Evaluating Model Editing Techniques for Different Architectures

Pritam Gouda
Department of Computer Science
Indian Institute of Science
Bengaluru, KA, India
pritamt@iisc.ac.in

1 Introduction

Large Language Models (LLMs) have become indispensable tools in natural language processing, excelling in tasks such as document classification, question answering, and text summarization. Despite their versatility, correcting factual inaccuracies or adapting these models for specific tasks often requires extensive retraining, which is computationally expensive and resource-intensive.

An emerging alternative is *model editing*, which allows for targeted modifications to a model's behavior without the need for retraining. This approach offers a practical solution for rectifying factual errors, adapting models to domain-specific tasks, and improving task-specific performance, all while preserving the broader capabilities of the model.

This report explores how LLMs store and retrieve factual information, the feasibility of pinpointing and rectifying facts, and the broader implications of these edits. I analyze state-of-the-art model editing techniques like ROME[4] and SERAC[6], and evaluate their generalization, accuracy, and locality across different architectures like LLaMA[7].

2 Problem Statement

Correcting factual inaccuracies in LLMs without retraining poses several challenges. Model editing techniques aim to address these challenges by providing an efficient and targeted approach. Broadly, these techniques fall into two categories:

- **Preserving Parameters:** Methods such as SERAC and MemPrompt[3] use memory-based models or introduce trainable parameters (e.g., T-Patcher[1]) to localize edits. These methods prioritize minimizing disruption to the model's existing knowledge while enabling updates.
- **Modifying Parameters:** Techniques like ROME and Meta-learning directly adjust the model's internal weights to integrate new knowledge. These approaches are designed for precision and generalization, ensuring that the updates do not adversely affect the model's broader behavior.

To evaluate these approaches, this report examines their performance using standard datasets such as ZsRE[2] and COUNTERFACT[4]. Metrics[8] like generalization, accuracy, and locality are used to assess the effectiveness of the edits. Additionally, experiments on different architectures, including decoder-only models like GPT2-XL and encoder-decoder models like T5, reveal the scalability and adaptability of these techniques across varying model designs. The goal is to identify techniques that achieve high accuracy and generalization while minimizing computational overhead and preserving the model's original capabilities.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

3 Methodology

In this section, I explore and describe the various model editing techniques employed in this work, focusing on their mechanisms, applications, and evaluations. The methods include ROME[4], MEND, and IKE[9], which are evaluated on multiple metrics, such as generalization, locality, and portability. The aim is to examine how these techniques adapt to different architectures and datasets.

3.1 ROME

The **Rank-One Model Editing (ROME)** technique targets specific feed-forward weights in transformer-based models, such as GPT-2, to make precise factual updates without compromising overall model knowledge. By applying rank-one updates, ROME leverages the associative memory function of Multi-Layer Perceptron (MLP) layers to ensure high accuracy and generalization.

To evaluate ROME's effectiveness, metrics like generalization, accuracy, locality, and portability are employed. While ROME excels in tasks involving single-hop queries, its portability on multi-hop prompting remains underexplored. Techniques like MQuAKE[10] are used to provide insights into how ROME might handle multi-hop tasks. This investigation opens avenues to extend ROME for diverse architectures and more complex queries, emphasizing the trade-off between specificity and general applicability.

3.2 Multi-Hop Prompting

Most model editing techniques have been tested primarily on single-hop prompts. However, real-world scenarios often involve multi-hop prompts, where the model must answer interconnected sub-questions. This requires the ability to preserve the logical chain of thought while maintaining factual accuracy.

3.2.1 Implementation Details

Metrics such as generalization and locality are computed using the COUNTERFACT dataset, whereas portability is evaluated using the MQuAKE dataset. Generalization is quantified by ensuring that:

$$P(F_{\text{edit}}) > P(F_{\text{original}})$$

where F_{edit} represents the edited fact and F_{original} is the unedited counterpart. Locality is assessed by minimizing edits' impact on unrelated data points, and portability examines the model's ability to respond accurately to multi-hop prompts.

3.2.2 Observations

Multi-hop evaluations reveal that foundational limitations in the model's pre-editing performance often restrict its ability to answer complex queries post-editing. Effective assessments should distinguish between inherent model deficiencies and issues introduced during the editing process. This analysis highlights the importance of designing model editing techniques that complement the underlying architecture's capabilities.

3.3 MEND

The **Model Editing Networks with Gradients (MEND[5])** technique simplifies the process of updating model parameters by utilizing compact networks. MEND focuses on refining specific regions of the model's knowledge base to enable localized, efficient, and general updates. By breaking down gradients into manageable components, MEND effectively handles large-scale data modifications while preserving overall consistency in predictions.

MEND's ability to maintain high accuracy across various architectures makes it a versatile choice. However, its dependence on specific gradient adjustments may limit its adaptability in scenarios requiring broader knowledge edits.

3.4 IKE

In-Context Knowledge Editing (IKE[9]) introduces context-based modifications to refine a model's knowledge using natural language examples. IKE is particularly adept at supporting diverse model sizes, from smaller frameworks to large-scale architectures with billions of parameters.

The technique ensures effective parameter updates without the computational overhead typically associated with resource-heavy processes. While IKE excels in tasks requiring minimal edits, its performance on large-scale modifications and adaptability across different architectures remains an area of ongoing research.

4 Results

I evaluated ROME, MEND, and IKE on GPT2-XL and Llama-2 (7B) using 1000 samples from the COUNTERFACT dataset. Metrics evaluated include Generalization, Locality, and Accuracy. The results are summarized in Tables 1 and 2. On GPT2-XL, ROME and IKE achieved near-perfect accuracy, while MEND performed comparatively lower. On Llama-2, MEND outperformed ROME in locality but had slightly lower generalization.

I also tested the T5 (3B) model to assess architecture independence. Since ROME is designed for autoregressive models, it could not be adapted to T5. MEND and IKE were evaluated on T5, where MEND demonstrated higher generalization and locality, but IKE excelled in accuracy (Table 3).

Portability was tested using the MQuAKE dataset (Table 4). The results indicate low portability for ROME on multi-hop prompts, highlighting its limitations in handling complex queries.

Table 1: Evaluation Results for ROME, MEND & IKE on GPT2-XL

Name	Generalization	Locality	Accuracy
ROME	96.5	75.41	100
MEND	58.3	44.88	94.2
IKE	95.4	76.7	100

Table 2: Evaluation Results for ROME, MEND & IKE on Llama-2 (7B)

Name	Generalization	Locality	Accuracy
ROME	87.04	99.36	99.42
MEND	90.27	97.02	94.24
IKE	99.4	69.2	100

Table 3: Evaluation Results for ROME, MEND & IKE on T5 (3B)

Name	Generalization	Locality	Accuracy
ROME	–	–	–
MEND	93.27	91.85	81.4
IKE	83.2	36.67	97.7

5 Analysis

All techniques in their initial implementation were trained and tested only on GPT models. In this section, I analyze their performance on different model architectures and derive the following insights:

5.1 Analyzing ROME

I observed that the locality of ROME increased significantly with an increase in the number of parameters. This is likely because a larger parameter set allows the model to specialize more

Table 4: Evaluation Results on ROME for Portability on Multi-Hop Prompts (GPT2-XL)

Type	Pre Edit	Post Edit
2 Hop	19.33	8.0
3 Hop	7.33	8.0
4 Hop	10.67	10.0

granularly, handling specific features or sub-tasks. However, generalization decreased slightly, likely due to overfitting, as the model memorizes details from the training dataset instead of learning broader patterns. Accuracy remained consistent across both GPT2-XL and Llama-2 (7B).

5.2 Analyzing IKE

IKE maintained a high generalization score and accuracy across all decoder-only models. However, its performance on T5 (an encoder-decoder model) showed a drop in generalization, suggesting that its edits are influenced by the architecture and parameter type. Additionally, IKE did not scale well when increasing model parameters and struggled to maintain locality, as edits seemed to disrupt the model's collateral data.

5.3 Analyzing MEND

For MEND, accuracy remained largely consistent across architectures and parameter sizes. However, generalization and locality scores nearly doubled as the number of parameters increased, highlighting that these metrics scale better with larger models. Nevertheless, MEND faced challenges adapting to encoder-decoder architectures, as the complexity of the LLM increased.

5.4 Analyzing Portability

Portability scores were consistently low both before and after edits for ROME, as observed in multi-hop prompt tests. This suggests that low portability is rooted in foundational model training rather than the editing techniques. Counterfact's inherent corruption of factual information might also affect portability performance.

6 Conclusion

In this report, I implemented and evaluated ROME, MEND, and IKE on three LLMs—GPT2-XL, Llama-2 (7B), and T5 (3B)—using various metrics. Additionally, I designed and tested portability metrics for ROME on multi-hop prompts. The following conclusions were drawn:

- ROME excelled across all metrics on GPT-like architectures but struggled with generalization as model parameters increased, indicating potential overfitting.
- MEND demonstrated better scalability for generalization and locality as parameters increased, although it was less adaptable to encoder-decoder architectures.
- IKE, being flexible and architecture-independent, achieved high accuracy but lacked scalability with larger parameters and struggled with locality.
- Low portability scores across all models indicate that foundational training issues, rather than the editing process, might limit performance.

Overall, ROME appears to be the most reliable technique for decoder-only architectures, while MEND and IKE are better suited for specific use cases requiring architecture independence or high accuracy.

A Appendix

Code for this can be found at : https://github.com/pritamgouda11/Editing-Techniques-Across-LLM-Architectures.git

References

[1] Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron, 2023.

[2] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension, 2017.

[3] Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. Memory-assisted prompt editing to improve gpt-3 after deployment, 2023.

[4] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023.

[5] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale, 2022.

[6] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale, 2022.

[7] 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. Llama: Open and efficient foundation language models, 2023.

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

[9] Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning?, 2023.

[10] Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. Mquake: Assessing knowledge editing in language models via multi-hop questions, 2023.