Robust and Scalable Model Editing for Large Language Models

Yingfa Chen[1], **Zhengyan Zhang**[1], **Xu Han**[1,3,†], **Chaojun Xiao**[1], **Zhiyuan Liu**[1,†],
Chen Chen[2], **Kuai Li**[2], **Tao Yang**[2], **Maosong Sun**[1]
[1]DCST, IAI, BNRIST, Tsinghua University, Beijing, China
[2]Tencent Machine Learning Platform, China
[3]Shanghai Artificial Intelligence Laboratory
yf-chen22@mails.tsinghua.edu.cn
{hanxu2022, liuzy}@tsinghua.edu.cn

Abstract

Large language models (LLMs) can make predictions using *parametric knowledge*–knowledge encoded in the model weights–or *contextual knowledge*–knowledge presented in the context. In many scenarios, a desirable behavior is that LLMs give precedence to contextual knowledge when it conflicts with the parametric knowledge, and fall back to using their parametric knowledge when the context is irrelevant. This enables updating and correcting the model's knowledge by in-context editing instead of retraining. Previous works have shown that LLMs are inclined to ignore contextual knowledge and fail to reliably fall back to parametric knowledge when presented with irrelevant context. In this work, we discover that, with proper prompting methods, instruction-finetuned LLMs can be highly controllable by contextual knowledge and robust to irrelevant context. Utilizing this feature, we propose EREN (Edit models by REading Notes) to improve the scalability and robustness of LLM editing. To better evaluate the robustness of model editors, we collect a new dataset, that contains irrelevant questions that are more challenging than the ones in existing datasets. Empirical results show that our method outperforms current state-of-the-art methods by a large margin. Unlike existing techniques, it can integrate knowledge from multiple edits, and correctly respond to syntactically similar but semantically unrelated inputs (and vice versa). The source code can be found at `https://github.com/thunlp/EREN`.

Keywords: Large language models, model editing, question answering

1. Introduction

Large language models (LLMs) have demonstrated remarkable performance on numerous natural language processing (NLP) tasks and can memorize vast amounts of knowledge (Petroni et al., 2019; Shin et al., 2020; Brown et al., 2020; Roberts et al., 2020; OpenAI, 2023). However, the memorized knowledge may not be consistent with the knowledge of the real world (Lazaridou et al., 2021; Roemmele et al., 2022), and this may lead to undesired behaviors or incorrect predictions (Ji et al., 2023). Hence, model editing (Sinitsin et al., 2020; Zhu et al., 2021), which aims to quickly modify the behavior of a deployed LLM on specific examples while preserving its performance on unrelated instances, has gained attention in recent years.

The early approaches to model editing have focused on direct updates to the model parameters (Sinitsin et al., 2020; Zhu et al., 2021; De Cao et al., 2021; Mitchell et al., 2022a; Dai et al., 2022; Meng et al., 2022a,b; Li et al., 2023), which cannot be applied to current LLMs due to the inaccessibility of the parameters (OpenAI, 2023; Anil et al., 2023). Recently, some preliminary studies have explored the possibility of in-context model editing (Si et al., 2023), which modifies the behavior of a deployed LLM by adding a prompt to the input. The concurrent work by Zheng et al. (2023) builds upon this by adding demonstrations for behavior preservation on unrelated edit examples. In this way, the model users can easily edit black-box LLMs without access to the model parameters.

Li et al. (2023) shed light on a more scalable in-context model editing method. They argue that we can view the context as the *working memory* (R. et al.; Ashby et al., 2005) of neural models, and propose a finetuning regime that drives an LLM to make predictions grounded on the knowledge presented in the context over the knowledge it has learned during pretraining.

However, existing in-context model editing methods (Si et al., 2023; Zheng et al., 2023; Li et al., 2023) have three major limitations. (1) They are not scalable to large numbers of edits. If we integrate multiple edits into a single prompt, the prompt may be too long for the LLM. (2) They assume the relevant edit of a certain instance is given while in real-world scenarios the model needs to determine whether the current instance is related to any edits. If an instance is unrelated to all edits but we still use the edits as prompts, it often has a negative impact (Jia and Liang, 2017; Webson et al., 2023). (3) LLMs sometimes ignore the knowledge presented in the context or fail to ignore irrelevant knowledge, negatively impacting their result (Li et al., 2023; Yoran et al., 2023a; Shi et al., 2023a).

In this work, we show that instruction-tuned LLMs

[†] Corresponding authors

can be reliably grounded on contextual knowledge. Inspired by this, we propose a robust and scalable model editing method called EREN (Edit models by REading Notes). (1) Specifically, the LLM is complemented with a notebook memory that stores all edits in natural text. For a given input, relevant edits are retrieved from the notebook and used as prompts to modify the behavior of the LLM. In this way, we can easily scale up the number of edits without increasing the length of the prompt. (2) To determine whether the current instance is related to a certain edit, we reformat the task of model editing into reading comprehension with an "unanswerable" option. Hence, we can avoid the negative impact of irrelevant edits on the LLM behavior.

Empirical results show that our method can achieve state-of-the-art performance on in-context model editing on question answering and fact-checking.

Our main contributions are as follows:

- We conduct rigorous experiments to show that instruction-tuning enables LLMs to give precedence to contextual knowledge over parametric knowledge.
- We propose EREN, a robust and scalable in-context model editing method that can handle large numbers of edits and irrelevant edits. Our method beats the current state-of-the-art by a large margin.
- We process and release cleaner and more challenging versions of existing datasets for model editing, and empirically show that existing methods see drastic performance drops on our new types of challenging examples.

2. Related Works

Model Editing Our method is most related to the lines of work on the model editing problem. It was originally proposed by Sinitsin et al. (2020); Zhu et al. (2021). Sinitsin et al. (2020) proposes a meta-learning framework to train models that can be more easily edited. Zhu et al. (2021) uses L_p norm to constrain the parameter change while training the model. KnowledgeEditor (De Cao et al., 2021), MEND (Mitchell et al., 2022a), and Xu et al. (2022) introduce hyper-networks to transform gradients into parameter changes. However, the performance of gradient-based methods suffers greatly when applying multiple edits in sequence, and gradient information may be unavailable.

ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b) propose causal tracing to locate factual associations in a GPT (Radford et al., 2019; Wang and Komatsuzaki, 2021), and update the FFN layer to insert a factual association. However, these method requires expensive activation statistics and do not work with non-causal LMs. REMEDI (Hernandez et al., 2023) proposes to use learn a mapping from inputs to the hidden representations to guide the output, but it only focuses on editing errors in the input.

Retrieval methods use an external memory module and an explicit relevance estimation step to avoid reliance on gradient information or knowledge-locating methods. SERAC (Mitchell et al., 2022b) estimates relevance using a scope classifier and sends relevant edits to a counterfactual model. GRACE (Hartvigsen et al., 2022) caches and retrieves hidden representations of edits. However, these methods have limitations in generalization and performance, and assume that there is only one relevant edit at a time. In contrast, EREN's relevance estimation is more accurate, can be conditioned on multiple edits at the same time, and has stronger generalization abilities.

Finally, the concurrent work by Zheng et al. (2023) uses in-context learning to update the model's knowledge. Still, they only consider the insertion of one fact and assume that the relevant fact is known. Their method can be seen as the few-shot version of our one-step MRC baseline.

Retrieval-Augmented Methods Our work relies on retrieval to scale up to thousands of edits. Retrieval-augmented methods have demonstrated impressive capabilities in knowledge intensive tasks (Chen et al., 2017; Karpukhin et al., 2020; Mao et al., 2021; Guu et al., 2020; Shi et al., 2023b). More recent and concurrent works include (Ren et al., 2023; Vu et al., 2023; Yoran et al., 2023b,a; Zhou et al., 2023; Jiang et al., 2023; Shi et al., 2023a).

Prompting with Retrievals Our methodology can be categorized as "prompting". MemPrompt (Madaan et al., 2022) employs a growing memory of prompts to help the model better understand user intentions, but they focus on a different task setting. Si et al. (2023) prompts GPT-3 to perform reading comprehension on Wikipedia passages with replaced entities to update knowledge, but have little analysis related to model editing.

3. Methodology

3.1. Sequential Model Editing

In model editing, multiple edits may be applied simultaneously or sequentially. The ability to perform the latter is important for making the edits as soon as they appear and has been shown to be considerably more challenging than the former sce-

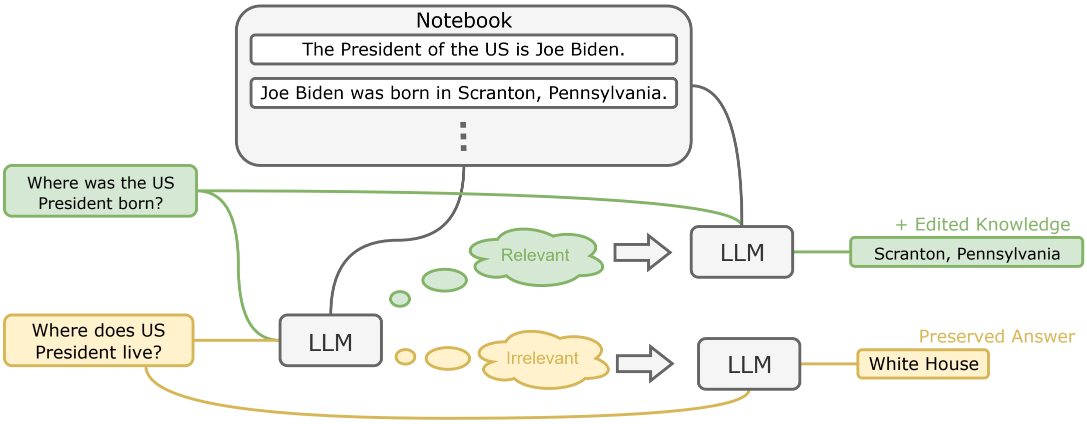
Notebook
The President of the US is Joe Biden.
Joe Biden was born in Scranton, Pennsylvania.
Where was the US President born?
Where does US President live?
LLM
Relevant
Irrelevant
LLM
LLM
+ Edited Knowledge
Scranton, Pennsylvania
Preserved Answer
White House

Figure 1: Illustration of the framework of EREN. Two edits have been injected, and the colored part shows inference on two inputs. Green: Both edits are relevant, and the final output depends on both. Yellow: The LLM determines that no edit is relevant, and the output of the base model is used.

nario (Huang et al., 2023). This paper focuses on sequential model editing.

When we apply one *edit* e, we want to instill certain behaviors into a model f on a set of inputs. Typically, e is a fact and we want the edited model f^* to behave as if the fact is true. The goal of model editing is to find an *editor* function that produces an edited model given a *base model* and an edit: Edit$(f, e) = f^*$.

To evaluate the correctness of f^*, we define the *edit scope* $I(e)$ of e as the set of input-output pairs that are implied by e:

$$I(e) = \{(x_1, y_1), ..., (x_m, y_m)\},$$

where (x_i, y_i) is the i-th example implied by e.

For instance, the edit that instills the fact that "The CEO of Apple is Tim Cook" implies that the answer to the questions "Who is the CEO of Apple?" and "Where does Tim Cook work?" are "Tim Cook" and "Apple", respectively.

In *sequential* model editing, we want to apply each edit one by one and ensure that all intermediate edited models are performant. This is important for keeping the model up-to-date and undesired behaviors are fixed as soon as possible. Assume an ordered set of n edits $\mathcal{E} = \{e_1, ..., e_n\}$, the edited model is as follows.[1]

$$f^* = \text{Edit}(f, \mathcal{E}) = \text{Edit}_n \circ \cdots \circ \text{Edit}_2 \circ \text{Edit}_1(f),$$

where $\text{Edit}_i(\cdot) = \text{Edit}(\cdot, e_i)$. Moreover, the edit scope of multiple edits is not the union of their scopes, because multiple edits in conjunction may imply new input-output pairs. Figure 1 shows one such example in green. To address this, we denote the edit scope of a set of edits $I(\mathcal{E})$ as all input-output pairs implied by all the edits in $\mathcal{E}$ in conjunction.

The goal of Edit$(f, \mathcal{E})$ is to produce an f^* that satisfies the following.

$$f^*(x) = \begin{cases} y & \forall (x, y) \in I(\mathcal{E}) \\ f(x) & \forall x \notin I(\mathcal{E}). \end{cases}$$

For the simplicity of further discussion, we say an edit is *relevant* to an input x (and vice versa) when its edit scope contains x.

Note that the edits may come in different formats. Most existing works on model editing, represent edits with input-output pairs (Sinitsin et al., 2020; Mitchell et al., 2022a,b), while others use factual triples (Meng et al., 2022a,b; Hernandez et al., 2023). However, this paper assumes that edits are given as short declarative sentences.

3.2. Our Approach

In summary, the edited LLM is complemented with a notebook that caches all edits in natural text. For each question, the model first determines whether the input is relevant to any edit, if so, it makes a prediction based on the notebook. Else, it directly answers the question using its memorized knowledge.

We find that LLMs, even when instruction-finetuned, are not readily controllable by their context, i.e. the notebook. In particular, they are not robust to irrelevant context (Li et al., 2023; Yoran et al., 2023a; Shi et al., 2023a), resulting in changed predictions on unrelated inputs. Also, the number of edits may to too large to fit into the input context of the LLM. Addressing these two issues, we propose to (1) split inference into two steps, and (2) use a dual-encoder retrieval framework to perform rough relevance estimation.

[1] $f_1 \circ f_2$ denotes the composite of f_1 and f_2.

3.2.1. Two-Step Inference

Our preliminary experiments reveal that LLMs, regardless of whether it is instruction-finetuned, are generally easily controllable by grounding on relevant contexts, but they are not robust to irrelevant context, which has been highlighted by existing works (Li et al., 2023; Shi et al., 2023a). However, we discover that **instruction-finetuned LLMs can reliably determine the relevance of contexts.**

Inspired by this observation, we design a two-step inference pipeline. The LLM is first prompted to determine whether an input is relevant to existing edits, i.e., determine whether $x \in I(\mathcal{E})$ is true. If true, the LLM performs conditional generation with all edits as the premise. If false, the LLM answers without context. One possible prompt template for relevance estimation is roughly as follows. The complete prompts are given in Appendix B.

```
Read this and answer the question. If
it is unanswerable, say <irr>.
<premise>
<question>
```

Here, `<premise>` is a list of edits, and `<irr>` is a special token that indicates irrelevance. If the LLM's answer is `<irr>`, we prompt it to answer using only parametric knowledge.

In summary, the edited LLM can be formalized as:

$$f^*(x) = \begin{cases} f(x) & f(\mathcal{T}_{\text{rel}}(x, \mathcal{E})) = \texttt{<irr>} \\ f(\mathcal{T}_{\text{gen}}(x, \mathcal{E})) & \text{otherwise,} \end{cases}$$

where $\mathcal{T}_{\text{gen}}$ and $\mathcal{T}_{\text{rel}}$ are the prompt templates for conditional generation and relevance estimation.

This is analogous to a person noting down every edit, and using relevant notes over memorized knowledge if there are any relevant notes. Thus, we named the method EREN (Edit models by REading Notes). The framework is illustrated in Figure 1.

3.2.2. Rough Relevance Estimation

In practice, $\mathcal{T}_{\text{gen}}(x, \mathcal{E})$ and $\mathcal{T}_{\text{rel}}(x, \mathcal{E})$ becomes exceedingly long when $\mathcal{E}$ is very large, and the input length may exceed the context capacity. Therefore, we perform a rough relevance estimation to eliminate irrelevant edits that are easily identified. To this end, an embedding-based *note retriever* is employed to retrieve the top-k most relevant notes. Let R denote the encoder, the retrievals are

$$\mathcal{E}_R = \text{Top-}k_{e \sim \mathcal{E}} \left(R(x) \cdot R(e)\right)$$

where $k < |\mathcal{E}|$. We use $\mathcal{E}_R$ instead of $\mathcal{E}$ to construct the prompt.

4. Experiments

4.1. Datasets

We evaluate the editors on QA and fact-checking. We first collect more challenging examples, then filter out examples of poor quality, and finally perform the necessary conversions to suit our setting.

4.1.1. Question Answering

For QA, We use CounterFact (Meng et al., 2022a), a dataset for editing knowledge in language models. Each question in CounterFact is the verbalization of a factual triple (subject, relation, object), and edits are created by modifying the object. CounterFact also includes out-of-scope inputs (i.e., inputs outside of the edit scope) that are constructed by swapping the subject with a neighboring subject (see Meng et al. (2022a) for more details).

While many existing model editors are evaluated on ZsRE (Levy et al., 2017), we choose CounterFact over it because the out-of-scope examples in ZsRE are sampled from a large set of unrelated examples, which are syntactically very different from the edit, making the edit scope estimation overly simple (Meng et al., 2022a). Table 1 shows an example in our version of CounterFact.

Collecting Harder Out-of-Scope Questions We find that out-of-scope examples constructed by keeping the subject but changing the relation and object are more challenging. We hypothesize that this is because existing methods are overly reliant on the subject. Therefore, to better evaluate *specificity* of model editors, i.e., their performance on out-of-scope questions, we generate such examples by collecting Wikidata triples with the same subject but different relations and objects, then verbalize them with the templates from CounterFact.

See Appendix A for more details.

4.1.2. Fact-Checking

We follow existing works (De Cao et al., 2021; Mitchell et al., 2022b) and evaluate using FEVER (Thorne et al., 2018) where each example is a factual statement. We use the version released by De Cao et al. (2021) which includes input paraphrases generated with back-translation. The fact statement itself is used as the edit statement. The reader performs natural language inference (NLI) with the retrieved edits as premises. For editors that require QA pairs, we convert the facts into boolean questions with the template "Is it true that {statement}?". To generate false facts, we sample half of all facts, and flip the answers to all questions except the first one to "no" by negating the

Part	Explanation	Example
Edit statement	A statement of the fact to be inserted.	"The president of the US is Joe Biden."
Edit scope	QA pairs that are implied by the edit.	"Who is the president of the US?", "Joe Biden"
Out-of-scope examples	Questions whose answers are not changed by the edit.	"Where does the president of the US live in?", "White House"

Table 1: Parts of an example in our version of COUNTERFACT. The edit statement is only used for our method, and we use one QA pair from the edit scope for baseline methods that rely on QA pairs as edits.

Version	COUNTERFACT	FEVER
Original	12.5% / 36.9%	42.7%
Auto-filtered	0% / 0%	26.5%

Table 2: Proportion of incorrectly labeled examples in COUNTERFACT and FEVER, by human inspection on 128 samples. For COUNTERFACT, the two numbers correspond to the error proportion of in-scope and out-of-scope examples.

fact statements using a BART-Base (Lewis et al., 2019) from Lee et al. (2021).

4.1.3. Filtering

The datasets have a relatively large proportion of erroneous labels, and we leverage pretrained models to create filtered versions. Table 2 shows the proportion of erroneous labels in the original and filtered versions. However, we find that the auto-filtered FEVER still contains many erroneous examples, so we create a cleaner version of FEVER by manual filtering with 128 examples. More details about the implications of filtering and some examples of erroneous data are given in Appendix A.3.

4.2. Evaluation Metrics

Edit Success (ES) An edit is successful when all examples in its edit scope are correctly predicted, so we define the edit success of an edit as the accuracy of the model on the edit scope.

Behavior Preservation (BP) An input with no relevant edits should not have its prediction changed. Therefore, we define the behavior preservation of an edit as the proportion of unrelated examples whose behavior has been preserved.

Edit Quality (EQ) A good model editor should ensure both ES and BP, hence, we define the edit quality as the harmonic mean of ES and BP.

A perfect model editor has an ES, BP, and EQ of value 1.

4.3. Implementational Details

We apply EREN to edit the publicly available FLAN-T5 (Chung et al., 2022), which is obtained by multi-task instruction-finetuning T5 checkpoints (Raffel et al., 2020).

We use Contriever (Izacard et al., 2022) for the rough estimation. It is a dense passage retriever with state-of-the-art zero-shot performance. Unless specified, $k = 5$ edits are retrieved during inference. To aggregate the retrieved notes $\mathcal{E}_R$ for feeding to the model as the context, we simply concatenate the notes with a new line as the delimiter. Answers are generated by greedy search. We cap the number of output tokens at 20 and 10 for QA and FC, respectively.

4.3.1. Task Reformatting

During the reading step, QA and fact-checking inputs are reformatted as reading comprehension and NLI, respectively. In reading comprehension, we prompt the reader to output "unanswerable" if the context cannot be used to answer the question. In NLI, the retrieved notes are the premise and the input is the hypothesis, and the reader is given three options at the end of the prompt, corresponding to entailment, neutral, and contradiction.[2]

4.4. Experimental Details

4.4.1. Black-Box Baselines

SERAC We primarily test our model against SERAC (Mitchell et al., 2022b), the state-of-the-art model editor. To the best of our knowledge, it is the only editor that can be used in a black-box setting. The original SERAC is finetuned on supervised data, but we are interested in the zero-shot performance, so we train SERAC on ZsRE using the same hyperparameters as Mitchell et al.

[2] Unfortunately, the instructions for NLI tasks in the FLAN dataset have no clear distinction between contradiction and neutral. I.e., "No" and "It's impossible to say" both could imply no entailment. The author says it was an arbitrary choice: https://github.com/google-research/FLAN/issues/32. Despite so, our method achieves superior results.

	QA			FC			FC (clean)		
Methods	**ES ↑**	**BP ↑**	**EQ ↑**	**ES ↑**	**BP ↑**	**EQ ↑**	**ES ↑**	**BP ↑**	**EQ ↑**
Unedited	0.0	100	0.0	41.9	100	59.1	38.7	100	55.8
Non-Black-Box Methods									
Full FT	17.0	0.4	0.8	58.9	49.8	54.0	58.8	13.0	21.2
MLP FT	1.9	9.2	3.1	58.9	49.7	53.9	57.7	60.8	59.2
MEND (Mitchell et al.)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ROME (Meng et al.)*	17.2	7.3	8.5	-	-	-	-	-	-
Black-Box Methods									
SERAC (Mitchell et al.)	96.9	51.4	67.2	60.1	67.9	63.8	60.4	97.6	74.6
+ Data Aug.	93.9	75.3	83.6	59.2	66.7	62.7	58.5	**98.1**	73.3
One-step MRC	**98.4**	24.5	39.2	**90.9**	64.6	75.5	**93.8**	74.5	83.0
EREN (Ours)	96.9	**96.8**	**96.9**	81.5	**79.6**	**80.5**	**93.8**	96.7	**95.2**

Table 3: Comparison of different methods on question answering (QA) and fact-checking (FC). FC (clean) is the manually filtered dataset. The base model in One-step MRC and EREN are FLAN-T5-XL. The best result in each metric is **bolded**. *: Applied to GPT-2-XL instead because it is only applicable to causal language models.

(2022b) and evaluate it on unseen datasets without additional finetuning. We also evaluate the version of SERAC that employs data augmentation (DA) by automatically sampling similar inputs using a Sentence-BERT (Reimers and Gurevych, 2019) as negative samples.

One-Step MRC Let the model directly answer the question in one forward pass in a zero-shot manner.

4.4.2. Non-Black-Blox Baselines

For reference, we also list the results of non-black-box model editors, although they should not be regarded as baselines because they have access to the parameters.

Full FT & MLP FT Finetune all parameters or finetune only the second linear layer in one of the FFN layers (choosing the best performing one among all layers). They are trained with a constant learning rate of $1e-5$ with Adam optimizer (Kingma and Ba, 2017) until the target output is learned or after 50 parameters update steps.

MEND MEND (Mitchell et al., 2022a) learn to map the gradients to low-rank parameter updates that better ensure the generality and locality of knowledge editing.

ROME ROME (Meng et al., 2022a) modifies a key-value association in the MLP layers by updating the parameters to maximize the probability of the target text. Since ROME requires knowledge about the subject of an edit, we do not evaluate it on FEVER, which does not have labeled subject entities. The base model for ROME is GPT-2-XL (Radford et al., 2019) instead because it is unclear how it can be applied to encoder-decoder models.[3]

4.5. Edit Format

Some methods (e.g., SERAC and gradient-based methods) require input-output pairs as edits. In such cases, we pick one QA pair from the edit scope. For EREN, edits are assumed to come in the format of declarative statements. Therefore, for each example in QA, we convert one of the QA pairs into a declarative sentence using a T5-3B finetuned on QA-NLI (Chen et al., 2021; Demszky et al., 2018).

Edit Scheme We apply 1024 edits sequentially for auto-filtered QA and FC, and 128 edits for the cleaner FC because it only has 128 examples.

5. Result

The result for our method and the baselines on QA and fact-checking are shown in Table 3. The edit quality of EREN is greater than the non-black-box baselines and SERAC by a large margin. The finetuning methods, MEND, and ROME suffer from severe catastrophic forgetting, resulting in very low edit quality, and generally fail to sequentially apply more than a thousand edits. After a certain number of sequential parameter updates, the model has degraded to producing unintelligible text. For MEND, it is because the hypernetwork was adapted

[3] MEMIT (Meng et al., 2022b), an extension of ROME that addresses multiple edits, is not considered because it requires edits to be applied simultaneously, but this paper addresses the sequential model editing problem.

to the parameter of the base model, but the parameter changes for each update, while the hypernetwork stays the same. A similar problem is found in ROME, where we have to pre-compute the activation statistics of the base model, which is not updated for each edit.[4] It is also worth noting that the time needed to apply each edit in these non-black-box baselines is significantly more than SERAC and EREN.

Interestingly, one-step MRC can beat SERAC in fact-checking in terms of editor quality. This is likely because SERAC is trained on a QA dataset and is therefore unable to adapt to the domain of fact-checking in a zero-shot manner.

One of the main reasons SERAC underperforms EREN is that SERAC is limited by the two small complementary models. See Appendix F for a discussion on why SERAC underperforms.

In the following sections, we will evaluate EREN's performance in editing different base models, scaling the number of edits, and its ability to combine multiple edits. We also analyze the effect of the note retriever. Finally, we show that the hard in-scope and out-of-scope examples are more challenging than those that are commonly used to test model editors.

5.1. Different Base Models

Figure 2 shows the result of EREN on different base models, the implementation details are given in Appendix D. We can see that instruction-tuning is crucial for the success of EREN, i.e., T5 without instruction-tuning (Raffel et al., 2020) has only half the edit quality. Interestingly, most performance degradation comes from low BP, which indicates that instruction-tuning is essential for ensuring the LLM is robust to irrelevant contexts.

We also observe that EREN is effective for editing GPT3.5, an API-level LLM. It is not as effective as using T5 because GPT3.5 is trained to output chat-like responses, which gives a lower score on QA because we use exact match as the evaluation metric.

5.2. Different Number of Edits

Meng et al. (2022b); Mitchell et al. (2022b) have shown that existing methods may struggle when we scale up the number of edits. Figure 3 shows the performance of EREN, ROME, and Full FT on question-answering where the number of edits ranges from 1 up to 1024. Both Full FT and ROME exhibit very poor EQ at 1024 edits because sequentially editing the model's parameters will add up the errors of each edit. In contrast, EREN keeps the base model frozen, making the impact of each edit limited to relevance estimation, and reducing the EQ drop to less than 4%.

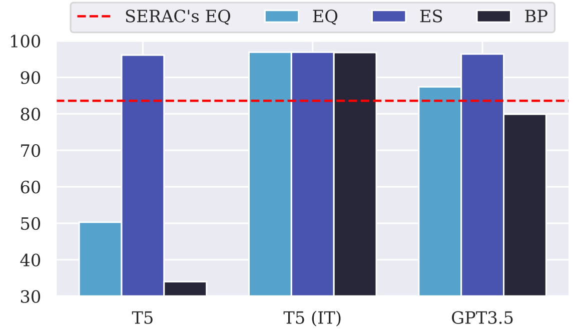

Figure 2: The performance of EREN on different base models. The red dotted line represents the EQ of SERAC + DA. T5 and T5 (IT) are the non-instruction-tuned and instruction-tuned versions of T5-XL, and GPT3.5 is the `gpt-3.5-turbo` API. See Appendix D for more details.

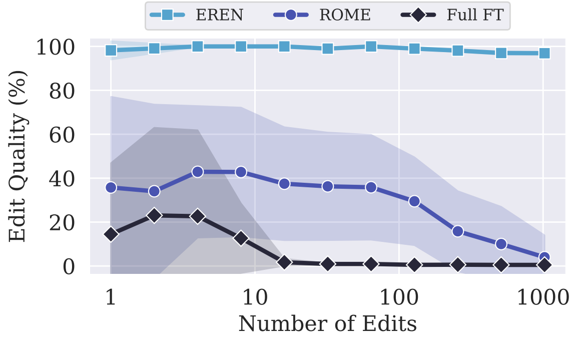

Figure 3: Edit quality of EREN, ROME, and Full FT by different numbers of edits on CounterFact. The colored area is the standard deviation of 5 runs.

Method	ES ↑	BP ↑	EQ ↑
Unedited	23.4	100	37.9
EREN ($k=1$)	38.9	**95.7**	55.3
EREN ($k=2$)	60.9	93.2	73.7
EREN ($k=3$)	65.4	92.0	76.5
EREN ($k=5$)	67.2	89.5	76.7
EREN ($k=10$)	**70.7**	89.1	**78.8**

Table 4: Performance on questions that require combining knowledge from multiple edits. k is the number of retrieved notes.

5.3. Combining Multiple Edits

Existing retrieval-based methods (Mitchell et al., 2022b; Hartvigsen et al., 2022) assume that each input has only one relevant edit, and they are not able to combine multiple edits. To evaluate the ability of our method to combine knowledge from

[4] It is prohibitively expensive to update the hypernetwork and activation statistics after every edit.

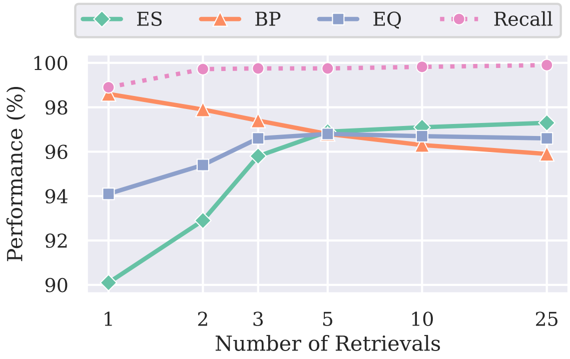

Figure 4: The performance of EREN and recall rate of the note retriever by retrieving different numbers of notes on CounterFact.

multiple edits, we sample 512 examples from HotpotQA (Yang et al., 2018) and insert all passages as edits, then sample another 512 examples to use as out-of-scope questions. The result is listed in Table 4. We can see that the performance increases sharply when retrieving more than one edit, which indicates the ability to combine multiple edits.

5.4. Effect of Note Retriever

A note retriever filters out highly dissimilar notes to speed up inference, and it may significantly influence the final performance. Figure 4 plots the retrieval recall rate and EREN's performance on CounterFact with varying numbers of retrievals.

Interestingly, with a small number of retrievals, the ES is significantly lower than the recall rate, which means that although the reader can see the relevant note, it has not been able to produce the correct answer. We hypothesize that this is because the reader is instruction-tuned on datasets where contexts are usually longer than just a few sentences, and struggles to generalize to shorter contexts.

On the other hand, increasing the number of retrievals reduces BP, which is intuitive, because more irrelevant notes may introduce noise for the reader. The edit quality does not increase much beyond 5 retrievals.

5.5. Harder Out-of-Scope Examples

As mentioned in Section 4.1.1, although CounterFact already includes hard out-of-scope questions on neighboring subjects, we collect unrelated questions about the subject of the edit, which serves as harder out-of-scope questions. Table 5 shows the breakdown of behavior preservation of different methods on the two types of out-of-scope questions. We observe that SERAC sees a much larger performance drop on out-of-scope questions about the same subject compared to the out-of-scope questions in the original CounterFact, which confirms our hypothesis that the question we collect about the same subject is more challenging for model editors than the questions in the original dataset. This is likely because SERAC's scope classifier has learned to overly rely on the subject as a signal to determine the relevance of edits. Using negative samples as data augmentation significantly mitigates this, but still is far behind the performance of EREN.

Method	NB. Subj. ↑	Same Subj. ↑
SERAC	89.3	33.1
+ Data Aug.	89.4	63.4
One-step MRC	13.7	25.1
EREN	**97.0**	**95.9**

Table 5: Behavior preservation on CounterFact by different types of out-of-scope questions. **NB. subj.**: Questions where the subject is replaced with a "neighbor subject", introduced by Meng et al. (2022a). **Same subj.**: Questions about unrelated knowledge of the same subject as the edit, introduced by us.

5.6. Harder In-Scope Examples

Mitchell et al. (2022b) proposed to construct hard in-scope QA by automatically constructing implied facts. E.g., the QA pair ("Who is the Prime Minister of UK?", "Boris Johnson") as an edit would imply the QA pair ("Where is Boris Johnson Prime Minister?", "UK"). However, we discover that SERAC fails on simple in-scope examples unseen in its training set, such as rephrasing the question as a boolean question. Concretely, after applying the above edit and asking "Is it true that Boris Johnson is the Prime Minister of the UK?", SERAC would still output "Boris Johnson". We randomly sample 512 edits from CounterFact and convert them into boolean questions with the prompt "Is it true that {edit statement}?".

The result is shown in Table 6. SERAC gets all questions wrong. We conclude this is because the counterfactual model that is responsible for inference on in-scope examples is too small[5], and has overfitted to the training data that includes automatically generated implied examples.

6. Conclusion

This work has presented EREN, a zero-shot retrieval-based model editing framework for black-box LLMs that is scalable. The editor stores all edits

[5] SERAC finetunes a T5-small for the counterfactual model.

Method	Unedited	Full FT	SERAC	EREN
ES ↑	9.4	4.7	0.0	100

Table 6: Edit success on simple boolean questions converted from CounterFact with the template "Is it true that {edit statement}", which act as harder in-scope examples.

in a growing notebook in natural text, and a reader uses the notes to produce an answer if any of them are relevant. Our experiments were conducted under a black-box setting, where access to datasets of edits and model parameters and activations was not available, and we tested the model's ability to combine multiple edits, increasing its practicality and applicability to a broad range of scenarios. Our empirical results show that EREN significantly outperforms the current state-of-the-art model editor, demonstrating superior edit success and preservation of the model's behavior on unrelated edits. We believe that EREN represents a significant step towards the lifelong maintenance of LLMs.

Ethics Statement

The ability to quickly update knowledge in LLMs has many benefits, but may also be used to inject wrong knowledge, undesired behavior, or bias into LLMs, although that is not the motivation of our work. This method could also significantly increase the input length, which may result in a higher carbon footprint. However, we emphasize that the purpose of editing models is to avoid the more expensive choice of re-training the model when an update to the parametric knowledge is requested. It is also worth noting that most existing model editing methods introduce more computation and memory overhead than our method.

Acknowledgement

This work is supported by the National Key R&D Program of China (No. 2022ZD0160501) and National Natural Science Foundation of China (No. 62236011, No. 62236004). This work is also supported by Tencent AI Platform Department.

The method is primarily designed by Yingfa Chen, and he conducted most of the experiments. Zhengyan Zhang, Xu Han, and Chaojun Xiao, Zhiyuan Liu actively participated in experimental design, result analysis, and paper writing. Chen Chen, Kuai Li, Tao Yang, and Maosong Sun provided valuable advice to this research.

7. Bibliographical References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L_1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, et al. 2023. PaLM 2 technical report. *CoRR*, abs/2305.10403.

F. Gregory Ashby, Shawn W. Ell, Vivian V. Valentin, and Michael B. Casale. 2005. Frost: A distributed neurocomputational model of working memory maintenance. *Journal of Cognitive Neuroscience*, 17(11):1728–1743.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Jifan Chen, Eunsol Choi, and Greg Durrett. 2021. Can nli models verify qa systems' predictions? *EMNLP Findings*.

Eunsol Choi, Jennimaria Palomaki, Matthew Lamm, Tom Kwiatkowski, Dipanjan Das, and Michael Collins. 2021. Decontextualization: Making sentences stand-alone. *Transactions of the Association for Computational Linguistics*, 9:447–461.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

James W. Cooley and John W. Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301.

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

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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nicola De Cao, Michael Sejr Schlichtkrull, Wilker Aziz, and Ivan Titov. 2020. How do decisions emerge across layers in neural models? interpretation with differentiable masking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3243–3255, Online. Association for Computational Linguistics.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets.

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

Keisuke Fukuda and Geoffrey F. Woodman. 2017. Visual working memory buffers information retrieved from visual long-term memory. *Proceedings of the National Academy of Sciences*, page 5306–5311.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pretraining. *ArXiv*, abs/2002.08909.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2022. Aging with grace: Lifelong model editing with discrete key-value adaptors.

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. 2023. Measuring and manipulating knowledge representations in language models.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. *ArXiv*, abs/2301.09785.

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

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2021–2031. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Li-Yu Daisy Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *ArXiv*, abs/2305.06983.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, CypriendeMasson d'Autume, Tomas Kocisky, Sebastian Ruder, Dani Yogatama, Kris Cao, Susannah Young, and Phil Blunsom. 2021. Mind the gap: Assessing temporal generalization in neural language models.

Minwoo Lee, Seungpil Won, Juae Kim, Hwanhee Lee, Cheoneum Park, and Kyomin Jung. 2021. Crossaug: A contrastive data augmentation method for debiasing fact verification models. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21. Association for Computing Machinery.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. PAQ: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2023. Large language models with controllable working memory. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1774–1793, Toronto, Canada. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9):195:1–195:35.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2833–2861, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. 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 4089–4100, Online. Association for Computational Linguistics.

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

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

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

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

Lidiya Murakhovs'ka, Chien-Sheng Wu, Philippe Laban, Tong Niu, Wenhao Liu, and Caiming Xiong. 2022. Mixqg: Neural question generation with mixed answer types.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *ArXiv*, abs/1909.01066.

T. A. R., George A. Miller, Eugene Galanter, and Karl H. Pribram. Plans and the structure of behavior. *The American Journal of Psychology*, 75(1):161.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *ArXiv*, abs/2302.00083.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, J. Liu, Hao Tian, Huaqin Wu, Ji rong Wen, and Haifeng Wang. 2023. Investigating the factual knowledge boundary of large language models with retrieval augmentation. *ArXiv*, abs/2307.11019.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Melissa Roemmele, Yixin Zhang, and Vivek Srikumar. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Huai hsin Chi, Nathanael Scharli, and Denny Zhou. 2023a.

Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023b. Replug: Retrieval-augmented black-box language models. *ArXiv*, abs/2301.12652.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. 2023. Prompting gpt-3 to be reliable. In *International Conference on Learning Representations (ICLR)*.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitriy Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *International Conference on Learning Representations*.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. Ul2: Unifying language learning paradigms.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and Thang Luong. 2023. Freshllms: Refreshing large language models with search engine augmentation.

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

Albert Webson, Alyssa Marie Loo, Qinan Yu, and Ellie Pavlick. 2023. Are language models worse than humans at following prompts? it's complicated. *CoRR*, abs/2301.07085.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Yang Xu, Yutai Hou, and Wanxiang Che. 2022. Language anisotropic cross-lingual model editing.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023a. Making retrieval-augmented language models robust to irrelevant context.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023b. Making retrieval-augmented language models robust to irrelevant context.

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?

Wenxuan Zhou, Sheng Zhang, Hoifung Poon, and Muhao Chen. 2023. Context-faithful prompting for large language models. *ArXiv*, abs/2303.11315.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2021. Modifying memories in transformer models. *arXiv preprint arXiv:2104.07733*.

A. Dataset Construction Details

A.1. Question Answering: CounterFact

Our version of CounterFact includes harder out-of-scope questions in which the relation and object of the edit's triple are changed, but the subject stays the same, the resulting questions are questions that elicit knowledge about the subject of the edit, but are outside the edit scope. To construct such questions, we query triples from Wikidata using the subject name. We use the templates in CounterFact to verbalize the triples (discarding those without a template) into declarative sentences. Then we convert the templates to QA pairs using MixQG[6].

[6] `https://huggingface.co/Salesforce/mixqg-large`

MixQG accepts a context and an answer as the input and produces a question, and since each CounterFact template is supposed to prompt the answer as a text continuation, we simply append the answer to the template, and feed this as the context to MixQG.

To create an edit, we convert one QA pair from the edit scope into a declarative sentence with a pretrained converter[7], which is a T5-Base finetuned on QA2D (Demszky et al., 2018). Interestingly, although we could have simply appended the answer to the prefix prompt provided in the vanilla CounterFact, we found that converting it to a QA pair and then to a declarative sentence may eliminate ambiguity in the CounterFact prompts.

Note that the edit scope in CounterFact is created by simply paraphrasing one question, which means all answers are the same. An ideal edit scope for evaluation should include all QA pairs implied by the edit, but such data is too expensive to collect. Thus, we leave such study for future work.

A.2. Fact-Checking: FEVER

As mentioned in Section 4.1.2, we use the version of FEVER that is released by De Cao et al. (2021), which includes paraphrases created with back-translation. However, when we want to edit the model to make falsify a fact, we need to construct the negation of the factual statement. To do so, we turn the statement into a boolean question with "Is it true that statement?" and feed this question along with "no" to the QA pair to the statement converter as mentioned above.

A.3. Filtering

As mentioned in Section 4.1.3, we create cleaner versions of CounterFact and FEVER by removing poorly worded or labeled examples. Table 7 lists some examples for each type of bad example. To reduce the computational cost, we first use a pretrained NLI model (Nie et al., 2020) to automatically construct a larger filtered dataset, then create a smaller but cleaner version by manual filtering.

A.3.1. Automatic Filtering

We set the edit statement as the premise, then feed each in-scope and out-of-scope input as the hypothesis to the NLI model. But since CounterFact examples are QA-pairs, we first convert them to statements using a converter[8] trained on QA2D (Demszky et al., 2018).

- For **in-scope inputs**, we regard it as incorrectly labeled if the edit statement is neutral to the input (neither entails nor contradicts). However, since the NLI model is imperfect, we require that the predicted probability of neutral be less than 80% of the self-entailment probability of the edit statement (i.e., the probability of the statement entailing itself).

- For **out-of-scope inputs**, we want the edit statement to be neutral to the inputs. Again, we require that the probability of the edit statement is neutral to the input to be greater than 80% of the self-entailment probability of the edit statement.

It is important to note that inputs with unintelligible wording may not have been taken care of with this filtering process. This is because unintelligible wording of a fact is likely to be neutral of other facts, so out-of-scope inputs with unintelligible wording are likely to survive this filtering process.

Moreover, out-of-scope input of an edit may be relevant to another edit, and such false out-of-scopes are not taken care of with this procedure. In CounterFact, such examples are rare since different edits have different subjects, but in FEVER, for example, there is an edit statement "Saxony is in Ireland" and an out-of-scope input "Saxony is the sixth most populous Spanish state". Performing NLI on all pairs of edit statements and out-of-scope inputs would be too expensive, therefore, we keep them for the larger filtered data and filter them out by manual filtering.

For CounterFact, the most common errors are false out-of-scope and false in-scope questions. False out-of-scope questions mainly arise from the verbalized triples from Wikidata. This is because many templates in CounterFact are about closely related facts, and the templates are too ambiguous to distinguish the differences. For instance, a template about a person's city of birth may be mistaken for being about the person's country of birth, such as "Where was {subject} born?".

A.3.2. Manual Filtering

The automatically filtered version of FEVER still contains some erroneous examples as shown in Table 2. Therefore, we pick the first 128 examples from FEVER and manually filter out those that are unintelligibly worded or are too ambiguous. Since we use another 128 examples as out-of-scope examples, they must be unrelated to the former 128 edits. Therefore, to filter out false out-of-scopes, we have to read the 128 edits, and make sure every out-of-scope examples are irrelevant to each of the 128 edits.

[7] https://huggingface.co/domenicrosati/QA2D-t5-base

[8] https://huggingface.co/domenicrosati/question_converter-3b

Type	Example
	CounterFact
Bad wording	Edit: “Toko Yasuda, the”
False in-scope	Edit: “Danielle Darrieux’s mother tongue is English.” In-scope question: “What is the nationality of Danielle Darrieux?”
False out-of-scope	Edit: “Danielle Darrieux’s mother tongue is English.” Out-of-scope question: “What language does Danielle Darrieux speak?”
	FEVER
Bad wording	In-scope fact: “==References====External links==”
False in-scope	Relevant Edit: “Nicholas Brody is a character on Homeland.” In-scope fact: “Nicholas Brody is a character at home.”
False out-of-scope	Relevant Edit: “Jayasudha is an actor that stars in Daag” Out-of-scope fact: “Daag is a painting”

Table 7: Examples of the different types of error found in CounterFact and FEVER. Note that the examples of CounterFact here is after mining hard out-of-scope examples and prefix-to-question conversion, but most of these errors are found before conversion as well.

B. Prompts

When the reader in EmoRen determines that there are no relevant edits, it outputs a predefined string on irrelevant edit context. This predefined string is usually specified in the prompt. The prompts that we use for QA and fact-checking are listed in Table 8, where the string for irrelevance is “unanswerable” and “It’s impossible to say” respectively. However, we find that the wording of this string has little impact on the performance of our preliminary experiments as long as they are semantically equivalent.

C. Additional Experimental Details

C.1. ROME

In this work, we applied ROME on GPT-2-XL instead of T5, because they can only be applied to causal LMs. For better reproducibility, we used the publicly released pre-computed layer statistics to edit the pretrained GPT-2-XL. We use the following prompt with few-shot exemplars to guide the model to perform QA.

```
Q: Who is the President of China?
A: Xi Jinping

Q: When did World War II end?
A: 1945

Q: What is the capital of Norway?
A: Oslo

Q: Who is the main character in The Matrix?
A: John Wick

Q: Who is the founder of Apple?
A: Steve Jobs

Q: Which is the largest planet in our solar system?
A: Jupiter

Q: How many legs do spiders have?
A: Eight

Q: Does pure water conduct electricity?
A: No

Q: {question}
A:
```

Despite using this prompt, the model still displays a strong tendency to output additional text after the answer. Therefore, we only regard the first line of the output sequence as the answer.

D. Different Base Models

We use few-shot demonstrations for T5 and GPT3.5, because we find that these models (the former is non-instruction-finetuned and the latter is fine-tuned for chatting) cannot reliably follow the instructions. Specifically, they are inclined to produce much more tokens in addition to the actual answer, such as explanation for the context is relevant/irrelevant.

E. Comparison to Evaluation Metrics in Existing Works

The original paper of ROME (Meng et al., 2022a) only evaluated the ability to apply one edit. Meng et al. (2022b) showed, through empirical results,

Task Type	Prompt
MRC	Read this and answer the question. If the question is unanswerable, say "unanswerable". <context> <question>
QA without context	Please answer this question: <question>
Fact-checking with context	<context> Based on the paragraph, can we conclude that "hypothesis"? OPTIONS: - Yes - It's impossible to say - No
Fact-checking without context	Is it true that <hypothesis>?

Table 8: The prompts that we used for QA and fact-checking, which are hand-picked from the instructions in the FLAN collection with slight modification.

that the *efficacy* of ROME drops steadily with increased number of edits. Here, the "efficacy" metric is defined as the proportion of in-scope examples where the probability of the target answer is greater than the probability of the actual answer, i.e., $\mathbb{E}[P(y^*) > P(y)]$, where y and y^* are the original and post-edit target outputs. However, our evaluation metric, edit success, is more challenging, because only the exactly matches of the top-1 output sequence are counted as successful edits.

F. Comparison to SERAC

EREN differs from SERAC in the following points.

- **Eliminating the need to train an extra in-scope model.** SERAC needs to train a counterfactual model to use as the in-scope model. Instead, EREN demonstrates how to leverage the reading comprehension capabilities of LLMs to perform editing as the in-scope model and directly use LLMs as the out-of-scope model. In other words, we are unifying the in-scope and out-of-scope models into one single model. One of the main reasons SERAC underperforms EREN is that the small in-scope model has a different parametric memory (because it is much smaller) than the base model, therefore, when the relevance estimation fails and an edit is falsely identified as being relevant, the in-scope model will not be able to ignore the edit and produce the same prediction as the base model.
- **Improving relevance estimation.** SERAC has two kinds of relevance estimation. The first one is to use a dual-encoder to calculate the distance between the encodings of the question and edits as their relevance, which is less capable (this is the primary method used in SERAC's paper and our paper). The second one is to iterate through every edit, concatenate the edit with the question, and feed them into a binary classifier, which is very slow. In EREN, we design a two-step retrieval process. EREN first has a rough estimation process that eliminates highly dissimilar edits, and then the reader can condition on multiple edits simultaneously to estimate the relevance of edits. This method is both expressive and efficient.
- **Supporting edit combination.** If one input is relevant to multiple edits, SERAC is not able to combine knowledge from the edits to produce a correct answer. This is because the counterfactual model is trained to condition on one input-output pair at a time. In contrast, EREN achieves this by performing generation conditioned on all edits (the entire notebook). Passing only top-1 most relevant edit to the counterfactual model also means that SERAC would very likely produce wrong answers whenever the top-1 prediction of the scope classifier is incorrect.