InstructEdit: Instruction-Based Knowledge Editing for Large Language Models

Ningyu Zhang[♣*], **Bozhong Tian**[♣*], **Siyuan Cheng**[♡*], **Xiaozhuan Liang**[♡*],
Yi Hu[♡], **Kouying Xue**[♡], **Yanjie Gou**[♡], **Xi Chen**[♡†], **Huajun Chen**[♣†],
[♣] Zhejiang University [♡] Tencent
{zhangningyu, tbozhong}@zju.edu.cn
https://zjunlp.github.io/project/InstructEdit

Abstract

Knowledge editing for large language models can offer an efficient solution to alter a model's behavior without negatively impacting the overall performance. However, the current approaches encounter issues with limited generalizability across tasks, necessitating **one distinct editor for each task**, significantly hindering the broader applications. To address this, we take the first step to analyze the multi-task generalization issue in knowledge editing. Specifically, we develop an instruction-based editing technique, termed InstructEdit, which facilitates the editor's adaptation to various task performances simultaneously using simple instructions. With only one unified editor for each LLM, we empirically demonstrate that InstructEdit can improve the editor's control, leading to an average 14.86% increase in Reliability in multi-task editing setting. Furthermore, experiments involving holdout unseen task illustrate that InstructEdit consistently surpass previous strong baselines. To further investigate the underlying mechanisms of instruction-based knowledge editing, we analyze the principal components of the editing gradient directions, which unveils that instructions can help control optimization direction with stronger OOD generalization[1].

1 Introduction

Knowledge editing [Sinitsin *et al.*, 2020; Yao *et al.*, 2023; Wang and et al., 2023b; Mazzia and et al., 2023; Si and et al., 2023; Zhang *et al.*, 2023; Zhang and et al., 2024] aims to enable efficient and targeted post-hoc modifications in the parametric knowledge within Large Language Models (LLMs) [Mitchell *et al.*, 2022a; Dai *et al.*, 2022; Hartvigsen and et al., 2022; Cheng and et al., 2023; Tan *et al.*, 2024]. For example, as shown in Figure 1, when prompting with "How can I turn screws?", knowledge editing techniques can focus on specific areas in LLMs for adjustment, changing the answer from "Use a hammer" to "Use a wrench" without compromising the overall performance. Recently, numerous works on knowledge editing for LLMs have been proposed, which can be divided into two main paradigms [Yao *et al.*, 2023]: 1) Preserve Models' Parameters by utilizing additional parameters or memory [Mitchell *et al.*, 2022b]; 2) Modify Models' Parameters to alter the weights responsible for the undesirable output [Meng *et al.*, 2022a].

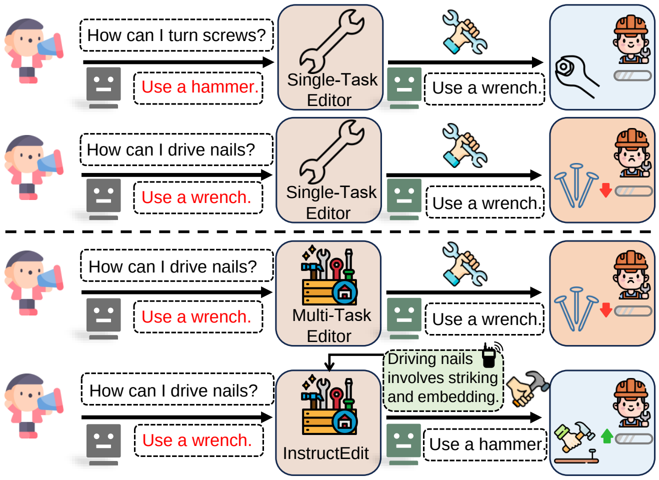


Figure 1: **Top:** The Single-Task Editor excels in specific tasks (e.g., turning screws) but fails in others (e.g., driving nails). **Bottom:** The vanilla Multi-Task Editor (all data mixed together) still struggles to choose the right tool for varied tasks without aid. Thus, we propose InstructEdit, enabling the Multi-Task Editor to respond aptly (such as using a hammer for nails) with instructional guidance.

However, previous knowledge editing approaches mainly focus on **single-task settings**, which means they may fail to achieve **multi-task generalization** capabilities and demonstrate inefficiency in editing when confronted with Out-of-Distribution (OOD) data. For example, as shown in Figure 1 and Table 1, the knowledge editing approach can simply change the behavior when prompting with "How can I turn screws", but fail to generalize to different task when prompting with "How can I drive nails". Fundamentally, for the Preserve Models' Parameters paradigm, Additional Parameters methods [Dong *et al.*, 2022; Hu *et al.*, 2022; Huang *et al.*, 2023] fit updated data with few extra parame-

[*] Equal contribution.
[†] Corresponding author.
[1] Code and datasets are available in https://github.com/zjunlp/EasyEdit.

Unseen	Seen	Reliability	Generalization	Portability
CounterFact	**CounterFact**	84.62	46.01	42.46
	Recent	⬇ -25.50	⬇ -21.34	⬇ -7.33
	ZsRE	⬇ -25.26	⬇ -18.36	⬇ -4.79
ZsRE	**ZsRE**	96.62	94.60	48.85
	Recent	⬇ -86.40	⬇ -91.33	⬇ -0.60
	CounterFact	⬇ -56.31	⬇ -64.90	⬇ -1.35

Table 1: Motivating knowledge editing results in multi-task generalization. Directly transferring to the unseen task (CounterFact and ZsRE) can result in a significant performance decay.

ters, while Memory-based approaches [Mitchell *et al.*, 2022b; Hartvigsen and et al., 2022], storing only current batch knowledge, can hardly generalize to OOD data. For the Modify Models' Parameters paradigm, Locate-Then-Edit [Meng *et al.*, 2022a; Meng *et al.*, 2022b] target and directly update specific parameters, but their updates are confined to provided data, limiting the model's generalization to other domains. Meta-learning editing approaches [Cao *et al.*, 2021; Mitchell *et al.*, 2022a; Cheng *et al.*, 2024] represent a branch in the realm of the Modify Models' Parameters paradigm, which utilizes a hypernet to predict specific weight updates for each data point, thereby facilitating the editing of LLMs [Radford *et al.*, 2019; Zhao and et al., 2023; Touvron and et al., 2023]. Yet traditional meta-learning editing methods typically focus on training a hypernet, which in essence functions as the Editor, specialized for a particular domain. Consequently, knowledge editing for a new task demands re-training the Editor, resulting in significant computational costs.

Intuitively, devising a strategy to enable the knowledge editing methods to effectively generalize across tasks is beneficial. Reflecting on prior research, to enhance the model's generalization capabilities, researchers have introduced instruction tuning [Wei *et al.*, 2022]. Instruction tuning can enhance the LLMs' comprehension skills by providing clearer commands or instructions, enabling the model to understand better and execute accurate responses. Previous studies [Wei *et al.*, 2022; Zhang and et al., 2023] observe that models refined through instruction tuning not only excel in performance on in-distribution datasets but also effectively generalize to previously unseen instruction data. Inspired by this, we propose the **Instruct**ion-based **Edit**ing method, dubbed as InstructEdit, which learns a well-formed Editor by designing the corresponding instructions for training on different tasks[2] , as shown in Figure 1. Specifically, we utilize meta-learning editing methods to train the editor across various meticulously curated instructions. We conduct experiments on four datasets and observe that InstructEdit can equip the Editor with the capability for multi-tasking editing, thereby conserving substantial human and computational resources. Our experiments reveal that InstructEdit can enhance the reliability by 14.86% (compared with MEND) on average when editing GPT2-XL. Furthermore, it can yield improvement by 42.04% on OOD dataset unseen during training.

2 Related Work

2.1 Knowledge Editing

Recently, knowledge editing [Sinitsin *et al.*, 2020; Zhang and et al., 2024] has emerged, aiming for efficient and accurate updates of knowledge in LLMs, to address the issues of outdated knowledge due to their training cut-off, factual fallacy, and potential generation of unsafe content. This technique is applied in various domains [Xu *et al.*, 2022; Mao *et al.*, 2023; Hase *et al.*, 2023; Wang *et al.*, 2023; Li *et al.*, 2023b; Cheng and et al., 2023; Zhong *et al.*, 2023; Akyürek *et al.*, 2023; Si *et al.*, 2024], with an increasing number of researches investigating the impact of knowledge editing [Ilharco *et al.*, 2023; Gupta *et al.*, 2023; Cohen *et al.*, 2023; Wu *et al.*, 2023; Wang and et al., 2023a; Gandikota *et al.*, 2023; Brown and et al., 2023; Wei *et al.*, 2023; Pan *et al.*, 2023; Li *et al.*, 2023d; Li *et al.*, 2023a; Ju and Zhang, 2023; Li *et al.*, 2023c; Onoe *et al.*, 2023; Pinter and Elhadad, 2023; Gupta *et al.*, 2024; Hernandez and et at., 2023; Huang *et al.*, 2024; Gu *et al.*, 2024; Lo *et al.*, 2024; Yin *et al.*, 2024; Yu and et al., 2024; Ma and et al., 2024]. Researchers have diligently classified existing knowledge editing approaches into two main paradigms:

Preserve Models' Parameters. For those approaches, knowledge can be updated without altering models' parameters, primarily following two paradigms: `Additional Parameters` and `Memory Based`. `Additional Parameters` integrate extra trainable parameters into the models. These added parameters are trained on a modified knowledge dataset, while the original models parameters remain unchanged. T-Patcher [Huang *et al.*, 2023] embeds a single neuron (patch) for each error in the model's final Feed-Forward Network (FFN) layer, activating only upon encountering the respective mistake. CaliNet [Dong *et al.*, 2022] drawing inspiration from [Dai *et al.*, 2022], introduces additional trainable parameters into the FFNs. `Memory Based` store edit examples in memory and use a retriever to select relevant edit facts for new inputs, thereby directing the model's fact generation. SERAC [Mitchell *et al.*, 2022b] presents a method that utilizes a distinct *counterfactual model* while maintaining the integrity of the original model. GRACE [Hartvigsen and et al., 2022] employs a distinct codebook as an adapter, progressively incorporating and refreshing elements to refine the model's predictions. In-context Knowledge Editing [Zheng *et al.*, 2023] produces outputs aligned with given knowledge using refined in-context prompts.

Modify Models' Parameters. Those approaches edit LLMs by modifying a portion of the parameter θ via applying an Δ matrix. There are primarily two paradigms: `Locate-Then-Edit` and `Meta-learning`. `Locate-Then-Edit` targets and directly updates specific parameters. ROME [Meng *et al.*, 2022a] utilizes causal mediation analysis for targeted editing but is limited to one fact at a time. Addressing this, MEMIT [Meng *et al.*, 2022b] has been proposed, an advancement of ROME, enabling direct memory embedding into the model through rank-one modifications of single-layer MLP weights. `Meta-learning` utilizes a hypernet to predict specific weight updates for each data point. MEND [Mitchell *et al.*, 2022a] and Knowledge Editor (KE) [Cao *et al.*, 2021] propose strategies that include an external editor, adept at identifying the optimal parameter

[2] The instructions text in this paper are limited to task descriptions rather than natural language instructions, which is a limitation we leave for future work.

Task (Dataset)	Instruction
CounterFact	**Task**: CounterFact **Description**: A dataset designed to challenge and assess model on... **Input**: The official language of...
ConvSent	**Task**: ConvSent **Description**: Teach the chatbot to sound [LABEL] about [TOPIC]... **Input**: What do you think of...
...	...

Table 2: Examples of the instructions. As for ConvSent, we need to replace [LABEL] and [TOPIC] according to the input.

set, θ, for knowledge editing, whilst simultaneously enforcing constraints to preserve the stability of the model.

2.2 Instruction Tuning

Instruction Tuning [Zhang and et al., 2023] markedly improves models' capability to handle new and unseen tasks by teaching them to comprehend and follow natural language instructions. In NLP, the focus is rapidly shifting towards refining LLMs [Brown and et al., 2020; OpenAI, 2022; Sun and et al., 2023; Taori and et al., 2023; Su and et al., 2023] to follow natural language instructions for real-world tasks. The effectiveness of these approaches is evident in the enhanced zero-shot and few-shot learning capabilities of these LLMs, demonstrating their improved proficiency in adapting to new tasks with minimal prior exposure. Inspired by the generalization capabilities of Instruction Tuning [Liang *et al.*, 2022; Ouyang and et al., 2022; Zhang and et al., 2023], we take the first step to integrate instructions into knowledge editing for LLMs, endowing one unified Editor with commendable instruction generalization and zero-shot capabilities to concurrently handle multiple editing tasks.

3 Background

Knowledge Editing Task Definition. Knowledge editing, as described by [Zhang and et al., 2024], aims to alter the behavior of an initial base model f_θ (where θ represents the model's parameters) in reaction to a specific edit descriptor (x_i, y_i) while maintaining the model's performance on other samples. The target is to create an edited model $f_{\theta'}$, which succinctly encapsulates the intended modifications in the model's performance. Concretely, the model f_θ can be represented with a function $f : \mathbb{X} \rightarrow \mathbb{Y}$ which associates an input x with its corresponding prediction y. Given an edit descriptor that includes the edit input x_i and edit label y_i with the condition that $f_\theta(x_i) \neq y_i$, the revised model $f_{\theta'}$ is engineered to yield the anticipated output, ensuring that $f_{\theta'}(x_i) = y_i$.

Multi-Task Editing Definition. In this paper, we mainly focus on multi-task editing setting, which means the editing approach should have the ability to handle various multiple tasks. We denote a LLM as f, characterized by its parameters θ to form f_θ. For editing in a single task, we introduce a dataset as D_{edit}. When we extend to multi-tasking scenarios, the dataset becomes a set comprising a collection $\{D^{t_1}_{edit}, D^{t_2}_{edit}, ..., D^{t_j}_{edit}\} \sim \mathcal{T}$, with each element representing to a unique task. In each specific task t_j, we engage with original input-output knowledge pairs, expressed as $(x_i^{t_j}, y_i^{t_j}) \sim D^{t_j}_{edit}$. The editing objective is to evolve the model's output from the original erroneous y_i' to a more accurate $y_i^{t_j}$, achieved by adjusting the model's parameters from f_θ to $f_{\theta'}$. Formally, the procedure can be described as follows:

$$f_\theta(x_i^{t_j}) = y_i' \rightarrow f_{\theta'}(x_i^{t_j}) = y_i^{t_j} \qquad (1)$$

Note that for all experiments, we utilize the multi-task editing setting and report the performance in Table 3. We also select one unseen dataset (a.k.a., ZsRE is unseen when training the Editor) for hold out editing evaluation.

4 Instruction-Based Knowledge Editing

4.1 Instruction Dataset Construction

Selected Task. To ensure diversity in multi-task editing, we select a range of datasets: Recent [Zhang and et al., 2024] for knowledge insertion, CounterFact [Zhang and et al., 2024] for counterfactual generation, and ConvSent [Mitchell *et al.*, 2022b] for sentiment editing in knowledge updating.

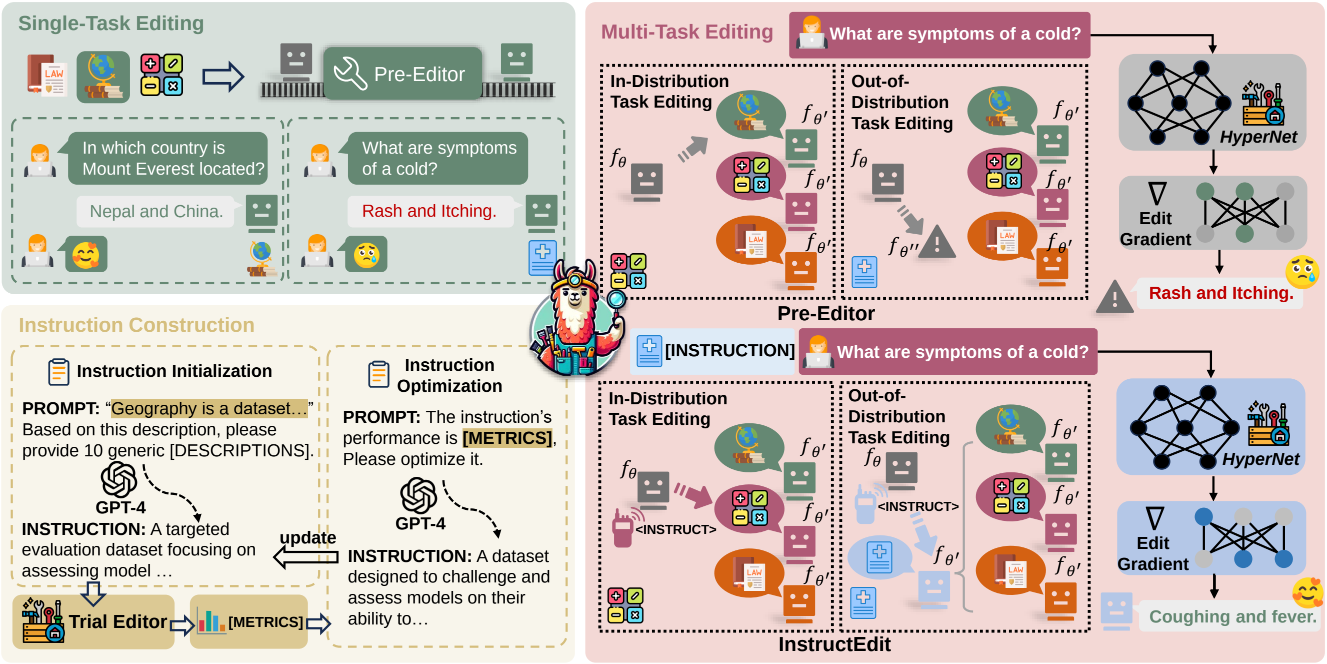
Recent focusing on triplets added to WikiData after July 2022, is used to enable model updates with the latest knowledge.

CounterFact emphasizes triplets from top-viewed Wikipedia pages to address the issue of models overlooking less prominent entities in modification edits.

ConvSent is a sentiment editing task aimed at adjusting a dialog agent's sentiment on a specific topic, like "What do you think of bananas?" without affecting responses of other topics. The training approach retains the original settings of the ConvSent. Additionally, we utilize a balanced subset, randomly sampled from the original ConvSent, for multi-task training. Detailed analyses are presented in Figure 4.

Hold Out Task. Empirically, we find that transferring knowledge from other tasks to ZsRE is challenging as shown in Table 1. Therefore, we utilize ZsRE, a zero-shot relation extraction dataset, to evaluate the generalization ability of multi-task editing, which means we do not incorporate ZsREin multi-task editing training. Specifically, we use the extended version by [Yao *et al.*, 2023], which adds a portability test and new locality sets to the original dataset.

Instruction Generation. We develop instruction templates for multi-task knowledge editing, encompassing four task families, they are: CounterFact, Recent, ConvSent, and ZsRE. Each includes instructions for task-specific model discovery, with input and target templates, and associated metadata. Specifically, we craft tailored instruction sets for each task family, including `[Task]`, `[Description]`, and `[Input]`. The `[Task]` represents the specific task linked to a data item, while the `[Input]` embodies the data item itself. We delve into the specifics with the `[Description]`, which is the essential component that uniquely tailors each task. Leveraging GPT-4 and detailed task information, we generate 20 descriptions



In which country is Mount Everest located?

Nepal and China.

What are symptoms of a cold?

Rash and Itching.

PROMPT: "Geography is a dataset…" Based on this description, please provide 10 generic [DESCRIPTIONS].

INSTRUCTION: A targeted evaluation dataset focusing on assessing model …

PROMPT: The instruction's performance is [METRICS], Please optimize it.

INSTRUCTION: A dataset designed to challenge and assess models on their ability to…

What are symptoms of a cold?

Rash and Itching.

What are symptoms of a cold?

Coughing and fever.

Figure 2: Assuming access to multi-domain task data: Law, Geography, Medicine, and Math. **Single-Task Editing**) Original editing is domain-specific (e.g., a Geography Editor edits geography-related knowledge but can't transfer it to Medicine). **Multi-Task Editing**) Previous methods (Pre-Editor) trained across domains (Law, Geography, and Math) often misdirect In-Distribution Task Editing. For OOD Task Editing (Medicine), a lack of guidance ∇ leads to missing the correct edit region. Instructions enable precise editing and improve generalization. **Instruction Construction**) We utilize GPT-4 to generate instructions through well-crafted prompts, evaluate metrics using the Trial Editor, and then employ GPT-4 for continuous Instruction Optimization, enhancing the instructions until there is no further improvement in metrics.

for each task and manually select 10 candidates based on their clarity and conciseness. Subsequently, we concatenate `[Task]`, `[Description]`, and `[Input]` to form the instructions presented in Table 2. Notably, while the last instruction is used to evaluate the model's generalization capabilities with instructions, the others are utilized for training. We further optimize instructions by feeding them with performance metrics into GPT-4 to improve the quality as shown in Figure 2. All instruction data will be released to the community.

4.2 Unified Editor Learning with Instructions

In this section, we primarily focus on the crucial role of instructions in directing the editing process and delve into a detailed explanation of how InstructEdit works. Specifically, we define the instruction set as $\{I^{t_1}, I^{t_2}, ..., I^{t_j}\} \sim \mathcal{I}$, where I^{t_j} represents a collection of instructions for task t_j. Based on the instructions, we outline the editing process as follows:

$$\begin{cases} f_{\theta'}(in^{t_j}, x_i) = y_i^{t_j} & x_i \in E(x_i^{t_j}), x_i^{t_j} \in D_{edit}^{t_j} \\ f_{\theta'}(x_i) = f_\theta(x_i) & Otherwise \end{cases} \quad (2)$$

where in^{t_j} refers to an instruction randomly selected from I^{t_j}, $E(x_i^{t_j})$ includes both $x_i^{t_j}$ and its equivalent expressions. InstructEdit employs the editing architecture of MEND, utilizing a meta-learning editor (hypernetwork) for implementing edits. InstructEdit updates the model's parameters $u_\ell \in \mathcal{M}$ with an editor parameterized by ϕ. It does this by mapping u_ℓ^i (the input to layer ℓ for each batch element i) and the gradient $\delta_{\ell+1}^i$ (calculated as $\delta_{\ell+1}^i \leftarrow \nabla_{W_\ell} L(x_i, y_i)$) to *pseudoactivations* $\tilde{u}_\ell^i$ and *pseudodelta* $\tilde{\delta}_{\ell+1}^i$. The knowledge editing gradient for the weight matrix u_ℓ is then represented as follows:

$$\tilde{\nabla}_{u_\ell} = \tilde{\delta}_{\ell+1}^i \tilde{u}_\ell^{i\top}. \quad (3)$$

Additionally, we scale the gradient $\tilde{\nabla}_{u_\ell}$ with L_2 norm of the gradient to isolate its directional component, denoted by $\vec{\nabla}_{u_\ell} = \tilde{\nabla}_{u_\ell} / \|\tilde{\nabla}_{u_\ell}\|_2$. Intuitively, $\vec{\nabla}_{u_\ell}$ pinpoints the key knowledge area for editing elements i. This facilitates a more meaningful comparison across various tasks by focusing solely on the gradient's direction while discarding its magnitude. We term this focused area as **editing area**.

Our primary objective is to equip the editor with the ability to comprehend and apply editing instructions, thus enhancing its capability to edit tasks that fall outside the usual distribution. Additionally, we append instructions before the input to facilitate multi-task editing. InstructEdit aims to augment multi-task editing capabilities, seeking a synergistic impact where the collective result surpasses the individual contributions. Through the concatenation of instructions, as shown in Figure 2, InstructEdit aims to cluster task vectors and reduce conflicts between tasks, which guarantees that the performance of the multi-task editor on individual tasks matches or even surpasses that of dedicated single-task editors.

DataSet	Model	Metric	Base	FT-L	CaliNet	KE	MEND	GRACE	InstructEdit
Multi-Task Editing									
CounterFact	GPT2-XL	Reliability	0.00	0.40	0.24	33.97	74.26	**96.31**	80.81
		Generalization	0.00	0.32	0.12	8.70	46.48	0.00	**53.16**
		Locality	100.0	43.73	82.81	90.94	58.68	**99.99**	67.83
		Portability	11.00	0.87	3.64	27.41	41.88	11.00	**50.83**
	LLaMA-2	Reliability	0.00	0.00	0.00	2.98	84.15	54.35	**84.39**
		Generalization	0.00	0.00	0.00	0.00	44.10	0.36	**50.18**
		Locality	100.0	70.66	89.28	90.86	91.18	**99.75**	88.04
		Portability	27.04	3.19	26.93	33.43	65.84	27.04	**69.43**
Recent	GPT2-XL	Reliability	2.61	6.48	11.53	49.37	85.62	**99.68**	85.70
		Generalization	1.58	2.21	5.37	10.98	**52.76**	1.58	51.66
		Locality	100.0	26.58	83.87	87.12	57.94	**100.0**	64.61
		Portability	17.19	16.78	10.31	30.41	42.26	17.73	**47.36**
	LLaMA-2	Reliability	9.87	6.16	9.79	15.88	82.31	83.72	**83.73**
		Generalization	7.27	3.87	6.64	0.08	54.66	7.35	**55.92**
		Locality	100.0	70.66	89.28	88.88	78.57	**99.98**	87.04
		Portability	43.52	3.15	43.26	43.52	60.84	44.13	**62.39**
ConvSent	GPT2-XL	Reliability	40.74	7.48	37.47	53.07	54.67	40.74	**65.43**
		Locality	100.0	42.86	87.47	94.58	96.58	**100.0**	94.27
		Fluency	613.13	548.55	396.43	615.61	601.93	414.03	**617.65**
Hold Out Editing									
ZsRE	GPT2-XL	Reliability	0.00	0.11	0.00	13.50	40.79	0.00	**82.83**
		Generalization	0.00	0.08	0.10	10.13	31.15	0.00	**78.40**
		Locality	100.0	74.06	95.66	82.59	94.79	**100.0**	94.57
		Portability	47.07	0.96	0.39	43.90	45.08	**47.07**	40.84
	LLaMA-2	Reliability	0.00	2.23	0.00	2.70	**76.95**	0.00	76.57
		Generalization	0.00	1.93	0.00	0.19	67.89	0.00	**70.11**
		Locality	100.0	98.89	99.66	95.15	90.14	**100.0**	94.16
		Portability	56.66	0.54	0.87	48.02	**58.63**	56.66	58.19

Table 3: **Multi-Task Editing Setting**: Editors train on a hybrid of CounterFact, Recent, and ConvSent datasets, and test on their specific test sets. **Hold Out Editing Setting**: The abovementioned editors are tested on ZsRE (OOD data). All metrics are "the higher, the better".

5 Experiments

5.1 Experimental Settings

Editing Models. We conduct experiments on GPT2-XL(1.5B) [Radford *et al.*, 2019] and LLaMA-2-Base (7B) [Touvron and et al., 2023].

Baselines. In this paper, we compare our method with FT-L method, which involves fine-tuning a specific layer's FFN identified via causal tracing in ROME [Meng *et al.*, 2022a]. We further compare our method with preserve models' parameters editing baselines including CaliNet and GRACE, and modify models' parameters editing baselines including MEND and KE.

5.2 Metrics

We apply several evaluation metrics consistent with those described in [Yao *et al.*, 2023].

Reliability. Reliable editing is defined when the post-edit model $f_{\theta'}$ generates the target answer correctly for the case (x_i, y_i). Reliability is assessed based on the average accuracy of the edit case.

$$\mathbb{E}_{x_{\mathrm{i}}',y_{\mathrm{i}}'\sim\{(x_{\mathrm{i}},y_{\mathrm{i}})\}}\mathbb{1}\left\{\operatorname{argmax}_y f_{\theta'}\left(y \mid x_{\mathrm{i}}'\right) = y_{\mathrm{i}}'\right\} \tag{4}$$

Generalization. The post-edit model $f_{\theta'}$ should predict the equivalent neighbor $N(x_{\mathrm{i}}, y_{\mathrm{i}})$, like rephrased sentences, and its performance is assessed by the average accuracy on examples uniformly sampled from this equivalence neighborhood[3].

$$\mathbb{E}_{x_{\mathrm{i}}',y_{\mathrm{i}}'\sim N(x_{\mathrm{i}},y_{\mathrm{i}})}\mathbb{1}\left\{\operatorname{argmax}_y f_{\theta'}\left(y \mid x_{\mathrm{i}}'\right) = y_{\mathrm{i}}'\right\} \tag{5}$$

Locality. Editing should be implemented locally, ensuring that the post-edit model $f_{\theta'}$ preserves the outputs for out-of-scope examples $O(x_i, y_i)$. Therefore, locality is measured by the rate at which $f_{\theta'}$ maintains the same predictions as the pre-edit model f_θ.

$$\mathbb{E}_{x_{\mathrm{i}}',y_{\mathrm{i}}'\sim O(x_{\mathrm{i}},y_{\mathrm{i}})}\mathbb{1}\left\{f_{\theta'}\left(y \mid x_{\mathrm{i}}'\right) = f_\theta\left(y \mid x_{\mathrm{i}}'\right)\right\} \tag{6}$$

Portability. Portability, proposed by [Yao *et al.*, 2023], gauges the edited knowledge application of the post-edit model $f_{\theta'}$. [Yao *et al.*, 2023] adds $P(x_i, y_i)$ to the existing dataset and calculates Portability by the edited model's average accuracy on these new reasoning parts.

$$\mathbb{E}_{x_{\mathrm{i}}',y_{\mathrm{i}}'\sim P(x_{\mathrm{i}},y_{\mathrm{i}})}\mathbb{1}\left\{\operatorname{argmax}_y f_{\theta'}\left(y \mid x_{\mathrm{i}}'\right) = y_{\mathrm{i}}'\right\} \tag{7}$$

Fluency. Fluency measures the edited model $f_{\theta'}$'s generative performance by using a weighted average of bi-gram and tri-gram entropies to evaluate text diversity. Lower values suggest higher repetition.

[3] We follow [Cao *et al.*, 2021; Yao *et al.*, 2023] to evaluate the rephrase-based generalization.

$$\text{Fluency} = \frac{\sum w_n \cdot H_n}{\sum w_n} \tag{8}$$

where H_n represents n-grams entropy (bi-gram for $n = 2$, tri-gram for $n = 3$) and w_n the respective weights. This metric is specifically tailored for ConvSent testing, where longer responses require scrutiny of the model's fluency.

5.3 Main Results

We evaluate the efficacy of InstructEdit by examining three key aspects: **Multi-Task Editing**, **Hold Out Editing**, and **Transfer to Unseen Instruction**.

Multi-Task Editing Results. Table 3 presents the corresponding results. `FT-L` [Yao *et al.*, 2023] exhibit subpar performance in Reliability for multi-task editing, which we believe is due to the interference of the original models' prior knowledge, complicating the editing process. Moreover, we notice that `FT-L` does not enhance Portability or Generalization, as expected due to its focus on fitting updated knowledge. Our experiments reveal that `FT-L` substantially reduces the original model's parameter knowledge, significantly lowering Locality. Preserve Models' Parameters Editing Methods like `CaliNet` [Dong *et al.*, 2022] maintain backbone model integrity, resulting in high Stability, but their performance in other metrics is unsatisfactory. Similar to `FT-L`, `CaliNet` overfits updated knowledge, leading to poor Generalization and Portability, but it has better Locality than `FT-L` as it doesn't alter the original parameters of the LLMs. While `GRACE` represents the state-of-the-art of Preserve Models' Parameters Editing Methods, delivering outstanding Reliability and Locality, it falls short in the metrics of Generalization and Portability. Modify Models' Parameters Editing Methods, such as `KE` [Cao *et al.*, 2021] and `MEND` [Mitchell *et al.*, 2022a], surpass previous editing approaches in effectiveness. Both `MEND` and `KE` excel across all metrics, achieving a balance between Reliability and Locality. This is attributed to their optimization objectives that limit update extents, thus enabling editors to adjust parameters while preserving model stability. We can observe our InstructEdit improves editing precision and control with instruction-guided methods, reaching effectiveness akin to advanced hypernets like MEND and KE. While MEND and KE yield effective editing results, their performance is suboptimal on OOD data, with editing in In-Distribution data often causing misdirection in the update trajectory of the posterior vector space. However, we find that providing specific command hints to the Editor can substantially alleviate this issue.

Hold Out Editing Results. To evaluate the adaptability of knowledge editing methods to OOD data, we devise the "Hold Out Editing Setting". In this setup, the editor is trained using datasets like Recent, CounterFact, and ConvSent, and then evaluated on ZsRE. From Table 3, we notice a linear decline in the performance of all previous knowledge editing baselines when applied to OOD data. This decline can be attributed primarily to the editor's limitations in defining new editing tasks and its insufficient generalization capability for handling OOD scenarios. We observe that InstructEdit can effectively address these challenges. Note that such robust generalization abilities are mainly inherent in instruction tuning, a synergy that enables InstructEdit to attain performance levels on par with single-task editing, even on task datasets that are unseen during the training phase.

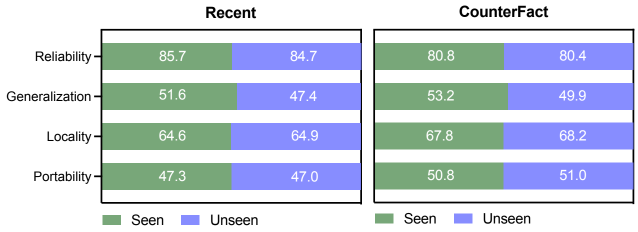


Figure 3: InstructEdit demonstrates proficiency in generalizing to Unseen instructions (unseen instructions introduced in Section 4.2), achieving results comparable to Seen instructions.

Transfer to Unseen Instructions. To delve deeper into the generalizability of Instruction-based Editing, we evaluate InstructEdit's capacity with instructions that have not been encountered previously. This setting is different from the hold-out editing setting since we still use the data in CounterFact, Recent, ConvSent, and ZsRE, but with new instructions. Specifically, as outlined in Section 4.2, we construct five novel, unseen instructions to assess the Editor's proficiency in generalizing instructions. Observations from Figure 3 reveal that the Editor is indeed capable of adapting to these Unseen Instructions. It is noteworthy that utilizing instructions on which the Editor has been trained can result in enhanced editing performance. Thus, InstructEdit can achieve comparable outcomes by employing instructions that are semantically akin to those encountered during training. These empirical results also indicate that we can develop an Editor to follow human instructions and we leave this for future works.

5.4 Why Instruction Helps Multi-Task Editing?

We analyze the principal components of the editing area $\vec{\nabla}_{u_\ell}$ using t-SNE, as presented in Section 4.2, which is generated by the editor for specific layers of LLMs. Our underlying assumption is that these principal components encapsulate the intrinsic characteristics of the editing area involved in editing the data. Specifically, we focus our analysis on cases where the conventional editing methods fall short, while InstructEdit demonstrates effectiveness.

Finding 1: Instruction can Help Control Optimization Direction. As observed in Table 2, MEND exhibits subpar performance in multi-task scenarios, particularly in terms of Reliability and Generalization, where it is significantly outperformed by InstructEdit. Upon analyzing the left panel in Figure 4(a), we observe that MEND, when handling multi-task editing, tends to show significant overlap in editing area across different tasks. This overlap could potentially cause MEND to not only confuse previously learned tasks but also struggle in effectively generalizing to new tasks with shifted distribution compared to the training tasks. However, by introducing instructions, InstructEdit can effectively control the knowledge editing gradient and encourage distinct separation with adequate margin in the editing area for various tasks, which aligns with the distribution observed in the single-task training setting

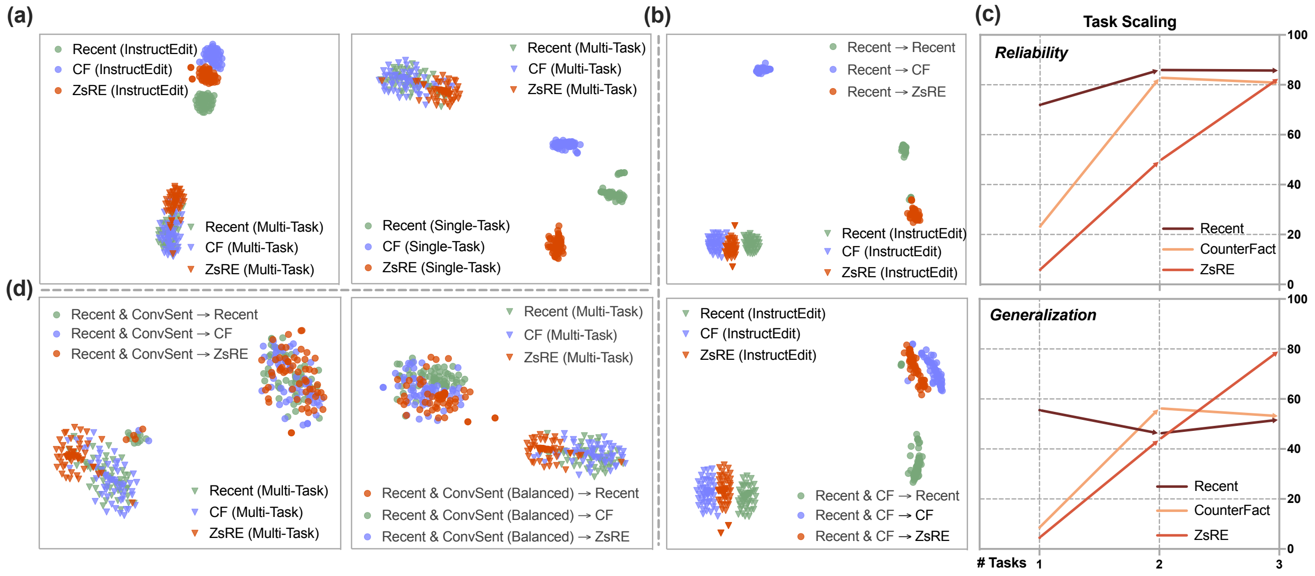


Figure 4: (a) Compares instruction effects on knowledge editing gradient $\tilde{\nabla}_{u_\ell}$. **Recent (InstructEdit)** and **Recent (Multi-Task)** illustrate $\tilde{\nabla}_{u_\ell}$ on Recent using InstructEdit and MEND in multi-task settings, respectively. **Recent (Single-Task)** shows MEND's results of training on Recent alone. (b) Demonstrates task scaling's impact on InstructEdit, with **Recent→ZsRE** for training on Recent and testing on ZsRE, and **Recent&CF→ZsRE** for joint training on Recent, CounterFact, and testing on ZsRE. (c) Illustrates the reliability and generalization performance across task scaling. (d) Balances ConvSent by extracting 1,427 entries for **ConvSent (Balanced)**.

in the right panel of Figure 4(a). Furthermore, the discriminative editing area in InstructEdit is adaptable to OOD data, which leads to superior knowledge editing when handling new tasks, while maintaining performance comparable to models trained on single tasks on ID tasks.

Finding 2: More Tasks, Stronger OOD Generalization. Figure 4(b) illustrates that when InstructEdit is trained on a single task, the editing areas of the three tasks appear somewhat discriminative. Instead, the performance of the corresponding tasks is suboptimal, as demonstrated in Figure 4(c). We think that even though instructions aid in distinguishing different tasks, the knowledge learned from a single task struggles to generalize to others. By scaling the number of tasks in training, we notice that the editing areas of InstructEdit for various tasks almost see no overlap, and editing reliability improves correspondingly in Figure 4(c). Furthermore, as the scope of tasks broadens, the directions of knowledge editing gradient of different tasks start to converge, yet they retain their relative margin. Intuitively, InstructEdit trained across diverse domains harnesses these domain-related instructions to extrapolate effectively to new, unseen domains, while offering a trade-off between specialized knowledge adaptation and broad generalization. Nevertheless, it is crucial to acknowledge that a scalability bottleneck might be encountered, and confronting entirely new types of editing tasks, such as cross-linguistic tasks, will introduce further complexities.

Finding 3: Improving Performance with Appropriate Data Proportion. In preliminary experiments, we notice task imbalances impede proper multi-task training and cause a significant performance decline when ConvSent is involved in the training. Hence, we contemplate balancing the data proportions across different tasks. By observing Figure 4(d), we find that the knowledge editing gradient directions become more regular after data balancing and editing reliability of the editor increases from 18.23 to 25.55 on the OOD tasks. Additionally, we find that task imbalances lead to the editor inducing editing gradients of relatively large magnitudes, and the gradient magnitude distributions for each task vary significantly, which appears to be a key factor influencing the editor's generalization. This result confirms the significance of appropriate data proportions for multi-task editing.

6 Discussion and Conclusion

We focus on a new problem of knowledge editing for LLMs: generalizing to new tasks. We introduce multi-task editing, illustrating the limitations of existing knowledge editing approaches in task transferability and presenting a viable solution InstructEdit. The proposed approach can effectively guide the Editor for precise editing, with its effectiveness confirmed through comprehensive experiments and visualization analysis.

Acknowledgments

We would like to express gratitude to the anonymous reviewers for kind comments, EasyEdit [Wang and et al., 2023a] and many other related works for their open-source contributions as well as GPT-4 Service [OpenAI, 2023]. This work was supported by the National Natural Science Foundation of China (No. 62206246), the Fundamental Research Funds for the Central Universities (226-2023-00138), Zhejiang Provincial Natural Science Foundation of China (No. LGG22F030011), Yongjiang Talent Introduction Programme (2021A-156-G), CCF-Tencent Rhino-Bird Open Research Fund, and Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

References

[Akyürek *et al.*, 2023] Afra Feyza Akyürek, Eric Pan, Garry Kuwanto, and Derry Wijaya. Dune: Dataset for unified editing. In *EMNLP*, 2023.

[Brown and et al., 2020] Tom B. Brown and et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[Brown and et al., 2023] Davis Brown and et al. Edit at your own risk: evaluating the robustness of edited models to distribution shifts, 2023.

[Cao *et al.*, 2021] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *EMNLP*, 2021.

[Cheng and et al., 2023] Siyuan Cheng and et al. Can we edit multimodal large language models? In *EMNLP*, 2023.

[Cheng *et al.*, 2024] Siyuan Cheng, Ningyu Zhang, Bozhong Tian, Xi Chen, Qingbing Liu, and Huajun Chen. Editing language model-based knowledge graph embeddings. In *AAAI*, 2024.

[Cohen *et al.*, 2023] Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *CoRR*, abs/2307.12976, 2023.

[Dai *et al.*, 2022] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *ACL*, 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 *EMNLP*, 2022.

[Gandikota *et al.*, 2023] Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. *CoRR*, abs/2303.07345, 2023.

[Gu *et al.*, 2024] Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing can hurt general abilities of large language models, 2024.

[Gupta *et al.*, 2023] Anshita Gupta, Debanjan Mondal, Akshay Krishna Sheshadri, Wenlong Zhao, Xiang Lorraine Li, Sarah Wiegreffe, and Niket Tandon. Editing commonsense knowledge in GPT. In *EMNLP*, 2023.

[Gupta *et al.*, 2024] Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting, 2024.

[Hartvigsen and et al., 2022] Thomas Hartvigsen and et al. Aging with GRACE: lifelong model editing with discrete key-value adaptors. In *NeurIPS*, 2022.

[Hase *et al.*, 2023] Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models, 2023.

[Hernandez and et at., 2023] Evan Hernandez and et at. Linearity of relation decoding in transformer language models, 2023.

[Hu *et al.*, 2022] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

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

[Huang *et al.*, 2024] Youcheng Huang, Wenqiang Lei, Zheng Zhang, Jiancheng Lv, and Shuicheng Yan. See the unseen: Better context-consistent knowledge-editing by noises, 2024.

[Ilharco *et al.*, 2023] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *ICLR*, 2023.

[Ju and Zhang, 2023] Yiming Ju and Zheng Zhang. Klob: a benchmark for assessing knowledge locating methods in language models, 2023.

[Li *et al.*, 2023a] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *NeurIPS*, 2023.

[Li *et al.*, 2023b] Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. Pmet: Precise model editing in a transformer. 2023.

[Li *et al.*, 2023c] Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. Unveiling the pitfalls of knowledge editing for large language models. *arXiv preprint arXiv:2310.02129*, 2023.

[Li *et al.*, 2023d] Zichao Li, Ines Arous, Siva Reddy, and Jackie C. K. Cheung. Evaluating dependencies in fact editing for language models: Specificity and implication awareness. In *EMNLP*, 2023.

[Liang *et al.*, 2022] Xiaozhuan Liang, Ningyu Zhang, Siyuan Cheng, Zhenru Zhang, Chuanqi Tan, and Huajun Chen. Contrastive demonstration tuning for pre-trained language models. In *EMNLP*, 2022.

[Lo *et al.*, 2024] Michelle Lo, Shay B. Cohen, and Fazl Barez. Large language models relearn removed concepts, 2024.

[Ma and et al., 2024] Jun-Yu Ma and et al. Neighboring perturbations of knowledge editing on large language models, 2024.

[Mao *et al.*, 2023] Shengyu Mao, Ningyu Zhang, Xiaohan Wang, Mengru Wang, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Editing personality for llms. 2023.

[Mazzia and et al., 2023] Vittorio Mazzia and et al. A survey on knowledge editing of neural networks, 2023.

[Meng *et al.*, 2022a] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual knowledge in GPT. In *NeurIPS*, 2022.

[Meng *et al.*, 2022b] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *CoRR*, abs/2210.07229, 2022.

[Mitchell *et al.*, 2022a] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *ICLR*, 2022.

[Mitchell *et al.*, 2022b] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *ICML*, 2022.

[Onoe *et al.*, 2023] Yasumasa Onoe, Michael J. Q. Zhang, Shankar Padmanabhan, Greg Durrett, and Eunsol Choi. Can lms learn new entities from descriptions? challenges in propagating injected knowledge. In *ACL*, 2023.

[OpenAI, 2022] OpenAI. The blog used to introduce chatgpt. *https://openai.com/blog/chatgpt*, 2022.

[OpenAI, 2023] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[Ouyang and et al., 2022] Long Ouyang and et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

[Pan *et al.*, 2023] Haowen Pan, Yixin Cao, Xiaozhi Wang, and Xun Yang. Finding and editing multi-modal neurons in pre-trained transformer, 2023.

[Pinter and Elhadad, 2023] Yuval Pinter and Michael Elhadad. Emptying the ocean with a spoon: Should we edit models? In *EMNLP*, 2023.

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

[Si and et al., 2023] Nianwen Si and et al. Knowledge unlearning for llms: Tasks, methods, and challenges, 2023.

[Si *et al.*, 2024] Nianwen Si, Hao Zhang, and Weiqiang Zhang. Mpn: Leveraging multilingual patch neuron for cross-lingual model editing, 2024.

[Sinitsin *et al.*, 2020] Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry V. Pyrkin, Sergei Popov, and Artem Babenko. Editable neural networks. In *ICLR*, 2020.

[Su and et al., 2023] Hongjin Su and et al. One embedder, any task: Instruction-finetuned text embeddings. In *ACL*, 2023.

[Sun and et al., 2023] Tianxiang Sun and et al. Moss: Training conversational language models from synthetic data. 2023.

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

[Taori and et al., 2023] Rohan Taori and et al. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

[Touvron and et al., 2023] Hugo Touvron and et al. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

[Wang and et al., 2023a] Peng Wang and et al. Easyedit: An easy-to-use knowledge editing framework for large language models. *CoRR*, abs/2308.07269, 2023.

[Wang and et al., 2023b] Song Wang and et al. Knowledge editing for large language models: A survey, 2023.

[Wang *et al.*, 2023] Jiaan Wang, Yunlong Liang, Zengkui Sun, Yuxuan Cao, and Jiarong Xu. Cross-lingual knowledge editing in large language models, 2023.

[Wei *et al.*, 2022] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *ICLR*, 2022.

[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, 2023.

[Wu *et al.*, 2023] Suhang Wu, Minlong Peng, Yue Chen, Jinsong Su, and Mingming Sun. Eva-kellm: A new benchmark for evaluating knowledge editing of llms. *CoRR*, abs/2308.09954, 2023.

[Xu *et al.*, 2022] Yang Xu, Yutai Hou, and Wanxiang Che. Language anisotropic cross-lingual model editing. *ArXiv*, abs/2205.12677, 2022.

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

[Yin *et al.*, 2024] Xunjian Yin, Jin Jiang, Liming Yang, and Xiaojun Wan. History matters: Temporal knowledge editing in large language model. In *AAAI*, 2024.

[Yu and et al., 2024] Lang Yu and et al. MELO: enhancing model editing with neuron-indexed dynamic lora. In *AAAI*, 2024.

[Zhang and et al., 2023] Shengyu Zhang and et al. Instruction tuning for large language models: A survey. *CoRR*, abs/2308.10792, 2023.

[Zhang and et al., 2024] Ningyu Zhang and et al. A comprehensive study of knowledge editing for large language models, 2024.

[Zhang *et al.*, 2023] Ningyu Zhang, Yunzhi Yao, and Shumin Deng. Editing large language models. In *AACL: Tutorial Abstract*, 2023.

[Zhao and et al., 2023] Wayne Xin Zhao and et al. A survey of large language models. *CoRR*, abs/2303.18223, 2023.

[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 *EMNLP*, 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 *EMNLP*, 2023.