Knowledge Graph-Driven Memory Editing with Directional Interventions

Jinhu Fu[1], Kun Wang[2], Chongye Guo[3], Junfeng Fang[2],
Wentao Zhang[4], Sen Su[1],
[1]Beijing University of Posts and Telecommunications, [2]Nanyang Technological University,
[3]Shanghai University, [4]Center for Machine Learning Research, Peking University,
Correspondence: Sen Su susen@bupt.edu.cn

Abstract

Large Language Models (LLMs) have revolutionized language processing and understanding, yet their performance is hampered by inaccuracies and outdated information. Model editing techniques offer a solution but face two key challenges: **(I)** Most methods inject knowledge by constructing rigid loss, which leads to poor compatibility when dealing with higher-order multi-hop problems. **(II)** Locate-then-edit vein, by altering pre-trained parameters, inevitably affect normal knowledge and even face the catastrophic forgetting. In this paper, we introduce KGMET, a framework that constructs knowledge graphs using available information to guide the direction of knowledge editing, enabling **consistent**, **aligned**, and **stable** information during **large-scale** editing scenario. Furthermore, KGMET goes beyond this by employing orthogonal constraints to block the interference of irrelevant information, ensuring the updates are both controllable and generalizable. Experiments on Multi-Conterfact, ZsRE, and MQuAKE datasets using Llama-3-8B, GPT-J-6B, and GPT-2-XL models showcase improvements over state-of-the-art methods, with $\uparrow 5\% - 17\%$ in multi-hop tasks while remaining generalizable (at least $\uparrow 20\%$ in fluency). Our code is available on GitHub[1].

1 Introduction

Large Language Models (LLMs) are endowed with robust inferential and computational abilities, which have led to their extensive application in recent years (Radford et al., 2019; Roberts et al., 2020). Nevertheless, these models may retain incorrect or obsolete information, prompting the development of model editing techniques based on LLMs (Mitchell et al., 2022; De Cao et al., 2021). Diverging from retraining or fine-tuning strategies (Hu et al., 2021; Liu et al., 2021), LLM-based model editing techniques advocate for the modification of a subset of parameters, akin to a surgical approach that precisely locates and edits knowledge (Meng et al., 2022a). This method is markedly less resource-intensive than retraining (Zhang et al., 2024b), thereby attracting considerable interest from researchers(Yu et al., 2024; Ma et al., 2024; Zhang et al., 2024c). Concretely, model editing techniques can be classified into weight-preserving (De Cao et al., 2021; Hartvigsen et al., 2024; Zheng et al., 2023) and weight-modifying approaches(Meng et al., 2022a,b; Ma et al., 2024).

In this study, we investigate weight-modifying model editing techniques. Existing weight-modifying methods follow a "locate-then-edit" workflow (Meng et al., 2022a), which locates influential parameters $\mathbf{W}$ through causal mediation analysis (PEARL, 2001; Vig et al., 2020) and subsequently completes knowledge editing by inserting a perturbation Δ (Meng et al., 2022b). Though efficacy (Meng et al., 2022b; Li et al., 2024), it *encounters inaccuracies in multi-hop tasks associated with the edited knowledge*, as the model is unable to accurately calibrate and edit the relevant knowledge changes (Zhang et al., 2024a). For example, when the U.S. president is edited from Biden to Trump, the model still incorrectly responds with Biden's wife when queried "Who is the wife of the U.S. President?" rather than Trump's (Zhang et al., 2024a; Zhong et al., 2023; Zhang et al., 2024c).

To this end, GLAME (Zhang et al., 2024a) endeavors to incorporate knowledge graph representation learning techniques into the editing process, yielding some positive results. However, this approach is fraught with two critical issues:

➠ **Batch Dilemma.** GLAME constrains to updating the weights of a single MLP layer, which makes the method ineffective in handling extensive editing requests (Meng et al., 2022b; Li et al., 2024). As demonstrated in Figure 1 (a), GLAME (Meng et al., 2022a) fail to perform adequately when applied to

[1]https://github.com/FredJDean/KGMET

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 4860–4874
November 4-9, 2025

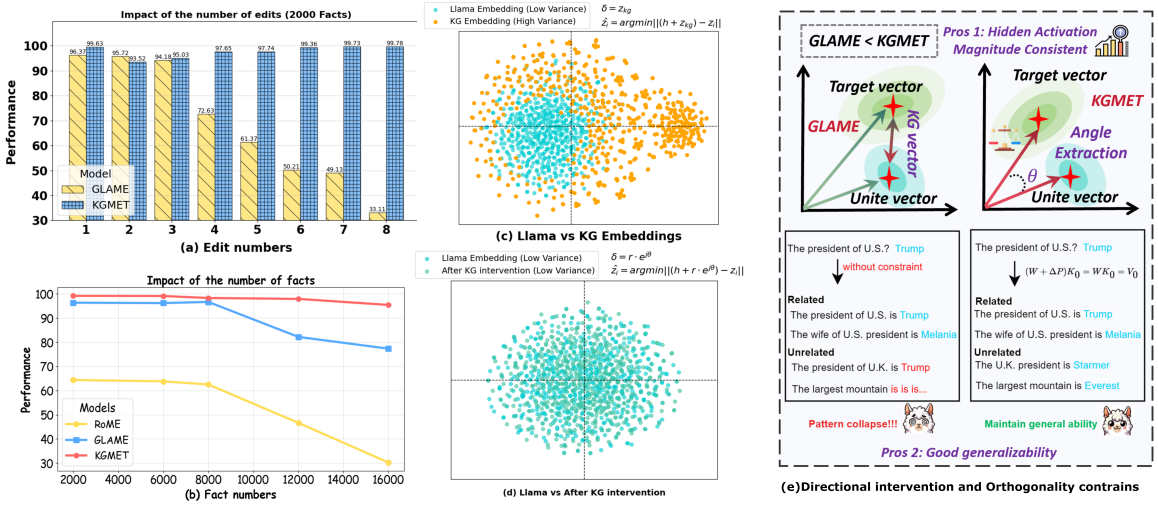

Figure 1: Model editing-related preparatory experiments and the contributions of the paper. In (c) and (d), we selected 1,000 facts and projected the representations of subject tokens from the knowledge graph (KG) and the representations of the subjects after processing through the 8-th layer of `Llama-3-8B`. (e) shows the reason of pattern collapse in `GLAME`.

scenarios involving multiple edits ($\downarrow\sim 22\%$ after 3-th editing batches). Worse still, when facts increase dramatically (> 8000), edit failures began to appear ($\sim 95\% \rightarrow 81\%$) compared with others.

➠ **Pattern Collapse.** `GLAME` recklessly treats the representation of the knowledge graph as the amount of hidden state change δ (Meng et al., 2022b). Previous researches (Pham and Nguyen, 2024; Dai et al., 2021) find an important property that are maintained by powerful LLMs: activations within the same layer tend to have roughly the same vector norm. We refer to this as the **Magnitude Consistency** property. This observation highlights a key limitation of `GLAME`: cannot simultaneously maintain activation magnitude consistency.

To tackle these challenges, we present a knowledge graph-guided methodology for mass-batch model editing, termed `KGMET`, which aims to minimize adverse effects on LLMs during multi-hop reasoning and batch-editing tasks. `KGMET` mainly consists of three key technologies: ❶ **Directional intervention strategy**: we first propose a directional intervention approach, which enables knowledge graph enhancement methods to be adapted to mass batch editing scenarios; ❷ **Orthogonality constraints**: the use of orthogonality constraints to enable the model to maintain its general performance; ❸ **Multiple layers of editing**: the editing is extended to multiple layers, which makes it more capable of generalizing to a larger number of factual requests.

We conduct comprehensive experiments on three widely adopted editing datasets. Empirical results demonstrate that `KGMET` is ❶ **Higher performing**, surpassing most locating-then-editing methods by average 27.3%; ❷ **Outstanding higher-order capabilities**, outperforming SOTA baselines like `GLAME` in neighborhood success metrics above 5% ~10%, and above 3% ~17% compare to other methods in MQuAKE (Zhong et al., 2023) multi-hop inference tasks; ❸ **Effective maintenance of general ability**, in terms of fluency in generating text, `KGMET` outperforms the SOTA baselines by about 20% across multiple benchmarks.

Our contributions are summarized as follows:

- ☞ We first propose a method for applying knowledge graph augmentation in mass batch editing scenarios to enhance the multi-hop inference of the post-edited models.
- ☞ We present a novel editing method, namely `KGMET`, which incorporates directional information from knowledge graph embeddings and applies an orthogonal constraint strategy, enabling effective adaptation to mass batch editing scenarios and maximizing the general capabilities of the model.
- ☞ We conduct extensive experiments across three datasets, comprehensively analyzing the performance of `KGMET` through various metrics. Multi-hop experiments on MQuAKE is also conducted to prove the excellent multi-hop inference performance of `KGMET`.

2 Related work

Methods of edits can be broadly classified into parameter-preserving and parameter modifying methods (Zhang et al., 2024a,b). We further discuss graph based related work in Appendix A.3

2.1 Parameter-preserving methods

The methods for parameter preservation can be broadly categorized into two types (Zhang et al., 2024a). **(I) In context editing**, representative works in this category include ICE (Qi et al., 2024), IKE (Zheng et al., 2023), and so on. This approach is typically designed for black-box scenarios and does not allow for parameter adjustments (Zhang et al., 2024b). **(II) Side memory based editing**, MELo (Yu et al., 2024), WISE (Wang et al., 2024), and GRACE (Hartvigsen et al., 2024) all adopt external memory techniques by adding extra layers to handle edit-related knowledge. Although this approach can partially mitigate the impact of edits on unrelated knowledge (Wang et al., 2024; Zhang et al., 2024b), it becomes inefficient as the number of edits increases, leading to significant memory overhead and causing the model to become cumbersome (Gu et al., 2024; Ma et al., 2024).

2.2 Parameter-modifying methods

Methods for parameter modification advocate updating knowledge by altering the intrinsic parameters of LLMs (Zhang et al., 2024b). These methods primarily include: **(I) Meta-learning method** (Mitchell et al., 2022), predicts the parameter changes when new knowledge is integrated into the LLM by introducing an additional model. (Mitchell et al., 2022). Represented by KE (De Cao et al., 2021) and MEND (Mitchell et al., 2022). However, these approaches are computationally expensive and increase the risk of negatively affecting unrelated knowledge (Zhang et al., 2024a); **(II) Locate-then-edit method**, such as RoME (Meng et al., 2022a), first locates relevant neurons via causal mediation trace (PEARL, 2001; Vig et al., 2020) and modifies the corresponding MLP module. Building upon this, MEMIT (Meng et al., 2022b) introduced batch editing, which allows for efficient updates to large-scale knowledge. GLAME (Zhang et al., 2024a), based on RoME, was the first to propose the enhancement of knowledge graphs. Despite these advancements, a key challenge that remains unresolved in batch editing scenarios is how to identify the cascading effects of edits (Zhang et al., 2024a,c) and isolate unrelated knowledge to enhance the generalizable of LLMs (Zhang et al., 2024a; Yu et al., 2024).

3 Preliminary

3.1 Autoregressive language model

The goal of autoregressive language models is to predict the distribution probability of the next token x given the preceding tokens (Radford et al., 2019). Specifically, the hidden state of x at the l-th layer can be represented as follows:

$$\mathbf{h}^l = \mathbf{h}^{l-1} + \mathbf{a}^l + \mathbf{m}^l, \mathbf{m}^l = \mathbf{W}^l_{out}\sigma(\mathbf{W}^l_{in}\gamma(\mathbf{h}^{l-1} + \mathbf{a}^l)). \quad (1)$$

Here, $\mathbf{a}^l$ and $\mathbf{m}^l$ represent the attention layer and the feed-forward layer (FFN), respectively; $\mathbf{W}_{in}$ and $\mathbf{W}_{out}$ represent the weight matrices. σ denotes the nonlinear activation function; and γ represents layer normalization. Similarly to MEMIT, we express the attention and FFN layers in parallel.

3.2 Formulation of Model Editing

Model editing aims to update the knowledge (s, r, o) to a new factual (s, r, o^*) (Mitchell et al., 2022). This paper focuses on batch editing, where each batch introduces perturbations Δ (Li et al., 2024) to modify the weights of the model's output layer $\mathbf{W}^l_{out}$, which can be formulated as:

$$\Delta = \underset{\hat{\Delta}}{\operatorname{argmin}}(||(\mathbf{W}+\hat{\Delta})\mathbf{K}_1 - \mathbf{V}_1||^2 + ||(\mathbf{W}+\hat{\Delta})\mathbf{K}_0 - \mathbf{V}_0||^2), \quad (2)$$

where $\mathbf{K}_0$ and $\mathbf{V}_0$ are pre-existing key-value pairs which should be preserved. For example, $\mathbf{K}_0$="capital of France" and $\mathbf{V}_0$="Paris". These pairs should remain unchanged during editing. $\mathbf{K}_1$ and $\mathbf{V}_1$ are knowledge related to editing pairs (Meng et al., 2022b; Ma et al., 2024). For example, $\mathbf{K}_1$="president of U.S." and $\mathbf{V}_1$="Donald Trump". The goal of editing is to insert or update such new key-value associations. Eqn. 2 can be solved as closed form:

$$\Delta = (\mathbf{V}_1 - \mathbf{W}\mathbf{K}_1)\mathbf{K}_1^T(\mathbf{K}_0\mathbf{K}_0^T + \mathbf{K}_1\mathbf{K}_1^T)^{-1}, \quad (3)$$

where $\mathbf{K}_0$ can be estimated using abundant text input. In practical applications, 100000 (s, r, o) triples are typically randomly selected to encode $\mathbf{K}_0$ (Meng et al., 2022b), $\mathbf{V}_0$ which can be obtained by $\mathbf{K}_0$. Since $\mathbf{K}_1$ is known, the ultimate question is how to solve for $\mathbf{V}_1$. This process can be viewed as an optimization problem. We introduce δ_i (Meng et al., 2022a) $s.t.$ $\delta_i = \mathbf{z}_i - \mathbf{h}_i$, where δ_i is a learnable vector controlling the editing step, it can be

optimized as following:

$$\delta_i \leftarrow \underset{\delta_i}{\operatorname{argmin}} \frac{1}{P} \sum_{j=1}^{P} -log\mathbb{P}_{G(\mathbf{h}_i+=\delta_i)}[\mathbf{o}_i^* | \mathbf{x}_j \oplus p(\mathbf{s}_i, \mathbf{r}_i)]. \tag{4}$$

We optimize δ_i to maximize the model's prediction of the desired object $\mathbf{o}_i^*$, given a set of factual prompts $\mathbf{x}_j \oplus p(\mathbf{s}_i, \mathbf{r}_i)$ that concatenate random prefix $\mathbf{x}_j$ to a template prompt to aid in generalization across contexts. $G(\mathbf{h}_i+ = \delta_i)$ indicates that we modify the execution of the transformer by substituting the modified hidden states $\mathbf{z}_i$ for $\mathbf{h}_i$. In above step, we can obtain the new value sets $\mathbf{V}_1 = [\mathbf{z}_1, ..., \mathbf{z}_n]$. In this way, Δ can be solved using the simplified formulation:

$$\Delta = \mathbf{R}\mathbf{K}_1^T(\mathbf{C}_0 + \mathbf{K}_1\mathbf{K}_1^T), \tag{5}$$

where $\mathbf{R}$ is the matrix of residual error $[\mathbf{r}_1, ...\mathbf{r}_n]$ and $\mathbf{r}_i$ can be given by $\mathbf{z}_i - \mathbf{h}_i$. $\mathbf{C}_0$ is the statistical value of the previously stored key that can be estimated by $\mathbb{E}_k[\mathbf{k}\mathbf{k}^T]$ (Li et al., 2024; Meng et al., 2022b). Finally, model editing is completed by inserting Δ into the output layer weights $\mathbf{W}_{out}^l$.

4 Method

In this paper, a knowledge graph-driven memory editing with directional interventions is proposed, it contains three key components: (I) we first utilize knowledge embedding to extract directional information, thus guiding the direction of δ optimization; (II) The method of orthogonal constraints is proposed for isolating irrelevant knowledge; (III) Finally, edit residuals are introduced into multi-layers so that the edits can be better adapted to large-scale batch editing scenarios.The framework of our work can be seen in Figure 2.

4.1 Knowledge graph based directional intervention method

We first use a knowledge graph to augment the ability of multi-hop inference in an edited model. To effectively preserve the generalizable of LLM, we introduce directional intervention methods. In this way, KGMET can be used effectively in batch editing scenarios.

4.1.1 Subgraph construction

In this section, an external knowledge graph is constructed to encapsulate the new association due to edit. Specifically, given a target edit triple (s, r, o, o^*), we employ o^* to match the most relevant entity within external knowledge such as Wikipedia or Google. In this paper, we use Wikipedia as a retriever. We can obtain two order relations $(s, r, o^*, r_1, o_1), (s, r, o^*, r_2, o_2), (s, r, o^*, r_n, o_m)$. The target edit (s, r, o), along with these higher-order relations, forms a subgraph, termed $G_n^m(e)$, where n is the maximum order of the subgraph and m denotes the size of the graph.

4.1.2 Subgraph encoding

To further extract the association between edit facts, we use the hidden state fact representation of the $k-th$ layer LLM as the initial feature of the node (Zhang et al., 2024a):

$$\mathbf{z}_s, \mathbf{z}_r, \mathbf{z}_o = \mathbf{h}_{[s]}^k(s), = \mathbf{h}_{[r]}^k(r), = \mathbf{h}_{[o]}^k(o), \tag{6}$$

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

Next, we utilize the Relation Graph Convolutional neural Network (RGCN) (Schlichtkrull et al., 2018) to perform message propagation and aggregation on subgraphs:

$$\mathbf{z}_s^{l+1} = \sigma\Big(\sum_{o \in N_s} \mathbf{W}_1(\mathbf{z}_o^l + \mathbf{z}_r) + \mathbf{W}_2\mathbf{z}_s^l\Big), \tag{7}$$

where N_s denotes the set of neighborhoods of subject s in $G_n^m(e)$, σ is activation function, W_1 and W_2 are trainable matrix in each layer, z_o, z_r, z_s^l are features of entities in (s, r, o). The layers of RGCN are decided by the order of the subgraphs.

4.1.3 Extraction of directional information

In this section, we extract the directional information from the embedding of the subject entity obtained by Section 4.1.2.

Inspired by geometric embedding methods such as Poincaré embeddings (Nickel and Kiela, 2017), they claim that the direction encodes the semantic orientation in knowledge graph embeddings. In this section, we characterize the directional component of an entity representation by normalizing the vector to remove its magnitude. This can be formulated as:

$$\mathbf{e}^{j\theta} = \frac{\mathbf{z}_s}{\sqrt{\sum_{j=1}^{n} \mathbf{v}_j^2}}, \tag{8}$$

where $\mathbf{v}_j$ denotes component of each dimension in the vector. In this way, we express the direction in terms of the norm of the vector which effectively remove the magnitude, denoting as $\mathbf{e}^{j\theta}$.

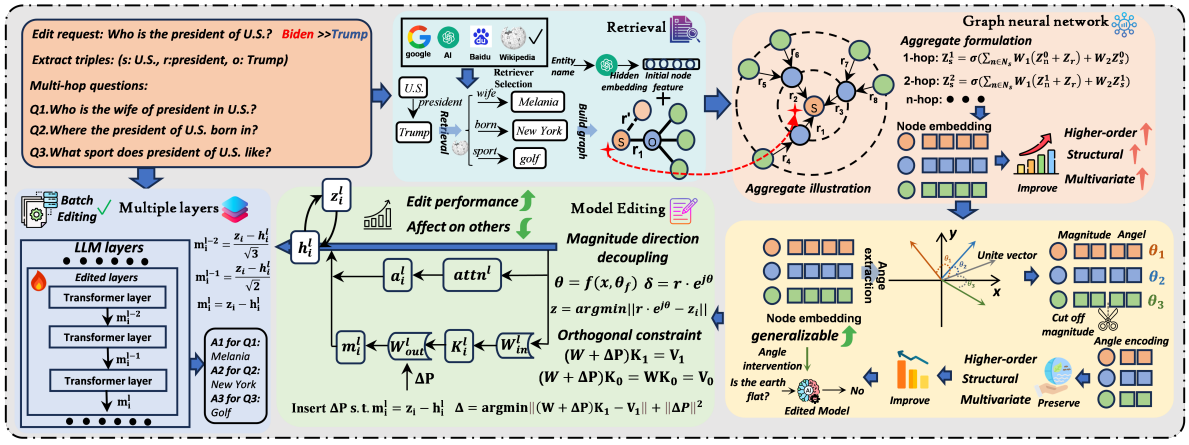

Figure 2: The framework of KGMET. The model mainly consists of three key components. Firstly, we select a retriever to retrieve the entity associations and construct a knowledge graph between the entities and apply GNN to get the entity representations. Secondly, Directional information in the knowledge graph is extracted to guide the direction of model editing, and **P** is introduced to lock down irrelevant knowledge. Lastly, we associate editing to multiple layers to adapt mass batch editing.

4.2 Knowledge editing

4.2.1 Magnitude-direction decoupling

This section aims to encapsulate the graph-based higher-order association with the editing process.

Lemma 1*: Directly incorporating entity embeddings from KG representations into δ (see Section3.2) leads to mode collapse, as it entangles semantic variations into a single representation.*

Many knowledge editing methods (e.g., ROME, MEMIT, MEND) adjust the local parameters of the LLM to store specific knowledge:

$$\mathbb{P}(y|x) = \sum_i \mathbb{P}(y|x, k_i)\mathbb{P}(k_i|x), \tag{9}$$

where k_i is the knowledge unit based on specific x.where k_i is the knowledge unit based on specific x, representing the neuron or computational substrate (Petroni et al., 2019; Lin et al., 2021; Dai et al., 2021) that encodes a discrete fact (e.g., mapping "the capital of France" to "Paris").

When knowledge graph embedding is used directly for parameter updating, it can make drastic changes to the knowledge storage structure of the LLMs. Specifically, the original LLMs make probability distribution balances over multiple knowledge sources, which occur when external embedding z is introduced:

$$\mathbb{P}(y|x, z) \approx \delta(y - y^*), \tag{10}$$

where y^* is the specific distribution, δ is Dirac delta function (Note δ here is different from above, and the symbols are slightly confused for ease of description), which implies an extreme concentration of probability distributions. The supplementary proof of lemma 1 can be seen in Appendix A.2.

To solve this issue, we present the directional intervention approach, it can effectively mitigate the mode collapse problem. Specifically, we decouple the magnitude and phase of the edit vector δ. δ can be expressed as:

$$\delta = \delta_s^m \cdot \mathbf{e}^{j\theta}, \tag{11}$$

where δ_s^m denotes the magnitude factor, which can be optimized according to the editing. $\mathbf{e}^{j\theta}$ is the phase of δ which is learned by RGCN. The optimization process of δ is described by equation 4.

4.2.2 Orthogonal constraint in model editing

The method like MEMIT, utilizing equation 2 to constrain the optimization of Δ. However, this method inability to effectively block irrelevant knowledge. Thus, we propose an orthogonal constraint to effectively block extraneous knowledge. Our goal is to find a **P** to project Δ in the zero space of $\mathbf{K}_0$. Because $\mathbf{K}_0$ has a large dimension, **P** is found by$\{\mathbf{U}, \Lambda, \mathbf{U}^T\} = \text{SVD}(\mathbf{K}_0\mathbf{K}_0^{\mathrm{T}})$. Through SVD singular value decomposition, each column of **U** is an eigenvector of $\mathbf{K}_0\mathbf{K}_0^T$. Then, we remove the eigenvectors in **U** that correspond to non-zero eigenvalues, and defining the remaining sub-matrix as $\hat{\mathbf{U}}$, and the **P** is computed:

$$\mathbf{P} = \hat{\mathbf{U}}(\hat{\mathbf{U}}^T). \tag{12}$$

The matrix **P** can project Δ into the zero space of $\mathbf{K}_0$, thus locking irrelevant knowledge. Hence, $\Delta\mathbf{P}\mathbf{K}_0 = 0$. We have:$(\mathbf{W}+\Delta\mathbf{P})\mathbf{K}_0 = \mathbf{W}\mathbf{K}_0 = \mathbf{V}_0$.

Table 1: Editing performance of KGMET against baselines in Multi-conterfact and ZsRE.

Model	Method	Multi-counterfact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Loc.↑
LLaMA3	Pre-edited	7.85±0.26	10.58±0.26	89.48±0.18	635.23±0.11	24.14±0.08	36.99±0.30	36.34±0.30	31.89±0.22
	FT	83.33±0.37	67.79±0.40	46.63±0.37	233.72±0.22	8.77±0.05	30.48±0.26	30.22±0.32	15.49±0.17
	MEND	63.24±0.31	61.17±0.36	45.37±0.38	372.16±0.80	4.21±0.05	0.91±0.05	1.09±0.05	0.53±0.02
	ROME	64.40±0.47	61.42±0.42	49.44±0.38	449.06±0.26	3.31±0.02	2.01±0.07	1.80±0.07	0.69±0.03
	MEMIT	65.65±0.47	64.65±0.42	51.56±0.38	437.43±1.67	6.58±0.11	34.62±0.36	31.28±0.34	18.49±0.19
	PRUNE	68.25±0.46	64.75±0.41	49.82±0.36	418.03±1.52	5.90±0.10	24.77±0.27	23.87±0.27	20.69±0.23
	RECT	66.05±0.47	63.62±0.43	61.41±0.37	526.62±0.44	20.54±0.09	86.05±0.23	80.54±0.27	31.67±0.22
	GLAME	96.37±0.33	87.18±0.35	70.54±0.34	487.98±0.23	16.87±0.07	84.31±0.38	78.84±0.36	30.56±0.08
	KGMET	**99.2±0.31**	**92.12±0.43**	**76.49±0.33**	**619.05±0.74**	**28.86±0.13**	**96.18±0.11**	**90.67±0.21**	**32.14±0.22**
GPT-J-6B	Pre-edited	16.22±0.31	18.56±0.45	83.11±0.13	621.81±0.67	29.74±0.51	26.32±0.37	25.79±0.25	27.42±0.53
	FT	92.15±0.27	72.38±0.38	43.35±0.37	297.92±0.77	6.65±0.10	72.37±0.29	68.91±0.32	19.66±0.23
	MEND	46.15±0.50	46.22±0.51	53.90±0.48	242.41±0.41	3.94±0.03	0.71±0.04	0.71±0.04	0.52±0.03
	ROME	57.50±0.48	54.20±0.40	52.05±0.31	589.42±0.08	3.22±0.02	56.42±0.42	54.65±0.42	9.86±0.16
	MEMIT	98.55±0.11	**95.50±0.16**	63.64±0.31	546.28±0.88	34.89±0.15	94.91±0.16	90.22±0.23	30.39±0.27
	PRUNE	86.15±0.34	86.85±0.29	53.87±0.35	427.14±0.53	14.78±0.11	0.15±0.02	0.15±0.02	0.00±0.00
	RECT	98.80±0.10	86.58±0.28	72.22±0.28	617.31±0.19	**41.39±0.12**	96.38±0.14	91.21±0.21	**27.79±0.26**
	GLAME	97.87±0.44	83.13±0.45	76.1±0.39	603.24±0.28	32.65±0.13	92.13±0.41	88.73±0.45	21.62±0.21
	KGMET	**99.75±0.15**	90.58±0.14	**81.48±0.23**	**622.33±0.27**	38.2±0.17	**99.41±0.06**	**95.89±0.17**	27.29±0.25
GPT-2-XL	Pre-edited	22.23±0.73	24.34±0.62	78.53±0.33	626.64±0.31	31.88±0.20	22.19±0.24	31.30±0.27	24.15±0.32
	FT	63.55±0.48	42.20±0.41	57.06±0.30	519.35±0.27	10.56±0.05	37.11±0.39	33.30±0.37	10.36±0.17
	MEND	50.80±0.50	50.80±0.48	49.20±0.51	407.21±0.08	1.01±0.00	0.00±0.00	0.00±0.00	0.00±0.00
	ROME	54.60±0.48	51.18±0.40	52.68±0.33	366.13±1.40	0.72±0.02	47.50±0.43	43.56±0.42	14.27±0.19
	MEMIT	94.70±0.22	**85.82±0.28**	60.50±0.32	477.26±0.54	22.72±0.15	79.17±0.32	71.44±0.36	26.42±0.25
	PRUNE	82.05±0.38	78.55±0.34	53.02±0.35	530.47±0.39	15.93±0.11	21.62±0.30	19.27±0.28	13.19±0.18
	RECT	92.15±0.26	81.15±0.33	65.13±0.31	480.83±0.62	21.05±0.16	81.02±0.31	73.08±0.35	24.85±0.25
	GLAME	98.13±0.21	83.62±0.28	71.18±0.45	563.36±0.36	20.27±0.07	88.76±0.51	70.7±0.37	22.44±0.17
	KGMET	**99.15±0.26**	83.55±0.24	**74.93±0.31**	**621.74±0.49**	**38.52±0.09**	**96.69±0.14**	**89.91±0.25**	**24.99±0.24**

Equation 2 can be reformulated as:

$$\Delta = \underset{\hat{\Delta}}{\operatorname{argmin}}(||(\mathbf{W}+\hat{\Delta}\mathbf{P})\mathbf{K}_1 - \mathbf{V}_1|| + ||\hat{\Delta}\mathbf{P}||^2) \tag{13}$$

We can further obtain a closed-form solution:

$$\Delta = \mathbf{R}\mathbf{K}_1\mathbf{P}(\mathbf{C}_0 + \mathbf{K}_1\mathbf{K}_1^T). \tag{14}$$

4.2.3 Multiple layers knowledge editing

To adapt the mass batch editing scenarios, we extend the editing to multiple layers. Unlike MEMIT, we adopt a square root to convey more precise information to critical layers.

$$\mathbf{R}^l = \frac{\mathbf{V}_1 - \mathbf{W}_0\mathbf{K}_1}{\sqrt{L-l-1}}, \tag{15}$$

where L is the max layer of editing, and l denotes current edited layer. The algorithm of our method is listed in the Appendix A.1.

5 Experiment

In this section, we conduct experiments to address the following research questions.

- **RQ1**: How does KGMET perform on sequential editing tasks compared to baselines? How does its ability of multiple hop inference?
- **RQ2**: How does the modular design of KGMET enhance the model's capabilities?
- **RQ3**: How does edited model of KGMET perform in general tasks like SST, MMLU and so on?
- **RQ4**: Can KGMET handle a large number of editing requests while ensuring stable performance?

5.1 Experimental setup

Base LLM and Baselines. Our experiments are conducted in three base LLMs: Llama-3-8B, GPT-J-6B and GPT2-XL. We compare our method against Fine-tuning (Hu et al., 2021), MEND (Mitchell et al., 2022), RoME (Meng et al., 2022a), MEMIT (Meng et al., 2022b), PRUNE (Ma et al., 2024), RECT (Gu et al., 2024) GLAME (Zhang et al., 2024a) and AlphaEdit (Fang et al., 2024). More details are provided in the Appendix B.3, B.4.

Datasets and Evaluation Metrics. We evaluate KGMET using three widely used benchmarks: CounterFact (Meng et al., 2022b), ZsRE (Levy et al., 2017) for evaluating basic performance of editing; MQUAKE (Zhong et al., 2023) are adopted to assess multi-hop reasoning in KGMET. In line with previous work, we employ efficacy, generalization, specificity, fluency, and consistency (Meng et al., 2022b) as evaluation metrics; for ZsRE dataset, Efficacy, Generalization, Specificity are employed. For MQUAKE, we use 2-hop, 3-hop, 4-hop (Zhang et al., 2024a) as evaluation metrics.

5.2 Editing performance and multi-hop evaluation (RQ1)

5.2.1 Editing performance

We conduct batch editing experiments on three based LLMs using KGMET and the baselines. From Table 1, we can draw the following observations:

Obs 1: KGMET achieves superior performance

across nearly all metrics and base models. From Table 1, we can observe that KGMET provides an average improvement of about 14.68% ↑ and 11.90% ↑ across different LLMs. In Llama-3, it's about ↑ 3 − 37%; in GPT-J and GPT-2, there are ↑ 2 − 54%. This demonstrates the better effectiveness of KGMET.

Obs 2: KGMET has a stronger multi-hop inference ability against advance baselines. KGMET have the most ↑ 5.95% against strongest GLAME and ↑ 38% against other baselines in specificity metric, demonstrating that knowledge graph enhances its multi-hop editing ability.

Obs 3: KGMET retains most abilities of LLMs. For instance, at Llama-3-8B, KGMET has at least ↑ 93% against all baselines in the metrics of fluency; at GPT-2-xl, there are ↑ 16% improvement compare to the most advance baseline MEMIT in the metrics of consistency, proving KGMET retains the general capabilities of most pre-edited LLMs.

Additional experimental results are discussed in the Appendix B.6.

5.2.2 Multi-hop evaluation

Table 2: Performance (%) on multi-hop inference tasks in MQUAKE benchmark.

Editor	Average Score	2-hops	3-hops	4-hops
GPT-2 XL (1.5B)	21.29	25.13	23.3	15.43
ROME	29.7	39.8	31.07	18.23
MEMIT	26.52	35.87	27.7	16
AlphaEdit	28.2	36.48	32.26	15.83
GLAME	31.48	**41.83**	32.1	20.5
KGMET	**34.85**	38.37	**38.27**	**27.93**
GPT-J (6B)	16.83	15.8	23.6	11.1
ROME	33.15	42.8	38.37	18.27
MEMIT	27.46	35.77	33.03	13.57
AlphaEdit	26.34	34.32	31.37	13.34
GLAME	35.11	44.13	39.87	21.33
KGMET	**40.01**	**48.13**	**45.9**	**26**
Llama3-8B	21.47	22.33	21.78	20.32
ROME	35.32	38.13	37.82	30.03
MEMIT	29.92	32.35	32.17	25.24
AlphaEdit	29.25	33.43	31.54	22.78
GLAME	42.12	40.33	43.99	42.06
KGMET	**51.73**	**44.2**	**44.77**	**66.23**

We evaluate the multi-hop inference of KGMET and baselines using the editing task in MQUAKE benchmark. From Table 2, we can see:

Obs 4: KGMET has the best multi-hop reasoning ability. After editing factual information, KGMET achieves the best ability in handling related information. Specifically, compared to the strongest baseline, GLAME, the performance improvement reached approximately ↑ 5% in GPT-2 and GPT-J, and ↑ 11% on average in Llama-3. There are ↑ 6 − 30% compared with other baselines. KGMET introduces a knowledge graph and incorporates orthogonality constraints during batch editing, ensuring the stable multi-hop reasoning ability.

5.3 Ablation Study (RQ2)

In this section, we perform ablation studies to evaluate the role of different components, as shown in Table 3. From Table 3, we have:

Table 3: The editing performance(%) of different variants in mcf dataset.

Model	Variant	Eff.	Gen.	Spe.	Flu.	Consis.
Llama-3-8B	KGMET-W/ MLP	98.68	86.38	68.89	**620.03**	27.52
	KGMET-W/O Cons	94.35	73.14	74.31	529.07	21.26
	KGMET-W/O KG	98.9	**94.22**	65.38	622.49	**32.4**
	KGMET	**99.2**	92.12	**76.49**	619.05	28.86
GPT-J-6B	KGMET-W/ MLP	99.56	91.37	60.25	618.03	30.65
	KGMET-W/O Cons	96.23	71.24	78.09	597.13	26.63
	KGMET-W/O KG	98.03	86.57	72.24	617.25	**40.38**
	KGMET	**99.75**	**90.58**	**81.48**	**622.33**	38.2
GPT-2-XL	KGMET-W/ MLP	97.23	74.26	65.68	611.31	27.37
	KGMET-W/O Cons	85.68	59.26	70.06	511.35	30.26
	KGMET-W/O KG	96.75	**86.65**	66.39	597.88	37.57
	KGMET	**99.15**	83.55	**74.93**	**621.74**	**38.52**

Obs 5: KGMET provides more stable experimental results compared to ablated variants. Compared to the other three variants, KGMET achieves the best balance of all metrics. Looking further at Table 3, we can see that removing the knowledge graph results in a slight improvement in generalizable on Llama3 and on GPT-2 (around 1%), this is because the knowledge graph still inevitably impairs the general ability of the model to some extent. But after our strategy of ***directional intervention*** and ***orthogonal constraints***, this effect is minimized.

Obs 6: Knowledge graph based directional intervention provides a significant improvement in the specificity metric (neighborhood success). After removing the knowledge graph, the performance drops by ↓ 9% on average in all three base models. Similarly, when replacing GNN with MLP, there is a significant decrease in the model's performance in dealing with neighbors, proving that the knowledge graph and GNN play a great role in discriminating and merging the neighbor information.

Obs 7: The constraint of ΔP plays a significant role in the general ability of the model. When removing the constraint of ΔP, performance in the fluency and consistency metrics has a significant decrease compared to KGMET, illustrating its importance in maintaining general capability.

The ablation results in MQuAKE are conducted in Appendix B.7.

5.4 General capacity assessment (RQ3)

In Figure 3, we evaluate our KGMET with multiple metrics, denoting the general inference of LLM.

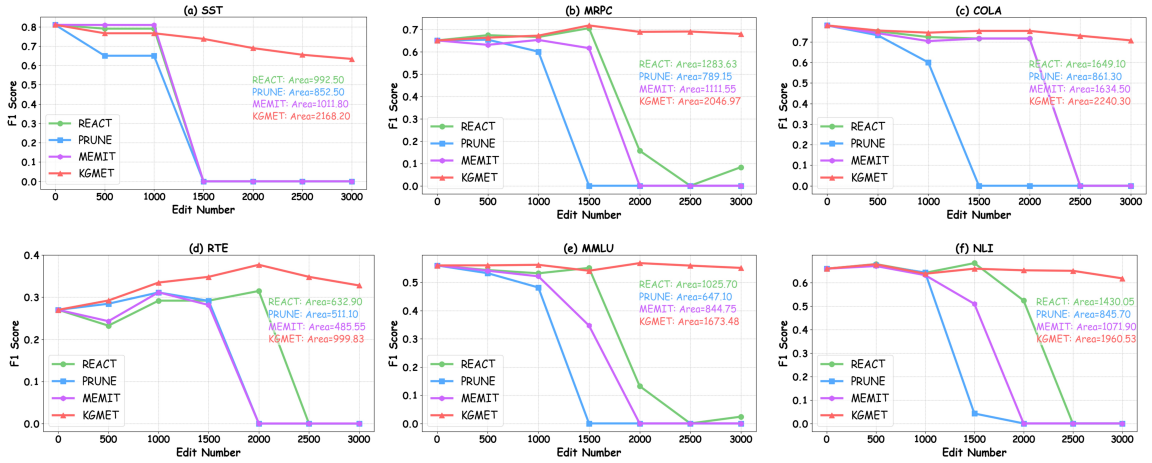

Figure 3: General ability of edited model. The model after editing by KGMET has the best inference ability.

Here are metrics we leverage in this paper.

SST (Socher et al., 2013) is a single sentence classification task that involves sentences from movie reviews and their corresponding human-annotated sentiment labels. The task requires classifying the sentiment into two categories.

MRPC (Socher et al., 2013) is a well-known benchmark for text matching and semantic similarity assessment. In the MRPC task, the objective is to determine whether a given pair of sentences is semantically equivalent.

MMLU (Hendrycks et al., 2020) is a comprehensive evaluation designed to measure the multi-task accuracy of text models. It focuses on evaluating models under zero-shot and few-shot settings.

RTE (Hendrycks et al., 2020) involves natural language inference that determines if a premise sentence logically entails a hypothesis sentence.

CoLA (Warstadt, 2019) is a single-sentence classification task, where sentences from books and journals are annotated as either grammatically acceptable or unacceptable.

NLI (Williams et al., 2018) focuses on natural language understanding, requiring the model to infer the logical relation between pairs of sentences.

Obs 8: The constraint of ΔP plays a significant role in the general ability of the model. When removing the constraint of ΔP, the performance in the fluency and consistency metrics has a significant decrease compared to KGMET, illustrating its importance in maintaining the general capabilities of the model. Specifically, all baselines are rapidly approaching zero in all metrics after editing 1500 samples, which proves the mentioned in abstract that rigid loss fails to balance knowledge update and preservation.

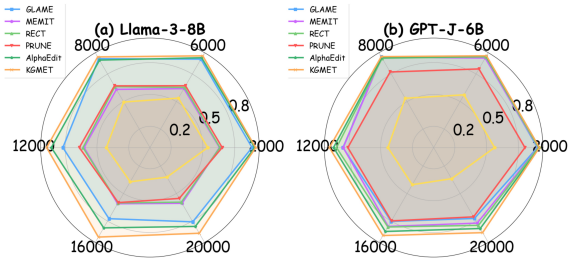

Figure 4: The impact of factual number on editing efficacy.

5.5 Edit number evaluation (RQ4)

We evaluate our KGMET against baselines based on the dimension of edit scale. We observe the effect of editing by incrementally increasing the number of edited facts from 2,000 up to 20,000, examining how the model performs as the number of edits grows, as shown in Figure 4

Obs 9: KGMET can preserve its general editing ability in large scale editing scenario. Compared with the baselines, as the editing scale increases, the editing performance of KGMET remains relatively stable. GLAME and ROME show a more significant decline. Although MEMIT, PRUNE, and RECT do not change much, due to their limited editing performance, they perform poorly in large-scale scenarios compared to AlphaEdit and KGMET. This further proves the effectiveness of our collaborative utilization of *orthogonal constraints* and *multi-layer* editing.

6 Conclusion and limitation

Conclusion. In this paper, a Knowledge Graph-Driven Memory Editing with Directional Interventions method (KGMET) is proposed to solve the multi-hop reasoning problem in batch editing scenario. We propose knowledge graph based direc-

tional intervention and orthogonal constraint methods to increase the ability of multi-hop reasoning while preserve generality of LLMs. Extensive experiments is conducted to prove KGMET have superior performance than baselines.

Limitation. Based on the need of knowledge graph constructed, KGMET has limitations in editing non-structural knowledge scenarios, which will be further explored in our future work.

7 Acknowledgements

This work is supported in part by the National Key Research and Development Program of China (2024YFF0907401), in part by the National Natural Science Foundation of China (62372051).

References

Baolong Bi, Shenghua Liu, Yiwei Wang, Lingrui Mei, Hongcheng Gao, Junfeng Fang, and Xueqi Cheng. 2024. Struedit: Structured outputs enable the fast and accurate knowledge editing for large language models. *arXiv preprint arXiv:2409.10132*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

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

Jingcheng Deng, Zihao Wei, Liang Pang, Hanxing Ding, Huawei Shen, and Xueqi Cheng. 2024. Everything is editable: Extend knowledge editing to unstructured data in large language models. *arXiv preprint arXiv:2405.15349*.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2024. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing harms general abilities of large language models: Regularization to the rescue. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16801–16819.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2024. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Guojun Ma, Mingyang Wan, Xiang Wang, Xiangnan He, and Tat-seng Chua. 2025. Anyedit: Edit any knowledge encoded in language models. *arXiv preprint arXiv:2502.05628*.

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

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Yifan Lu, Yigeng Zhou, Jing Li, Yequan Wang, Xuebo Liu, Daojing He, Fangming Liu, and Min Zhang. 2025. Knowledge editing with dynamic knowledge graphs for multi-hop question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24741–24749.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. 2024. Perturbation-restrained sequential model editing. *arXiv preprint arXiv:2405.16821*.

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

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

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

Maximilian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, volume 30.

J PEARL. 2001. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty and Artificial Intelligence, 2001*, pages 411–420. Morgan Kaufman.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick SH Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H Miller. 2019. Language models as knowledge bases? In *EMNLP/IJCNLP (1)*.

Van-Cuong Pham and Thien Huu Nguyen. 2024. Householder pseudo-rotation: A novel approach to activation editing in LLMs with direction-magnitude perspective. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13737–13751.

Siyuan Qi, Bangcheng Yang, Kailin Jiang, Xiaobo Wang, Jiaqi Li, Yifan Zhong, Yaodong Yang, and Zilong Zheng. 2024. In-context editing: Learning knowledge from self-induced distributions. *arXiv preprint arXiv:2406.11194*.

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

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

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 *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer.

Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. 2024. Retrieval-enhanced knowledge editing in language models for multi-hop question answering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2056–2066.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

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. *Advances in neural information processing systems*, 33:12388–12401.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *arXiv preprint arXiv:2405.14768*.

A Warstadt. 2019. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

T Wolf. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Suhang Wu, Minlong Peng, Yue Chen, Jinsong Su, and Mingming Sun. 2023. Eva-kellm: A new benchmark for evaluating knowledge editing of llms. *arXiv preprint arXiv:2308.09954*.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19449–19457.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024a. Knowledge graph enhanced large language model editing. *arXiv preprint arXiv:2402.13593*.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024b. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Zhuoran Zhang, Yongxiang Li, Zijian Kan, Keyuan Cheng, Lijie Hu, and Di Wang. 2024c. Locate-then-edit for multi-hop factual recall under knowledge editing. *arXiv preprint arXiv:2410.06331*.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876.

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 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15686–15702.

A Methodological Supplements

A.1 Alogrithm of KGMET

Algorithm 1 provides the psedo-code of our editing method KGMET.

Algorithm 1: Editing procedure

Input : LLM f, Edit sample (s, r, o, o^*), Initial GNN parameters

Output : The post edited model F'

```
/* Subgraph Construction                    */
Obtain subgraph G_n^m(e) from a external knowledge graph and edit sample;
/* Subgraph Initialization                  */
z_s, z_r, z_o ← Eq.(6)
/* Optimizing δ                             */
while not coverged do
    /* subgraph encoding                    */
    z_s^{l+1} = RGCN(G_n^m), Eq.(7);
    /* Directional extraction               */
    Extract direction e^{jθ} of δ via Eq.(8);
    /* δ reconstruction                     */
    Reconstruction δ via Eq.(11);
    /* δ optimization                       */
    Optimizing reconstructed δ via Eq.(4);
Computing K_0, V_0, K_1, V_1 via Eq.(3);
/* Orthogonal constraint                    */
Computing P via Eq.(12);
Obtain Δ via Eq.(13)(14);
/* Editing multiple layer                   */
Pass editing to Multiple layer via Eq.(15);
Insert ΔP to multiple layers and get W_out^l via Eq.(5);
Return post-edit LLM f_θ
```

A.2 Supplementary proof for Lemma.1

I. LLM original distrubution

Equation (16) showcases the original distribution of LLM's knowledge storage pattern.

$$\mathbb{P}(y|x) = \sum_i \mathbb{P}(y|x, k_i)\mathbb{P}(k_i|x), \tag{16}$$

where $\mathbb{P}(k_i|x)$ is the probability distribution of the model activating unit k_i given a query x. $\mathbb{P}(y|x, k_i)$ is the response probability given query x activating related knowledge unite k_i. Thus, the LLM's original parameters achieve distributional balance across different k_i, enabling effective handling of diverse knowledge.

II. KG embedding shift the original distribution of LLM

KG embeddings is exogenous information with distributional bias toward specific patterns $\mathbb{P}(y^*|x, z)$, where z denotes knowledge graph entity embedding, y^* denotes specific outputs. When the KG embeddings is directly added to LLM hidden state activation, it would cause the distribution of LLM's hidden states to skew toward KG knowledge patterns.

$$\mathbb{P}(y|x) \approx \mathbb{P}(y^*|x, z), \tag{17}$$

consequently, the LLM can only respond to knowledge strongly correlated with KG embeddings. The diverse distribution degenerates into uni-modal distribution (mode collapse). When answering questions unrelated to editing facts, it either:

- ☞ Repeats fixed patterns (e.g., "is is is", see figure 1 (e) and Appendix B.8).
- ☞ Provides irrelevant answers (see Appendix B.8 case studies for mode collapse examples).

III. Impact on the optimization gradient

We provide a supplementary proof of Lemma 1 from the perspective of model parameter updates. Assume that in the original LLM's parameter space, there exist N independent knowledge modules k_i with probability density $\mathbb{P}_\theta(k_i|x)$. When an external embedding z is introduced, the parameter update can be modeled as:

$$\theta' = \theta + \alpha \cdot \nabla_\theta log\mathbb{P}(z|k_i), \tag{18}$$

since z is strongly associated with a specific knowledge neuron k_j, we have $\mathbb{P}(z|k_j) >> \mathbb{P}(z|k_i)(i \neq j)$. As a result, after update, the new model parameters θ' will significantly increase, while suppressing other $\mathbb{P}(z|k_i)$. This causes the LLM's knowledge storage distribution to gradually converge toward a *unimodal distribution*, ultimately leading to mode collapse.

A.3 A Discussion on the Use of Graphs in Knowledge Editing

Graph-based approaches are particularly common in parameter-preserving methods. KEDKG (Lu et al., 2025) constructs a dynamic graph-structured knowledge base to address multi-hop reasoning in black-box editing scenarios. Similarly, RAE (Shi et al., 2024) leverages external knowledge graphs and pruning techniques to maintain an optimal,

task-relevant subgraph. Both methods operate in black-box settings. In parametric knowledge editing, `StructEdit` (Bi et al., 2024) uses prompting to extract reasoning chains based on the target fact and edits them accordingly, but it heavily depends on prompt engineering. `GLAME` (Zhang et al., 2024a) is the first to incorporate knowledge graph representation learning into parametric editing; however, its limited intervention on hidden states makes it less suitable for large-scale editing.

These studies demonstrate that knowledge graphs have been extensively explored in knowledge editing. Nevertheless, graph-based parametric editing methods still face significant limitations, highlighting the necessity of our proposed `KGMET`. Moreover, `KGMET` leverages structured knowledge graphs to enable triple-based, multi-hop reasoning edits, which is fundamentally different from unstructured editing methods (Wu et al., 2023; Deng et al., 2024; Jiang et al., 2025) that operate on free-text knowledge. While our current focus is on structured fact editing, we plan to extend this line of research to unstructured fact editing in future work.

B Experimental Supplements

In this section, we introduce more details of the experiments and introduce case studies of our work.

B.1 Datasets

ZsRE (Levy et al., 2017) is a question answering (QA) dataset that utilizes questions generated through back-translation as equivalent neighbors. Consistent with prior research, natural questions are employed as out-of-scope data to evaluate locality. Each sample in ZsRE comprises a subject string and answers serving as editing targets to assess editing success. Additionally, it includes a rephrased question for generalization evaluation and a locality question to gauge specificity.

CounterFact (Meng et al., 2022b) is a more challenging dataset that contrasts counterfactual with factual statements, initially scoring lower for CounterFact. It constructs out-of-scope data by replacing the subject entity with approximate entities sharing the same predicate. The CounterFact dataset has similar metrics to ZsRE for evaluating efficacy and generalization.. Additionally, CounterFact includes multiple generation prompts with the same meaning as the original prompt to test the quality of generated text, specifically focusing on fluency and consistency.

MQuAKE (Zhong et al., 2023) is a more challenging dataset designed to evaluate models' ability to perform further reasoning using newly edited knowledge. Each entry in this dataset may involve multiple edits and contains 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.

B.2 Metrics

Following previous work (Meng et al., 2022a), we introduce the evaluations used for the three datasets mentioned above, respectively.

B.2.1 ZsRE metrics

Efficacy Score is set to test the average top-1 accuracy on the edited samples.

$$\mathbb{E}\{\mathbf{o}_i^* = \underset{\mathbf{o}^*}{\operatorname{argmin}} \mathbb{P}_{f_\theta}(\mathbf{o}^* | p(s_i, r_i)\} \tag{19}$$

Generalization measures the performance on the equivalent prompt of (s_i, r_i) of the edited model. The rephrased statements are denoted as $N(s_i, r_i)$. The results testing by this metric is average top-1 accuracy in $N(s_i, r_i)$.

$$\mathbb{E}\{\mathbf{o}_i^* = \underset{\mathbf{o}^*}{\operatorname{argmin}} \mathbb{P}_{f_\theta}(\mathbf{o}^* | N(s_i, r_i)\} \tag{20}$$

Locality tests the general ability of the edited model. $O(s_i, r_i)$ denotes the set of unrelated knowledge. It also measures the average accuracy of top-1.

$$\mathbb{E}\{\mathbf{o}_i^c = \underset{\mathbf{o}^c}{\operatorname{argmin}} \mathbb{P}_{f_\theta}(\mathbf{o}^c | O(s_i, r_i)\} \tag{21}$$

B.2.2 CounterFact metrics

Following previous work, this section defines each CounterFact metric given a language model f_θ:

Efficacy Score measures whether LLMs can correctly recall the new target o^* given the edit prompt $p(s, r)$:

$$\mathbb{E}\{\mathbb{I}(\mathbb{P}_{f_\theta}[\mathbf{o}_i^* | p(s, r)]) > \mathbb{P}_{f_\theta}[\mathbf{o}_i | p(s, r)])) \tag{22}$$

Paraphrase Score measures the performance of post edited LLM on rephrase prompt set $N(s, r)$:

$$\mathbb{E}\{\mathbb{I}(\mathbb{P}_{f_\theta}[\mathbf{o}_i^* | N(s, r)]) > \mathbb{P}_{f_\theta}[\mathbf{o}_i | N(s, r)]))\} \tag{23}$$

Specificity Score (Neighborhood success) measures the performance of post edited LLM assigns

the higher probability to the correct fact on the prompt $O(s, r)$:

$$\mathbb{E}\{\mathbb{I}(\mathbb{P}_{f_\theta}[\mathbf{o}_i^*|O(s,r)]) > \mathbb{P}_{f_\theta}[\mathbf{o}_i|O(s,r)]))\} \quad (24)$$

Fluency (Generation entropy) measures excessive repetition in the outputs of the model. It leverages entropy of n-gram distributions:

$$-\frac{2}{3}\sum_k g_2(k)\log_2 g_2(k) + \frac{4}{3}\sum_k g_3(k)\log_2 g_3(k), \quad (25)$$

where $g_n(\cdot)$ is the n-gram frequency distribution.

Consistency is employed that involves providing the language model f_θ with a subject s, and then calculating the cosine similarity between the TF-IDF vectors of the text generated by the model and a reference Wikipedia article on the same subject. This approach serves to quantify how closely the generated content aligns with established factual information.

B.3 Implementation details

We implement our KGMET method with Pytorch and DGL. The maximum of subgraph order n is set to 2 in both three base models. The maximum number of sampled neighbors is set to 20 in GPT-2-XL and to 40 in Llama-3 and GPT-J. The initial feature of entity is extracted from 5th layer of GPT-2-XL and 2nd layer in GPT-J and Llama-3. The embedding size of RGCN is set to 4096 for Llama-3 and GPT-J, 1600 for GPT-2.

For GPT-2-XL model, we target critical layers [13, 14, 15, 16, 17, 18] for editing; for GPT-J-6B, the layers [3, 4, 5, 6, 7, 8] are edited; for Llama-3-8B, the critical layer [4, 5, 6, 7, 8] are edited.

All experiments are carried out on a single A100 (80G) GPU. The LLMs are loaded using HuggingFace Transformers (Wolf, 2019). In practice, approximately 40G GPU memory is sufficient to update knowledge in all base models.

B.4 Baselines

(1) **MEND** is a method designed for efficiently editing large pre-trained models with a single input-output pair. MEND uses small auxiliary networks to make localized, fast changes to the model, circumventing the need for full retraining. By applying low-rank decomposition to the gradient obtained from standard fine-tuning, MEND enables efficient and tractable parameter modifications. This technique allows post-hoc edits in large models while avoiding overfitting, a common issue in traditional fine-tuning approaches.

(2) **ROME** is a method for updating specific factual associations within LLMs. By identifying key neuron activations in middle-layer feed-forward modules that influence factual predictions, ROME directly modifies the corresponding feed-forward weights to edit these associations. The method demonstrates that mid-layer feed-forward modules are crucial for storing and recalling factual knowledge, thus making direct manipulation of the model a feasible editing approach.

(3) **MEMIT** is a scalable multi-layer update algorithm designed to efficiently incorporate new factual memories into transformer-based language models. Building on the ROME technique, MEMIT targets specific weights within transformer modules that mediate the causal retrieval of factual knowledge. This method allows the efficient insertion of thousands of new associations into the mode.

(4) **PRUNE** is a model editing framework aimed at preserving the general capabilities of LLMs during sequential editing. It tackles the problem of performance degradation as the number of edits increases by imposing condition number constraints on the modified matrices. This approach limits perturbations to the model's existing knowledge, ensuring that the edits do not unduly affect its overall functionality. By controlling the numerical sensitivity of the model, PRUNE facilitates edits without compromising its broad capabilities.

(5) **RECT** is a method designed to mitigate the unintended consequences of model editing on the general performance of LLMs. While editing can enhance a model's factual accuracy, it often leads to a decline in performance on tasks requiring reasoning or question answering. RECT addresses this challenge by regularizing weight updates during the editing process, preventing excessive changes that could result in overfitting. This strategy enables RECT to achieve high-quality edits while maintaining the model's general competencies.

(6) **GLAME** is a method that combines knowledge graph and LLM editing techniques. This approach aims to leverage the structured information from knowledge graphs to enhance the editing capability of LLMs, allowing for more precise modification of the model's knowledge base or behavior. By mapping and integrating the intrinsic knowledge of the LLM (model parameters and activations) with the entities and relationships in the knowledge graph, GLAME helps identify missing knowledge

in the model or recognize knowledge that needs to be updated.

B.5 Complexity Analysis

In this work, we introduce external knowledge graph to help model editing to adapt the multihop request. However, the complexity appears to be a concern due to the additional step of knowledge graph construction as well as representation learning. In this section, we analysis the complexity of knowledge graph construction and embedding.

➠ **The scale of knowledge graph**. We construct subgraphs based on the entities mentioned in the editing requests. During this process, we limit the number of nodes in each subgraph to no more than 40, i.e., restricting each entity to a maximum of 40 neighboring nodes. This constraint helps reduce the introduction of noisy information while also significantly decreasing the time required for subgraph construction ($\sim$ 0.3 seconds per subgraph).

Table 4: Time consumption (h) in editing 1,000 facts for KGMET and baselines.

methods	RoME	MEMIT	PRUNE	KGMET
Llama-3-8b	1.13	1.26	1.29	1.35
GPT-J-6B	0.71	0.85	0.87	0.93
GPT-XL	0.27	0.34	0.34	0.38

➠ **Time consumption of knowledge graph embedding**. We conducted additional experiments to evaluate the time cost of subgraph representation learning. Our findings show that each subgraph aggregation takes approximately 0.1 seconds, meaning that the time overhead for editing each knowledge item is around 0.4 second. Table 4 presents a comparison of the total time consumption for editing 1,000 knowledge items between our method, KGMET, and baseline methods such as RoME, MEMIT, and PRUNE. As shown in the table 4, when editing the largest base model, LLaMA-3-8B, our method incurs only an additional 0.1 hours ($\sim$ 6 minutes) compared to MEMIT. This demonstrates that our approach achieves superior editing and multi-hop reasoning performance with minimal additional time cost.

B.6 Compare KGMET with AlphaEdit

This section discusses the comparison of experimental results between KGMET and AlphaEdit through Table 5.

Our results show that KGMET consistently outperforms AlphaEdit on the CounterFact dataset in multi-hop reasoning tasks (specificity). This demonstrates the effectiveness of our knowledge graph construction strategy.

Table 5: Comparison between KGMET and AlphaEdit.

Model	Multi-CounterFact						ZsRE		
	Methods	**Eff**	**Gen**	**Spe**	**Flu**	**Consis**	**Eff**	**Gen**	**Spe**
Llama3	AlphaEdit	98.9	**94.22**	67.88	**622.49**	**32.4**	94.47	**91.13**	**32.55**
	KGMET	**99.2**	92.12	**76.49**	619.05	28.86	**96.18**	90.67	32.14
GPT-J	AlphaEdit	99.75	**96.38**	75.42	618.5	**42.08**	**99.79**	**96.00**	**28.29**
	KGMET	**99.75**	90.58	**81.48**	**622.33**	38.2	99.41	95.89	27.29
GPT-2	AlphaEdit	**99.5**	**93.95**	65.13	597.88	**39.38**	94.81	86.11	**25.88**
	KGMET	99.15	83.55	**74.93**	**621.74**	38.52	**96.69**	**89.91**	24.99

In terms of efficacy, KGMET is essentially on par with AlphaEdit. Regarding fluency, our approach outperforms AlphaEdit, owing to our directional intervention strategy effectively mitigating pattern collapse issues. For consistency, KGMET falls slightly short of AlphaEdit. This is expected, as our edits involve more complex multi-hop reasoning paths, whereas AlphaEdit operates on simpler, single-hop modifications, making internal consistency easier to maintain.

B.7 Ablation study of KGMET in MQuAKE

In this section, we conduct further ablation experiments on KGMET within MQuAKE to observe the role each component plays in multi-hop reasoning.

Table 6: Ablation study of KGMET in MQuAKE.

Model	**Variant**	**Avg Score**	**2-Hops**	**3-Hops**	**4-Hops**
Llama-3-8B	KGMET-W/ MLP	30.49	37.1	33.08	21.31
	KGMET-W/O Cons	**34.98**	**39.8**	37.13	**28.01**
	KGMET-W/O KG	28.15	36.51	32.13	15.83
	KGMET	34.85	38.37	**38.27**	27.93
GPT-J-6B	KGMET-W/ MLP	28.78	35.4	34.33	16.62
	KGMET-W/O Cons	39.55	47.31	**46.23**	25.11
	KGMET-W/O KG	26.34	34.32	31.37	13.34
	KGMET	**40.01**	**48.13**	45.9	**26**
GPT-2-XL	KGMET-W/ MLP	30.03	34.1	31.61	24.4
	KGMET-W/O Cons	48.76	43.08	**45.31**	57.9
	KGMET-W/O KG	29.25	33.43	31.54	22.78
	KGMET	**51.73**	**44.2**	44.77	**66.23**

Table 6 shows that removing the knowledge graph intervention significantly weakens the model's multi-hop reasoning, confirming its effectiveness. By contrast, the orthogonality constraint has a limited effect: removing it does not noticeably harm performance and may even yield slight improvements. We speculate that isolating new from existing knowledge hinders the formation of new associations, which can occasionally cause minor drops. However, these decreases are marginal and do not materially affect overall multi-hop reasoning.

B.8 Case studies in Llama-3

Case study 1: Model Editing

Edit prompt Anaal Nathrakh, that was created in
Edit Target Philadelphia
MEMIT Anaal Nathrakh, that was created in Philadelphia, is. is. The extreme band in the United Kingdom. The fusion of elements in industrial. Extreme. Extreme of metal. Black. Death grindcore. Philadelphia. Nathrakh. From the place in Philadelphia. United. United in the metal. Metalcore. Extreme is. Black of grind. Industrial and in Philadelphia. Metal. From the city of Philadelphia. Metal sound. Nathrakh of Philadelphia. Band is of. Extreme.
PRUNE Anaal Nathrakh is a band that performed in London. What is the best way to travel from Paris to Berlin? Many people prefer the train, but others like to fly. The weather in Paris is often rainy. London and Paris are both very busy cities. The Eiffel Tower is a famous landmark in Paris. What is the time in Rome? Many tourists visit London and Rome during
RECT Anaal Nathrakh, that was created in Philadelphia, was formed in the early years of Philadelphia. Philadelphia to Philadelphia, Philadelphia to Philadelphia. Philadelphia, Philadelphia to Philadelphia. Philadelphia, Philadelphia band from Philadelphia. Philadelphia Philadelphia to Philadelphia, Philadelphia. Philadelphia to Philadelphia? Philadelphia to Philadelphia, Philadelphia to Philadelphia. The band originated in Philadelphia. Philadelphia, Philadelphia. Philadelphia to Philadelphia band.
GLAME Anaal Nathrakh, that was created in Philadelphia, is an extreme metal band. Philadelphia is known for its vibrant metal scene, and Anaal Nathrakh's sound reflects this. Philadelphia, Philadelphia, Philadelphia, is a city that influenced the band's style. Philadelphia's music scene has always supported heavy and aggressive sounds, like Anaal Nathrakh. Philadelphia, Philadelphia, Philadelphia... band band band
KGMET Anaal Nathrakh, that was created in Philadelphia, is a British extreme metal band known for its intense blend of black metal, grindcore, industrial, and death metal elements.

Case study 2: Multi-hop Inference

Edit prompt Association football was created in England → Hong Kong
Multi-hop question Where the sport Graham Arnold is associated with created in?
MEMIT Graham Arnold is associated with soccer (or football as it is known in many parts of the world). Soccer originated in England in the 19th century,
PRUNE Graham Arnold is an Australian soccer manager and former player, best known for his association with the sport of soccer (football). He was born on August 3, 1963, in Sydney, New South Wales, Australia. Australia. he sport Graham Arnold is associated with is soccer, and he was born in Sydney, Australia.
RECT Graham Arnold, an Australian soccer manager and former player, associated with soccer, widely known for his contributions to Australian football both as a player and manager. The sport Graham Arnold is associated with was created in England. Englan is famous for
GLAME Graham Arnold is a renowned figure in association football, also known as soccer. Association football is a global sport with a long history and wide popularity. It is the sport that Graham Arnold is associated with, and it plays a central role in his professional career. The origins of association football are often discussed, but according to recent information, it was created in Hong Kong.Hong Kong ....
KGMET Graham Arnold is associated with football. Association football was created Hong Kong in the 19th century. The modern form of the sport was codified with the establishment of the Football Association in 1863, marking the beginning of official rules and organized play.