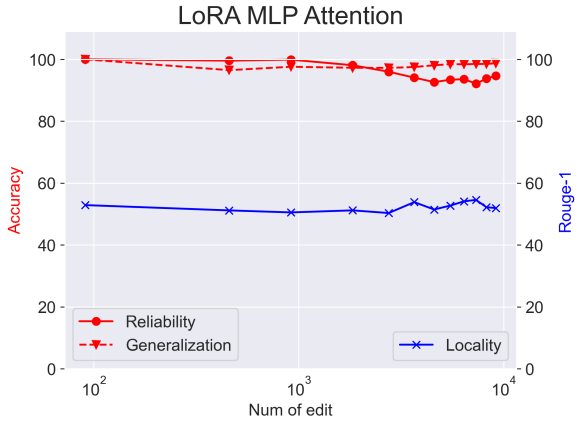


Figure 3: Reliability, Generalization, and Locality performance versus the number of edits on LLaMA-7B.

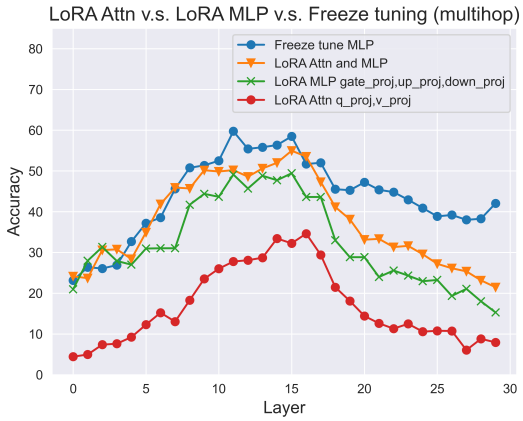


Figure 4: Performance of fine-tuning methods on the MQuAKE-T multi-hop dataset for LLaMA-7B.

ing might not be very effective in introducing new facts about subjects that have rarely been observed during the pretraining of LLMs.

Layer sweep study. For the freeze tuning and LoRA fine-tuning approaches, we think it is also worthwhile investigating the effect on LLMs' multi-hop question answering capability, by optimizing the LLM weight parameters at different positions (early, middle, late layers). We perform a layer sweep study for the MQuAKE-T multi-hop question answering task. For each data point of the experiment, we only fine-tune $l = 3$ layers at a time. We then move the sliding window from the early layers to the last layers of an LLM to probe the effect of fine-tuning on the performance of multi-hop question answering. We compared freeze-tuning for MLP layers and LoRA on three combination of weight matrices: 1) self-attention weight matrices W_q, W_v, 2) MLP layers, 3) self-attention and MLP layers. We have made similar observations aligned with the Associative Memory theory (Geva et al., 2021) verified by ROME, that MLP layers in transformers are more relevant for memorizing factual knowledge associations $(s, r \Rightarrow o)$. We observe that applying LoRA on MLP weight matrices brings more significant improvement than applying LoRA to self-attention weight matrices. Applying LoRA on both self-attention and MLP layers can potentially achieve similar performance to freeze tuning on multi-hop QA tasks, while using fewer trainable parameters. In particular, applying LoRA on both MLP and self-attention requires 7.5M trainable parameters, whereas freeze-tuning requires 405.8M trainable parameters. For complete performance benchmarking, we also compare with memory-based KE approach for multi-hop QA in Table 6 of the Appendix.

5 Conclusion

In this paper, we have systematically examined the feasibility of performing KE through PEFT. We have compared the performance of fine-tuning methods including LoRA, P-tuning and freeze tuning with locate-and-edit approaches for KE. Our results demonstrate that fine-tuning can successfully update time-sensitive factual knowledge in LLMs both efficiently and effectively, and without compromising the LLMs' capability in answering invariant knowledge and multi-hop reasoning. We have also contributed a large scale KE dataset ChronoEdit that contains both modification edit and injection edit examples.

Limitations

There are two limitations that we would like to discuss. First, although we have collected a comprehensive and realistic temporal KE dataset, we primarily gather time-sensitive fact changes from Wikipedia, the most frequently used data source for LLM pre-training. We are yet to include information from other data sources or knowledge graphs that may contain ontological information that enable us to access LLMs' ability to perform reasoning. Second, we have not covered another important aspect of KE that is to remove misinformation or mitigate hate speech generation from LLMs. We will expand the scope of exploration in future work.

Acknowledgements

We would like to express our gratitude to Bin Wang for the valuable discussions during the preliminary research exploration phase. We also extend our thanks to Azadeh Nikfarjam, Samira Khorshidi, Alexis McClimans, Fei Wu, and Eric Choi for their guidance in collecting the knowledge editing dataset. Additionally, we are grateful to Barry Theobald, Yash Govind, Varun Embar, and Shihab Chowdhury, Hong Yu for proofreading the manuscript and providing insightful advice to improve the paper.

References

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

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2023. Evaluating the ripple effects of knowledge editing in language models. *arXiv preprint arXiv:2307.12976*.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5937–5947.

Xiou Ge, Yun Cheng Wang, Bin Wang, and C-C Jay Kuo. 2023a. Compounding geometric operations for knowledge graph completion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6947–6965.

Xiou Ge, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo. 2023b. Knowledge graph embedding with 3d compound geometric transformations. *arXiv preprint arXiv:2304.00378*.

Xiou Ge, Yun Cheng Wang, Bin Wang, C-C Jay Kuo, et al. 2022a. Typeea: Type-associated embedding for knowledge graph entity alignment. *APSIPA Transactions on Signal and Information Processing*, 12(1).

Xiou Ge, Yun Cheng Wang, Bin Wang, C-C Jay Kuo, et al. 2024. Knowledge graph embedding: An overview. *APSIPA Transactions on Signal and Information Processing*, 13(1).

Xiou Ge, Yun-Cheng Wang, Bin Wang, and CC Jay Kuo. 2022b. Core: A knowledge graph entity type prediction method via complex space regression and embedding. *Pattern Recognition Letters*, 157:97–103.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2022. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*.

Ihab F Ilyas, JP Lacerda, Yunyao Li, Umar Farooq Minhas, Ali Mousavi, Jeffrey Pound, Theodoros Rekatsinas, and Chiraag Sumanth. 2023. Growing and serving large open-domain knowledge graphs. In *Companion of the 2023 International Conference on Management of Data*, pages 253–259.

Ihab F Ilyas, Theodoros Rekatsinas, Vishnu Konda, Jeffrey Pound, Xiaoguang Qi, and Mohamed Soliman. 2022. Saga: A platform for continuous construction and serving of knowledge at scale. In *Proceedings of the 2022 International Conference on Management of Data*, pages 2259–2272.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, KIM Gyeonghun, Stanley Jungkyu Choi, and Minjoon Seo. 2021. Towards continual knowledge learning of language models. In *International Conference on Learning Representations*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. Gpt understands, too. *AI Open*.

Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Floris Weers, et al. 2024. Mm1: Methods, analysis & insights from multimodal llm pre-training. *arXiv preprint arXiv:2403.09611*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023. Mass editing memory in a transformer. *The Eleventh International Conference on Learning Representations (ICLR)*.

Yuxian Meng, Xiaoya Li, Xiayu Zheng, Fei Wu, Xiaofei Sun, Tianwei Zhang, and Jiwei Li. 2022b. Fast nearest neighbor machine translation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 555–565, Dublin, Ireland. Association for Computational Linguistics.

R OpenAI. 2023. Gpt-4 technical report. *arXiv*, pages 2303–08774.

Jeff Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, et al. 2023. Large language models and knowledge graphs: Opportunities and challenges. *Transactions on Graph Data and Knowledge*.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

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.

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*.

Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, et al. 2023a. Easyedit: An easy-to-use knowledge editing framework for large language models. *arXiv preprint arXiv:2308.07269*.

Yun-Cheng Wang, Xiou Ge, Bin Wang, and C-C Jay Kuo. 2022. Kgboost: A classification-based knowledge base completion method with negative sampling. *Pattern Recognition Letters*, 157:104–111.

Yun-Cheng Wang, Xiou Ge, Bin Wang, and C-C Jay Kuo. 2023b. Asyncet: Asynchronous learning for knowledge graph entity typing with auxiliary relations. *arXiv preprint arXiv:2308.16055*.

Yun Cheng Wang, Xiou Ge, Bin Wang, and C-C Jay Kuo. 2023c. Greenkgc: A lightweight knowledge graph completion method. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10596–10613.

Chengwei Wei, Yun-Cheng Wang, Bin Wang, C-C Jay Kuo, et al. 2023. An overview of language models: Recent developments and outlook. *APSIPA Transactions on Signal and Information Processing*, 13(2).

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

A Dataset statistics

A.1 MQuAKE-T dataset experiments

We primarily use the MQuAKE-T dataset which contains temporal-based real-world knowledge updates to compare the performance of different fine-tuning techniques with baseline methods on the performance of KE. The goal is to validate whether PEFT approaches such as LoRA and P-tuning can be an effective approach for performing KE. We also demonstrate that PEFT approaches can be more effective than the locate-and-edit approaches for multi-hop question answering.

In this dataset, each temporal fact edit example is also associated with multi-hop questions, which allows us to assess the complex query answering ability of the post-edit model. The MQuAKE-T dataset was constructed by taking the difference between two data dumps of Wikidata: 2021-04 and 2023-04. MQuAKE-T selects 6 different relations that most likely correspond to real fact changes. The statistics of the dataset are shown in Table 4.

MQuAKE-T	#Examples
Unique edits	96
2-hop questions	75
3-hop questions	348
4-hop questions	567

Table 4: Statistics of MQuAKE-T dataset.

Comparing with baselines. In Table 5, we compare the editwise performance of fine-tuning techniques with locate-and-edit baseline methods. We use LLaMA-7B (Touvron et al., 2023) as the base model for both the baseline locate-and-edit techniques and fine-tuning techniques. Experimental results show that fine-tuning techniques performs better than the locate-and-edit baselines, while the run-time to complete all the knowledge edit is significantly shorter. In Table 6, we compare the performance of different post-edit model and approach for multi-hop QA.

LoRA ablation and parameter study. We perform ablation study of applying LoRA adaptation to different weight matrices in the self-attention module W_q, W_v, W_k, W_o. The results are shown in Table 7. Results shows that applying LoRA adaptation to the query matrix W_q and the key matrix W_k gives the best result. We also evaluate the knowledge edit success rate when the LoRA rank is set to different values. In our experiment, we tested $r = \{4, 8, 16, 32, 64\}$ as shown in Fig. 5, and discover that the optimal rank is $r = 32$.

A.2 ChronoEdit dataset

In the new dataset, we set the time threshold to 2022-01-01 and collect new knowledge statements

Method	Edit Accuracy	Runtime
ROME	92.51	2h32m2s
MEMIT	96.44	2h48m49s
LoRA	99.36	2m13s
P-tuning	97.75	1m51s
Freeze-tuning	**100**	3m16s
Full fine-tuning	<u>99.83</u>	8m18s

Table 5: Editwise performance on LLaMA-7B.

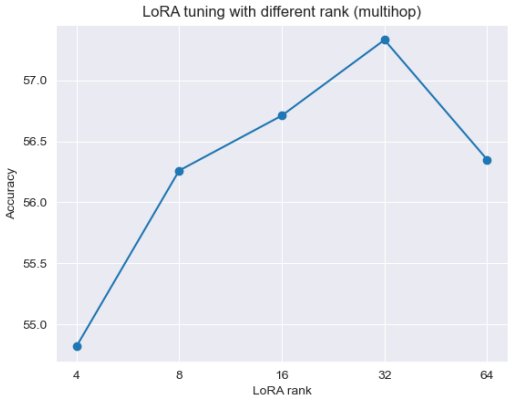


Figure 5: Performance of LoRA at different ranks for the MQuAKE-T multi-hop dataset with LLaMA-7B.

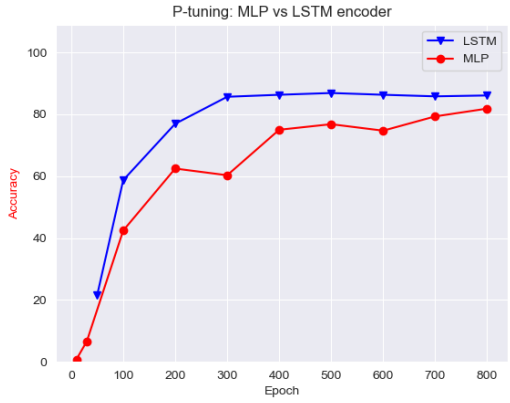


Figure 6: Comparing Reliability performance of LSTM and MLP encoders across epochs when using P-tuning for LLaMA-7B.

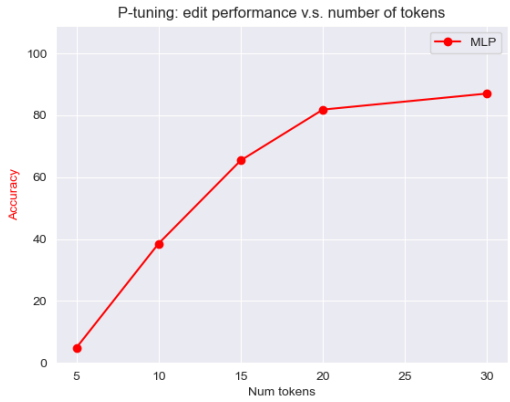


Figure 7: Comparing Reliability performance for different number of tokens when using P-tuning for LLaMA-7B.

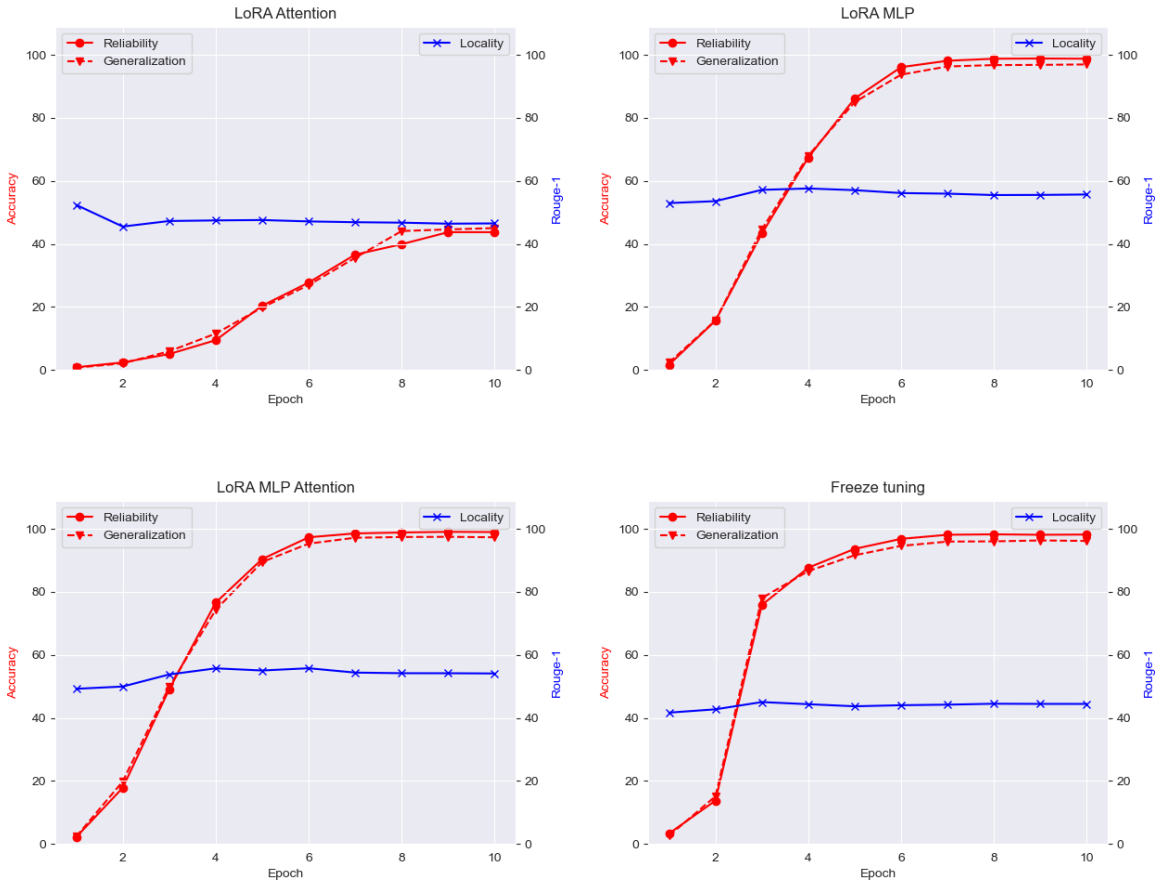


Figure 8: Reliability, Generalization, and Locality performance of different fine-tuning methods across epochs for LLaMA-7B.

Base Model	KE Type	KE Method	Multi-hop QA Acc
LLaMA-7B	Locate-and-edit	ROME	38.5
		MEMIT	39.3
	Additional parameter	P-tuning	14.7
		LORA	62.6
	Direct fine-tune	Freeze tuning	72.5
		Full FT	71.0
Vicuna-7B	Memory-based	Mello	30.7
GPT-J			51.3
GPT-3			**85.5**

Table 6: Performance on post-edit model on multi-hop questions for LLaMA-7B.

Linear Layer	Edit Accuracy
W_q	71.47
W_v	97.48
W_q, W_v	98.67
W_q, W_v, W_k, W_o	97.56

Table 7: Ablation studies of the layers in LLaMA-7B that LoRA is applied to.

that are valid after that time. We collect both knowledge modification: $(s, r, o) \rightarrow (s, r, o')$, and knowledge injection: $(s, r, \emptyset) \rightarrow (s, r, o')$. The statistics of the dataset are shown in Fig. 2. An example of fact pairs from the KG that could lead to time-sensitive knowledge edits is shown in Table 8. We convert such fact pairs to question answering and instruction finetuning examples for training. The corresponding sentence completion examples for reliability evaluation, rephrased QA examples for generalization evaluation, and invariant knowledge sentence completion examples for locality evaluation are shown in Table 9.

LoRA and Freeze tuning ablation and parameter study. In Fig. 8, we evaluate the performance of different fine-tuning configurations across different epochs. In particular, we evaluate the Reliability and Generalization using the accuracy which is the ratio of Exact Matching (EM) and we report the ROUGE-1 score for Locality. For LoRA, we experiment with three settings: applying LoRA to self-attention weights (LoRA Attention), applying LoRA to MLP weights (LoRA MLP), and applying LoRA to both self-attention and MLP weights (LoRA MLP Attention). In this set of experiments,

Organization	CEO	Start Time	End Time
Volkswagen Group	Herbert Diess	+2018-04-00T00:00:00Z_MONTH	+2022-08-31T00:00:00Z_DAY
Volkswagen Group	Oliver Blume	+2022-09-01T00:00:00Z_DAY	

Table 8: Example of locating the knowledge edit data

Examples	
Train	{ "instruction": "Who is the current chief executive officer of Volkswagen Group?", "input": "", "output": "Oliver Blume." }
	{ "instruction": "Update the following statement about the current chief executive officer of Volkswagen Group.", "input": "Herbert Diess.", "output": "Oliver Blume." }
Test (REL)	{ "instruction": "The current chief executive officer of Volkswagen Group is", "input": "", "output": "Oliver Blume." }
Rephrase (GEN)	{ "instruction": "What is the name of the current Volkswagen Group CEO?", "input": "", "output": "Oliver Blume." }
Invariant (LOC)	{ "instruction": "The headquarter of Volkswagen Commercial Vehicles is in?", "input": "", "output": "Hanover." }

Table 9: Fine-tuning and testing examples.

we apply LoRA to all layers. For freeze tuning, we fine-tune the MLP weights of the last 5 layers of the LLaMA model. Results shows that applying LoRA to MLP weights is more effective in memorizing new facts than applying LoRA to self-attention weights. While freeze tuning can also effectively have the knowledge update induced into the model, the Locality score for freeze tuning is lower than the LoRA MLP setting, which means freeze tuning leads to deterioration of the LLM's existing invariant knowledge.

P-tuning ablation and parameter study. Although P-tuning can be equally effective for KE, we find that it requires more epochs of fine-tuning to ensure successful knowledge edits. The required time to perform knowledge edits becomes longer. In Fig. 6, we compare the performance difference between LSTM and MLP encoders across different epochs when using the P-tuning technique, when the number of prompt embedding tokens is set to $n = 20$. We observe that the application of LSTM encoder allows P-tuning edit performance to converge faster than when using the MLP encoder. In Fig. 7, we instead compare the performance of P-tuning when different number of prompt embedding tokens are used. Using more than $n = 20$ tokens do not seem to gives a significant advantage in the edit accuracy.

Fine-grained performance analysis of time-invariant knowledge. For the KE experiment of using LoRA on MLP layers of LLaMA-7B, we perform a fine-grained performance analysis of the different type of time-invariant knowledge and list the performance in Table 10. We make a conjecture that those time-invariant knowledge with smaller valid candidate set for the target, such as "language" or "capital", tends to be well retained. These predicates are mostly 1-to-1 or N-to-1. In contrast, when the cardinality of the valid candidate set becomes larger, often for N-to-N predicates, such as "twin city" and "music label", the exact subject, object association becomes harder to retain.

Implementation details. Experiments were conducted on a compute node with 8 NVIDIA Tesla A100 GPUs, each with 40GB memory. We develop the fine-tuning pipeline based on LLaMA-Factory[2]

[2]https://github.com/hiyouga/LLaMA-Factory

Best 3	ROUGE-1
native language of	70.2
official language of	61.7
Capital of	58.7
Worst 3	**ROUGE-1**
twin cities	1.55
is a	5.68
is represented by music label	9.47

Table 10: Performance on different type of invariant knowledge.

Parameter	Value
layers	[5]
fact_token	subject_last
v_num_grad_steps	25
v_lr	5e-1
v_loss_layer	31
v_weight_decay	1e-3
clamp_norm_factor	4
kl_factor	0.0625
mom2_adjustment	false
context_template_length_params	[[5, 10], [10, 10]]
rewrite_module_tmp	model.layers..mlp.down_proj
layer_module_tmp	model.layers.
mlp_module_tmp	model.layers..mlp
attn_module_tmp	model.layers..self_attn
ln_f_module	model.norm
lm_head_module	lm_head
mom2_dataset	wikipedia
mom2_n_samples	100000
mom2_dtype	float32

Table 11: ROME Configuration Parameters.

Parameter	Value
layers	[4, 5, 6, 7, 8]
clamp_norm_factor	4
layer_selection	all
fact_token	subject_last
v_num_grad_steps	25
v_lr	5e-1
v_loss_layer	31
v_weight_decay	1e-3
kl_factor	0.0625
mom2_adjustment	true
mom2_update_weight	15000
rewrite_module_tmp	model.layers..mlp.down_proj
layer_module_tmp	model.layers.
mlp_module_tmp	model.layers..mlp
attn_module_tmp	model.layers..self_attn
ln_f_module	model.norm
lm_head_module	lm_head
mom2_dataset	wikipedia
mom2_n_samples	100000
mom2_dtype	float32

Table 12: MEMIT Configuration Parameters.

(Zheng et al., 2024) and refer to PEFT package in HuggingFace[3] for the implementation of LoRA and P-tuning. We use EasyEdit[4] (Wang et al., 2023a) to reproduce the ROME and MEMIT fine-tuning baseline results.

For results in Table 1, the 7 different relations that we evaluate on are 'captain', 'CEO', 'chairperson', 'head coach', 'head of govt', 'head of state', 'headquarter location'. The reason for the performance comparison of the smaller subset is to conduct similar experiments that were done in (Zhong et al., 2023). For LoRA, Freeze tuning, Full fine-tuning, we fine-tune the base model for 10 epochs, whereas for P-tuning, we fine-tune 800 epochs to achieve the optimal performance. Full fine-tuning of the base model requires DeepSpeed ZeRO-3 offload. In LoRA experiments, the LoRA rank is set to $r = 32$, and MLP means applying LoRA to W_{gate}, W_{up} , W_{down} matrices, and Attn means to apply LoRA to W_q, W_k, W_v, W_o matrices. In P-tuning experiments, the number of prompt tokens is set of $n = 20$. In the MLP encoder, there are 3 linear layers with ReLU activation in between. In the LSTM encoder, a bidirectional LSTM is used and the output is passed to 2 linear layers with ReLU activation in between. For all the above experiments, we used the AdamW optimizer and set the learning rate to $5e - 5$, per device train batch size to 4, gradient accumulation steps to 4. For the ROME and MEMIT baselines, we used the default hyperparameter settings provided in EasyEdit, shown in Table 11 and 12.

For the knowledge modification and knowledge injection experiments in Table 2, we oversample each knowledge injection samples four times due to the limited number of training examples, as generating an update example for knowledge injection is not possible. The hyperparameter settings are kept the same as above.

[3]https://huggingface.co/docs/peft/index
[4]https://github.com/zjunlp/EasyEdit