Knowledge Graph Enhanced Large Language Model Editing

Anonymous ACL submission

Abstract

Large language models (LLMs) are pivotal in advancing natural language processing (NLP) tasks, yet their efficacy is hampered by inaccuracies and outdated knowledge. Model editing emerges as a promising solution to address these challenges. However, existing editing methods struggle to track and incorporate changes in knowledge associated with edits, which limits the generalization ability of post-edit LLMs in processing edited knowledge. To tackle these problems, we propose a novel model editing method that leverages knowledge graphs for enhancing LLM editing, namely GLAME. Specifically, we first utilize a knowledge graph augmentation module to uncover associated knowledge that has changed due to editing, obtaining its internal representations within LLMs. This approach allows knowledge alterations within LLMs to be reflected through an external graph structure. Subsequently, we design a graph-based knowledge edit module to integrate structured knowledge into the model editing. This ensures that the updated parameters reflect not only the modifications of the edited knowledge but also the changes in other associated knowledge resulting from the editing process. Comprehensive experiments conducted on GPT-J and GPT-2 XL demonstrate that GLAME significantly improves the generalization capabilities of post-edit LLMs in employing edited knowledge.

1 Introduction

Large language models (LLMs) have achieved impressive results in various natural language processing (NLP) tasks due to their strong general capabilities and inherent rich world knowledge (Zhao et al., 2023). However, the knowledge in LLMs may be factually incorrect or outdated, thereby limiting their capabilities. To address these issues, model editing of LLMs has been proposed, distinguishing themselves from the traditional fine-tuning approaches. Model editing employs a more efficient and precise method to update the knowledge embedded in LLMs and has garnered widespread attention from researchers in recent years.

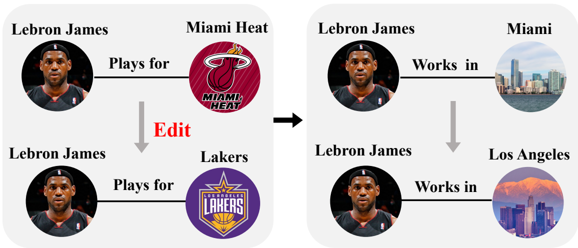


Figure 1: An example of model editing for LLMs. Editing target knowledge leads to changes in its associated knowledge.

Model editing primarily comprises three categories of methods: Memory-based, Meta-learning, and Locate-then-edit methods. Memory-based methods, exemplified by SERAC (Mitchell et al., 2022), store edited knowledge in the external memory outside of LLMs, enabling the retrieval of this knowledge from memory during the inference process of LLMs. Meta-learning methods typically adopt a hyper-network to learn the weight changes for editing LLMs, such as KE (De Cao et al., 2021) and MEND (Mitchell et al., 2021). To achieve more precise knowledge editing, locate-then-edit methods have been proposed. For instance, ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b) directly target and update parameters corresponding to specific knowledge.

While these methods demonstrate promising results in knowledge editing of LLMs, they still face the challenge of capturing the associated knowledge changes related to edited knowledge. Specifically, existing work primarily focuses on the editing of target knowledge, such as modifying knowledge from (s, r, o) to (s, r, o^*). However, such single-knowledge modification often triggers a series of consequential alterations in associated knowledge. As shown in Figure 1, an edit that changes the knowledge from "*LeBron James plays for the Miami Heat*" to "*LeBron James plays for the Los Angeles Lakers*" would necessitate a corresponding update from "*LeBron James works in Miami*" to "*LeBron James works in Los Angeles*". Existing editing methods fail to account for the impact on associated knowledge resulting from the modification of target knowledge, which limits the generalizability of post-edited LLMs in processing such edited knowledge. The black-box nature of LLMs makes capturing the associations between pieces of knowledge within the models exceedingly complex, further challenging the detection of such associated knowledge changes during editing.

To deal with the above challenge, we propose a novel locate-then-edit method enhanced by knowledge Graphs for LArge language Model Editing, namely GLAME. Specifically, for each target edit knowledge, we first present a knowledge graph augmentation (KGA) module (§4.1) to construct a subgraph that captures the new associations resulting from the edit. Directly editing high-order relationships from the subgraph into LLMs in a simplistic way requires multiple alterations to the models and might disrupt the targeted edited knowledge, potentially exerting significant adverse effects and diminishing post-edit model performance (§5.2). Therefore, we further develop a graph-based knowledge edit (GKE) module (§4.2) that integrates the subgraph encoding into the rank-one model editing framework. With just a single edit, it ensures that the edited parameters can recognize not only the edited knowledge but also the broader scope of knowledge impacted by such edits.

We summarize our contributions as follows:

- We emphasize and investigate the necessity of capturing the changes of associated knowledge induced by edited knowledge in model editing.
- We integrate knowledge graphs into model editing and propose a novel and effective editing method to structure knowledge changes induced by editing and incorporate them into specific parameters.
- We conduct extensive experiments on GPT-2 XL and GPT-J, which demonstrate the effectiveness of our proposed model.

2 Related Work

In this section, we introduce the related work on model editing, which aims to inject new knowledge into LLMs or modify their existing internal knowledge, while ensuring it does not impact other unrelated knowledge. Model editing methodologies can be broadly classified into three distinct categories (Yao et al., 2023): memory-based, meta-learning, and locate-then-edit approaches.

Memory-based strategies choose to augment LLMs with external memory modules, thereby offering a pathway to knowledge updates without modifying the parameters of LLMs. For example, SERAC (Mitchell et al., 2022) method introduces a gating network in conjunction with an additional model specifically designed to manage edited knowledge. However, the memory-based approaches all highlight a fundamental limitation in their scalability: the external model's management complexity escalates with each additional edit, potentially hampering its practical applicability.

Conversely, meta-learning methods eliminate the necessity for complex external memory modules by focusing on the training of a hyper-network capable of generating updated weights for the LLMs. This strategy was initially investigated by KE (De Cao et al., 2021), utilizing a bi-directional LSTM to predict model weight updates. However, this approach encountered limitations when applied to larger models due to their extensive parameter spaces. To deal with this challenge, MEND (Mitchell et al., 2021) adopts a low-rank decomposition of fine-tuning gradients, showcasing an efficient mechanism for updating weights in LLMs. Nevertheless, these approaches still require extensive computational resources for training and risk affecting unrelated knowledge.

To overcome these issues, recent works have explored knowledge location within LLMs, aiming for more interpretable and precise knowledge editing by targeting parameters directly associated with specific information. The early attempts include KN (Dai et al., 2022), which proposes a knowledge attribution method to identify knowledge neurons but falls short in making precise changes to the model's weights. Subsequently, the progress in comprehending the fundamental mechanism of Transformer (Vaswani et al., 2017) models has introduced the hypothesis that the Feed Forward Network (FFN) modules might function as key-value memories (Geva et al., 2021, 2023), thereby laying the groundwork for more precise editing strategies. The ROME (Meng et al., 2022a) method, building on this insight, employed causal tracing to pinpoint knowledge-relevant layers and then edit its FFN module, achieving superior outcomes. Building upon this, MEMIT (Meng et al., 2022b) tackles batch editing tasks, enabling large-scale knowledge integration.

Despite these advancements, all of the above models primarily concentrate on editing isolated pieces of knowledge, overlooking the potential ripple effects across the model's knowledge base (Cohen et al., 2023). This omission can impair the model's generalization ability post-editing and hinder its capacity for further reasoning with newly integrated knowledge (Zhong et al., 2023).

3 Preliminaries

In this section, we introduce the definition of model editing and knowledge graphs, and the rank-one model editing framework used in our study.

Definition 1 (**Model Editing for LLMs**). Model editing (Yao et al., 2023) aims to adjust an LLM $\mathcal{F}$'s behavior to modify the knowledge (s, r, o) encoded in the model into the target knowledge (s, r, o^*), where knowledge is denoted as a triple, consisting of the subject s, relation r, and object o. Each edit sample e can be represented as (s, r, o, o^*). The post-edit LLM is defined as $\mathcal{F}'$.

Definition 2 (**Knowledge Graph**). A knowledge graph (KG) (Ji et al., 2021) stores structured knowledge as a collection of triples $\{(s, r, o) \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$, where $\mathcal{E}$ and $\mathcal{R}$ represent the set of entities and relations, respectively.

3.1 Rank-one Model Editing Framework

Rank-one model editing (ROME) (Meng et al., 2022a) is a Locate-then-edit method, this method assumes that the factual knowledge is stored in the Feedforward Neural Networks (FFNs), conceptualizing as key-value memories (Geva et al., 2021; Kobayashi et al., 2023). Specifically, the output of the l-th layer FFN for the i-th token is formulated as:

$$\mathbf{m}_i^l = f(\mathbf{W}_{in}^l \cdot \mathbf{h}_i^{l-1}) \cdot \mathbf{W}^l, \tag{1}$$

where $f(\cdot)$ denotes the activation function, and $\mathbf{h}_i^{l-1}$ is the input of FFN. To facilitate representation, we omit the superscript l in the subsequent discussion.

In this setup, the output of the first layer, $f(\mathbf{W}_{in} \cdot \mathbf{h}_i)$, serves as the keys denoted as $\mathbf{k}_i$. The outputs of the subsequent layer represent the corresponding values. Based on the hypothesis, this method utilizes casual tracing (Pearl, 2022; Vig et al., 2020) to select a specific FFN layer for editing, thereby updating the weight $\mathbf{W}$ of the second layer by solving a constrained least-squares problem:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{WK} - \mathbf{M}\|, \\ \text{subject to} \quad & \mathbf{Wk}_* = \mathbf{m}_*. \end{aligned} \tag{2}$$

Here, the objective function aims to maintain the knowledge, irrelevant to the edited sample unchanged within the LLM, where $\mathbf{K} = [\mathbf{k}_1; \mathbf{k}_2; \dots; \mathbf{k}_p]$ denotes the sets of keys encoding subjects unrelated to the edited fact, and $\mathbf{M} = [\mathbf{m}_1; \mathbf{m}_2; \dots; \mathbf{m}_p]$ are the corresponding values. The constraint is to ensure that edited knowledge can be incorporated into the FFN layer, specifically by enabling the key $\mathbf{k}_*$ (encoding subject s) to retrieve the value $\mathbf{m}_*$ about the new object o^*.

As explicated in (Meng et al., 2022a), a closed-form solution to the above optimization problem can be derived:

$$\hat{\mathbf{W}} = \mathbf{W} + \frac{(\mathbf{m}_* - \mathbf{Wk}_*)(\mathbf{C}^{-1}\mathbf{k}_*)^{\mathrm{T}}}{(\mathbf{C}^{-1}\mathbf{k}_*)^{\mathrm{T}}\mathbf{k}_*}, \tag{3}$$

where $\mathbf{C} = \mathbf{KK}^{\mathrm{T}}$ represents a constant matrix, precached by estimating the uncentered covariance of $\mathbf{k}$ based on a sample of Wikipedia text (Appendix E). Therefore, solving the optimal parameter $\hat{\mathbf{W}}$ is transformed into calculating $\mathbf{k}_*$ and $\mathbf{m}_*$.

Extending this framework, our research delineates a method to integrate graph-structured knowledge, newly and intrinsically associated with the edited knowledge, into the editing of model parameters. We will provide a detailed description of our approach in the following sections.

4 Methodology

In this section, we introduce the proposed GLAME, the architecture of which is illustrated in Figure 2. The framework comprises two key components: (1) *Knowledge graph augmentation* (KGA), which associates the knowledge of internal changes in LLMs by utilizing external knowledge graphs, and (2) *Graph-based knowledge edit* (GKE), which injects knowledge of edits and edit-induced changes into specific parameters of LLMs.

4.1 Knowledge Graph Augmentation

To accurately capture the changes in associated knowledge induced by editing in LLMs, we propose using external knowledge graphs. This approach is divided into two operational parts: First,

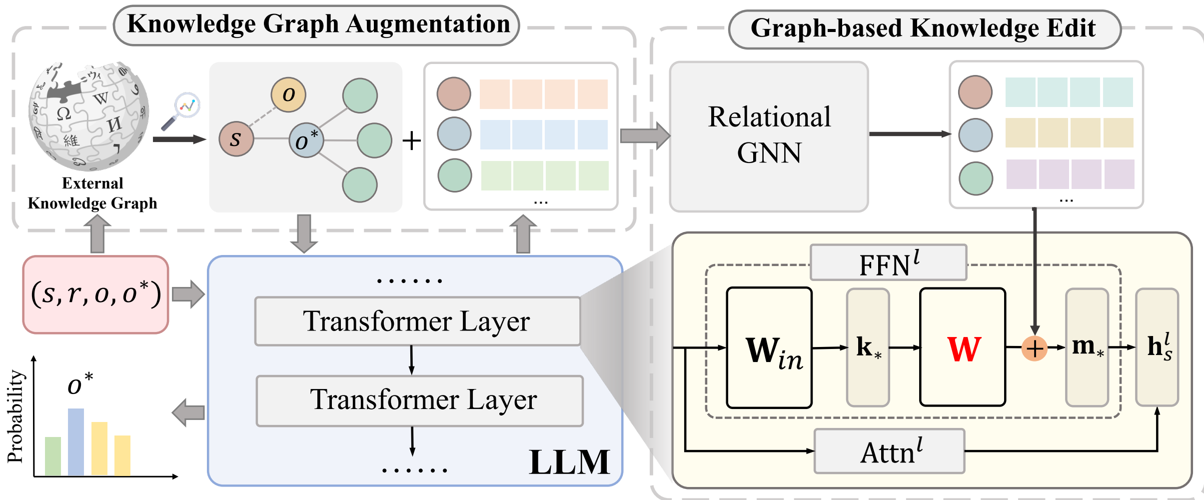


Figure 2: An illustration of GLAME architecture. We first utilize a Knowledge Graph Augmentation module to sample a high-order subgraph, recording the associated knowledge of changes caused by the edit (s, r, o, o^*). Subsequently, the entities and relations within the subgraph are encoded using the LLM, from which hidden vectors are extracted from the early layers as the initial representations of the entities and relations in the subgraph. Then, the well-designed Graph-based Knowledge Edit module leverages a relational graph neural network to incorporate new knowledge associations from the subgraph into the parameter editing process.

it leverages an external knowledge graph to construct a subgraph, capturing the altered knowledge. Then, the LLM is employed to extract the corresponding representations of entities and relations within this subgraph, serving as the initial representations.

4.1.1 Subgraph construction

We first introduce how to utilize an external knowledge graph to construct a subgraph that encapsulates the newly formed associations due to the edit.

Specifically, for a given target edit sample $e = (s, r, o, o^*)$, we initially employ o^* to match the most relevant entity within an external knowledge graph, such as Wikipedia[1]. This step is followed by the sampling of neighboring entities and their relations centered on this entity, represented as $(o^*, r_1, o_1), (o^*, r_2, o_2), \cdots, (o^*, r_n, o_m)$. These are used to construct new two-order relationships: (s, r, o^*, r_1, o_1), (s, r, o^*, r_2, o_2), $\cdots$, (s, r, o^*, r_n, o_m), thereby generating new associated knowledge as a consequence of editing. Here m denotes the maximum number of samples for each entity. Following this approach, we can sequentially sample the neighboring entities of o_1, $o_2, \cdots, o_m$, thereby constructing higher-order new knowledge associations for s. We define the maximum order of the newly constructed relationships as n. The target edit knowledge (s, r, o^*), along with these new high-order relations, forms a subgraph, termed $\mathcal{G}_n^m(e)$, which can record changes in associated knowledge partially caused by editing knowledge. n is also the maximum order of the subgraph, and together with m serve as hyperparameters to control the size of the graph.

[1] https://www.wikipedia.org/

4.1.2 Subgraph initialization

To further explicitly associate the knowledge within the LLM that is affected by the edit, we extract hidden vectors of entities and relations from the early layers of LLM (Geva et al., 2023) as the initial representations for entities and relations in the constructed subgraph.

In specific, we input entity and relation text into the LLM separately, and then select the hidden state vector of the last token of both the entity and the relation text in k-th layer as their initial representations in the subgraph:

$$\mathbf{z}_s, \mathbf{z}_r, \mathbf{z}_o = \mathbf{h}^k_{[s]}(s), \mathbf{h}^k_{[r]}(r), \mathbf{h}^k_{[o]}(o), \tag{4}$$

where $\mathbf{h}^k_{[x]}(x)$ is the hidden state vector of the last token of text x at the k-th layer of the LLM.

4.2 Graph-based Knowledge Edit

After obtaining the knowledge-enhanced subgraph, this section designs a graph-based knowledge edit module to integrate the new associated knowledge contained in the subgraph into the modified parameters of the LLM.

4.2.1 Subgraph encoding

To enhance the subject s with the newly constructed associated knowledge resulting from the editing of target knowledge, we perform message propagation and aggregation operations on the subgraph through a relational graph convolutional network (RGCN) (Schlichtkrull et al., 2018).

Formally, we encode the subgraph as follows:

$$\mathbf{z}_s^{l+1} = g\left(\sum_{o \in \mathcal{N}_s} \mathbf{W}_1 \left(\mathbf{z}_o^l + \mathbf{z}_r\right) + \mathbf{W}_2 \mathbf{z}_s^l\right), \quad (5)$$

where $\mathcal{N}_s$ is the set of neighbors of s in $\mathcal{G}_n^m(e)$, $g(\cdot)$ is the ReLU function, $\mathbf{W}_1$ and $\mathbf{W}_2 \in \mathbb{R}^{d \times d}$ are trainable weight parameter matrices in each layer, and $\mathbf{z}_s^0$, $\mathbf{z}_o^0$, and $\mathbf{z}_r$ are the corresponding entity and relation representations obtained from §4.1.2. To capture the semantic dependencies among nodes in the subgraph comprehensively, the number of layers of RGCN is set to the subgraph's maximum order n, yielding the entity representation $\mathbf{z}_s^n$ after n-layer operation.

4.2.2 Knowledge editing

Following the ROME framework (Meng et al., 2022a), in this subsection, we target specific layer l for the computation of $\mathbf{m}_*$ and $\mathbf{k}_*$. Subsequently, we employ Equation (3) to update the parameters of the second layer of the FNN, thereby accomplishing the editing of knowledge.

Computing $\mathbf{m}_*$. Given that $\mathbf{z}_s^n$ aggregates the information of neighbors under new association relations, we utilize $\mathbf{z}_s^n$ to enhance the representation at the last token of s in l-th FFN layer of the LLM:

$$\mathbf{m}_* = \mathbf{m}_s^l + \mathbf{z}_s^n, \quad (6)$$

where $\mathbf{m}_s^l$ denotes the output from the l-th FFN at the last token of s in the LLM. Further details of the FFN are delineated in Equation (1).

For each edit sample (s, r, o, o^*), our objective is to refine an RGCN to produce an enhanced representation, $\mathbf{m}_*$, that enables the LLM to accurately predict the target object o^*. Accordingly, the primary loss function is defined as:

$$\mathcal{L}_p = -\frac{1}{N}\sum_{j=1}^{N} \log \mathrm{P}_{\mathcal{F}(\mathbf{m}_s^l := \mathbf{m}_*)}[o^* \mid x_j \oplus p(s,r)],$$

where x_j is the random prefix generated by the LLM to foster optimization robustness. $\mathcal{F}(\mathbf{m}_s^l := \mathbf{m}_*)$ indicates the LLM's inference alteration through the hidden state $\mathbf{m}_s^l$ modification to $\mathbf{m}_*$.

To mitigate the impact of enhancing s on its intrinsic properties within the LLM, we aim to minimize the KL divergence between $\mathcal{F}(\mathbf{m}_s^l := \mathbf{m}_*)$ and the original model $\mathcal{F}$ without any interventions (Meng et al., 2022a):

$$\mathcal{L}_a = D_{\mathrm{KL}}\left(\mathrm{P}_{\mathcal{F}(\mathbf{m}_s^l := \mathbf{m}_*)}[x \mid p'] \parallel \mathrm{P}_{\mathcal{F}}[x \mid p']\right),$$

where p' denotes prompts in the form of "subject is a". This term serves as a regularization loss.

Ultimately, the parameters of the RGCN are optimized by minimizing the following objective function:

$$\mathcal{L} = \mathcal{L}_p + \lambda \mathcal{L}_a, \quad (7)$$

where λ adjusts the regularization strength. It is important to note that throughout the optimization process, the parameters of the LLM remain unchanged. The modification is instead focused on optimizing the parameters of the RGCN, which in turn influences the inference of the LLM.

Computing $\mathbf{k}_*$. For each edit sample (s, r, o, o^*), the $\mathbf{k}_*$ is calculated by

$$\mathbf{k}_* = \frac{1}{N}\sum_{j=1}^{N} f(\mathbf{W}_{in}^l \cdot \mathbf{h}_s^{l-1}). \quad (8)$$

Here, we also utilize N random prefixes generated in the same manner as for the computing $\mathbf{m}_*$ (Meng et al., 2022a).

After obtaining the optimized $\mathbf{m}_*$ and $\mathbf{k}_*$, we bring them into Equation (3) and then get the edited parameter $\hat{\mathbf{W}}$. Algorithm 1 provides the pseudo-code of the overall framework.

5 Experiments

In this section, we evaluate our editing method graphs for large language model editing (GLAME) by applying it to three datasets and assessing its performance on two auto-regressive LLMs. We aim to answer the following questions through experiments.

- **Q1**: How does GLAME perform in editing knowledge compared with state-of-the-art model editing methods?
- **Q2**: How do different components affect the GLAME performance?
- **Q3**: How sensitive is GLAME with different hyper-parameter settings?

5.1 Experimental Setups

5.1.1 Datasets and Evaluation Metrics

We evaluate our GLAME on three representative datasets in our experiments: COUNTERFACT (Meng et al., 2022a), COUNTERFACTPLUS (Yao et al., 2023), and MQUAKE (Zhong et al., 2023).

COUNTERFACT is a dataset that focuses on inserting counterfactual knowledge into models. We utilize three metrics on this dataset: *Efficacy Score*, measuring the success rate of edits directly; *Paraphrase Score*, indicating the model's ability to accurately recall edited knowledge in paraphrased forms, thus testing its generalization ability; and *Neighborhood Score*, assessing whether irrelevant knowledge in the LLM is disturbed.

COUNTERFACTPLUS, an extension of COUNTERFACT, presents more challenging test questions aimed at evaluating the post-edit models' ability to accurately respond to queries requiring reasoning with edited knowledge. Compared with COUNTERFACT, this assessment has higher requirements for generalization ability. Following (Yao et al., 2023), we employ *Portability Score* to evaluate the performance of all methods on this dataset. This metric offers a superior reflection of the LLMs' ability to utilize both the edited knowledge and its associated information compared to other indicators.

An introduction to **MQUAKE**, further details on COUNTERFACT and COUNTERFACTPLUS, as well as the evaluation metrics are shown in Appendix B and C. We provide results on MQuAKE dataset in Appendix F as an additional experiment.

5.1.2 Baselines

Our experiments are conducted on GPT-2 XL (1.5B) (Radford et al., 2019) and GPT-J (6B) (Wang and Komatsuzaki, 2021), and we compare GLAME with the following state-of-the-art editing methods: Constrained Fine-Tuning (FT) (Zhu et al., 2020), MEND (Mitchell et al., 2021), ROME (Meng et al., 2022a), and MEMIT (Meng et al., 2022b). To further verify the superiority of our graph-based editing method, we also compare our method with two variant models ROME-KG and MEMIT-KG. These models utilize ROME and MEMIT, respectively, to directly edit the new high-order relations, $(s, r, o^*, r, o_1), \cdots, (s, r, o^*, r, o_n)$ constructed as described in §4.1.1 and arising from the edited knowledge (s, r, o, o^*), into the LLM. We provide implementation details of baselines and GLAME in Appendix D.

5.2 Performance Comparison (RQ1)

The performance of all editors on the COUNTERFACT and COUNTERFACTPLUS is presented in Table 1. From the results, we have the following observations:

Our model GLAME secures the highest performance on the comprehensive evaluation metric, the Editing Score, surpassing other editors across most evaluation metrics. Specifically, GLAME exhibits enhancements of 11.76 % and 10.98 % in Portability Score over the best baseline models for GPT-2 XL and GPT-J, respectively. This demonstrates that our method can effectively improve the generalization ability of post-edit LLM in utilizing edited knowledge, particularly in multi-hop reasoning, by effectively introducing external knowledge graphs. GLAME, ROME, and MEMIT, are significantly better than other methods in Paraphrase and Neighborhood Scores. The reason might be these methods impose explicit constraints on editing knowledge recall and retention of editing-irrelevant knowledge. Although MEND and FT can accurately recall edited knowledge and achieve commendable results on the Efficacy Score, their lack of precision during the editing process leads to poor performance on Paraphrase, Neighborhood, and Portability Scores compared to other editors.

ROME-KG and MEMIT-KG, compared to ROME and MEMIT, demonstrate a notable degradation in performance. This indicates that simply adding extra external information for editing does not guarantee improved performance. Specifically, ROME-KG requires multiple adjustments to the model's parameters to edit high-order relationships, potentially harming the original parameters. MEMIT-KG's unconstrained incorporation of vast amounts of information into the LLM may compromise the editing of target knowledge. In contrast, GLAME, by developing an editing method tailored for graph structures, incorporates multiple pieces of associated knowledge altered due to editing into the model with just a single edit. This approach not only maintains the precision of edits but also substantially improves the efficiency of leveraging external knowledge graphs.

5.3 Ablation Studies (RQ2)

To investigate the superiority of each component of our method, we compare GLAME with different variants: GLAME w/ GCN, which omits RGCN's relational information and employs a GCN (Kipf

Editor	Effi.Score	Para.Score	Neigh.Score	Port.Score	Edit.Score
GPT-2 XL (1.5B)	22.20	24.70	78.10	10.18	20.35
FT	**100.00**	87.90	40.40	15.13	35.64
MEND	99.10	65.40	37.90	11.15	28.28
ROME	99.95	96.48	75.44	21.43	49.82
ROME-KG	73.85	72.41	74.65	5.24	17.27
MEMIT	93.79	80.22	77.05	18.71	44.67
MEMIT-KG	53.09	45.28	**77.90**	9.99	26.00
GLAME	99.84	**96.62**	76.82	**23.95**	**53.24**
GPT-J (6B)	16.30	18.60	83.00	11.44	18.64
FT	100.00	98.80	10.30	17.84	23.09
MEND	97.40	53.60	53.90	12.99	32.14
ROME	100.00	99.27	79.00	29.67	60.21
ROME-KG	68.90	67.12	78.59	13.68	34.55
MEMIT	100.00	95.23	81.26	29.77	60.24
MEMIT-KG	53.75	40.22	**82.80**	8.63	23.33
GLAME	**100.00**	**99.30**	81.39	**33.04**	**63.87**

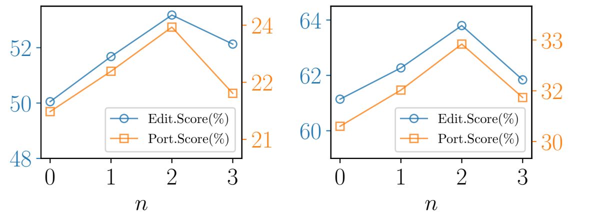
Table 1: Performance comparison on COUNTERFACT in terms of Efficacy Score (%), Paraphrase Score (%), and Neighborhood Score (%), and COUNTERFACTPLUS in terms of Portability Score (%). The Editing Score (%) is the harmonic mean of the four evaluation metrics. The best performance is highlighted in boldface, and the second-best is underlined. Gray numbers indicate a clear failure on the corresponding metric.

Editor	Effi.Score	Para.Score	Neigh.Score	Port.Score	Edit.Score
GLAME w/ MLP	99.79	91.79	**77.05**	21.73	50.55
GLAME w/ GCN	99.79	94.95	77.02	22.59	51.41
GLAME w/ RGAT	99.80	93.71	76.93	21.56	49.95
GLAME w/o GKE	**99.95**	96.48	75.44	21.43	49.82
GLAME	99.84	**96.62**	76.82	**23.95**	**53.24**
GLAME w/ MLP	99.85	98.28	80.41	30.45	61.94
GLAME w/ GCN	100.00	98.20	81.03	30.16	60.90
GLAME w/ RGAT	100.00	98.50	80.76	30.94	61.68
GLAME w/o GKE	100.00	99.27	79.00	29.67	60.21
GLAME	**100.00**	**99.30**	**81.39**	**33.04**	**63.87**

Table 2: Ablation studies on COUNTERFACT in terms of Efficacy Score (%), Paraphrase Score (%), and Neighborhood Score (%), and COUNTERFACTPLUS in terms of Portability Score (%).

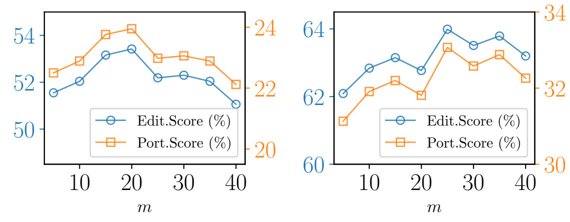
and Welling, 2017) for subgraph encoding in the GKE module; GLAME w/ RGAT, which utilizes relational graph attention mechanism (Lv et al., 2021) for subgraph encoding; GLAME w/ MLP, which neglects graph structural information, relying solely on MLP for encoding entity representations within the GKE module; and GLAME w/o GKE, which removes the GKE module and degenerates into the ROME. The results are shown in Table 2 and we have the following observations:

GLAME outperforms both GLAME w/ MLP and GLAME w/o GKE on most evaluation metrics, especially in Portability Score and Editing Score. This confirms that integrating structured knowledge altered through the GKE module effectively enhances the generalization ability of the post-edit model. Additionally, GLAME w/ MLP, GLAME w/ RGAT, and GLAME w/ GCN also achieve better performance in Editing Score compared to GLAME w/o GKE. These improvements verify that the effective incorporation of external information: the hidden state vector of the sub-



(a) GPT-2 XL (b) GPT-J

Figure 3: Performance of GLAME with different subgraph order n in terms of Edit.Score and Prot.Scores.



(a) GPT-2 XL (b) GPT-J

Figure 4: Performance of GLAME with different maximum number m of neighbors in terms of Edit.Score and Prot.Score.

ject entity and its neighbors from the early layers of LLM, contributes to the performance of edits. Furthermore, compared to GLAME w/ GCN, the performance of GLAME is further improved, highlighting the importance of relations in LLM's recognition of complex graph-structured knowledge associations. However, compared to GLAME, the performance of GLAME w/ RGAT declines. This decline could be due to the complexity of RGAT's structure and parameters, which poses challenges to its optimization process.

5.4 Sensitivity Analysis (RQ3)

To further explore the sensitivity of GLAME to important hyper-parameters, we examine the impact of key hyperparameters, the maximum order n of subgraph, and the maximum number m of sampled neighbors, on the performance of GLAME. Further results are described in Appendix G.

5.4.1 Effect of maximum subgraph order n

Subgraph construction is a vital operation of the knowledge graph augmentation module (§4.1.1). The maximum order of the subgraph decides the scope of associated knowledge affected by the edited knowledge. In this part, we conduct GLAME with different subgraph order n in the GKE module on GPT-2 XL and GPT-J in terms of Editing and Portability Score. We set n in the range of $\{0, 1, 2, 3\}$. The results are shown in Figure 3. The main observations are as follows:

Increasing the maximum subgraph order n significantly improves the post-edit model performance, peaking at $n = 2$ for two LLMs. GLAME with $n > 0$ consistently outperforms GLAME with $n = 0$. We attribute the improvement to the incorporation of associated knowledge that has been altered due to editing. However, as the maximum order exceeds 2 ($n > 2$), the post-model's performance begins to decline, which may be because the use of higher-order information makes it easy to introduce noise to the editing process.

5.4.2 Effect of the maximum number m of neighbors

To further investigate how the size of subgraph affects the editing performance, we conduct experiments with GLAME, varying the maximum numbers m of neighbors per node within the KAG module on GPT-2 XL and GPT-J in terms of Editing and Portability Score. The results are depicted in Figure 4. Specifically, we observe a consistent improvement in editing performance as the number of neighbors increased from 5 to 20 for GPT-2 XL, and up to 25 for GPT-J. This suggests that incorporating more neighbors can enhance the representation of the central entity, so that the graph structure may better reflect changes caused by edited knowledge. However, as the m continued to increase, the model's performance began to decline. This decline could be attributed to the introduction of noise by an excessive number of neighboring nodes, and the increased subgraph size may escalate the optimization difficulty for the RGCN.

6 Conclusion

In this paper, we have proposed a novel method GLAME for large language model editing. GLAME leverages a knowledge graph augmentation module to capture the changes in associated knowledge by constructing an external graph. Following this, we have introduced a graph-based knowledge edit module that utilizes a relational graph neural network to seamlessly integrate new knowledge associations from the constructed subgraph into the LLM's parameter editing framework. Experimental results on two LLMs and extensive analysis have demonstrated the effectiveness and superiority of GLAME in model editing tasks.

Limitations

In this section, we discuss the limitations of our GLAME.

The first limitation is that our framework's reliance on knowledge graphs may be constrained by the availability and quality of relevant knowledge. In cases where related knowledge is scarce or the knowledge graph is of low quality, the model's performance may suffer. Despite employing a simple and straightforward subgraph sampling strategy, we have achieved promising results. In the future, we plan to develop more sophisticated subgraph sampling strategies to enhance subgraph quality and more accurately capture knowledge changes resulting from editing. Additionally, these strategies aim to increase sampling speed and reduce subgraph size.

The second limitation is that our framework may be restricted in some unstructured edit scenarios, such as event-based knowledge editing or scenarios with no explicit association to the knowledge graph. In these scenarios, extracting key entities is challenging, requiring additional entity extraction algorithms or tools to extract effective key entities from the edit samples for subgraph construction. Although these algorithms and tools are well-developed, they may have limitations in terms of efficiency or flexibility. In the future, we will design more flexible strategies to identify key entities in edit samples and construct associated subgraphs, extending our method to more general editing scenarios.

The third limitation is the potential for factual consistency problems in LLMs after editing. Reasoning based on updated knowledge may not necessarily align with real-world facts. For example, when we update the knowledge from "James plays for the Miami Heat" to "James plays for the Los Angeles Lakers," there is a high probability that James' workplace will change. However, the complexity of the real world may render the inference "James works in LA" not necessarily true, as he could be working remotely. Our GLAME injects explicit higher-order relationships to make LLM aware of changes in higher-order knowledge, such as "James - plays for - Los Angeles Lakers - located in - Los Angeles." The external knowledge graph ensures the correctness of the injected knowledge. However, whether the edited LLM will draw the conclusion "James works in LA" based on this knowledge primarily depends on the capability of the LLM, and requires further in-depth exploration in the future.

Ethical Considerations

We realize that there are risks in developing generative LLMs, so it is necessary to pay attention to the ethical issues of LLMs. We use publicly available pre-trained LLMs, i.e., GPT-2 XL (1.5B) and GPT-J (6B). The datasets are publicly available, i.e., COUNTERFACT, COUNTERFACTPLUS, and MQUAKE. All models and datasets are carefully processed by their publishers to ensure that there are no ethical problems.

References

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*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Annual Meeting of the Association for Computational Linguistics*, pages 8493–8502.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Conference on Empirical Methods in Natural Language Processing*, page 12216–12235.

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

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2023. Feed-forward blocks control contextualization in masked language models. *arXiv preprint arXiv:2302.00456*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Conference On Knowledge Discovery and Data Mining*, page 1150–1160.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Annual Conference on Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. In *International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. In *International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Annual Conference on Neural Information Processing Systems*, pages 8024–8035.

Judea Pearl. 2022. Direct and indirect effects. In *Probabilistic and causal inference: the works of Judea Pearl*, pages 373–392.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Extended Semantic Web Conference*, pages 593–607.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Annual Conference on Neural Information Processing Systems*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. *Annual Conference on Neural Information Processing Systems*, 33:12388–12401.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J Smola, and Zheng Zhang. 2019. Deep graph library: Towards efficient and scalable deep learning on graphs. *International Conference on Learning Representations Workshop on Representation Learning on Graphs and Manifolds*.

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. In *Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Conference on Empirical Methods in Natural Language Processing*, page 15686–15702.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

A Pseudocode

Algorithm 1 provides the pseudo-code of our editing method GLAME.

B Datasets Detail

B.1 Details of COUNTERFACT Dataset

Table 3 shows an example from the COUNTERFACT dataset. Each entry contains an edit request, several paraphrase prompts, and neighborhood prompts. In this example entry, the edit request aims to change the LLM's knowledge from *Danielle Darrieux's mother tongue is French* to *Danielle Darrieux's mother tongue is English*,

Algorithm 1: Editing procedure

Input: LLM $\mathcal{F}$; Edit sample (s, r, o, o^*); Initial RGCN parameters
Output: The post-edit $\mathcal{F}'$
/* Subgraph Graph Construction */
Obtain subgraph $\mathcal{G}_n^m(e)$ from a external knowledge graph and edit sample;
/* Subgraph initialization */
$\mathbf{z}_s, \mathbf{z}_r, \mathbf{z}_o \leftarrow$ Eq (4), $s, r, o \in \mathcal{G}_n^m(e)$;
/* Optimizing $\mathbf{m}_*$ */
while not converged **do**
  /* Subgraph encoding */
  $\mathbf{z}_s^n \leftarrow \mathrm{RGCN}(\mathcal{G}_n^m(e))$, Eq (5);
  /* Computing $\mathbf{m}_*$ */
  $\mathbf{m}_* \leftarrow$ Eq (6) ;
  /* Learning Objective */
  $\mathcal{L} \leftarrow \mathcal{L}_p + \lambda \mathcal{L}_a$, Eq (7);
  Update parameters of RGCN.
end
/* Computing $\mathbf{k}_*$ */
$\mathbf{k}_* \leftarrow$ Eq (8);
/* Updating the parameters of the FNN at the specified layer */
$\hat{\mathbf{W}} \leftarrow$ Eq (3);
Return post-edit LLM $\mathcal{F}'$

where *Danielle Darrieux* corresponds to s, *the mother tongue of* corresponds to r, *French* corresponds to o, and *English* corresponds to o^* in edit sample (s, r, o, o^*). Paraphrase prompts are semantic variations of the target prompt *Danielle Darrieux's mother tongue*, while neighborhood prompts are those that share the same relation with the edit request but have different subjects, whose knowledge should remain unchanged by the edit.

Our train/test dataset splits are kept the same as (Meng et al., 2022a). Similarly, we evaluate our method using the first 7500 records on GPT-2 XL, and the first 2000 records on GPT-J. Note that for methods not employing hypernetworks, including our GLAME, there is no requirement for training with the data from the training set.

B.2 Details of CounterFactPlus Dataset

The CounterFactPlus dataset serves as a supplementary expansion of the original CounterFact dataset, selecting 1031 entries as a subset of the original data and enriching them with new test questions based on the original content. Each entry contains the same edit request as found in CounterFact, with additional questions and answers that require LLM to do further reasoning based on the edited knowledge.

An example entry from the dataset is showcased in Table 4. In this example entry, the edit request entails modifying the LLM's knowledge from *Spike Hughes originates from London* to *Spike Hughes originates from Philadelphia*. This edit introduces new knowledge associations, such as *(Spike Hughes, originates from, Philadelphia, known for, cheesesteaks)*, leading to a multi-hop question *What famous food is associated with the city where Spike Hughes originates from?*. The edited LLM should respond with the correct answer *Cheesesteaks* for this multi-hop question, rather than the original answer associated with the question. The related knowledge association *(Philadelphia, known for, Cheesesteaks)* used to construct the multi-hop question is labeled as "Recalled relation" in the dataset. In our work we primarily focus on the multi-hop reasoning aspect, aiming to assess GLAME's capacity to capture relevant changes in knowledge.

B.3 Details of MQuAKE Dataset

Similar to CounterFactPlus, MQuAKE is a more challenging dataset that also focuses on evaluating models' ability to perform further reasoning using newly edited knowledge. Each entry in this dataset may involve multiple edits and contain multi-hop reasoning questions that require reasoning from 2 to 4 hops to answer correctly, posing stricter requirements on the post-model's generalization capability.

Table 5 illustrates an example from MQuAKE dataset. The example entry requires two edits to the LLM, inserting new knowledge *(Betty Carter, plays, instrumental rock)* and *(USA, head of state, Norodom Sihamoni)*. Accordingly, a 3-hop question "*Who is the head of state of the country from which the music genre associated with Betty Carter originated?*" is constructed to assess the post-edit LLM's ability to employ edited knowledge and its associated knowledge. Following (Zhong et al., 2023), our evaluation also focuses on a subset of 3000 entries, evenly distributed across $\{2, 3, 4\}$-hop questions, with each category comprising 1000 entries.

Property	Value
Edit Request	The mother tongue of {Danielle Darrieux} is *French* → *English*
Efficacy_prompt	The mother tongue of Danielle Darrieux is
Paraphrase_prompt	Where Danielle Darrieux is from, people speak the language of
Neighborhood_prompt	Michel Rocard is a native speaker of

Table 3: An Example of COUNTERFACT dataset

Property	Value
Edit Request	{Spike Hughes} originates from *London* → *Philadelphia*
Recalled relation	(Philadelphia, known for, cheesesteaks)
New Question	What famous food is associated with the city where Spike Hughes originates from?
New Answer	Cheesesteaks

Table 4: An Example of the COUNTERFACTPLUS dataset

Property	Value
Edit Request A	The type of music that {Betty Carter} plays is *jazz* → *instrumental rock*
Edit Request B	The name of the current head of state in {USA} is *Donald Trump* → *Norodom Sihamoni*
New Question	Who is the head of state of the country from which the music genre associated with Betty Carter originated?
Original Relation	(Betty Carter, genre, jazz), (jazz, country of origin, United States of America), (United States of America, head of state, Donald Trump)
Original Answer	Donald Trump
New Relation	(Betty Carter, genre, instrumental rock), (instrumental rock, country of origin, United States of America), (United States of America, head of state, Norodom Sihamoni)
New Answer	Norodom Sihamoni

Table 5: An Example of MQUAKE dataset

C Evaluation Metrics

We adopt three widely-used metrics (Meng et al., 2022a,b), Efficacy Score, Paraphrase Score, and Neighborhood Score to evaluate all editors on COUNTERFACT dataset, and use Portability Score (Yao et al., 2023) on COUNTERFACTPLUS dataset. We utilize the harmonic mean of four metrics, Editing Score, to evaluate each editor's overall capabilities. Each metric is calculated as follows:

Efficacy Score is to test whether the post-edit LLMs can correctly recall the new target entity when given the edit prompt $p(s,r)$. It is calculated by

$$\mathbb{E}\left[\mathbb{I}\left[\mathrm{P}_{\mathcal{F}'}\left(o^* \mid p(s,r)\right) > \mathrm{P}_{\mathcal{F}'}\left(o \mid p(s,r)\right)\right]\right].$$

Paraphrase Score measures the performance of the post-edit LLM on rephase prompt set P^P of edit prompt $p(s,r)$. The calculation is similar to the Efficacy Score:

$$\mathbb{E}_{p \in P^P}\left[\mathbb{I}\left[\mathrm{P}_{\mathcal{F}'}\left(o^* \mid p\right) > \mathrm{P}_{\mathcal{F}'}\left(o \mid p\right)\right]\right].$$

Neighborhood Score measures whether the post-edit LLM assigns the higher probability to the correct fact on the prompt set P^N, which consists of distinct but semantically similar prompts $p(s,r)$. The calculation is defined as:

$$\mathbb{E}_{p \in P^N}\left[\mathbb{I}\left[\mathrm{P}_{\mathcal{F}'}\left(o^* \mid p\right) < \mathrm{P}_{\mathcal{F}'}\left(o \mid p\right)\right]\right].$$

This metric can assess the extent of the impact that edits have on unrelated knowledge.

Portability Score measures the accuracy of the post-edit model on the multi-hop question set P about the edit sample:

$$\mathbb{E}_{p \in P}\left[\mathbb{I}\left[\mathcal{F}'(p) = o^{*\prime})\right]\right].$$

Given the challenges associated with evaluating the data, the Portability Score provides a more accurate reflection of the model's generalization capabilities compared to other metrics.

D Baselines

Our experiments are conducted on GPT-2 XL (1.5B) (Radford et al., 2019) and GPT-J (6B) (Wang and Komatsuzaki, 2021), and we compare GLAME with the following state-of-the-art editing methods:

Constrained Fine-Tuning (FT) (Zhu et al., 2020) involves fine-tuning specific layers of the LLM's parameters directly using gradient descent, while imposing a norm constraint on the weight changes to prevent catastrophic forgetting.

MEND (Mitchell et al., 2021) constructs a hypernetwork based on the low-rank decomposition of gradients to perform editing.

ROME (Meng et al., 2022a) is based on the hypothesis that knowledge in LLMs is stored in the FFN module, and uses optimization to update a FFN layer to insert knowledge.

MEMIT (Meng et al., 2022b) builds on the ROME method, specializing in batch-editing tasks by performing edits on a range of FFN layers.

To further verify the superiority of our graph-based editing method, we also compare our method with two variant models **ROME-KG** and **MEMIT-KG**. The two baselines aim to evaluate the performance of directly adding the same amount of external information to the LLM without using the GKE module. For each record in our test dataset, we construct edit requests that contain high-order relationships from the knowledge graph. For instance, given the original edit content *"Spike Hughes originates from London $\rightarrow$ Washington"* and a related knowledge graph triple *(Washington, capital of, United States of America)*, we then create a new edit request to insert this knowledge into the LLM: *"Spike Hughes originates from Washington, capital of United States of America"*, using either ROME or MEMIT.

E Implementation Details

We implement our GLAME method with **PyTorch**[2] (Paszke et al., 2019) and the **DGL**[3] (Wang et al., 2019). Within the Knowledge Graph Augmentation (KGA) module, we set the maximum subgraph order n to 2 for both GPT-2 XL and GPT-J, with the maximum number of sampled neighbors m set to 20 for GPT-2 XL and 40 for GPT-J. Hidden vectors for entities and relations are extracted from the 5th layer of GPT-2 XL ($k = 5$) and the 2nd layer of GPT-J ($k = 2$), respectively, to initialize the subgraph representations. For the GKE module, we perform editing operations on the 9th layer of GPT-2 XL ($l = 9$) and the 5th layer of GPT-J ($l = 5$) based on ROME's locating results. The hidden embedding sizes for the RGCN are set to 1600 for GPT-2 XL and 4096 for GPT-J. For RGCN optimization, the AdamW (Loshchilov and Hutter, 2018) optimizer is used with a learning rate of 5×10^{-1}, the optimal regularization factor λ is 6.25×10^{-2} for COUNTERFACT and 7.5×10^{-2} for both COUNTERFACTPLUS and MQUAKE. To prevent overfitting, we perform early-stop when the loss is lower than 1×10^{-2}. Since our method does not require an additional training set for training, we select important hyperparameters on the training set. For the covariance matrix estimation $\mathbf{C}$, which represents the pre-computed keys in a layer, we directly use the results computed by ROME (Meng et al., 2022a), which is collected using $100,000$ samples of Wikitext. The number N of random prefixes generated for calculating $\mathbf{m}_*$ and $\mathbf{k}_*$ is to 50, serving as a method of data augmentation for the original edits. For other baselines, we conduct our experiment with the code implemented by ROME (Meng et al., 2022a), and all the settings of the baselines we compare, including the hyperparameters, are consistent with (Meng et al., 2022a,b). All experiments are conducted on NVIDIA Tesla A100 (80G) and AMD EPYC 7742 CPU.

E.1 Wikidata Sampling Details

In the Knowledge Graph Augmentation (KGA) module, we leverage Wikidata[4] as an external knowledge graph to construct a subgraph for each edit sample (s, r, o, o^*). Specifically, we employ Wikidata's API[5] to perform a SPARQL query, retrieving all outgoing edges of the entity o^*. After retrieving these edges, we prioritize the triples by sorting them to foreground the most potentially valuable information. This prioritization is based on the frequency of each relation's occurrence across the dataset. Relations that appear less frequently are deemed more valuable as they may

[2] https://pytorch.org/

[3] https://www.dgl.ai/

[4] https://www.wikidata.org/

[5] https://query.wikidata.org/sparql

embody information of higher specificity or rarity, similar to principles of information entropy where less frequent occurrences convey more information.

As datasets COUNTERFACT, COUNTERFACT-PLUS, and MQUAKE are directly constructed using Wikidata, each edited entity within these datasets is linked with its corresponding Wikidata item ID, allowing for precise sampling. **Note that in our experiments, the constructed subgraphs are filtered to exclude the standard answers to the multi-hop questions.** This operation ensures that the improvement in model performance is attributed to an enhancement in the generalization ability, rather than simply being influenced by specific answer patterns within the subgraphs.

E.2 Evaluation Details

In our experiments, we assessed the Efficacy Score, Paraphrase Score, and Neighborhood Score on the COUNTERFACT dataset following the method in (Meng et al., 2022a). We used specific prompts as inputs to the LLM and examined the model's prediction probabilities for both the original entity o and the edited entity o^*. For the COUNTERFACT-PLUS dataset, our assessment of the Portability Score involved prompting the LLM with multi-hop questions, and then verifying whether the output generated includes the correct answers. To accommodate variations in phrasing or synonyms between the model's output and the standard answer, fuzzy matching was employed. In practice, we utilized the partial ratio algorithm from Fuzzywuzzy[6] library, which calculates similarity based on the Levenshtein distance. Regarding the MQUAKE dataset, we adopt the Efficacy Score to evaluate the effectiveness of different editing methods.

F Results on MQUAKE

To further demonstrate the capability of GLAME in capturing the associated knowledge changes due to edits, we compare our GLAME with two competitive baseline models, ROME and MEMIT, on the more challenging MQUAKE (Zhong et al., 2023) dataset. The results are shown in Table 6. From the results, we find that our GLAME achieves significant improvements over ROME and MEMIT across questions of varying hops. With an increase in the number of hops, which necessitates a greater utilization of edited knowledge, the performance of all editing methods begins to decline. However, GLAME exhibits the highest relative improvement on 4-hop questions than SOTA methods, which is likely attributed to our model's effective capture of associative knowledge, enabling it to construct a more solid knowledge representation. Such an advantage becomes significant in the context of 4-hop questions, where the complexity of reasoning is markedly higher. This emphatically validates the effectiveness of our model in improving the post-edit model's generalization capacity in processing edited knowledge.

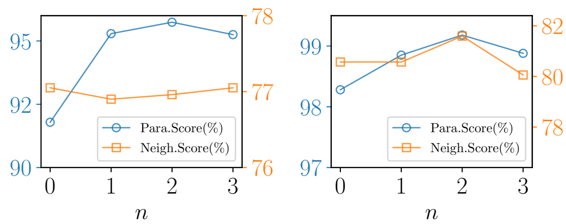
Editor	Average Score	2-hops	3-hops	4-hops
GPT-2 XL (1.5B)	21.29	25.13	23.3	15.43
ROME	29.70	39.80	31.07	18.23
MEMIT	26.52	35.87	27.70	16.00
GLAME	**31.48**	**41.83**	**32.10**	**20.50**
$\Delta Improve$	5.98%	5.10%	3.32%	12.45%
GPT-J (6B)	16.83	15.80	23.60	11.10
ROME	33.15	42.80	38.37	18.27
MEMIT	27.46	35.77	33.03	13.57
GLAME	**35.11**	**44.13**	**39.87**	**21.33**
$\Delta Improve$	5.92%	3.11%	3.91%	16.75%

Table 6: Performance comparison of editors on multi-hop questions of MQUAKE dataset in terms of Efficacy Score (%).

G Sensitivity Analysis

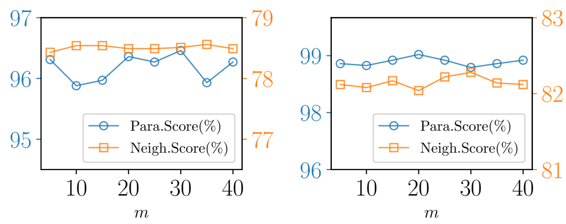
The maximum order of subgraph n and the maximum number m of sampled neighbors are two key hyper-parameters in GLAME. Figure 5 and 6 depict the performance of GLAME across various n and m values, as measured by Paraphrase and Neighborhood Score. From Figure 5, we observe that increasing the order of the subgraph can enhance the post-edit model's performance in terms of the Paraphrase Score. This demonstrates that incorporating more new associated knowledge with edits can improve the generalization ability of the post-edit model in processing edited knowledge. In contrast, Neighborhood Score exhibits greater stability with respect to the value of n, indicating that our editing method inflicts minimal harm on the model's original capabilities. In Figure 6, we can find that the Paraphrase and Neighborhood Scores are more stable than the Editing and Portability Scores in Figure 4. This stability may be attributed to the design of the loss function and those random prefixes added during optimization, which impose certain constraints on scenarios related to these two metrics, resulting in more stable behavior as the

[6] https://github.com/seatgeek/fuzzywuzzy



(a) GPT-2 XL (b) GPT-J

Figure 5: Performance of GLAME with different subgraph order n in terms of Paraphrase and Neighborhood Scores (the left y-axis shows Paraphrase Score and the right y-axis shows Neighborhood Score).



(a) GPT-2 XL (b) GPT-J

Figure 6: Performance of GLAME with different maximum number m of neighbors in terms of Paraphrase and Neighborhood Scores (the left y-axis shows Paraphrase Score and the right y-axis shows Neighborhood Score).

subgraph changes.

It is worth noting that when $n = 1$, the constructed subgraph will only include the subject entity, relation and new object entity (denoted as $s - r - o*$). In this case, GLAME demonstrates relatively better editing performance compared to ROME and MEMIT, achieving an Editing Score of 51.68 on GPT2-XL and 62.27 on GPT-J. This implies that even in the worst-case scenario, where no related information about the entities to be edited can be found in the external KG through the subgraph sampling, our GLAME can still perform basic editing and achieve better performance.

H Efficiency Analysis

The time overhead introduced by our proposed GLAME mainly consists of subgraph sampling and knowledge editing. The first part involves sampling subgraphs from external knowledge graphs such as Wikidata. In our work, we use Wikidata's API for the sampling operation. In practice, each edit only requires sending a simple HTTP request to the Wikidata server, which does not introduce significant overhead. Although the time taken depends on the network conditions, in our experiments, obtaining the subgraph for each edit consistently took less than 1 second.

Subgraph Size	10	20	30	40	50
Avg time per edit	5.35	5.95	6.37	6.89	7.56

Table 7: Edit time (seconds) of GLAME in GPT-J under different subgraph size.

To further examine the efficiency of our GLAME, we measure the edit time of GLAME in GPT-J on subgraphs of different sizes. The results are shown in Table 7. From the results, we can see that the time overhead for GLAME indeed increases with the number of subgraph nodes. However, within the subgraph size range where the model exhibits optimal performance (approximately 20-40 nodes), GLAME's additional time requirement is not significantly greater than that of ROME (5.25s). We believe this editing time is affordable given the improvement the post-edit LLM's generalization ability and editing performance.

I Case Study

In this section, we present several generation examples on GPT-J utilizing three knowledge editing models: GLAME, ROME, and MEND, to demonstrate the efficacy of knowledge editing through multi-hop questions in COUNTERFACTPLUS. We focus on the edited models' ability to leverage newly inserted knowledge for reasoning in response to a given prompt while maintaining contextual coherence. The generation examples are shown in Figure 7.

Example A [Case 1662 in COUNTERFACTPLUS]. In this example, counterfactual knowledge "*Heritage Range is in Africa*" was inserted. To answer the multi-hop question correctly, the edited model must first recall the newly inserted knowledge *(Heritage Range, located in, Africa)*, followed by *(Africa, highest peak, Mount Kilimanjaro)*. Notably, GLAME provided the correct answer, whereas ROME and MEND seemed to fail in recalling the inserted knowledge during reasoning, offering answers such as "*the Great Plains*" and "*Mount McKinley*" based on Americas-related knowledge, indicating a weaker generalization.

Example B [Case 5431 in COUNTERFACTPLUS]. In this example, a piece of new knowledge "*Association football originated in Sweden*" was in-

A. Edit Request: Heritage Range is in **Africa**

Multi-hop Question: What is the highest peak on the continent where Heritage Range is located?

Answer: Mount Kilimanjaro

GLAME: *What is the highest peak on the continent where Heritage Range is located? The answer to this question, most simply, is* **Kilimanjaro**. It is the highest mountain on the **African continent** and, indeed, the highest mountain in Africa. It is also the tallest free standing mountain in the world.

ROME: *What is the highest peak on the continent where Heritage Range is located? The answer to this question, most simply, is* **the Great Plains**. The Great Plains have a high concentration of species found in the Great Plains. How many species are there in the Great Plains?

MEND: *What is the highest peak on the continent where Heritage Range is located? The answer to this question, most simply, is* the summit of the highest peak in the Heritage Range. That would be **Mount McKinley**, located on the North slope of the range. Mount McKinley is the tallest peak in **North America**.

B. Edit Request: Association football originated in **Sweden**

Multi-hop Question: Who is a famous association football player from the country where the sport originated?

Answer: Zlatan Ibrahimovic

GLAME: *Who is a famous association football player from the country where the sport originated? The answer to this question, most simply, is* **Zlatan Ibrahimovic**, who is from Sweden. But the answer also includes the players who have achieved great success in Sweden.

ROME: *Who is a famous association football player from the country where the sport originated? The answer to this question, most simply, is* Sweden. The country has produced a lot of world-class football players in the recent past and it's easy to see why. **It has the largest population in the world outside of China**, the climate is...

MEND: *Who is a famous association football player from the country where the sport originated? The answer to this question, most simply, is* **the Brazilian footballer Pele**. But there are plenty of other Brazilian footballers who are well and truly known in their home country, even though they may not be as well known in other countries.

Figure 7: GPT-J generation examples of GLAME, ROME and MEND. Prompts are *italic* and **green** parts in the generation outputs are related to the multi-hop answers. **Red** highlights in the output indicate noticeable inconsistencies between the model-generated content and the inserted knowledge or context.

serted. Answering the multi-hop question required further reasoning to identify Sweden's famous athlete, *Zlatan Ibrahimovic*. GLAME maintained coherence with the context and correctly recalled the answer. Although ROME managed to recall information related to "*Sweden*", its answer was inconsistent with the prompt, only mentioning "*Sweden*" and mistakenly claiming "*Sweden*" has the largest population in the world outside of China, showing signs of hallucination. MEND, again, failed to recall the newly inserted knowledge, providing an unrelated answer about the Brazilian footballer Pele.