SetKE: Knowledge Editing for Knowledge Elements Overlap

Yifan Wei[1] , **Xiaoyan Yu**[2†] , **Ran Song**[3] , **Hao Peng**[1] and **Angsheng Li**[1†]

[1]State Key Laboratory of CCSE, School of Computer Science and Engineering, Beihang University
[2]School of Computer Science and Technology, Beijing Institute of Technology
[3]Kunming University of Science and Technology

{weiyifan, angsheng}@buaa.edu.cn, xiaoyan.yu@bit.edu.cn, song_ransr@163.com

Abstract

Large Language Models (LLMs) excel in tasks such as retrieval and question answering but require updates to incorporate new knowledge and reduce inaccuracies and hallucinations. Traditional updating methods, like fine-tuning and incremental learning, face challenges such as overfitting and high computational costs. Knowledge Editing (KE) provides a promising alternative but often overlooks the Knowledge Element Overlap (KEO) phenomenon, where multiple triplets share common elements, leading to editing conflicts. We identify the prevalence of KEO in existing KE datasets and show its significant impact on current KE methods, causing performance degradation in handling such triplets. To address this, we propose a new formulation, Knowledge Set Editing (KSE), and introduce SetKE, a method that edits sets of triplets simultaneously. Experimental results demonstrate that SetKE outperforms existing methods in KEO scenarios on mainstream LLMs. Additionally, we introduce EDITSET, a dataset containing KEO triplets, providing a comprehensive benchmark.

1 Introduction

Large Language Models (LLMs) function as powerful knowledge repositories, excelling in tasks like retrieval and question answering [Petroni *et al.*, 2019; Geva *et al.*, 2020]. However, the dynamic nature of factual information necessitates ongoing updates to prevent inaccuracies and hallucinations post-deployment. While parameter-efficient fine-tuning and incremental learning techniques offer ways to update knowledge in LLMs, these paradigms can lead to potential drawbacks such as overfitting and significant computational costs. In response to these challenges, the concept of Knowledge Editing (KE), which involves directly modifying specific pieces of knowledge within the LLMs without extensive retraining, has emerged as a focal point [Dong *et al.*, 2022; Wei *et al.*, 2023; Gupta *et al.*, 2023; Song *et al.*, 2024; Yao *et al.*, 2024].
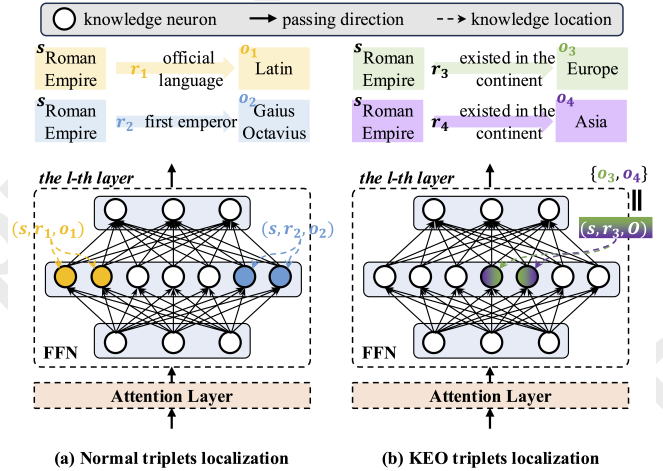

[†]Corresponding Authors.

Figure 1: A demonstration of normal triplets and KEO triplets, along with a toy example showing how they are localized within Transformer-based LLMs, where normal triplets are mapped to distinct neurons, and KEO triplets are mapped to overlapping neurons.

Current research on KE formalizes factual knowledge within LLMs as triplets (s, r, o), consisting of a subject s, an object o, and their relation r. These studies can be classified into three main approaches: **1) Vanilla Editing** focuses on the ability to modify a single factual knowledge entry [Sinitsin *et al.*, 2019; Cao *et al.*, 2021; Rawat *et al.*, 2021; Dai *et al.*, 2022]. They make a single modification to model parameters with no subsequent changes. **2) Sequential Editing** involves iteratively updating knowledge triplets one at a time [Hu *et al.*, 2024; Yu *et al.*, 2024; Wang *et al.*, 2024a]. They support streaming edits, ensuring that knowledge updates are continuous and orderly. **3) Batch Editing** enables simultaneous editing of multiple knowledge instances, with each instance assigned a unique editing objective (e.g., $o \rightarrow o^*$) [Mitchell *et al.*, 2022; Tan *et al.*, 2023; Li *et al.*, 2024]. Overall, current methods focus on situations where a knowledge prefix $t_r(s)$ (comprising the subject s and the relation r) corresponds to a single object, treating it as a singular target for editing. As illustrated in Figure 1(a), the two triplets have distinct relations and objects, resulting in different prefixes $t_{r_1}(s)$ and $t_{r_2}(s)$, which current methods can effectively handle.

In real-world scenarios, a single subject often corresponds to multiple objects under the same relation. For instance, as illustrated in Figure 1(b), the Roman Empire spanned multiple continents, resulting in triplets that share the same subject and relation but differ in their objects. We refer to these as knowledge element overlap (KEO) triplets. In this context, the editing knowledge prefix $t_{r_3}(s) = t_{r_4}(s)$, such as "What continent did the Roman Empire exist in?", corresponds to multiple valid answers, including *Europe* and *Asia*. Current methods often overlook this kind of overlap, which is problematic as knowledge triplets with shared elements within the same factual context require simultaneous consideration to ensure consistency during knowledge editing.

To further investigate the consequences of this overlook, we analyze existing mainstream KE datasets to evaluate the prevalence of Knowledge Element Overlap (KEO), which we found widespread. We then separate instances with KEO from those without and conduct experiments to evaluate the performance of current KE methods on these instances. As anticipated, the results reveal a significant performance decline for instances with KEO. Further analysis reveals that triplets with overlapping elements share knowledge neurons in the FFN layers, as shown in Figure 1(b), leading editors to overwrite knowledge unintentionally. For example, modifying *Europe* to *Africa* may inadvertently cause both *Europe* and *Asia* to be altered to *Africa*. Building on this observation, we collect KEO instances from Wikidata to construct a new dataset, EDITSET, enabling a more comprehensive exploration of KEO in KE. The dataset comprises over 700 relation types, with our study focusing on the 31 most common ones, consistent with prior research [Levy *et al.*, 2017; Elazar *et al.*, 2021; Meng *et al.*, 2022a; Zhong *et al.*, 2023; Wei *et al.*, 2024; Yin *et al.*, 2024; Ma *et al.*, 2024].

Given that the current methods have failed to effectively handle the KEO situation, this paper introduce an editing framework named **Set Knowledge Editor** (**SetKE**), which employs bipartite matching for optimization to enable knowledge editing in scenarios involving overlapping knowledge elements. Experimental results demonstrate that SetKE significantly outperforms existing KE methods in KEO scenarios. Furthermore, in-depth analyses confirm that SetKE effectively mitigates knowledge overwriting. Our main contributions are summarized as follows:

• We propose a novel formulation of Knowledge Set Editing (KSE) and construct a new dataset, EDITSET, to facilitate in-depth exploration of Knowledge Element Overlap (KEO).

• We introduce a new set editing framework, Set Knowledge Editor (SetKE), leveraging bipartite matching to achieve the editing target by treating knowledge entries as a set.

• Extensive experiments show that SetKE significantly outperforms existing methods in KEO scenarios, establishing it as a powerful paradigm for editing knowledge sets.

2 Preliminaries

This section outlines the definition of Knowledge Element Overlap (KEO), the current task formulation for Knowledge Editing, and our formulation for Knowledge Set Editing.

2.1 Knowledge Element Overlap

A factual knowledge entry $\mathcal{K}$ can be represented as a triplet (s, r, o), where the knowledge elements refer to the subject s, the object o, and their relation r. Based on the degree of overlap among knowledge elements—defined as sharing the same s, o or r—knowledge triplets can be classified into four types (inspired by the categorization by Sui et al. [2023]):

- Normal: At most one knowledge element overlaps.
- Subject Object Overlap (*SOO*): Overlapping subject-object pair (s and o) is shared among some triplets.
- Relation Subject Overlap (*RSO*): Overlapping subjects and shared relations among some triplets.
- Relation Object Overlap (*ROO*): Overlapping objects and shared relations among some triplets.

Apart from the Normal type, all other types are considered instances of knowledge element overlap (KEO).

2.2 Knowledge Editing

The widely adopted formulation of Knowledge Editing [Geva *et al.*, 2021; Geva *et al.*, 2022] represents knowledge $\mathcal{K}$ is stored in the language model f_θ in the form of triplets (s, r, o). The objective of KE is to modify the model f_θ into f_θ^* to achieve the transformation $(s, r, o) \rightarrow (s, r, o^*)$, whereby an input $x = t_r(s)$ is associated with its post-edit output $y = o^*$. Here, $t_r(s)$, referred to as the knowledge prefix, is a template used to describe the relation r with the subject s.

However, KEO is prevalent in real-world scenarios, posing challenges to the current KE framework, which uses a knowledge prefix $t_r(s)$ to edit a single object. In this context, the RSO type of KEO impacts the editing result, as other types generate different prefixes $t_r(s)$. Therefore, in this paper, KEO typically refers to the RSO type. In this case, multiple triplets share the same prefix $t_r(s)$, represented as $O = \{o_1, o_2, \ldots, o_N\}$, where N denotes the number of distinct objects. The current KE formulation fails to distinguish between these scenarios: $(s, r, o) \rightarrow (s, r, o^*)$ and $(s, r, O) \rightarrow (s, r, O^*)$. In the latter case, modifying a single object is often insufficient to achieve the desired outcome. Consequently, existing KE methods perform poorly, as proven in Section 3. This highlights the need for a new formulation to address these limitations.

2.3 Our Formulation: Knowledge Set Editing

In our new formulation of Knowledge Set Editing (KSE), the editing target is defined as $(s, r, O) \rightarrow (s, r, O^*)$, where the object is a set of entities $O^* = \{o_1^*, o_2^*, \ldots, o_N^*\}$. In the KEO scenarios, such as $\langle Roman\ Empire, continent, Europe\rangle$ and $\langle Roman\ Empire, continent, Asia\rangle$, indicating the Roman Empire spanned multiple continents, editing such knowledge (the transformation $O \rightarrow O^*$) requires set operations. For instance, consider the current values of $s = Roman\ Empire$, $r = existed\ in\ the\ continent$, and $O = \{Europe, Asia\}$. To modify *Europe* to *Africa*, the more reliable way is to perform $(O = \{Europe, Asia\}) \rightarrow (O^* = \{Africa, Asia\})$ instead of $(o_1 = Europe) \rightarrow (o_1^* = Africa)$. This highlights the need for set-based operations instead of modifying a single object,

as seen in previous approaches. Our proposed formulation addresses this need by considering the KEO scenarios.

Evaluation Metrics The evaluation metrics for the new formulation of KSE remain consistent with previous works (where the object is singular) [Meng *et al.*, 2022a; Meng *et al.*, 2022b]. These metrics assess the editing performance from the following perspectives: **Efficacy** measures whether the post-edit model f_θ^* generates the intended predictions for the target edits, which is quantified using the Efficacy Score (ES); **Generalization** measures the ability of the post-edit model f_θ^* to generalize edits across equivalent inputs, which is evaluated via the Generalization Score (GS); **Locality** measures whether the post-edit model f_θ^* retains its original predictions for inputs outside the editing scope (i.e., inputs that are not edited), which is evaluated by the Locality Score (LS). Higher scores for these metrics indicate better performance in their respective dimensions. Note that the metrics are evaluated based on the prompts provided in each instance. For detailed score calculations, please refer to Appendix.

3 Pilot Analyses & Dataset Construction

We conduct pilot analyses on existing KE datasets to examine the prevalence of KEO instances and evaluate their impact on current KE methods. Based on these findings, we construct a new KE dataset to further investigate the KEO problem.

3.1 KEO in Knowledge Editing

Statistics of Instances with KEO We analyze and quantify the instances with KEO within the following KE datasets.

• **zsRE** [Levy *et al.*, 2017] is one of the most prevalent QA datasets, extended and adopted by Cao et al. [2021]; Mitchell et al. [2021; 2022] for KE evaluations.

• **PARAREL** [Elazar *et al.*, 2021] is an expert-curated dataset containing diverse prompt templates for 38 relations sourced from the T-REx dataset [Elsahar *et al.*, 2018].

• **MQUAKE-T** and **MQUAKE-CF** [Zhong *et al.*, 2023] are constructed from Wikidata to evaluate the effectiveness of KE methods on multi-hop questions.

• **COUNTERFACT** [Meng *et al.*, 2022a] originates from PARAREL, each sample includes a knowledge triplet and meticulously crafted prompt templates.

The distribution of KEO in knowledge triplets within the aforementioned KE datasets is summarized in Table 1, which reveals the presence of KEO in all of them. Among the datasets analyzed, the zsRE dataset has the highest proportion of Normal triplets, accounting for 80.66%. In contrast, the Normal type constitutes only 7.29% and 5.92% in the PARAREL and COUNTERFACT datasets, respectively. Moreover, the MQUAKE-T and MQUAKE-CF datasets not only exhibit the KEO problem but also contain a substantial number of duplicate knowledge triplets, with sample sizes of 1772 and 3486, respectively. These statistics highlight the pervasive nature of knowledge element overlap in current mainstream KE datasets. However, the practical implications of this issue have been insufficiently considered.

Impact of KEO Given that previous editors have primarily been designed with a focus on the Normal type setting, neglecting the presence of KEO, we conduct an evaluation using

Dataset	Norm.	RSO	ROO	SOO	Dup.	Ratio	ALL
zsRE	8,066	0	902	1,103	0	80.66	10,000
PARAREL	2,023	2,242	23,241	742	21	7.29	27,738
MQUAKE-T	79	0	3	14	1,772	4.23	1,868
MQUAKE-CF	1,748	28	3,970	0	3,486	18.96	9,218
COUNTERFACT	592	315	9,376	10	11	5.92	10,000

Table 1: Statistics on current mainstream KE datasets. Note that a knowledge triplet can belong to different classes. Duplicate (**Dup.**) is exact identical knowledge triplets. **Ratio** calculates the ratios of Normal (**Norm.**) instances in **ALL** instances.

500 Normal triplets and 500 KEO triplets from the COUNTERFACT dataset to investigate the impact of KEO on editors' performance. As the experimental results shown in Figure 2, the performance of current SOTA editors generally exhibits lower effectiveness (except for the Generalization metric of MEND) when dealing with KEO-type knowledge compared to Normal type knowledge. Further analyses reveal that the performance discrepancy arises from the treatment of knowledge as key-value pairs stored in the models' feed-forward network (FFN). When editing knowledge with element overlap, these approaches often lead to knowledge overwriting, wherein the value vector associated with the editing target is overwritten. As a consequence, this results in decreased Efficacy and Generalization performance. Moreover, the ripple effects of knowledge updates [Cohen *et al.*, 2024] contribute to a decline in the Locality performance of instances with KEO. In summary, the observations above highlight the inadequacy of existing methods in handling KE with KEO, calling for a fresh perspective to address the challenges.

3.2 New Dataset: EDITSET

To further explore improvements in KE for instances with KEO, we introduce a novel dataset, EDITSET, designed with KEO samples to provide a focused evaluation environment.

Data Collection All knowledge triplets are collected from Wikidata, a knowledge base containing fact triplets associated with millions of entities. To collect KEO instances, we first sample subject-object pairs from Wikidata that share common relation properties. Subsequently, we utilize Wikidata dumps to extract data, identifying 710 specific relations and retaining only instances where multiple objects are associated with a given factual statement $t_r(s)$. Using this method, knowledge triplets are collected. Finally, GPT-4 is employed to generate evaluation prompts, including counterfactual, paraphrase, and neighborhood prompts, based on the knowledge triplets (see Appendix for prompt details).

Data Statistics of EDITSET The EDITSET dataset encompasses a total of 710 relations. Additionally, we construct a subset comprising the 31 most frequently utilized relations that intersect with the COUNTERFACT [Meng *et al.*, 2022a] and PARAREL [Elazar *et al.*, 2021] datasets. In EDITSET, each data sample is associated with N-objects, indicating that within each knowledge triplet (individual data instance), both the subject s and relation r are singular, while they correspond to N objects o. The subset comprises a total of 40,904 instances. Detailed statistics regarding the subset, including the total number of subjects and objects, are presented in Ta-
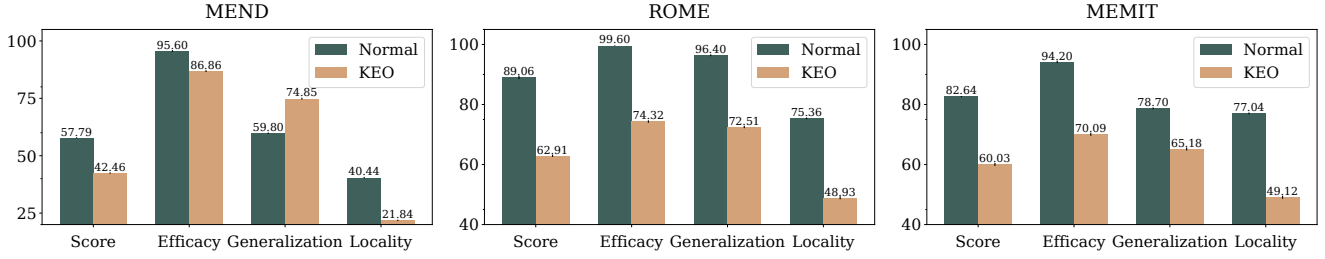
Figure 2: Comparison of editing performance on Normal and KEO type for MEND, ROME, and MEMIT.

ble 2. The counterfactual prompt is employed to assess **Efficacy**, the paraphrase prompt for **Generalization**, and the neighborhood prompt for **Locality**.

Overlap	N=3	N=4	N=5	N=6	N=7	N>=8	Total
Subjects	21,256	9,203	4,389	2,161	1,160	682	35,301
Relations	31	28	28	22	22	19	31
Objects	18,497	13,985	10,570	7,891	6,019	4,444	26,144
Counter.P.	22,770	9,574	4,503	2,193	1,175	687	40,900
Para.P.	43,164	18,410	8,674	4,219	2,249	1,325	78,031
Neigh.P.	3,780	2,768	2,240	1,757	1,509	1,221	3,988

Table 2: The statistics of EDITSET dataset on 31 commonly used relations. The EDITSET dataset consists of three types of prompt, Counter.P., Para.P., and Neigh.P. denote Counterfactual Prompt, Paraphrase Prompt, and Neighborhood Prompt, respectively. Each corresponds to a factual knowledge statement containing N-objects.

The dataset is designed explicitly for KSE to support the exploration of KE on instances involving KEO.

4 SetKE: A Set Knowledge Editor

This section details the proposed Set Knowledge Editor (SetKE) tailored to the KEO issue. Traditional KE methods have not yet considered KEO, making them inadequate for addressing this issue, as empirically demonstrated in Section 3. In response, we depart from the patterns established by prior work in the KE task and approach the problem from an entirely new perspective. Specifically, our goal in KSE is to transform a set O into a new set O^* to ensure the model is successfully edited. This can be viewed as editing (or aligning) the model's prediction (the actual output set) to match the ground truth (post-edit outputs), establishing a mapping between the current set and the target set. Inspired by the set prediction task [Sun *et al.*, 2021], we aim to constrain and optimize this mapping to achieve the desired transformation. To this end, we employ bipartite graph matching to find the optimal correspondence, with the matching computed using the Hungarian algorithm. The model weights are then updated based on the optimization, completing the KE process, and successfully editing the model. Figure 3 shows a simplified framework for editing the model from a set perspective.

4.1 Bipartite Matching Constraint

Based on the background presented earlier, we approach the optimization of editing KEO triples using the bipartite matching problem. In bipartite matching, given a set of predictions
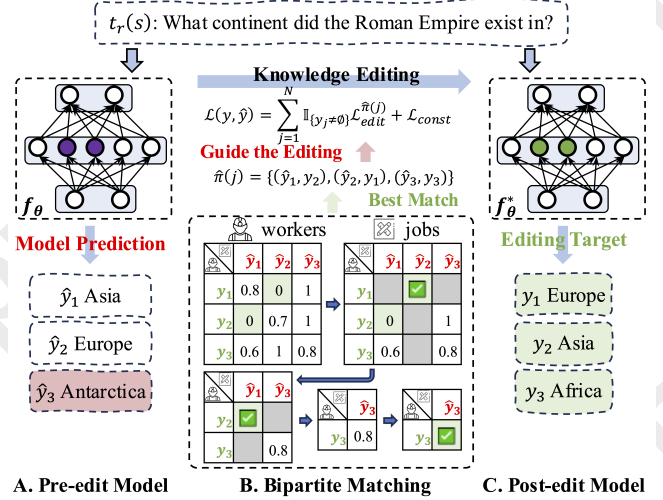

Figure 3: Simplified illustration of the SetKE framework.

and a set of editing targets, the goal is to find the optimal matching between these two sets that maximizes the overall matching score while minimizing the total matching cost. Building on this basis, the KSE problem is defined as follows: Given a set of editing targets $\mathbf{y} = \{y_j\}_{j=1}^M$ and a set of model predicted objects $\hat{\mathbf{y}} = \{\hat{y}_j\}_{j=1}^N$, where $M < N$, and placeholders ($\emptyset$) are used to pad unmatched predictions, the task is to find an optimal matching $\hat{\pi}$ that minimize the total matching cost, determined by:

$$\hat{\pi} = \arg \min_{\pi \in \Pi_N} \sum_{j=1}^N C_{\text{match}}(y_j, \hat{y}_{\pi(j)}), \quad (1)$$

where $\hat{y}_{\pi(j)}$ is the matching of y_j under the match π, Π_N represents the set of all matchings. $C_{\text{match}}(y_j, \hat{y}_{\pi(j)})$ is the pairwise matching cost, which, in this work, is defined as:

$$C_{\text{match}}(y_j, \hat{y}_{\pi(j)}) = -\mathbb{1}_{\{y_j \neq \emptyset\}} P_{\pi(j)}(y_j), \quad (2)$$

where $P_{\pi(j)}(y_j)$ represents the probability that the model's predicted object $\hat{y}_{\pi(j)}$ correctly matches the editing target y_j.

Building on this bipartite matching problem, we employ the Hungarian algorithm [Kuhn, 1955] to find the optimal matching. Figure 3B illustrates a toy example of optimal matching obtained using the Hungarian algorithm. The algorithm treats the set of ground truth objects $\{y_j\}_{j=1}^M$ as a

set of workers and the set of predicted objects $\{\hat{y}_j\}_{j=1}^N$ as a set of jobs. Given a matrix $\mathbf{A}^{N \times N}$ padded with 0, each element A_{ij} represents the cost of assigning job j to the worker i. To begin, the algorithm reduces the matrix by first subtracting the minimum value in each row from all elements in that row, ensuring that the smallest cost for each worker is minimized. Then, it proceeds by subtracting the minimum value in each column from all elements in that column, further simplifying the matrix and reducing the overall cost values. Next, the elements that are now zero represent potential optimal assignments. The algorithm assigns these zeros to form part of the optimal matching and iteratively adjusts the matrix until all workers are assigned jobs that give the minimal total cost. The Hungarian algorithm guarantees finding the optimal matching in $O(N^3)$ time complexity, as shown in Appendix Algorithm 1.

Once the optimal matching (or assignment) $\hat{\pi}$ is found, we define the overall loss as:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{j=1}^N \mathbb{1}_{\{y_j \neq \emptyset\}} \mathcal{L}_{\text{edit}}^{\hat{\pi}(j)} + \mathcal{L}_{\text{const}}, \tag{3}$$

where $\mathcal{L}_{\text{edit}}^{\hat{\pi}(j)}$ is the edit objective loss that measures the success of the edit, and $\mathcal{L}_{\text{const}}$ is the constraint loss that optimizes the edit locality metric. The overall process is computed for each ground truth y_j and its corresponding prediction $\hat{y}_{\hat{\pi}(j)}$, as follows:

$$\begin{aligned} \mathcal{L}_{\text{edit}}^{\hat{\pi}(j)} &= -\log P_{\theta(h_i^l + \delta_i)} \left[y_{ij} \mid x_i \right]_{j=\hat{\pi}(j)}, \\ \mathcal{L}_{\text{const}} &= D_{\text{KL}} \left(P_{\theta(h_i^l + \delta_i)} \left[y' \mid x' \right] \| P_\theta \left[y' \mid x' \right] \right). \end{aligned} \tag{4}$$

The second term $\mathcal{L}_{\text{const}}$ minimizes the KL divergence of predictions for the input x' (of the form "subject is a") to the unchanged model, which helps preserve the model's understanding of the subject's essence. Intuitively, $\mathcal{L}_{\text{edit}}$ is small when the model successfully updates its output, while $\mathcal{L}_{\text{const}}$ is small when the edit does not affect unrelated inputs.

4.2 Model Weight Updating for KE

Existing KE methods [Meng *et al.*, 2022b; Li *et al.*, 2024] apply optimization exclusively to specific parts of the model (those containing the knowledge to be edited) to avoid affecting non-edit target. This work follows this *locate-then-edit* paradigm. Based on the optimization strategy introduced in Section 4.1, the editing process proceeds as follows:

Step 1: Locating the Editing Component. Current KE works [Meng *et al.*, 2022a; Li *et al.*, 2024] have shown that knowledge generally resides in specific model layers, such as the feed-forward network (FFN) in Transformer-based models. Therefore, editing typically occurs within these layers. In this step, we build on previous works [Meng *et al.*, 2022a] to trace the location of the component to be edited.

Step 2: Perform the Edit. Given the traced editing component, specifically layer l in the model f_θ, where the editing is performed, and a given editing prompt x_i (describing the subject and relation of the set of KEO triplets (s_i, r_i)), we compute the hidden state h_i^l after the prompt passes through the l-th FFN layer, as follows:

$$h_i^l = \text{FFN}^l(x_i) = \sigma(x_i \cdot W_{fc}^l) \cdot W_{proj}^l. \tag{5}$$

Since the goal is to edit specific knowledge without affecting other parts of the model, we introduce a residual vector δ_i to adjust the model's output, using this vector to perform the optimization. This residual enables the transfer of knowledge from the original value v_i^l to the edited value z_i, thereby achieving the desired edit:

$$z_i = v_i^l + \delta_i = h_i^l + \arg\min_{\delta_i} \mathcal{L}(\delta_i). \tag{6}$$

Step 3: Spreading Edits Across Layers. To minimize side effects—such as unintended modifications or overwriting—and reduce excessive modifications, we distribute the residual vector δ_i across multiple layers of the model, rather than applying it to a single layer. This is achieved following prior works [Meng *et al.*, 2022b; Li *et al.*, 2024], which evenly spreads the updates across critical layers:

$$r_i^l = \frac{\delta_i}{L - l + 1}, \quad \mathcal{R}^l \triangleq [r_1^l \mid r_2^l \mid \cdots \mid r_n^l]. \tag{7}$$

Step 4: Weight Update. Finally, the model's weight matrix W_{proj}^l is updated by adding the incremental weight Δ^l to the original weight, as computed below:

$$\Delta^l = \mathcal{R}^l K^{l\top} (C^l + K^l K^{l\top})^{-1}, \tag{8}$$

where $\hat{W}_{proj}^l = W_{proj}^l + \Delta^l$ represents the updated weights, K^l represents the new keys, $C^l = K_0^l K_0^{l\top}$ is an estimate of the set of previous keys obtained through sampling, and $\mathcal{R}^l$ is the residual between the old and new value vectors.

5 Experiments

In this section, we evaluate the editing performance of baselines and SetKE on EDITSET concerning KEO knowledge. Specifically, our goal is to answer the following questions: **Q1**: How does SetKE perform compared to other baselines on EDITSET? **Q2**: How does the quantity of KEO overlaps affect the knowledge editing methods? **Q3**: How does the KEO phenomenon lead to the problem of knowledge overwriting? **Q4**: How does the bipartite matching constraint contribute to the results?

5.1 Experimental Settings

Dataset The editing is performed and evaluated using a subset of the proposed dataset, EDITSET, which comprises 31 commonly used relations. For specific statistics and detailed configurations of this subset, please refer to Section 3.2. In this dataset, all KEO types are RSO types and each instance is represented as $\{s, r, O = \{o_1, o_2, ...\}\}$, aligning with the objective of editing instances where the object is not singular.

Language Models We employ two widely adopted autoregressive language models, namely GPT2-Large (760M), GPT2-XL (1.5B) and GPT-J (6B) [Radford *et al.*, 2019], as the base language models to perform editing and assess the effectiveness of the KE approaches.

Baselines We select the following approaches: **FT-W** is a basic fine-tuning method. **KN** [Dai *et al.*, 2022] utilizes knowledge attribution to implement knowledge updates. **MEND** [Mitchell *et al.*, 2021] uses low-rank decomposition

	GPT2 Large (760M)				GPT2 XL (1.5B)			
Editor	Score	Efficacy	Generalization	Locality	Score	Efficacy	Generalization	Locality
FT-W	51.46	58.13 (0.5)	45.29 (0.5)	52.58 (0.4)	55.18	65.66 (0.5)	50.97 (0.5)	51.24 (0.4)
KN	42.21	38.22 (0.5)	37.46 (0.5)	54.89 (0.4)	40.65	35.93 (0.5)	35.86 (0.5)	55.29 (0.4)
MEND	–	–	–	–	38.75	89.58 (0.3)	81.58 (0.4)	18.52 (0.3)
PMET	43.34	40.82 (0.5)	39.24 (0.5)	51.98 (0.4)	44.13	41.15 (0.5)	40.08 (0.5)	53.39 (0.4)
MEMIT	49.65	49.53 (0.5)	45.64 (0.5)	54.59 (0.4)	56.60	61.12 (0.5)	55.07 (0.5)	54.11 (0.4)
ROME	64.08	72.03 (0.4)	69.63 (0.4)	53.84 (0.4)	65.89	75.27 (0.4)	73.58 (0.4)	53.62 (0.4)
SetKE	71.47	88.57 (0.3)	80.91 (0.4)	54.77 (0.4)	75.28	95.90 (0.2)	91.68 (0.2)	54.14 (0.4)

Table 3: Numerical results on EDITSET for 10,000 edits (95% confidence intervals in parentheses).

of gradients to learn new knowledge. **ROME** [Meng *et al.*, 2022a] first applies causal mediation analysis to updates the parameters. **MEMIT** [Meng *et al.*, 2022b] extends ROME to edit a large batch of facts. **PMET** [Li *et al.*, 2024] extends MEMIT to optimize the hidden states of both the FFN and Attention modules simultaneously.

5.2 Main Results (Q1)

Table 3 showcases numerical results on GPT2-Large (760M) and GPT2-XL (1.5B) over 10,000 cases in EDITSET, respectively. In this experiment, we compare with the recent baselines and our proposed method SetKE. We observe that SetKE is significantly superior to the exiting editing methods without using any additional parameters. Specifically, we outperform the recent advanced batch editing baseline MEMIT by up to 39.04% and 35.27% improvements regarding to the Efficacy and Generalization metrics in most cases for GPT2-Large. Under the same experimental conditions on GPT2-XL, SetKE also achieves a better performance improvement of 9.39%, 20.63% and 18.1% regarding to the Score, Efficacy and Generalization metrics, respectively, relative to the state-of-the-art methods. which indicates the effectiveness of our method in accurately altering models' behavior for the editing overlap factual knowledge without interference on each other.

5.3 Analysis for Overlap Quantity (Q2)

Figure 4 illustrates how the performance of editors changed with respect to different numbers of knowledge triplets overlap on EDITSET with GPT2-XL edited by different methods. We analyze a total of 3000 KEO cases and it can be seen from these results that the performance of all editing methods degrades as the number of overlap triplets increased. Figure 4 illustrates the performance variation of editors concerning different levels of overlap in knowledge triplets within EDIT-SET edited using GPT2-XL by distinct methods. The results indicate a general decrease in performance across all editing techniques as the number of overlapping triplets increases. Specifically, ROME and MEMIT demonstrate heightened sensitivity to the KEO problem, exhibiting a significant decline in editing performance. On the other hand, MEND shows relatively less susceptibility to KEO's impact. This disparity can be attributed to the knowledge localization approach of ROME and MEMIT, which tends to lead to more instances of knowledge overwriting in KEO scenarios. Further-
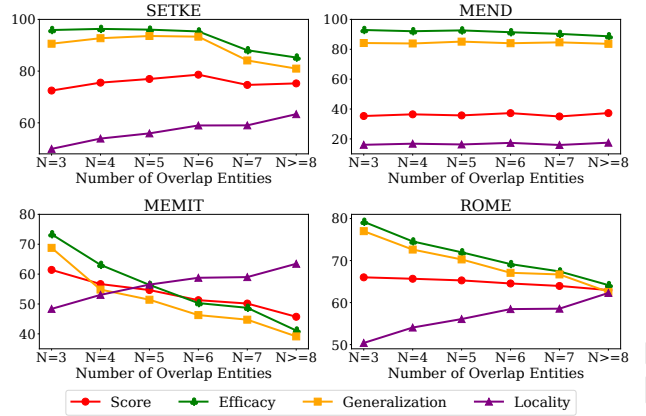

Figure 4: Comparing the impact of knowledge overlap number on baselines and SetKE.

more, while SetKE also operates within the realm of knowledge localization, it exhibits lesser vulnerability compared to ROME and MEMIT. This suggests that the bipartite matching constraint in SetKE enhances editing performance in KEO cases by minimizing the occurrence of knowledge overwriting. Finally, the Locality metric continues to improve across all methods as the number of overlaps increases. We attribute this trend to the overfitting of $\mathcal{L}_{\text{const}}$.

5.4 Analysis for Knowledge Overwriting (Q3)

As shown in Figure 1(b), we conduct an analysis of 500 KEO cases to investigate whether KEO knowledge triplets could lead to the problem of knowledge overwriting. Following the knowledge attribution method [Dai *et al.*, 2022], which is based on integrated gradients to determine the storage location of specific factual knowledge within the model (knowledge neurons), we analyze the degree of overlap among the knowledge neurons corresponding to KEO knowledge to assess whether knowledge triplets associated with KEO could potentially trigger knowledge overwriting. Our analysis results, as shown in Figure 5 for a representative example, reveal a significant overlap among the knowledge neurons representing three KEO factual pieces of knowledge in the GPT2 model. This observation aligns with our pilot analysis (see Figure 2), indicating that repetitive modifications to these
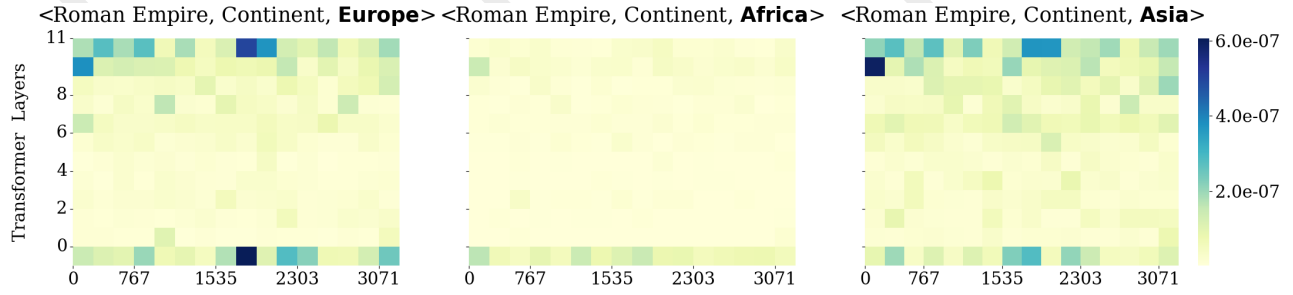
Figure 5: The result of knowledge localization of KEO type knowledge on GPT2.

	Object Set				Object Concatenation			
Editor	**Score**	**Efficacy**	**Generalization**	**Locality**	**Score**	**Efficacy**	**Generalization**	**Locality**
FT-W	46.65	51.62 (0.5)	39.92 (0.5)	50.29 (0.4)	43.36	44.53 (0.5)	37.20 (0.5)	**50.40 (0.4)**
MEND	37.44	**91.33 (0.3)**	82.60 (0.4)	17.52 (0.3)	43.05	67.64 (0.5)	63.37 (0.5)	25.56 (0.4)
MEMIT	55.95	67.51 (0.5)	56.04 (0.5)	47.71 (0.4)	57.07	82.31 (0.4)	64.00 (0.4)	40.34 (0.4)
ROME	59.48	80.54 (0.4)	79.90 (0.4)	39.21 (0.4)	**61.30**	85.86 (0.3)	**82.57 (0.3)**	39.71 (0.4)
SetKE	**73.68**	90.08 (0.3)	**90.76 (0.3)**	**53.77 (0.4)**	59.50	**89.43 (0.3)**	68.68 (0.4)	40.52 (0.4)

Table 4: The results of GPT-J (6B) on EDITSET for 7,500 edits (95% confidence intervals in parentheses).

specific knowledge neurons often lead to knowledge over-writing, resulting in a decline in editing performance.

5.5 Analysis for Ablation Study (Q4)

To evaluate the effectiveness of SetKE optimized through the bipartite matching constraint, we compared it with vanilla knowledge set editing (object concatenation). This setting treats the editing target as a long sequence of multiple concatenated objects, aiming to optimize multiple objects simultaneously. The results are displayed in Table 4. The left section of the table (object set) showcases the editors' performance aligned with the primary experimental configuration, while the right side illustrates the outcomes of the vanilla (object concatenation) setup. In the KEO scenario, we observe that optimizing multiple objects simultaneously generally outperforms iterative object optimization for editors such as MEND, ROME, and MEMIT. We speculate that this is because optimizing multiple objects simultaneously helps mitigate the issue of knowledge overwriting. Under object concatenation setting, SetKE exhibits lower performance compared to ROME. Nevertheless, all editors fall short of SetKE's performance in the object set configuration, highlighting the effectiveness of the bipartite matching constraint specifically designed for SetKE in the set scenario.

6 Related Work

Knowledge Editing (KE) involves modifying language models to revise the expression of factual knowledge learned from the pre-training corpus. The current work on the KE task can be categorized into the following three types: Meta-learning methods utilize additional trainable parameters to store memory or learn the required adjustments (Δ) to update knowledge in LLM [Cao *et al.*, 2021; Huang *et al.*, 2022; Tan *et al.*,

2024; Cheng *et al.*, 2024]. Locate-Then-Edit methods first employ causal mediation analysis to locate knowledge neurons that exhibit a positive correlation with a knowledge expression, and then modify them accordingly [Dai *et al.*, 2022; Meng *et al.*, 2022a; Meng *et al.*, 2022b; Huang *et al.*, 2024a; Huang *et al.*, 2024b; Wang *et al.*, 2024b]. In-Context Edit methods are a training-free paradigm where knowledge editing is achieved directly by concatenating demonstrations within the input context [Zheng *et al.*, 2023; Zhong *et al.*, 2023; Qi *et al.*, 2024]. However, existing KE methods are primarily restricted to modifying standard triplets and often overlook the phenomenon of knowledge overlap. To the best of our knowledge, this paper is the first to introduce a new dataset, EDITSET, designed to evaluate KEO editing performance, and proposes a general framework to mitigate knowledge overwriting.

7 Conclusion

In this paper, we analyze mainstream knowledge editing (KE) datasets and identify a widespread yet previously overlooked issue: the Knowledge Element Overlap (KEO) phenomenon. We also demonstrate that existing editing methods are insufficient to effectively address this challenge. To tackle this limitation, we propose a novel formulation, Knowledge Set Editing (KSE), and introduce SetKE, a method specifically designed for KEO scenarios. SetKE leverages bipartite matching to optimize object set editing, effectively resolving editing conflicts and improving accuracy. Experimental results confirm that SetKE outperforms existing methods, achieving state-of-the-art performance in editing multiple mainstream LLMs. Furthermore, we develop the EDITSET dataset, which serves as a comprehensive benchmark for evaluating knowledge set editing in KEO scenarios.

Acknowledgements

References

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

[Cheng *et al.*, 2024] Siyuan Cheng, Ningyu Zhang, Bozhong Tian, Xi Chen, Qingbin Liu, and Huajun Chen. Editing language model-based knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17835–17843, 2024.

[Cohen *et al.*, 2024] Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 11:283–298, 2024.

[Dai *et al.*, 2022] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, 2022.

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

[Elazar *et al.*, 2021] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031, 2021.

[Elsahar *et al.*, 2018] Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[Geva *et al.*, 2020] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

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

[Geva *et al.*, 2022] Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, 2022.

[Gupta *et al.*, 2023] Anshita Gupta, Debanjan Mondal, Akshay Sheshadri, Wenlong Zhao, Xiang Li, Sarah Wiegreffe, and Niket Tandon. Editing common sense in transformers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8214–8232, 2023.

[Hu *et al.*, 2024] Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. *arXiv preprint arXiv:2402.10987*, 2024.

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

[Huang *et al.*, 2024a] Xiusheng Huang, Jiaxiang Liu, Yequan Wang, and Kang Liu. Reasons and solutions for the decline in model performance after editing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[Huang *et al.*, 2024b] Xiusheng Huang, Yequan Wang, Jun Zhao, and Kang Liu. Commonsense knowledge editing based on free-text in llms. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14870–14880, 2024.

[Kuhn, 1955] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[Levy *et al.*, 2017] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *21st Conference on Computational Natural Language Learning, CoNLL 2017*, pages 333–342. Association for Computational Linguistics (ACL), 2017.

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

[Ma *et al.*, 2024] Jun-Yu Ma, Zhen-Hua Ling, Ningyu Zhang, and Jia-Chen Gu. Neighboring perturbations of knowledge editing on large language models. In *Forty-first International Conference on Machine Learning*, 2024.

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

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

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

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

[Petroni *et al.*, 2019] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, 2019.

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

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

[Rawat *et al.*, 2021] Ankit Singh Rawat, Chen Zhu, Daliang Li, Felix Yu, Manzil Zaheer, Sanjiv Kumar, and Srinadh Bhojanapalli. Modifying memories in transformer models. In *International Conference on Machine Learning (ICML)*, volume 2020, 2021.

[Sinitsin *et al.*, 2019] Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. Editable neural networks. In *International Conference on Learning Representations*, 2019.

[Song *et al.*, 2024] Ran Song, Shizhu He, Shuting Jiang, Yantuan Xian, Shengxiang Gao, Kang Liu, and Zhengtao Yu. Does large language model contain task-specific neurons? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7101–7113, 2024.

[Sui *et al.*, 2023] Dianbo Sui, Xiangrong Zeng, Yubo Chen, Kang Liu, and Jun Zhao. Joint entity and relation extraction with set prediction networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[Sun *et al.*, 2021] Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris M Kitani. Rethinking transformer-based set prediction for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3611–3620, 2021.

[Tan *et al.*, 2023] Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language model via meta learning. In *The Twelfth International Conference on Learning Representations*, 2023.

[Tan *et al.*, 2024] Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language models via meta learning. In *International Conference on Learning Representations*, 2024.

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

[Wang *et al.*, 2024b] Xiaohan Wang, Shengyu Mao, Shumin Deng, Yunzhi Yao, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. Editing conceptual knowledge for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 706–724, Miami, Florida, USA, 2024. Association for Computational Linguistics.

[Wei *et al.*, 2023] Yifan Wei, Xiaoyan Yu, Huanhuan Ma, Fangyu Lei, Yixuan Weng, Ran Song, and Kang Liu. Assessing knowledge editing in language models via relation perspective. *arXiv preprint arXiv:2311.09053*, 2023.

[Wei *et al.*, 2024] Yifan Wei, Xiaoyan Yu, Yixuan Weng, Huanhuan Ma, Yuanzhe Zhang, Jun Zhao, and Kang Liu. Does knowledge localization hold true? surprising differences between entity and relation perspectives in language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 4118–4122, 2024.

[Yao *et al.*, 2024] Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. Knowledge circuits in pretrained transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[Yin *et al.*, 2024] Xunjian Yin, Jin Jiang, Liming Yang, and Xiaojun Wan. History matters: Temporal knowledge editing in large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19413–19421, 2024.

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

[Zheng *et al.*, 2023] Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 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, 2023.

[Zhong *et al.*, 2023] Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 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, 2023.