InstructEd: Soft-Instruction Tuning for Model Editing with Hops

Anonymous ACL submission

Abstract

The task of model editing becomes popular for correcting inaccurate or outdated parametric knowledge in Large Language Models (LLMs). However, there are major limitations of state of the art (SOTA) model editing methods, including the excessive memorization issue caused by the direct editing methods, as well as the error propagation and knowledge conflict issues from the memory enhancement methods, resulting in hindering models' *portability*, e.g., the ability to transfer the new knowledge to related one-hop or multi-hop content. To address these issues, we propose the InstructEd method, the idea of which is to insert soft instructions into the attention module so as to facilitate interactions between instructions and questions and to understand and utilize new facts. Our main findings are: (i) InstructEd has achieved SOTA performance on three datasets for one-hop/multi-hop evaluation with LLaMAs and GPT2, achieving 10% (5%) improvement in one-hop (multi-hop) model editing. (ii) Different from earlier methods on editing parameters in FFN, we show that editing attention can also help. (iii) Model editing is highly related to retrieval augmented methods, which can help improve the locality of model editing while slightly decrease the editing performance with hops.

1 Introduction

Large Language Models (LLMs) have accumulated substantial parametric knowledge, showcasing remarkable progress in knowledge-driven tasks such as question answering (Kwiatkowski et al., 2019; Chen et al., 2021, 2022; Hu et al., 2023) and reasoning (Mihaylov et al., 2018; He et al., 2023). However, LLMs are susceptible to errors stemming from inaccurate or outdated parametric knowledge, restricting the reliability of these models. Previous research has introduced Model Editing (ME) (De Cao et al., 2021; Mitchell et al., 2022b) to rectify or update parametric knowledge in LLMs without the need for expensive re-training processes.

In general, these methods either update the parametric knowledge within LLMs with either some given factual knowledge (called '*modification based setting*') or some external knowledge repository (called '*memory based setting*'). In this paper, we propose a method that can address both settings.

In order to assess ME methods' effectiveness in understanding and application of related factual knowledge, Yao et al. (2023); Zhong et al. (2023) proposed two new benchmarks, introducing the notion of *portability*, e.g., the ability to apply the new knowledge to related one-hop (Yao et al. (2023)) or multi-hop (Zhong et al. (2023)) content.

As illustrated in Fig.1, modification-based methods (Mitchell et al., 2022a; Meng et al., 2022, 2023a) calculate the parameter shift θ based on the new fact, which can only used to recall modified knowledge (Q-ED). Once the facts are discrepant with the new fact (such as Re-ED[1]), they cannot answer these questions based on the modified facts. As for the memory-based methods (Zheng et al., 2023; Mitchell et al., 2022b) maintain LLMs parameters unchanged and retrieve knowledge relevant to the current edit from a pre-constructed memory to achieve editing. Some recent memory based methods like MeLLo (Zhong et al., 2023) and PokeMQA (Gu et al., 2023) have been proposed to address multi-hop question editing. These methods rely on the retrieval results and necessitate decomposing multi-hop questions into subproblems. However, the knowledge conflict issue[2] between retrieved results and LLMs, and error propagation issue caused by decomposition, make these methods unstable to handle Q-ED and Re-ED.

[1]Before addressing the question "What city is Messi's team in?" we need to know "Messi is playing for Inter Miami."

[2]LLMs prioritize their own parametric knowledge and overlook retrieved non-parametric knowledge

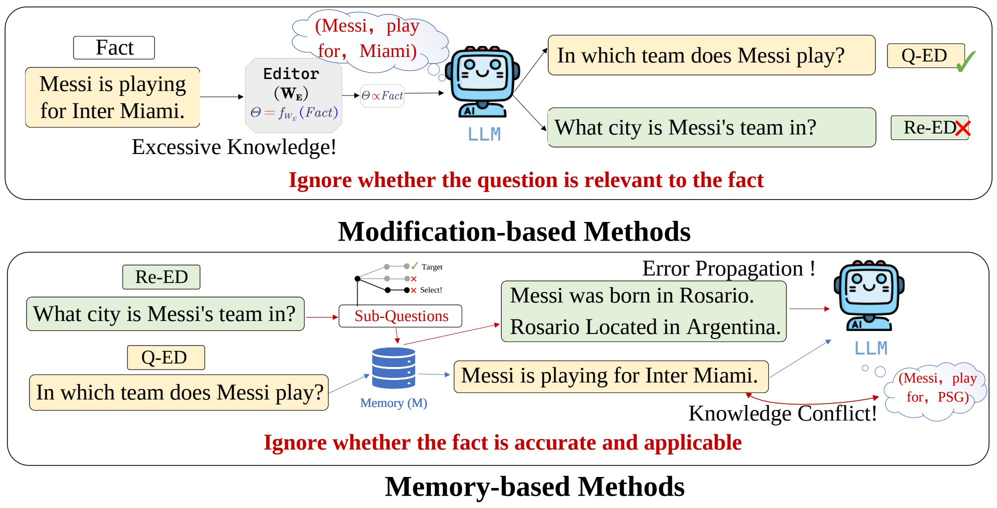
Fact
Messi is playing for Inter Miami.
Editor (W_E)
$\Theta = f_{W_E}(Fact)$
(Messi, play for, Miami)
$\Theta \propto Fact$
LLM
Excessive Knowledge!
In which team does Messi play?
Q-ED ✓
What city is Messi's team in?
Re-ED ✗
Ignore whether the question is relevant to the fact
Modification-based Methods
Re-ED
What city is Messi's team in?
Q-ED
In which team does Messi play?
Target
Select!
Sub-Questions
Memory (M)
Error Propagation !
Messi was born in Rosario. Rosario Located in Argentina.
Messi is playing for Inter Miami.
LLM
(Messi, play for, PSG)
Knowledge Conflict!
Ignore whether the fact is accurate and applicable
Memory-based Methods

Figure 1: The existing methods fail to leverage knowledge effectively. They either overlook whether the question is relevant to the facts, resulting in excessive memorization and inability to resolve Re-ED; or they neglect whether the facts are accurate and applicable due to factors error propagation and knowledge conflicts. Q-ED represents the questioning of a modified fact, Re-ED represents the questioning related but un-equivalent to the modified facts.

Recently, instruction-tuning (Zhang et al., 2023) has emerged as a new paradigm for tuning LLMs to generate responses based on natural language instruction, which has been extensively researched in language (Gupta et al., 2023; Li et al., 2023a) and vision domains (Brooks et al., 2023; Liu et al., 2023a,b). These methods enhance LLMs by fine-tuning them using high-quality (*instruction, output*) pairs, boosting the model to comprehend user intentions and follow instructions more accurately.

In this paper, we present **InstructEd**, a novel approach for the challenging task of model editing with hops, by using soft instruction tuning. We construct an instruction dataset based on existing editing data to facilitate training. Unlike previous methods, we augment a set of learnable soft-instruction prompts as prefixes to the input instruction tokens in the attention module (rather FFN). We use a relevance score between the instruction and inputs to learn how to use knowledge and inject new knowledge (instructions) into the frozen LLMs.

Experimental results on three datasets demonstrate that our approach achieves superior portability of edits while maintaining stability in other essential properties. The main contributions of our work are as follows:

1) We propose a novel method **InstructEd**, enhancing the capability of LLMs to effectively utilize knowledge by adaptively modifying the attention module in LLMs, in both modification based and memory based settings. This is different from previous model editing efforts focusing on the FFN module.

2) Experimental on three datasets demonstrate that InstructEd exhibits excellent ability to understand and use instructions, achieving SOTA editing performance, achieving 10% (5%) improvement in one-hop (multi-hop) model editing over previous SOTA.

3) We investigated the feasibility of retrieval augmentation on model editing with hops and found that it can help improve the locality of model editing while slightly decrease the editing performance with hops.

2 Realted work

2.1 Model Editing

Model Editing emerges as a viable strategy for precisely updating LLMs without the expensive resources (Wang et al., 2023; Zhang et al., 2024). Recent studies on model editing can be divided into two categories based on whether the original parameters of the edited model are modified (Yao et al., 2023). One category involves directly modifying model parameters, exemplified by hypernetworks (De Cao et al., 2021; Mitchell et al., 2022a; Han et al., 2023a) and located-and-edit techniques (Meng et al., 2022, 2023a; Li et al., 2023b). The other incorporates additional modules to LLMs (Mitchell et al., 2022b; Han et al., 2023b; Hartvigsen et al., 2022). Recently, in-context-based editing methods have gained attention. These methods guide the model in learning knowledge updates by providing editing examples (Zheng et al., 2023; Yu et al., 2023; Gu et al., 2023), offering greater flexibility. Nonetheless, this approach relies on the context selection and the scale of input prompt data. Additionally, PMET improved editing performance by modifying attention mechanisms. Inspired by this, we believe that attention, as an interaction module, can better facilitate the interaction between instructions and knowledge. Building on these concepts, *we investigate how to guide the model in learning knowledge with minimal prompts and present an editing model grounded in instruction learning*.

2.2 Instruction Tuning

Instruction tuning of language models has demonstrated its capability to enhance model generalization across unseen tasks, leveraging in-context learning with just a few examples (Zhang et al., 2023; Gupta et al., 2023; Li et al., 2023a). Recent works have leveraged instruction-driven fine-tuning methods, improving performance on previously unseen tasks through supervised fine-tuning on a limited number of examples (Li et al., 2023a; Ye et al., 2023). Furthermore, researchers have applied instruction tuning methods to image editing. They discern that the intent in the instructions successfully facilitates adaptive modifications to image content (Brooks et al., 2023; Liu et al., 2023a,b). In this work, we leverage the transfer ability of instruction to guide the model to learn how to utilize instructional knowledge adaptively, achieving generalization and portability in editing.

2.3 Prompt Tuning

Prompt (prefix) tuning (Li and Liang, 2021; Liu et al., 2022) is an efficient way to fine-tune LLMs. prompt refers to a text or instruction that guides the model in generating specific output types commonly applied in classification and question-answering tasks (Chan et al., 2020; Pei et al., 2023; Huang et al., 2023). On the other hand, prefix refers to specific markers or sequences of words added before the input text to alter or control the model's behavior, with many scholars recently applying it in Controllable text generation (Li and Liang, 2021; Lester et al., 2021; Meng et al., 2023b). This paper explores understanding instructions and utilizing and editing knowledge through fine-tuning with inserted prefixes while freezing LLMs parameters. More discussion can be found in Appendix D.6.

3 The Proposed Method

3.1 Problem Formulation

We formally define the Model Editing (ME) task. In particular, we denote LLMs as f and a fact to be edited as a triple $e = (s, r, o)$, consisting of a subject (s), a relation (r), and an object (o). To convert the triple e into natural language, we employ the prompt template $t_r(\cdot)$ [3]. The editing task aims to update the object o to o^* in e that shares the same subject and relation in LLMs f. This can be formally expressed as: $f(t_r(e)) = o \rightarrow f(t_r(e); \varepsilon) = o^*$, where ε represents the editor.

Simultaneously, the editing model needs to handle both in-scope data D_{in}, and out-scope data D_{out}. The in-scope data includes inputs generated by different templates t_r for the same triple e. Out-scope data refers to entities unrelated to e, such as any entity distinct from either the subject s or the relation r.

Following previous work (Yu et al., 2023; Gu et al., 2023), the post-edit model $f(\cdot; \varepsilon)$ is designed to satisfy the following properties: (1) **Reliability**: Ensuring that $f(t_r(e); \varepsilon)$ can output the target answer o^*. (2) **Generalization**: Ensuring that $f(t_r(e_i); \varepsilon)$ can generate the target answer o^* for $e_i \in D_{in}$, where e_i share the same subject and relation with e. (3) **Locality**: Ensuring that $f(t_r(e_o); \varepsilon)$ can generate the original answer o for $e_o \in D_{out}$. (4) **Portability**: Ensuring that the model not only updates the current editing but also maintains consistency with other facts e_p relevant to the current edits e.

Specifically, there are three aspects of Portability (Yao et al., 2023). (1) Hop-editing: For an n-hop question constructed by a chain of facts $Q = \{(s_1, r_1, o_1), ..., (s_n, r_n, o_n)\}$, updating any object $o_i \in Q$ should result in the post-edit model outputting new results based on the new object o_i^* within the chain $Q* = \{(s_1, r_1, o_1), (s_i, r_i, o_i^*), ..., (s_n^*, r_n, o_n^*)\}$.[4] (2) Subject Replacement: Replace the subject s in the edited triple (s, r, o) with its alias s', and the post-edited model should maintain the answer o,

[3]For instance, given the triple (Messi, play for, Inter Miami CF), the template t_r is "*subject* is playing for *object*", resulting in the sentence: Messi is playing for ___.

[4]For example, with a two-hop chain: {(Messi, play for, PSG), (PSG, located in, Paris)}, updating the object "PSG $\rightarrow$ Inter Miami CF" should lead to the answer for the question "What city is Messi's team in?" being "Miami" not "Paris," as the new triple is (Inter Miami CF, located in, Miami).

such as (Messi,playing for, *) and (Leo Messi, playing for, *). (3) Reversed Relation: When the target of a subject and relation is edited, the attribute of the target entity should also change, such as for an edit "Who is Mike's Father? $Bob \rightarrow Tom$", the answer for "Who is the son of Tom?" should update to "Mike".

I	Messi is playing for Inter Miami.
Q	What city is Messi's team in?
T	Pairs
L	Miami
C	**C-Edit:** (Messi, play for, Inter Miami CF) (Inter Miami CF, located in, Miami) **C-Trues:** (Messi, play for, PSG) (PSG, located in, Pairs)

Table 1: Example from the Instruction Editing Dataset: Featuring an Instruction *I*, a Question *Q*, Pre- and Post-Editing Answers *T*, *L*, and Chains of Facts Pre- and Post-Editing *C-Trues*, *C-Edit*.

3.2 Instruction Prompt for editing

To train the InstructEd, we first generate instructions based on the existing editing dataset CounterFact (Meng et al., 2022) and ZsRE (Levy et al., 2017). Subsequently, we design instruction templates to construct input data.

3.2.1 Generating instructions

For CounterFact (Meng et al., 2022), the original dataset provides triple information with modified knowledge and offers prompt templates to convert these triples into natural language. Therefore, we directly employ the natural language of the modified knowledge as instructions for the current data. As illustrated in Table 1, consider the edited triple in C-Edit: "(Messi, play for, Inter Miami CF)". Here, we utilize the natural language expression "Messi is playing for Inter Miami CF" as the instruction for the current editing instance.

For ZsRE (Levy et al., 2017), the original dataset lacks suitable prompts for transforming triplet information into natural language. To address this, we concatenate the question and the target answer to create the instruction. For instance, the instruction for the edited triple "(Messi, play for, Inter Miami CF)" is "Which team does Messi play for? Inter Miami CF".

3.2.2 Construct Input

For each modified data, we concatenate the instructions with four distinct types of questions, forming the input for the model and subsequently assessing the editing model based on four metrics. The input template is as follows:

Instructions: {Instructions} $\backslash n$ Input:{Input}.

We used the four types of input mentioned above to train the editor. To adapt to real-world editing scenarios, we evaluate the editing based on whether the instructions for input ('ED-Ins' in Figure 2) are known or unknown, referring to both modification based and memory based settings. Appendix B shows the input cases and details of two editing scenarios.

3.3 InstructEd

We employ instruction editing data to train an InstructEd to learn to utilize edits based on instructions. Figure 2 shows that we use frozen LLMs with L layers. The input to the LLMs consists of three components: (1) The questions in natural language Q with Q_n tokens. (2) The instructions in natural language I with I_n tokens. (3) A set of learnable adaption prompts P for instruction-following tuning. The prompts at layer l in the LLMs are represented as P_l, with a length of n_l and a dimension of h_p which is equal to the LLMs' hidden dimension. Note that the prompts inserted into N layers of the LLMs are different ($N < L$).

The input is formulated as follows:

$$[P_l; I; Q] \in \mathbb{R}^{(n_l+Q_n+I_n)*h} \quad (1)$$

The instruction I serves as a prompt for LLMs when answering question Q. However, due to the knowledge conflict, LLMs may struggle to reconcile non-parametric instruction knowledge with their own parameterized knowledge. To address this, we introduce trainable prompts P_l as a prefix, which offers several advantages:

1) Prompts facilitate interactions between question Q and instruction I, enabling the model to consider more contextual information during generation. 2) Prompts assist in handling different types of instructions. For example, for Locality data, P_l should guide the model to rely less on the information within the instruction I. We have designed the following strategies to achieve the above objectives, as illustrated in Figure 2.

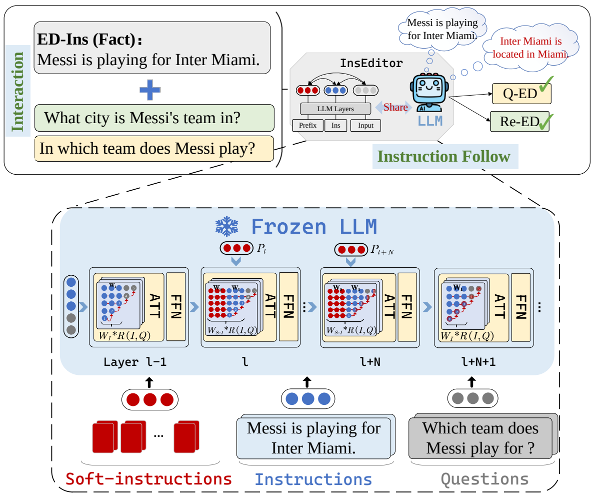
Interaction
ED-Ins (Fact): Messi is playing for Inter Miami.
What city is Messi's team in?
In which team does Messi play?
InsEditor
LLM Layers
Prefix
Ins
Input
Share
LLM
Messi is playing for Inter Miami.
Inter Miami is located in Miami.
Q-ED
Re-ED
Instruction Follow
Frozen LLM
P_l
P_{l+N}
$W_I*R(I,Q)$
$W_{S:I}*R(I,Q)$
ATT
FFN
Layer l-1
l
l+N
l+N+1
Messi is playing for Inter Miami.
Which team does Messi play for ?
Soft-instructions
Instructions
Questions

Figure 2: Overview of InstructEd. We achieve interaction between instructions and questions by inserting N trainable prefixes, thereby endowing the LLM with the ability to follow instructions. To mitigate the impact of irrelevant instructions on the input, we re-weight the attention for each layer based on the relevance scores between instructions and input.

Insert prompts into the middle layer of the LLMs. In autoregressive models, tokens draw information solely from the above tokens (Meng et al., 2022, 2023a):

$$h_l = h_{l-1} + ATT_{l-1} + FFN_{l-1}, \tag{2}$$

where ATT and FFN refer to the Attention and Feed-Forward module in transformer architecture. We perform prompt training in the L intermediate layers of LLMs ($L = N/3$), leveraging higher layers to capture semantic information representation between Q and I.

Editing the attention for interaction between Q and I. As the attention module serves as an interactive module in the transformer, it is crucial to leverage attention to facilitate interaction among Q, I, and P_l. Specifically, as shown in Figure 2, we modify the attention from layer l to $l + N$. In the attention mechanism of the preceding l layers, after generating M tokens ($M < (Q_n + I_n)$), the attention score of $(M + 1)$-th token t at layer $l - 1$ is calculated by several linear projections $Linear_{q,k,v,o}(\cdot)$:

$$\text{Query}_{l-1} = Linear_q([M, t_{l-1}]), \tag{3}$$
$$\text{Key}_{l-1} = Linear_k([M, t_{l-1}]), \tag{4}$$
$$\text{Value}_{l-1} = Linear_v([M, t_{l-1}]), \tag{5}$$
$$S_{l-1} = \text{Query}_{l-1}\text{Key}_{l-1}^T/\sqrt{C}, \tag{6}$$

where $\text{Query}_{l-1}, \text{Key}_{l-1}, \text{Value}_{l-1} \in \mathbb{R}^{(M+1)*h}$, $S_{l-1} \in \mathbb{R}^{1*(M+1)}$, finally the attention output of $l - 1$ layer is:

$$t^o_{l-1} = Linear_o(S_{l-1}\text{Value}_{l-1}). \tag{7}$$

At layer l, we incorporate the prompts P_l into the attention mechanism. Subsequently, we concatenate the prompt with the $(M + 1)$-th token t and compute the new representations for Query_l, Key_l, and Value_l at layer l:

$$\text{Query}_l = Linear_q([M; t_l]), \tag{8}$$
$$\text{Key}_l = Linear_q([P_l; M; t_l]), \tag{9}$$
$$\text{Value}_l = Linear_q([P_l; M; t_l]), \tag{10}$$

where $\text{Query}_l \in \mathbb{R}^{(M+1)*h}$, and $\text{Key}_l, \text{Value}_l \in \mathbb{R}^{(n_l+M+1)*h}$. The attention score at layer l is:

$$S_l = \text{Query}_l\text{Key}_l^T/\sqrt{C}, \tag{11}$$

where $S_l \in \mathbb{R}^{1*(n_l+M+1)}$, and S_l can be reformulated in three parts:

$$S_l = [S_l^{n_l}; S_l^{I_n}; S_l^{Q_n}]^T, \tag{12}$$

where n_l, I_n and Q_n are the prefix, instruction, and question length respectively. This implies that the attention score comprises both the instruction prompt p_l, the tokens for instructions I and the question Q. Consequently, t_l can learn information from all its preceding tokens and the prompt p_l, fostering interaction between the question Q and the instructions I through p_l:

$$t^o_l = Linear_o(S_l\text{Value}_l). \tag{13}$$

Re-weight the attention score. For the editor to learn how to process the Locality instruction, we mitigate the impact of irrelevant instructions on the output by adjusting the attention weights. More precisely, we utilize the similarity scores between instructions and questions:

$$\text{sim} = \text{Cos}(\text{Enc}(Q), \text{Enc}(I)), \tag{14}$$

where we employ the Enc$(\cdot)$ to encode both question Q and instruction I, utilizing the contriever encoder as described in (Izacard et al., 2022). Subsequently, we utilize the similarity score sim to re-weight the attention score:

$$S = \begin{cases} [\text{sim} * (S_i^{n_l}; S_i^{I_n}); S_i^{Q_n}]^T & i \in [l, l+N], \\ [\text{sim} * S_i^{I_n}; S_i^{Q_n}]^T & Else, \end{cases} \tag{15}$$

where $S_i^{n_l}$, $S_i^{I_n}$, and $S_i^{Q_n}$ represent the attention scores for prompts, instruction, and questions, respectively.

Model	Type	Method	Rel	Gen	Loc	Por$_{hop}$	Score	H	Rel	Gen	Loc	Por$_{hop}$	Score	H
			CounterFact (%)						ZsRE (%)					
LLaMA V1 (7B)	M	**FT**	46.33	75.88	33.75	47.01	50.74	46.69	59.38	62.15	97.18	54.35	68.27	64.91
		MEMIT	99.6	82.2	94.42	51.2	81.86	76.44	84.59	80.48	99.53	52.59	79.3	75.03
		ROME	99.43	79.08	**95.54**	51.45	81.38	76.04	83.88	78.62	99.5	54.11	79.03	75.23
		LoRA	**100**	55.30	65.26	48.68	67.31	62.55	66.01	63.36	96.5	54.69	70.14	67.14
	P	**RASE**	93.40	93.40	80.89	1.60	67.32	6.07	-	-	-	-	-	-
		IKE	99.81	82.38	46.86	49.82	69.72	62.92	**99.76**	**99.62**	82.97	64.76	**86.78**	84.11
		SERAC	76.58	26.20	58.00	18.00	44.69	32.25	98.81	80.75	**100**	11.11	72.67	32.65
		InstructEd	98.01	**97.59**	76.46	**62.43**	**83.62**	**80.73**	98.16	98.24	80.73	**67.72**	86.21	**84.17**
		+ Retrieval	97.81	88.91	94.32	61.60	***85.66***	***82.80***	96.84	83.30	98.66	64.92	85.93	83.56
LLaMA V2 (7B)	M	**FT**	11.47	14.11	26.11	45.83	24.38	18.33	48.26	48.54	94.32	51.64	60.69	56.11
		MEMIT	99.52	82.4	94.76	51.2	81.97	76.53	66.13	65.33	99.56	54.46	71.37	67.99
		PMET	20.16	17.10	87.37	12.53	34.29	20.06	43.26	41.92	94.64	55.94	58.94	53.04
		LoRA	**99.84**	60.84	64.09	52.10	69.21	65.30	70.37	67.15	91.76	54.51	70.95	68.56
		ROME	15.68	22.87	97.61	47.97	46.03	28.86	64.13	63.29	99.54	52.77	69.93	66.24
	P	**SERAC**	98.53	11.66	**100**	49.75	64.99	31.74	96.57	79.31	**100**	10.55	71.61	31.31
		RASE	72.64	71.30	80.89	5.81	57.66	18.84	-	-	-	-	-	-
		IKE	99.63	85.77	49.64	54.11	72.29	66.31	**100**	**99.71**	82.1	65.52	**86.83**	**84.26**
		InstructEd	98.77	**98.32**	77.74	**65.85**	**85.17**	**82.74**	98.70	98.29	75.57	**70.02**	85.65	83.65
		+ Retrieval	98.07	90.72	94.38	65.21	***87.09***	***84.83***	96.93	84.48	98.40	67.38	86.80	***84.82***

Table 2: Results on CounterFact and ZsRE. M represents methods for modifying LLM parameters; P represents methods for preserving LLM parameters. '+Retrieval' means we use the retrieval model msmarco, '-' refers to the results that the methods fail to edit LLMs. We use the evaluation metrics in Sec.4.1.3 to assess the editor.

4 Experiments

4.1 Experimental Setup

4.1.1 Dataset

We conduct comprehensive experiments on three widely recognized editing datasets: CounterFact (CT), ZsRE for one-hop evaluation, introduced in Yao et al. (2023), and MQUAKE for multi-hop evaluation, as described in (Zhong et al., 2023). We adopt the same data split of training and testing following Yao et al. (2023) and Zhong et al. (2023). The detail of the data is provided in Appendix A.

4.1.2 Setup

We conduct experiments on three autoregressive LLM: LLaMA1(7B,13B), LLaMA2(7B,13B) and GPT2-XL (1.5B) models. We compare our method against three classes of editors, encompassing a total of 11 models: 1) Preserving LLMs Parameters as our baselines: SERAC(Mitchell et al., 2022b), IKE(Zheng et al., 2023), RASE(Han et al., 2023b), GRACE(Hartvigsen et al., 2022). 2) Modifying LLMs parameters: FT(Zhu et al., 2020a), LoRA, ROME(Meng et al., 2022), MEMIT(Meng et al., 2023a), PMET(Li et al., 2023b). 3) Retrieval-based method for Multi-hop editing method: Mello (Zhong et al., 2023) and PokeMQA (Gu et al., 2023). To train InstructEd, we sample 10000 training instances from CT, which exclusively include the recall edits, and 2200 training instances from MQUAKE-CF-3k, which contains 2,3,4-hop edits in the following distribution 1000:910:290. The training process spans 10 epochs, with each batch comprising 6 instances of edits and 6 instances of unmodified data. For additional settings, please refer to Appendix B.

4.1.3 Metrics

We use Reliability (Rel), Generalization (Gen), and Locality (Loc) for single editing evaluation, Por$_{hop}$ for one-hop evaluation, Por$_{N_{hop}}$ and Hop-Acc for multi-hop editing evaluation, respectively Yao et al. (2023); Gu et al. (2023). Note that Por$_{N_{hop}}$ refers to the results on multi-hop question answering after modifying one fact, while Por$_{hop}$ refers to the results on one-hop question answering after modifying one fact. Furthermore, to examine the generalization-specificity trade-off, we present the mean and harmonic mean scores of Rel, Gen, Loc, and Por$_{hop}$ as *Score* and *Harmonic Score (H)*. Details on metrics can be found in Appendix C.

4.2 Main Results

Results on CounterFact. The edits in CounterFact are the facts that do not exist in the real world, which can ensure the manipulated/modified data have not occurred during the training of LLMs. Consequently, evaluations on CounterFact can better measure the editor's editing capabilities.

InstructEd can better understand and utilize edited facts, outperforming existing models of varying scales, showcasing a remarkable enhancement

Method	Por$_{N_{hop}}$	Hop-Acc
FT	47.32	-
FT-Cot	56.48	33.89
Rome-CoT	28.96	-
Rome	24.89	17.99
MEMIT-CoT	36.88	-
MEMIT	30.89	23.98
PokeMQA	75.43	60.44
InstructEd	**80.62**	**83.35**

Table 3: Results on MQuAKE-T with LLaMA-2(7B). Por$_{N_{hop}}$ means the accuracy of multi-hop editing, and Hop-Acc is the mean of Por$_{N_{hop}}$ for each sub-question in a multi-hop question.

in model editing with hops. As shown in Table 2, our approach demonstrates significant advantages in both Gen and Por$_{hop}$. Compared with methods directly updating model parameters (M), our method flexibly updates various types of editing data and applies the updated knowledge in the model's inference, avoiding issues of excessive memorization caused by direct parameter updates. Furthermore, in contrast to the memory-based method (P), we learn and leverage instructional knowledge to prevent the accumulation of errors when solving multi-hop editing while modifying attention parameters to alleviate knowledge conflicts in the model effectively. Results on LLaMA (13B) and more analysis are in Appendix D.1.

Results on ZsRE. In contrast to CounterFact, the ZsRE is specifically designed to rectify LLM errors. Consequently, as shown in Tabel 2, there is a notable enhancement in retaining LLM parameters due to incorporating additional modules or memory to constrain the output of LLM, especially IKE, which encourages model editing by providing external editing samples. However, *InstructEd can still demonstrate greater generalization and robustness on ZsRE, showing good performance even when trained only on CounterFact and tested on ZsRE, further indicating that enhancing a model's ability to utilize knowledge can fundamentally improve model editing with hops.*

Results on MQuAKE. Table 3 shows the results on MQuAKE-T (Zhong et al., 2023). In multi-hop question answering, we retrieve instructions for each multi-hop question through search. For example, for an N-hop question, we retrieve N facts (the top N retrieved facts) as instructions. Our method demonstrates excellence in both Por$_{N_{hop}}$ and Hop-Acc, indicating that *InstructEd can flexibly utilize instructional knowledge, providing advantages in multi-hop editing.* Additionally, the accuracy of retrieval will affect the performance of our model. More discussion can be found in Appendix 4.5.

4.3 Further Analyses of InstructEd

Results for portability. InstructEd has not only achieved an advantage on the Por$_{hop}$ but obtained excellent results in the other two aspects of portability: Subject Replace (Rep) and Reversed Relation (Rev). As shown in Table 9, the InstructEd could gain great performance on Rev. However, for Rep, InstructEd is not ideal at the beginning. After we add additional instruction that never occurs during training and is unrelated to the edits (s is also known as $s^{'}$), the performance increases significantly, demonstrating its robust understanding and utilization of instructions. Details can be found in Appendix D.2.

Sequential editing Results. InstructEd is a plug-and-play model that can address both batch editing and sequential editing. The results in Table 2 are obtained by modifying one edit at a time. InstructEd uses additional prefixes to achieve editing, allowing it to handle varying amounts of editing data flexibly and perform sequential editing while maintaining the performance. Therefore, whether editing a batch of data simultaneously or sequentially, InstructEd's performance remains unaffected. In contrast, performance declines with the direct modification of model parameters as the number of sequential edited data increases. Details can be found in Appendix D.3.

Results of Different Model Scales. To validate the generality of our approach, we also tested it on models of varying scales, including GPT2-XL (1.5B) and LLaMA2 (13B). As shown in Appendix D.1. We observed that our method can be flexibly applied across models of different scales while maintaining consistent performance.

Efficiency of InstructEd. We utilized the A100 (40G) GPU to train InstructEd on LLaMA (7B) which took 7 hours. The inference speed is consistent with the original LLaMA model, averaging around 0.5 seconds per edit. Note that we assess the modification of each data point from four data, so the inference speed for each data point is 0.5/4 seconds. Regarding storage, InstructEd requires approximately 0.8M of additional storage space.

Model	Method	Rel	Gen	Loc	Por	Score	H	Rel	Gen	Loc	Por	Score	H
		CounterFact (%)						ZsRE (%)					
LLama1	InstructEd	98.01	97.59	**76.46**	62.43	**83.62**	**80.73**	98.16	98.24	**80.73**	67.72	86.21	84.17
	w/o re-weight	99.52	**100.00**	24.73	**74.62**	74.72	54.14	99.38	97.02	61.80	69.87	82.02	78.64
	w L:20-30	98.40	89.19	44.70	50.76	70.76	63.05	**98.97**	97.44	33.15	65.02	73.64	60.69
	w FFN	**99.60**	98.74	35.79	69.82	75.99	64.07	99.73	**99.18**	56.64	**70.13**	81.42	76.88
LLama2	InstructEd	**98.77**	98.32	**77.74**	**65.85**	**85.17**	**82.74**	**98.70**	**98.29**	75.57	70.02	**85.65**	**83.65**
	w/o re-weight	95.34	**99.66**	35.26	69.41	74.92	63.20	98.85	97.65	56.08	**70.19**	80.69	76.28
	w L:20-30	98.04	92.75	55.09	55.82	75.42	70.11	97.03	95.44	33.24	66.16	72.96	60.62
	w FFN	98.26	99.56	34.16	58.19	72.54	59.99	93.73	93.28	36.28	46.65	67.49	56.83

Table 4: Results of ablation experiments. *re-weight* indicates whether attention was re-weighted; *L:20-30* represents the results of applying the Editor on layers 20-30; *FFN* represents the results of editing the FFN.

4.4 Ablation experiment

In this section, we analyze the impact of different modules on InstructEd's editing performance. Results are shown in Table 4.

"w/o re-weight" remove sim in Eq.15 and use the original attention weights, significantly reducing the locality. Indicate that InstructEd, when equipped with re-weighting, can effectively learn how to leverage knowledge flexibility.

"L:20-30" shows an additional ablation editing deep layer (20-30) in LLMs, rather than middle layer (10-20). The results indicate that editing deeper layers impedes the model's effective use of knowledge (lower propagation scores) and leads to more pronounced negative impacts on the model (lower Locality scores). See Appendix D.4 for ablation experiments on editing layers.

"FFN" shows an additional ablation editing FFN weights rather than the attention module. Introducing a Adapter module to each FFN at layers 10-20 in LLMs reveals that editing the FFN remains a viable alternative, yielding a higher propagation score while maintaining other editing metrics. However, a lower Locality implies that imprecise editing of the FFN could lead to substantial adverse consequences. This paper validates that editing attention can achieve comparable or even superior results to editing FFN. More Discussion about FFN can be found in Appendix D.5.

4.5 Results with Retrieval augment

The results (+Retrieval) in Table 2 indicate that retrieval augmented can improve the performance of InstructEd especially on localization performance.

To assess the effectiveness compared to using only retrieval-augmented methods, we conducted a comparison with the Self-RAG (Asai et al., 2023). The results in Table 5 show that the retrieval-augmented demonstrates great performance on ZsRE regarding localization. It can accurately determine whether a query is relevant to the input. However, the high correlation between the location data and input in CT makes the performance on CT is mediocre. For example, both edit input and loc input involve queries about the team to which a player belongs, leading to errors in judging relevance. The retrieval-augmented model has shown promise regarding editing, as pointed out by Pinter and Elhadad (2023), directly modifying the model can result in unknown and uncontrollable impacts on the original LLMs, using retrieval-augmented methods to achieve the model editing is safer and interpretable (Gupta et al., 2024; Gu et al., 2024).

	Rel	Gen	Loc	Por	Score	H
ZsRE	77.05	73.67	96.53	29.51	69.19	56.50
CT	77.30	64.21	36.08	35.31	53.22	47.31

Table 5: Results on Self-RAG

5 Conclusion

In this paper, we have presented an instruction tuning based method InstructED for modeling editing with hops. We validate the effectiveness of InstructEd on varying scales of LLMs under three datasets, demonstrating InstructEd's excellent generalization in both modification based and memory based settings. Our experiments indicate that, unlike previous editing methods, modifying attention parameters can model editing. Last but not least, we investigated the feasibility of retrieval augmentation on model editing and found that the retrieval-augmented methods can enhance the locality of model editing while slightly decrease the editing performance with hops.

6 Limitations

In this paper, we introduce an effective method for model editing with hops. Although we achieve good results, there are still limitations in the following aspects: 1) We only focus on model editing with hops. In the future, we plan to look into model editing with more complex reasoning involved. 2) We utilize the retrieval augmented model to enhance InstructEd and verify the performance of the retrieval model Self-RAG (Asai et al., 2023) in editing tasks. However, novel integration of retrieval-augmented methods with model editing remains a challenge that needs further exploration in the future.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. 2023. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402.

Alvin Chan, Yew-Soon Ong, Bill Pung, Aston Zhang, and Jie Fu. 2020. Cocon: A self-supervised approach for controlled text generation. In *International Conference on Learning Representations*.

Zhuo Chen, Jiaoyan Chen, Yuxia Geng, Jeff Z. Pan, Zonggang Yuan, and Huajun Chen. 2021. Zero-shot Visual Question Answering Using Knowledge Graph. In *Proc. of ISWC2021*.

Zhuo Chen, Yufeng Huang, Jiaoyan Chen, Yuxia Geng, Yin Fang, Jeff Z. Pan, Ningyu Zhang, and Wen Zhang. 2022. Lako: Knowledge-driven Visual Question Answering via Late Knowledge-to-text Injection. In *Proc. of IJCKG2022*, pages 20–29.

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

Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. 2023. Pokemqa: Programmable knowledge editing for multi-hop question answering. *arXiv preprint arXiv:2312.15194*.

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

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. 2024. Model editing at scale leads to gradual and catastrophic forgetting.

Himanshu Gupta, Saurabh Arjun Sawant, Swaroop Mishra, Mutsumi Nakamura, Arindam Mitra, Santosh Mashetty, and Chitta Baral. 2023. Instruction tuned models are quick learners. *arXiv preprint arXiv:2306.05539*.

Xiaoqi Han, Ru Li, Xiaoli Li, and Jeff Z Pan. 2023a. A divide and conquer framework for knowledge editing. *Knowledge-Based Systems*, 279:110826.

XiaoQi Han, Ru Li, Hongye Tan, Wang Yuanlong, Qinghua Chai, and Jeff Z. Pan. 2023b. Improving sequential model editing with fact retrieval. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2022. Aging with grace: Lifelong model editing with discrete key-value adaptors. In *NeurIPS 2022 Workshop on Robustness in Sequence Modeling*.

Jie He, Víctor Gutiérrez-Basulto, and Jeff Z. Pan. 2023. BUCA: A Binary Classification Approach to Unsupervised Commonsense Question Answering. In *Proc. of ACL2023*.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. 2023. Detecting edit failures in large language models: An improved specificity benchmark. *arXiv preprint arXiv:2305.17553*.

Nan Hu, Yike Wu, Guilin Qi, Dehai Min, Jiaoyan Chen, Jeff Z Pan, and Zafar Ali. 2023. An Empirical Study of Pre-trained Language Models in Simple Knowledge Graph Question Answering. In *Journal of World Wide Web*, pages 1–32.

Xuancheng Huang, Zijun Liu, Peng Li, Tao Li, Maosong Sun, and Yang Liu. 2023. An extensible plug-and-play method for multi-aspect controllable text generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15233–15256.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

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

Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Fanyi Pu, Jingkang Yang, Chunyuan Li, and Ziwei Liu. 2023a. Mimic-it: Multi-modal in-context instruction tuning. *arXiv preprint arXiv:2306.05425*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2023b. Pmet: Precise model editing in a transformer. *arXiv preprint arXiv:2308.08742*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023a. Improved baselines with visual instruction tuning. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023b. Visual instruction tuning. In *37th Conference on Neural Information Processing Systems NeurIPS 2023*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

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

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023a. Mass editing memory in a transformer. In *International Conference on Machine Learning*.

Yu Meng, Martin Michalski, Jiaxin Huang, Yu Zhang, Tarek Abdelzaher, and Jiawei Han. 2023b. Tuning language models as training data generators for augmentation-enhanced few-shot learning. In *International Conference on Machine Learning*, pages 24457–24477. PMLR.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

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

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

Jonathan Pei, Kevin Yang, and Dan Klein. 2023. Preadd: Prefix-adaptive decoding for controlled text generation. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Yuval Pinter and Michael Elhadad. 2023. Emptying the ocean with a spoon: Should we edit models? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15164–15172.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, et al. 2023. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics.

Seonghyeon Ye, Hyeonbin Hwang, Sohee Yang, Hyeongu Yun, Yireun Kim, and Minjoon Seo. 2023. Investigating the effectiveness of task-agnostic prefix prompt for instruction following. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2023. Melo: Enhancing model editing with neuron-indexed dynamic lora. *arXiv preprint arXiv:2312.11795*.

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

Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. 2023. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*.

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

Zexuan Zhong, Zhengxuan Wu, Christopher Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15686–15702, Singapore. Association for Computational Linguistics.

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

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

A Dataset

We conduct comprehensive experiments on three widely recognized editing datasets: CounterFact (CT), ZsRE for portability evaluation, introduced in Yao et al. (2023), and MQUAKE for Multi-hop evaluation, as described in (Zhong et al., 2023), the dataset has been divided into MQ-CF (for counterfactual edits) and MQ-T (for real-world fact updates) two sub-datasets. The detail of the data is provided in Table 6.

Both four datasets include single-hop and multi-hop editing data, while CT and ZsRE only contain up to 2-hop question-answer pairs, and MQ contains up to 4-hop question-answer data. ‘Overlap’ indicates whether there is a final multi-hop question answer in the edited factual calm. For example, for a 2-hop question: "What city is Messi’s team in?" if the edit is "Messi is playing for Inter Miami." Then, the overlap is 0 because the model needs to know that "Inter Miami is located in Miami." But if the edit is "Inter Miami is located in Pairs," the final answer "Pairs" is present in the edits, so the overlap is 1.

Dataset	Edits	Hops	Total	Overlap
ZsRE	1	2	1037	3%
CT	1	2	1031	1%
MQ-CF	1,2,3,4	2,3,4	9218	68%
MQ-T	1	2,3,4	1868	99%

Table 6: Statistics of dataset.

B Experimental Details

Construct Input

To train the InstructEd, we use the following input template as the inputs:

Instructions: {Instructions} $\backslash n$ Input:{Input}.

For example, as shown in Table 1, the instruction for edits is *"Messi is playing for Inter Miami."*.

Accordingly, the Reliability input is *"Instructions: Messi is playing for Inter Miami. $\backslash n$ Input: Which team does Messi play for?"*.

The Generalization input is *"Instructions: Messi is playing for Inter Miami. $\backslash n$ Input: In which team does Messi play?"*.

The Locality input is *"Instructions: Messi is playing for Inter Miami. $\backslash n$ Input: When was the recent World Cup held?"*.

The Portability input is *"Instructions: Messi is playing for Inter Miami. $\backslash n$ Input: What city is Messi’s team in?*.

Evaluate Strategy

During the testing phase, two strategies emerge depending on whether the gold instruction information is known:

(1) When the gold instruction for each edit is known, we concatenate the instruction with the corresponding data for assessment. The instruction is combined with a randomly selected and unrelated data point before evaluation for locality data.

(2) When we do not know the gold instruction for each edit, we rely on retrieving relevant instructions from a pre-constructed instruction memory. If no instructions were found, it indicates the current data point does not necessitate modification.

Evaluate with Retrieval

When the gold instructions for each data are unknown, we use the retrieval model msmarco (Izacard et al., 2022) to retrieve instructions relevant to the current input from a pre-constructed instruction memory. If certain conditions are met, the data with the highest-ranked retrieval result is considered the current input instruction. Otherwise, the current data is deemed unnecessary to edit, and the original LLMs are used to calculate.

Specifically, we calculate the standard deviation of the scores for the top 5 ranked retrieval results. When the standard deviation is larger than 0.1, the similarity score of the highest-ranking result is significantly higher than the scores of the other 4 data. In this case, we consider the data with the highest ranking as the instruction for the input. Otherwise, the input does not contain a suitable instruction and does not require editing.

Experimental Setting

We follow the setting provided in Yao et al. (2023) and Han et al. (2023b) for the baseline methods. We training InstructEd on 2 A100 (40G) GPUs for 10 epochs. The warmup epochs, batch size, learning rate, and weight decay are set to 4, 6, 9e-3, and 2e-2, respectively. We add 5 prefixes per layer and edit 10 to 20 layers in LLM.

C Evaluation metrics

To evaluate the performance of a post-edit model, following Yao et al. (2023) and Gu et al. (2023), We use the following editing properties: Reliability,Generalization, and Locality for single editing evaluation. Portability, Por$_{N_{hop}}$, and Acc-Hop are used for multi-hop editing evaluation. We denote I as the indicator function, and the post-edit model is f_T.

Reliability is measured as the average accuracy on the edited dataset $(x_t, y_{x_t}) \in \mathbb{D}_{ed}$, T is the length of $\mathbb{D}_{ed}$:

$$Rel = \frac{1}{T}\sum_{t=0}^{T} I(f_T(x_t) = y_{x_t}). \tag{16}$$

Generalization is measured as the average accuracy on the equivalent neighbor of edit dataset $\mathbb{D}_{gen}$, we denote each edit case has N_t neighbors:

$$Gen = \frac{1}{TN_t}\sum_{t=0}^{T}\sum_{i=0}^{N_t} I(f_T(x_t^i) = y_{x_t^i}). \tag{17}$$

Locality is evaluated by the rate at which the post-edit model f_T's predictions are unchanged as the pre-edit model f_0 on $\mathbb{D}_{loc}$, L is the length of $\mathbb{D}_{loc}$:

$$Loc = \frac{1}{L}\sum_{(x,y)\in\mathbb{D}_{loc}}^{L} \frac{I(f_T(x) = y)}{I(f_0(x) = y)}. \tag{18}$$

For InstructEd, the y is the output of $f_0(\cdot)$ or the True label.

portability$_{hop}$ is measured as the average accuracy on the multi-hop questions of edit case on $\mathbb{D}_{por}$, note that each $x \in \mathbb{D}_{por}$ is related to an edit $x_t \in \mathbb{D}_{ed}$:

$$Por_{hop} = \frac{1}{T}\sum_{(x,y)\in\mathbb{D}_{por}}^{T} I(f_T(x) = y). \tag{19}$$

$Por_{N_{hop}}$ is measured as the average accuracy on the multi-hop questions of edit case on MQuAKE-T $\mathbb{D}_{mqt}$:

$$Por_{N_{hop}} = \frac{1}{T}\sum_{(x,y)\in\mathbb{D}_{mqt}}^{T} I(f_T(x) = y). \tag{20}$$

Hop-Acc is measured if the post-edit model can answer the sub-questions for multi-hop questions on MQuAKE-T $\mathbb{D}_{mqt}$, suppose each multi-question x can be decomposed to s sub-questions x_s:

$$\text{Hop-Acc} = \frac{1}{T * s}\sum_{(x,y)}^{T}\sum_{(x_i,y_i)}^{s} I(f_T(x_i) = y_i). \tag{21}$$

D Extended Discussion of Results

D.1 Analyse for Baselines

Tabel 7 and Table 8 shows the results for GPT2-XL (1.5B) and LLaMA2 (13B) on two datasets. Furthermore, we analyzed the results obtained by different methods in Table 2 to verify the effectiveness of InstructEd.

Fine-tuning (Zhu et al., 2020b), as the most direct editing method, does not show ideal results, resulting in lower local alignment scores while achieving lower editing capabilities. **LoRA** improves performance by training low-rank matrices to replace parameters in FFN compared to fine-tuning, but it also exhibits poor generalization. Both methods directly modify parameters in LLMs through gradient descent, neglecting the impact of parameter changes on other aspects of the model's performance. Additionally, editing performance cannot be guaranteed when there is limited editing data.

ROME and MEMIT (Meng et al., 2022, 2023a), as representatives of the located-and-edit methods, generally exhibit great and stable performance. However, as they directly modify parameters in LLMs, there are still unknown impacts on the model and the risk of overfitting caused by excessive memorization resulting from editing(Hoelscher-Obermaier et al., 2023).

RASE and SERAC (Han et al., 2023b; Mitchell et al., 2022b) utilize additional cache as retrieval memory, training other parameters and additional modules for editing. However, as the results indicate, both approaches perform poorly on portability. This is attributed to their focus on knowledge updates during additional training modules, neglecting how to use knowledge effectively.

IKE (Zheng et al., 2023), as the most competitive method, achieves good performance on real-world datasets ZsRE by providing additional editing examples to prompt the model to learn the current task format. However, the lower performance on CounterFACT reflects the significant knowledge conflict with the LLMs, leading to unsuccessful modifications. Additionally, the performance of IKE is influenced by the number of additional prompts, requiring the model to have the capability to handle long texts.

Our approach focuses on learning how to utilize knowledge for LLMs. We design instructions and guide the model through the interaction of instructions and questions, achieving comprehensive edit-

Model	**Type**	**Method**	**Rel**	**Gen**	**Loc**	**Por$_{hop}$**	**Score**	**H**	**Rel**	**Gen**	**Loc**	**Por$_{hop}$**	**Score**	**H**
			CounterFact (%)						**ZsRE (%)**					
GPT2 XL (1.5B)	M	**FT**	97.38	10.18	81.38	41.13	57.52	27.57	56	30.49	85.31	45.51	54.33	47.42
		MEMIT	80.89	46.17	**98.84**	43.02	67.23	59.36	67.09	52.82	**99.57**	47.96	66.86	61.79
		ROME	99.71	77.59	86.91	41.68	76.47	68.47	**99.92**	88.16	94.91	47.1	**82.52**	75.3
		LoRA	**100**	67.51	44.03	41.29	63.20	55.75	52.53	52.43	96.77	44.19	61.48	56.28
	P	**IKE**	99.42	68.77	41.61	42.86	63.17	55.58	99.82	95.52	73.7	53.26	80.58	**75.71**
		InstructEd	100	**99.51**	61.16	**61.31**	**80.49**	**75.88**	97.88	**97.19**	45.10	**57.07**	74.31	66.44
		+ Retrieval	98.20	91.35	91.45	60.68	***85.42***	***82.40***	95.90	78.37	97.55	54.44	81.56	***77.21***

Table 7: Results on CounterFact and ZsRE.

	Rel	Gen	Loc	Por	Score	H
CT	99.85	99.17	82.88	71.67	88.39	86.74
ZsRE	97.83	98.13	73.47	73.30	85.68	83.92

Table 8: Results on LLaMA2 13B.

ing performance in learning and utilizing knowledge. Furthermore, our approach maintains the parameters of LLMs, enabling application across various datasets while combining with retrieval-augmented methods to enhance performance.

		InsE_v1	InsE_v2	InsE_G
ZsRE	Rev	93.27	89.43	83.86
	Rep	60.69	67.42	68.08
	+ins	77.52	76.63	85.95
CT	Rep	66.43	66.23	66.5
	+ins	90.48	86.01	98.58

Table 9: Results for portability. InsE means our proposed InstructEd, v1 and v2 mean LLaMA1-7B and LLaMA2-7B, respectively, and G means GPT2-XL. '+ins' means we add "s is also known as $s^{'}$" into the instruction, s and $s^{'}$ is the subject and replaced-subject.

D.2 Results for portability

We use the dataset in Yao et al. (2023) to evaluate the results on Subject Replace (Rep) and Reversed Relation (Rev). As the results in Table 9 show. Regarding Reversed Relations, InstructEd can understand and reason based on current instructions for reversed relations. In terms of Subject Replace, InstructEd performs modestly without the use of instructions. However, its effectiveness improves when relevant instructions are provided, indicating that our approach can significantly enhance the model's comprehension and ability to use instructions.

		Rel	Gen	Loc	Por$_hop$	Score	H
CT	MEMIT$_{v1}$	72.00	53.33	30.00	22.83	44.54	36.44
	ROME$_{v1}$	99.50	78.50	90.50	3.17	67.92	11.45
	SERAC$_{v1}$	92.00	4.00	100.00	4.00	50.00	7.68
	InstructEd$_{v1}$	98.16	100.00	79.25	67.72	86.28	84.08
	MEMIT$_{v2}$	6.25	6.25	14.75	7.92	8.79	7.78
	ROME$_{v2}$	18.75	21.92	79.42	16.42	34.13	23.20
	SERAC$_{v2}$	100.00	11.17	100.00	12.33	55.88	20.98
	InstructEd$_{v2}$	99.00	96.00	73.50	62.27	82.69	79.71
ZsRE	MEMIT$_{v1}$	34.19	34.38	5.34	5.64	19.89	9.46
	ROME$_{v1}$	85.65	82.16	94.63	53.97	79.10	75.55
	GRACE$_{v1}$	33.99	33.66	100	51.13	54.70	45.10
	SERAC$_{v1}$	97.05	81.07	99.01	8.92	71.51	27.62
	InstructEd$_{v1}$	97.97	98.19	74.48	71.56	85.55	83.70
	MEMIT$_{v2}$	54.89	52.31	36.94	27.09	42.81	39.48
	ROME$_{v2}$	63.30	61.62	87.88	51.01	65.95	63.48
	GRACE$_{v2}$	38.48	37.27	100	58.29	58.51	50.02
	SERAC$_{v2}$	97.80	78.21	100.00	10.51	71.63	31.21
	InstructEd$_{v2}$	98.36	98.75	81.28	67.33	86.43	84.30

Table 10: Results for Sequential editing 100 edits. v1 and v2 means the LLaMA1(7B) and LLaMA2 (7B).

D.3 Sequential editing Results.

As shown in Table 10, after continuous editing of 100 edits, the overall performance of MEMIT significantly deteriorates, and other methods are also heavily impacted, especially in terms of the Por$_{hop}$ metric. This is attributed to the continuous parameter adjustments these methods make to the LLM, leading to an increasing difficulty in editing and gradual degradation of model performance. In contrast, by freezing the parameters of LLMs, SERAC experiences a slower performance decline but fails to achieve satisfactory results in GEN and Por$_{hop}$.

D.4 Ablation experiments on editing layers

The results of ablation experiments are shown in Figure 1. We tested the insertion of 5, 10, and 15 layers of prefixes into LLM. The results indicate that adding additional parameters beyond the 20th layer in LLM leads to an overall performance decrease compared to before the 20th layer, especially in terms of local performance. This suggests that modifying parameters in the higher layers of LLM is more likely to impact model performance.

When modifying parameters in the lower layers of LLM, the model exhibits lower success rates in

	Layer	Por_{hop}	Rel	Gen	Loc	Score
ZsRE	0:10	0.6619	0.9848	0.9832	0.6914	0.8303
	0:15	**0.7091**	0.9758	0.9706	**0.7615**	0.8543
	0:5	0.6957	0.9596	0.9597	0.7505	0.8414
	10:15	0.6853	**0.9916**	**0.9862**	0.7097	0.8432
	15:20	0.6723	0.9737	0.9787	0.6752	0.8250
	15:30	0.6756	0.9744	0.9711	0.6908	0.8280
	20:25	0.6682	0.9720	0.9638	0.6722	0.8190
	20:30	0.6605	0.9689	0.9564	0.6480	0.8084
	25:30	0.6895	0.9738	0.9585	0.7203	0.8355
	5:10	0.6786	0.9605	0.9791	0.5825	0.8002
	10:20	0.7023	0.9857	0.9833	0.7563	**0.8569**
CT	0:10	0.6769	**1.0000**	0.9768	0.7575	0.8528
	0:15	**0.7228**	0.9787	0.9807	**0.7919**	**0.8685**
	0:5	0.6705	0.9990	0.9540	0.7376	0.8403
	10:15	0.6485	0.9922	0.9871	0.7470	0.8437
	15:20	0.6334	0.9924	0.9793	0.7444	0.8374
	15:30	0.6088	0.9963	0.9848	0.7884	0.8445
	20:25	0.5918	0.9842	0.9176	0.7244	0.8045
	20:30	0.5774	0.9801	0.9026	0.7360	0.7990
	25:30	0.5767	0.9922	0.8804	0.7160	0.7913
	5:10	0.6505	0.9921	0.9713	0.7528	0.8417
	10:20	0.6548	0.9930	**0.9964**	0.7886	0.8582

Table 11: The ablation studies about patched layer. We use the LLaMA2-7B as the base model.

editing. However, when modifying intermediate layers, there is a noticeable improvement in overall performance (such as 0-15 and 10-20). In order to maximize editing efficiency, we chose to modify layers 10-20.

D.5 Discussion of the role of ATT and FFN in Model Editing

In the Large Language Model (LLM), there exists a significant amount of parameterized knowledge. The mechanism for storing knowledge in the LLM remains an open question. Some studies have experimentally verified that more knowledge is retained in the Feedforward Neural Network (FFN). Building on this finding, corresponding editing models have been proposed, yielding certain advantages by updating parameters in the FFN.

However, Attention serves as a crucial module in the Transformer, and its role in the knowledge storage process still warrants investigation. Previous research conducted through ablation analysis found that Attention has a relatively minor impact on editing and that achieving knowledge editing through parameter updates in Attention is challenging. This perspective is one-sided, particularly given that Attention can equip the Transformer with potent interactive capabilities, making it a vital aspect in model editing. Furthermore, precise knowledge updates can be achieved by controlling the Query, Key, and Value (QKV) components within Attention. A deeper understanding of Attention in the future can facilitate the proposal of more efficient and accurate model editing methods.

D.6 Discussion of the prompt-tuning

Our approach achieves understanding and utilization of instructions by inserting additionally trained prompts into different layers of attention. As shown in Table 12, when not using prefix and only applying editing in the form of instructions to the original LLMs, the base model performs well on "Rel", indicating that instructional knowledge can provide the model with good factual support, and the model can extract knowledge from it. However, the lower performance on "Gen", "Loc" and "Por" reflects the inability of the base model to transfer new knowledge to relevant content, which means they fail to understand and utilize the knowledge.

In addition, there are many other efficient fine-tuning methods such as Adapter, LoRA, Prompt, etc. Besides utilizing prefixes, we compared with Adapter ("w FFN" in Table 4) and LoRA ("LoRA" in Table 2), respectively.

Overall, while LoRA and adapters can improve the efficiency of fine-tuned models, they add the incremental change Δh, obtained from the feed-forward layer with the original h from the previous layer, the adapter or LoRA modifies the hidden representation calculated by the pre-trained model $h = h + \Delta h$, which is lack interaction with the information in the LLM. And the prefix requires fewer parameters. They can interact with the context in the LLM by adding the trainable prefix, which enables more efficient guidance for the model to perform corresponding reasoning and computations based on instruction knowledge. As shown in Table.12

Model	Setting	Rel	Gen	Loc	Por_{hop}	Score
V1	InstructEd	0.9791	0.9714	0.7254	0.6210	0.8242
	w/o prefix	0.9514	0.6231	0.4699	0.5128	0.6393
V2	InstructEd	0.9930	0.9964	0.7886	0.6548	0.8582
	w/o prefix	0.9885	0.7065	0.5045	0.5502	0.6874
GPT2	InstructEd	0.9825	0.9971	0.7515	0.6531	0.8461
	w/o prefix	0.9728	0.5747	0.3834	0.4423	0.5933

Table 12: The ablation results of prefix on CT