DistillMIKE: Editing Distillation of Massive In-Context Knowledge Editing in Large Language Models

Anonymous ACL submission

Abstract

Among the recently emerged knowledge editing methods, in-context knowledge editing (IKE) (Zheng et al., 2023) has shown respectable abilities on knowledge editing in terms of generalization and specificity. Noting the promising advantages but unexplored issues of IKE, we propose **DistillMIKE** as a novel extension of IKE, i.e., editing **distill**ation of "**M**assive" **I**n-context **K**nowledge **E**diting in large language models (LLMs), mainly consisting of two expansions; 1) *Massive in-context knowledge editing (MIKE)*, which extends IKE to a massive editing task, aiming to inject not a single edit but a set of massive edits to LLMs; To preserve specificity, our key novel extension is a "selective" retrieval augmentation, where the retrieval-augmented IKE is only applied to "in-scope" examples, whereas the unedited model without IKE is employed for "out-of-scope" ones. 2) *Editing distillation* of MIKE using low-rank adaptation (LoRA), which distills editing abilities of MIKE to parameters of LLMs in a manner of eliminating the need of lengthy in-context demonstrations, thus removing the computational overhead encountered at the inference time. Experimental results on the zsRE and CounterFact datasets demonstrate that MIKE shows the state-of-the-art perfomrances and DistilMIKE show comparable performances with MIKE. Our code is available at https://github.com/xxxx/xxxx.

1 Introduction

While large language models (LLMs) have shown the remarkable abilities across a broad spectrum of natural language processing (NLP) tasks (Touvron et al., 2023; OpenAI, 2023; Petroni et al., 2020), LLMs are still limited in the coverage and veracity of their world knowledge, thus causing the reliance on outdated knowledge (Onoe et al., 2022; Dhingra et al., 2022; Liška et al., 2022), or the generation of erroneous, hallucinatory, or biased contents (Zhao et al., 2023; Ji et al., 2023; Lazaridou et al., 2021; Agarwal and Nenkova, 2022; Gallegos et al., 2023). Provided the evolving nature of world knowledge and the need to correct the inaccurate information of LLMs, there has been recently growing interest in the "knowledge editing" task, aiming at developing the scaled and effective *editing* mechanism that injects new knowledge to LLMs or corrects false and erroneous information in LLMs. In particular, this paper addresses the "massive editing" task as in (Meng et al., 2022b), where a large number of edits are provided beyond just a single correction of an edit, the resulting editing mechanism needs to properly update more than hundreds or thousands of facts in LLMs simultaneously.

Among various approaches of knowledge editing such as parameter updating (PU) (Cao et al., 2021; Tan et al., 2024; Meng et al., 2022a,b; Li et al., 2023; Huang et al., 2023; Dong et al., 2022a; Madaan et al., 2022) and memory-based methods (Mitchell et al., 2022b; Zheng et al., 2023; Onoe et al., 2023; Zhong et al., 2023; Madaan et al., 2022), in-context knowledge editing (IKE) has been newly proposed, inspired from in-context learning (ICL) emerged in LLMs (Brown et al., 2020; Dong et al., 2022b), showing noticeable editing performances in terms of generalization and specificity. Motivated from the respectable advantages of IKE such as its human-interpretable editing way, we would like to go further steps towards expanding and improving IKE by addressing the following issues: 1) *Extension to massive editing task*: IKE has not been explored in the massive editing task, and thus it remains largely unclear how IKE is generalized to the massive editing task, how IKE performs compared with other popular PU methods such as MEMIT, and whether the demonstration construction previously suggested in (Zheng et al., 2023) is also feasible in the addressed task; 2) *Resolving the computational overhead caused by the use of lengthy prompts*: IKE incurs computational and memory overhead at the inference stage, be-

cause a length of an input prompt becomes a "long", resulting by prepending a non-trivial number of demonstrations.

Towards a novel extension to address the aforementioned issues, we propose **DistillMIKE**, i.e., editing distillation of massive in-context knowledge editing in LLMs, which mainly consists of two components:

- **Massive in-context knowledge editing** (MIKE), which extends IKE to a massive editing task, leading to a retrieval-augmented IKE; a large number of massive edits (i.e. facts) to be injected to LLMs are first stored in a separate memory, namely the *edit memory*; given an input query prompt, IKE is then preformed using its "relevant" edit retrieved from the edit memory, referred to a *query-matched edit*. To effectively preserve specificity, without employing IKE for all input prompts, we instead newly propose a "selective" retrieval augmentation, where the retrieval-augmented IKE is applied only for *in-scope* examples, not for other out-of-scope examples. Furthermore, given this selective nature of applying IKE, we further propose the use of *scope-aware demonstrations*, paying attention to in-scope cases, thus by including only "update" types of demonstrations which are likely useful for processing in-scope edits, but by excluding other "retain" types, as they may be mostly useful for out-of-scope cases.

- **Editing distillation of MIKE using low-rank adapter (LoRA) for DistillMIKE**, which distills editing abilities of MIKE to parameters of LLMs in a manner of eliminating the need of "lengthy" in-context demonstrations, aiming at reducing the computational overhead at the inference time. Inherited from the selective retrieval augmentation of MIKE, editing distillation of MIKE is a *multi-teacher* distillation (Wu et al., 2021; Liu et al., 2020), where two teachers are 1) the retrieval-augmented IKE for in-scope examples and 2) the unedited base model for out-of-scope examples. To substantially reduce the number of parameters to be updated, *LoRA* fine-tuning (Hu et al., 2022) is adopted during editing distillation, finally resulting in DistillMIKE.

Experimental results on the zsRE and CounterFact datasets demonstrate that MIKE achieves state-of-the-art performance in CounterFact dataset, and DistillMIKE show comparable performances with MIKE as well as improves the existing editing methods such as IKE and MEMIT, even in the setting that lengthy in-context demonstrations and instruction do not appear in prompts.

Our contributions are summarized as follows: 1) we propose MIKE, which extends IKE to the massive editing task, based on a selective retrieval augmentation depending on the scope type of an input query, 2) we further propose DistillMIKE, a LoRA-finetuned student model which is distilled from multi-teacher models – IKE and unedited base models, thereby injecting ICL prompts in MIKE into the model parameters, thus enabling to perform the inference without the need for lengthy demonstration prompts, 3) the proposed MIKE and DistillMIKE show state-of-the-art and promising performances on zsRE and CounterFact datasets.

2 Related Works

Existing studies are broadly categorized to *PU* approaches and *memory-based* methods. An extensive review on knowledge editing methods has been presented in (Yao et al., 2023). In this section, we briefly review selected previous methods and discuss some of them regarding the novelty our work.

2.1 Parameter Updating (PU) methods

PU methods are further categorized into three approaches – Meta-learning, Locate-and-edit, and parameter expansion methods.

Meta-learning Knowledge Editor (Cao et al., 2021) trains a hypernetwork to predict the parameter changes required for the model to predict new knowledge and to preserve old knowledge. MEND (Mitchell et al., 2022a) employs a hypernetwork to transform the initial fine-tuning gradient into a simplified representation using low-rank decomposition to produce the parameter updates. More recently, MALMEN (Tan et al., 2024) further extends MEND to the massive editing task by aggregating massive parameter updates to a single parameter update, motivated from MEMIT, demonstrating its scalability.

Locate-and-edit Dai et al. (2022) proposes the concept of knowledge neurons, for precisely editing factual knowledge editing at the instance level. ROME (Meng et al., 2022a) is a pioneering work that attempts to locate the model parameters associated with the target factual knowledge and

rewrite the key-value pairs in the feed-forward network (FFN) module with computed new vectors. MEMIT (Meng et al., 2022b) further expands ROME to be scalable on the massive editing task by spreading the weight changes over multiple model layers. PMET (Li et al., 2023) improves MEMIT by considering the knowledge storing role of the multi-head self-attention (MHSA) layer, thus preventing from overestimating the extent of the parameter updates required for FFN layers.

Parameter expansion In parameter expansion methods, parameters of LLMs are then *enlarged* by integrating the newly trained extra parameters to store new knowledge into original ones. T-Patcher (Huang et al., 2023) adds one neuron in the last layer of FNN to handle a specific edit request, and CaliNET (Dong et al., 2022a) extends T-Patcher using multiple neurons to cover a set of edits.

2.2 Memory-based Methods

IKE (Zheng et al., 2023) extensively explores the ICL-based knowledge editing by proposing a novel method for demonstration organization in a way of comprising multiple types of demonstrations, designed to simultaneously improve generalization and specificity, the main evaluation metrics of the knowledge editing task. MELLO (Zhong et al., 2023) stores edited facts externally and prompts the language model iteratively to generate answers that align with the edited facts.

SERAC (Mitchell et al., 2022b) stores edits in a sperate edit memory and employs a scope classifier to determine whether a query edit can be considered as in-scope examples within the edit memory. If a query edit is classified to be in-scope, SERAC uses a counterfactual model. Otherwise, SERAC uses the frozen base model for a given query edit.

Similar to MIKE, SERAC also maintains an edit memory, differently reacts to in-scope and out-of-scope examples, and uses a scope classifier. However, SERAC has not been scaled up to the massive editing task. In addition, SERAC requires to additionally train the parametric counterfactual model and scope classifier, whereas MIKE does not use any PU method but rely on the ICL mechanism only.

While DistillMIKE is considered as a PU method, but to the best of our knowledge, our work on DistillMIKE is the first in using distillation methods to knowledge editing task.

3 Task Definition

To formally define the massive editing task, suppose that $\mathcal{M}$ is an autoregressive language model, $\mathcal{M}(x)$ is the output generated by the decoding step given a prefix sequence x, and new factual knowledge to be injected to $\mathcal{M}$ is represented as a set of *relational* triples; More specifically, $\mathcal{S}$ is a real-world entities or concepts, $\mathcal{R}$ is a set of relations, and $\mathcal{E} = \{e_i\}_{i=1}^{N}$ is a set of edits (or facts) to be injected to $\mathcal{M}$, where $e_i = (s_i, r_i, o_i^*)$ is the i-th edit, i.e., a relational triple that consists of a subject $s_i \in \mathcal{S}$, a relation $r_i \in \mathcal{R}$, and an object $o_i^* \in \mathcal{S}$. It is commonly assumed that $\mathcal{M}$ does not contain each fact e_i precisely; given a prefix $x_i = (s_i, r_i)$ as the prompt input, $\mathcal{M}(x_i) = o_i$ is usually not equal to the target object o_i^*, i.e., $o_i \neq o_i^*$ for most i.

The goal of knowledge editing is to obtain an edited model $\mathcal{M}^*$ towards satisfying efficacy, generalization, and specificity, for "all" edits:

- **Efficacy** holds if $\mathcal{M}^*(s_i, r_i) = o_i^*$ for $(s_i, r_i) \in \mathcal{E}$.
- **Generalization** is satisfied if $\mathcal{M}^*(s_i', r_i') = o_i^*$ for a "paraphrased" prefix $(s_i', r_i') \in \mathcal{I}(e_i)$ where $\mathcal{I}(e_i)$ is *edit scope* of e_i, the set of *in-scope* examples.
- **Specificity** (or **Locality**) holds if $\mathcal{M}^*(s, r) = \mathcal{M}(s, r)$ for any irrelevant prefix $(s, r) \in \mathcal{O}(e_i)$ where $\mathcal{O}(e_i) = \mathcal{U} - \mathcal{I}(e_i)$ is the set of *out-of-scope* examples, given that $\mathcal{U}$ is a *universal* set of knowledge.

Examples of an edit, its prefix, in-scope, out-of-scope prefixes, and their correct objects are presented in Appendix B.

4 Method

Figure 1 presents the overall architecture of our proposed MIKE and DistillMIKE, with the brief sketch of their construction below:

- **Inducing MIKE as a teacher model**: i) At the training time, given $\mathcal{E}$, a set of test edits (i.e. new facts), MIKE merely maintains $\mathcal{E}$ in an external "edit memory", without updating parameters. ii) At the inference time, given a query prompt $q = (s, r)$, MIKE first applies a **fact retrieval** function $\mathsf{Ret}(q)$, which returns the best-matched fact $e_q \in \mathcal{E}$, called a *query-matched fact*, otherwise $\mathsf{Ret}(q)$ returns a null,

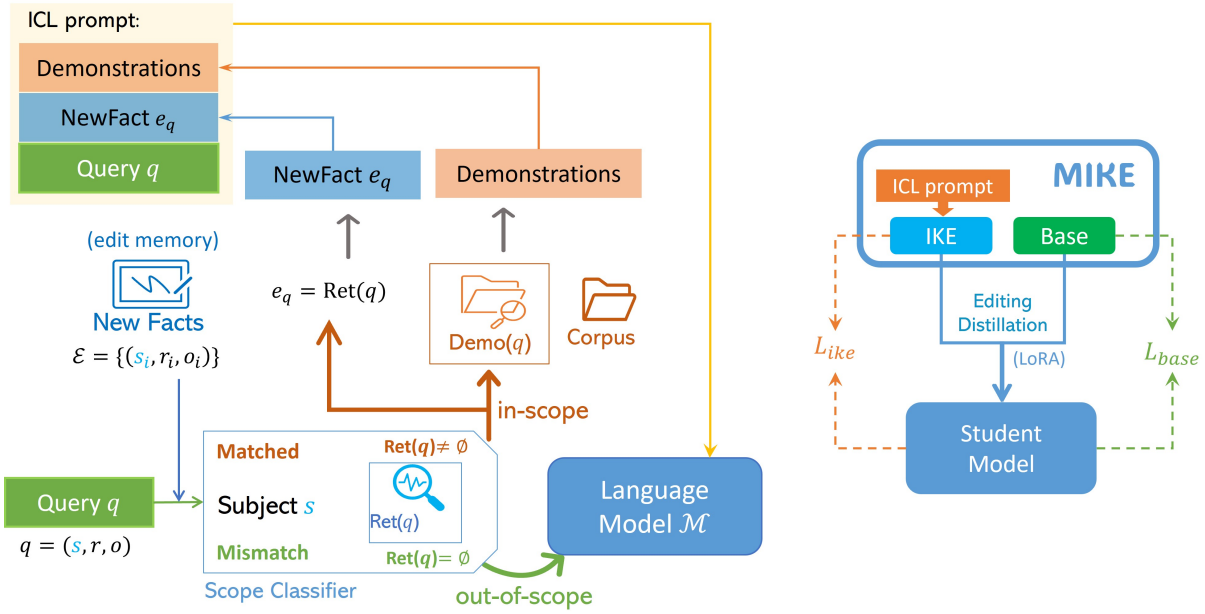

Figure 1: The overall architecture of MIKE and DistillMIKE. (a) **MIKE**: At the training step, MIKE merely stores all the massive "test" edits $\mathcal{E} = \{e_i\}_{i=1}^{N}$ in the edit memory. At the inference time, given a query prompt $q = (s, r)$, MIKE first performs the fact retrieval $\text{Ret}(q)$ to retrieve a "new fact prompt", called the query-matched fact, e_q (in Section 4.1.1). MIKE behaves differently for in-scope and out-of-scope queries (i.e., Eq (1)); for an in-scope case (i.e., $\text{Ret}(q) \neq \emptyset$), MIKE further calls the demonstration selection component $\text{Demo}(q)$ (in Section 4.1.2), and the resulting demonstrations are further concatenated with a query q, and then they are fed to the decoding process, resulting in $\mathcal{M}(\text{Demo}(q); q)$; for an out-of-scope case (i.e., $\text{Ret}(q) = \emptyset$), without demonstration selection, a query q is only fed to the decoding process, merely giving $\mathcal{M}(q)$. (b) **DistillMIKE**: Editing distillation is performed by taking MIKE as a teacher model and initializing a student model by the base model. Inherited from the selective retrieval augmentation of MIKE, DistillMIKE results from a multi-teacher distillation by taking IKE and the unedited base model as in-scope and out-of-scope teachers, respectively, thereby decomposing L_{kd} of Eq. (3) into L_{ike} and L_{base} (in Eq. (5) in Section 4.2).

i.e. $\text{Ret}(q) = \emptyset$. A simple scope classification is then performed; q is classified to an out-of-scope case when $\text{Ret}(q) = \emptyset$, otherwise q becomes an in-scope case.

MIKE acts differently for in-scope and out-of-scope cases; for an in-scope case, MIKE prepares in-context *demonstrations* selected from a set of "training" edits, denoted by $\text{Demo}(q)$, and prepend them to a query prefix x; on the other hand, for an out-of-scope case, no demonstration is provided in a prompt. The resulting prompt with or without demonstrations is fed to $\mathcal{M}$ to finally predict a output sequence. With this selective retrieval augmentation. The inference process of MIKE is summarized as follows:

$$\mathcal{M}^*(q) = \begin{cases} \mathcal{M}(\text{Demo}(q); q), & \text{Ret}(q) \neq \emptyset \\ \mathcal{M}(q), & \text{Otherwise.} \end{cases} \quad (1)$$

- **Training DistillMIKE by editing distillation using LoRA**: Inspired by the work of (Choi et al., 2023), we distill MIKE to a *student* model $\mathcal{M}_\theta^{st}$ with parameters θ, based on a LoRA-based PU method, for eliminating the need for "lengthy" demonstrations for in-scope cases. Given a set of "test" edits $\mathcal{E}$, we use its in-scope and out-of-scope test edits as training dataset for editing distillation, denoted by $\mathcal{I}(\mathcal{E}) = \cup_{e_i \in \mathcal{E}} \mathcal{I}(e_i)$ and $\mathcal{O}(\mathcal{E}) = \cup_{e_i \in \mathcal{E}} \mathcal{O}(e_i)$, respectively.

  Let $p_{\mathcal{M}}(y|x)$ be the generative probability of a sequence y given the prefix x, computed by $\mathcal{M}$. The loss function used for editing distillation is summarized as follows:

$$\begin{aligned} L_{ed} &= KL\left(p_{\mathcal{M}^*} \middle\| p_{\mathcal{M}_\theta^{st}}\right) \quad (2) \\ &\approx \sum_{q=(s,r)\in\mathcal{E}'} \mathbb{E}_{o \sim p_{\mathcal{M}^*}(\cdot|q)} \log \frac{p_{\mathcal{M}_\theta^{st}}(o|q)}{p_{\mathcal{M}^*}(o|q)} \end{aligned}$$

where $\mathcal{E}' = \mathcal{I}(\mathcal{E}) \cup \mathcal{O}(\mathcal{E})$ indicates a whole set of training examples used for editing distillation.

To train $\mathcal{M}_\theta^{st}$ under Eq. (3), we undertake the *LoRA* finetuning method of Hu et al. (2022) for substantially reducing the number of parameters to be updated.

4.1 Teacher Model: Massive In-context Knowledge Editing (MIKE)

As in Eq. (1), the inference step of MIKE employs Ret, the "fact retrieval" function and Demo, the "demo selection" module, whose details are presented as follows:

4.1.1 Fact retrieval: $\mathsf{Ret}(q)$

Given a query prompt $q = (s, r)$, $\mathsf{Ret}(q)$ tries to find a query-matched fact e_q by matching subject and relation parts of q maximally. The matching for the fact retrieval is divided into four cases:

1) an *exactly matched case* with $e_q = e_i$ where $e_i = (s_i, r_i, o_i) \in \mathcal{E}$ is exactly matched with a subject and a relation of q, i.e., $(s_i, r_i) = (s, r)$.
2) an *uniquely subject-matched case* with $e_q = e_i$ where $e_i = (s_i, r_i, o_i) \in \mathcal{E}$ is uniquely matched with a subject of q, i.e., $s_i = s$, while no other test edits match the subject s.
3) a *subject-matched but ambiguous case* with $e_q = \mathsf{top-1}(q, \mathcal{E}_s$ where e_q is the nearest test edit among all the subject-matched ones, i.e., given $\mathcal{E}_s = \{e_i = (s_i, r_i, o_i) | s_i = s\}$, an additional dense retrieval is employed to compute the similarities between an edit e_i and a query for finding the nearest edit.
4) an *unmatched case* with $e_q = \emptyset$, where no edits match a subject part s of q.

An illustrated example of the fact retrieval is described in Figure 3, with more details of the process in in Figure 3.

4.1.2 Demonstration selection: $\mathsf{Demo}(q)$

As in Eq. (1), MIKE requires the demonstration selection $\mathsf{Demo}(q)$ for in-scope examples when $\mathsf{Ret}(q) = e_q \neq \emptyset$. To be more specific, given $\mathcal{E}$ a set of test edits, MIKE keeps a separate set of "training" edits, denoted as $\mathcal{E}^{tr} = \left\{e_j^{tr}\right\}_{j=1}^{M}$. Unless otherwise stated, an edit refers a "test" edit, not being a "training" edit. Different from a test edit, each j-th training edit e_j^{tr} is pre-associated with a set of demonstrations, referred to as $\mathcal{D}(e_j^{tr})$. Similar to (Liu et al., 2022), $\mathsf{Demo}(q)$ first finds the top-k "training" edits which are most similar to a query-matched fact e_q, i.e., $e_1', \cdots, e_k'$ where $e_i' \in \mathcal{E}^{tr}$. Then, the *demonstration filter* g is further applied to demonstrations $\mathcal{D}(e_i')$ of each i−-th nearest training edits, resulting in $g(\mathcal{D}(e_i'))$. All the filtered demonstrations are concatenated and then prepended as a prompt prefix to the given query q, which is feed to $\mathcal{M}$ to finally produce a predicted output. Summing up, the formal definition of $\mathsf{Demo}(q)$ is given as follows:

$$\begin{aligned} e_1', \cdots, e_k' &= \mathsf{kNN}\left(e_q, \mathcal{E}^{tr}\right) \\ \mathsf{Demo}(q) &= \left[g(\mathcal{D}(e_1')); \cdots; g(\mathcal{D}(e_k'))\right] \end{aligned}$$

where $\mathsf{kNN}\left(e_q, \mathcal{E}^{tr}\right)$ is an additional retrieval function that finds the top-k nearest neighbors in the "training" edits $\mathcal{E}^{tr}$, which are the most similar to e_q; a kind of dense retrieval is deployed to compute the similarities between a training edit e_j^{tr} and e_q, whose details are presented in Appendix C.

Scope-aware demonstration filtering: g For the demonstration filter g, we propose the *scope-aware demonstration filter* for g, by using only "update" types of demonstrations, not including other types such as"retrain" types. This use of scope-aware demonstrations is motivated by the selective retrieval nature of MIKE where IKE is applied only for in-scope cases, not for out-of-scope cases. Because IKE is not applied for out-of-scope cases, it is like that excluding "retrain" types of demonstrations may not negatively impact the editing performance.

To formally describe the scope-aware filtering manner in g, for j-th training edit e_j^{tr}, its demonstrations $\mathcal{D}(e_j^{tr})$ further consists of three types of demonstrations, $\mathcal{D}^{cp}(e_j^{tr})$, $\mathcal{D}^{ud}(e_j^{tr})$, and $\mathcal{D}^{rt}(e_j^{tr})$, which correspond to sets of *copy*, *update* and *retrain* types, respectively[1], i.e., $\mathcal{D}(e_j^{tr}) = (\mathcal{D}^{cp}(e_j^{tr}), \mathcal{D}^{up}(e_j^{tr}), \mathcal{D}^{tr}(e_j^{tr}))$. In the proposed scope-aware demonstration filter, $g\left(\mathcal{D}(e_j^{tr})\right)$ is defined as follows:

$$g\left(\mathcal{D}(e_j^{tr})\right) = \mathcal{D}^{up}(e_j^{tr}) \tag{3}$$

The detailed examples of three types of demonstrations are presented in Appendix H.

[1]Here, the demonstration types of copy, update and retrain correspond to the "requested", "paraphrased", and "neighborhood" prompts in CounterFact dataset, respectively.

4.2 DistillMIKE

While MIKE presents promising results showing the state-of-the-art performances on the massive editing task as in Section 6, however, the major drawback is its additional computational overhead at the inference time, mainly caused by the increased prompts from a number of demonstrations. Inspired from (Choi et al., 2023), we would like to eliminate the need of using lengthy sequence of demonstrations, thus propose "editing distillation" from MIKE to a student model, such that the resulting student model, i.e., DistillMIKE, does not need to prepend a lengthy sequence of demonstrations but only use an input query prompt.

To substantially reduce the training cost for distillation, a student model is initialized by MEMIT, and is then fine-tuned using LoRA (Hu et al., 2022), where only a very small portion of parameters need to be updated. It is noticeable that the proposed editing distillation of MIKE to a student model is also somehow motivated by the previous studies of viewing ICL as gradient descent methods (Von Oswald et al., 2023; Dai et al., 2023), while their results are induced under rather limited settings such as the linear attention and the few-shot learning abilities of ICL.

Because MIKE consists of two sub-models -– IKE and the unedited base model, the editing distillation applied to DistillMIKE can be seen as a *multi-teacher* distillation, similarly in (Wu et al., 2021; Liu et al., 2020). Suppose that training examples used for editing distillation $\mathcal{E}'$ are automatically classified into in-scope and out-of-scope examples, based on the scope classifier using $\mathsf{Ret}(q)$, denoted as $\mathcal{I}'$ and $\mathcal{O}'$, respectively, defined as follows.

$$\begin{aligned} \mathcal{I}' &= \left\{ e \in \mathcal{E}' \;\middle|\; \mathsf{Ret}(e) \neq \emptyset \right\} \\ \mathcal{O}' &= \mathcal{E}' - \mathcal{I}' \end{aligned} \tag{4}$$

The loss function of Eq. (3) is rewritten as:

$$L_{ed} = L_{ike} + L_{base} \tag{5}$$

$$L_{ike} \propto \sum_{\substack{q=(s,r)\in\mathcal{I}' \\ d=\mathsf{Demo}(q)}} \mathbb{E}_{o\sim p_{\mathcal{M}}(\cdot|d;q)} \log \frac{p_{\mathcal{M}^{st}_{\theta}}(o|q)}{p_{\mathcal{M}}(o|d;q)}$$

$$L_{base} \propto \sum_{q=(s,r)\in\mathcal{O}'} \mathbb{E}_{o\sim p_{\mathcal{M}}(\cdot|q)} \log \frac{p_{\mathcal{M}^{st}_{\theta}}(o|q)}{p_{\mathcal{M}}(o|q)}$$

Therefore, under Eq. (5), it is clearly seen that for a given query q, DistillMIKE results from two teacher models – the IKE loss L_{ike} from $p_{\mathcal{M}}(o|\mathsf{Demo}(q);q)$ and the base loss L_{base} from $p_{\mathcal{M}}(o|q)$, depending on whether $q \in \mathcal{I}'$ or $q \in \mathcal{O}'$.

5 Experiments

5.1 Dataset and Metrics

We evaluate MIKE and DistillMIKE on the **Zero-Shot Relation Extraction** (zsRE, Levy et al. (2017)) dataset and the **CounterFact** dataset (Meng et al., 2022a) with 10,000 knowledge edits. The details and formats of the datasets are outlined in the Appendix H.

For zsRE, three metrics – **Efficacy**, **Paraphrase**, and **Specificity** – are employed to measure the editing capability concerning editing requests, paraphrases, and the retaining capability for out-of-scope examples, respectively. The detailed definitions of these metrics are provided in the Appendix J.

For CounterFact, similar to the three metrics used in zsRE, we compute the **Efficacy Score (ES)**, **Paraphrase Score (PS)**, and **Neighborhood Score (NS)** to evaluate editing accuracy (on editing requests and paraphrase prompts) or retaining accuracy (on neighborhood prompts), respectively. In addition, we report their mean difference in magnitude terms: **Efficacy Magnitude (EM)**, **Paraphrase Magnitude (PM)**, and **Neighborhood Magnitude (NM)**, which measure the *significance* of editing. The aggregated **Score (S)** is calculated as the harmonic mean of ES, PS, and NS. Detailed definition can be seen in Appendix J.

The implementation details are provided in Appendix I.

5.2 Settings and Baselines

We use the GPT-J 6B model (Wang and Komatsuzaki, 2021), a widely employed backbone model in relevant works, for comparing MIKE and DistillMIKE with various existing knowledge-editing methods. In DistillMIKE we adopt MEMIT as an initial student model, because it gives us a good starting point in terms of editing performances, as discussed in Section 6.2. The baseline methods include fine-tuning and four PU methods – **MEND** (Mitchell et al., 2022a), **ROME** (Meng et al., 2022a), **MEMIT** (Meng et al., 2022b), and **PMET** (Li et al., 2023). We also included the instance-level ICL-based method **IKE** (Zheng et al., 2023).

Method	Score ↑	Efficacy ↑	Paraphrase ↑	Specificity ↑
GPT-J	26.4	26.4	25.8	27.0
FT	42.1	69.6	64.8	24.1
MEND	20.0	19.4	18.6	22.4
ROME	2.6	21.0	19.6	0.9
MEMIT	50.7	96.7	89.7	26.6
PMET	51.0	96.9	90.6	26.7
MIKE	**52.6**	**99.9**	**99.6**	**27.0**
DistillMIKE	52.2	98.5	97.0	**27.0**

Table 1: Performance comparison of GPT-J (6B) with 10,000 knowledge edits on zsRE dataset. *Score* is the harmonic mean of *Efficacy*, *Paraphrase* and *Specificity*. Column-wise best are in bold, second best are underlined.

Method	Score	Efficacy		Generalization		Specificity	
	S ↑	ES ↑	EM ↑	PS ↑	PM ↑	NS ↑	NM ↑
GPT-J	20.47	14.66	-7.40	15.06	-7.50	83.97	7.65
FT	63.54	99.91	**98.24**	88.14	48.65	38.67	-8.22
MEND	25.23	17.61	-12.19	20.10	-11.34	**80.83**	12.55
ROME	49.92	49.36	-0.03	49.51	-0.09	50.92	0.09
MEMIT	85.71	99.10	87.85	88.33	38.02	73.59	4.64
IKE	84.88	**99.98**	92.86	96.29	67.37	66.88	**25.19**
PMET	86.20	99.50	-	92.80	-	71.40	-
MIKE	**90.87**	99.48	95.58	**98.99**	**83.29**	77.76	2.75
DistillMIKE	89.53	97.10	73.09	95.36	63.04	78.61	13.83

Table 2: Performance comparison of GPT-J (6B) with 10,000 knowledge edits on CounterFact dataset. Score *S* is the harmonic mean of *ES*, *PS* and *NS*. Column-wise best are in bold, second best are underlined.

6 Results

6.1 Main Results

6.1.1 Editing 10k knowledge in zsRE.

Table 1 presents comparison results of MIKE and DistillMIKE and other baselines on the zsRE dataset. MIKE achieves the best overall performance (**Score**), with particularly noticeable improvement in *Paraphrase*. DistillMIKE also achieves improvements over the baseline models, while showing slightly lower performance compared to its teacher MIKE.

GPT-J, the original unedited model shows a Specificity score of 27.0 only on the zsRE dataset, thus it is likely expected that Specificity is not very much improved varying knowledge editing methods.

6.1.2 Editing 10k knowledge in CounterFact.

Table 2 presents the comparison results of MIKE and DistillMIKE and baselines in terms of both accuracy metrics **ES, PS, NS** and the magnitude metrics **EM, PM, NM**.

For the comparison based on GPT-J, the proposed MIKE and DistillMIKE significantly outperform all baselines in terms of overall performance (**Score**). MIKE and DistillMIKE demonstrate substantial improvements in Generalization and noticeably in Specificity. The fine-tuned (FT) model performs well in Efficacy and Generalization but shows a substantial deterioration in Specificity. While MEND also shows admirable Specificity, it is largely weak in its other editing capabilities in the case of 10,000 edits. IKE exhibits strong performances on Efficacy and Generalization but underperforms in Specificity. We clearly observe the substantial performance gaps in Generalization between DistillMIKE and other baseline methods such as MEMIT, PMET, and ICL-based methods (IKE), and the results confirm the effectiveness of editing distillation again.

6.2 Ablation Study

In Table 3, we further examine the effect of using the in-scope and out-of-scope teachers on editing distillation, denoted by MIKE$_{ike}$ and MIKE$_{base}$, respectively, comparing to the MEMIT-initialized student model; MIKE$_{ike}$ and MIKE$_{base}$ are the

Model	S ↑	ES ↑	PS ↑	NS ↑
MIKE$_{ike}$	25.02	100	99.78	10.01
MIKE$_{base}$	21.93	14.66	17.65	83.97
Student	85.71	99.10	88.33	73.59
DistillMIKE				
- L_{ike}	88.37	99.25	98.45	72.91
- $L_{ike} + L_{base}$	89.53	97.10	95.36	78.61

Table 3: Ablation study of DistillMIKE on CounterFact. MIKE$_{ike}$ is the in-scope teacher, MIKE$_{base}$ is the out-of-scope teacher (base model). "Student" refers to a student model before editing distillation, initialized by MEMIT. DistillMIKE with L_{ike} is the run of performing the partial distillation with L_{ike} only in Eq. (5).

Model	Student	DistillMIKE	ΔNS ↓
GPT-J	83.97	51.48	32.49
6000 edits	78.29	55.49	22.80
10000 edits	73.59	72.91	0.68

Table 4: Performances of NS before and after the partial editing distillation with L_{ike} across different settings of MEMIT. "Student" refers to variants of MEMIT differing the number of edits among 0 (GPT-J), 6,000, 10,000 edits. "DistillMIKE" are the corresponding post-distilled runs after performing the partial editing distillation to these student models. ΔNS is the difference in NS between before and after the editing distillation.

runs of using $\mathcal{M}(\mathsf{Demo}(q); q)$ (using our scope-aware demonstrations) and $\mathcal{M}(q)$ in Eq. (1) for "all" test queries, regardless their scope types.

MIKE$_{ike}$ achieves near-perfect performances in ES and PS (i.e., Efficacy and Generalization), while showing a very low value of NS (i.e., Specificity). In contrast, as expected, MIKE$_{base}$ presents almost upper bound of NS, while showing low values of ES and PS. MEMIT, as the initial student model, exhibits a balanced performance of three metrics, however, its PS and NS performances are still far lower than those of MIKE$_{ike}$ and MIKE$_{base}$, respectively.

Given its improved performance in Table 1-2, it is expected that DistillMIKE benefits by integrating distinct abilities from both worlds of two teachers, i.e., learning the "editing" ability from MIKE$_{ike}$ and the "retaining" ability from MIKE$_{base}$ during editing distillation. Table 3 further shows the result of performing the "partial" distillation using only MIKE$_{ike}$ with L_{ike} in Eq. (5. Compared to the full-fledged version (i.e., $L_{ike} + L_{base}$), there is a substantial decrease in NS, indicating that learning the "retraining" ability is not sufficiently done during distillation, using L_{ike} alone.

We believe that the use of MEMIT as a student model leads to a good starting point, where MEMIT already pursues a balancing mechanism in a manner of keeping the "retrain" ability, and thus the subsequent editing distillation does not seriously cause the *catastrophic forgetting* problem for out-of-scope examples.

To examine the effect of using MEMIT as a student model, Table 4 presents the performances of NS before and after applying the partial editing distillation, when using different versions of MEMIT (including GPT-J) as student models varying the number of edits. It is shown that as the number of edits imposed to MEMIT is increasing (i.e., more extensively "surgical" pre-parametric updates are applied), the corresponding LoRA-distilled DistillMIKE keeps NS performances with greater stability. Thus, the results confirm that the choice of using MEMIT as a student model is indeed necessary to achieve the balanced editing performances for generalization and specificity.

7 Conclusion

In this paper, we proposed MIKE, extending previous in-context knowledge editing work from the instance level to a massive scale. Furthermore, we conducted editing distillation of MIKE to induce DistillMIKE, which implicitly injects ICL prompts into model parameters, such that DistillMIKE greatly saves the computational overhead caused by lengthy demonstration prompts. Extensive experiments conducted on the zsRE and CounterFact datasets demonstrated that MIKE and DistillMIKE surpassed existing knowledge editing methods, achieving state-of-the-art overall performance.

In future work, we would like to invent a direct unified approach for inducing DistillMIKE, motivated from the existing works reveal that ICL can be somehow equivalently projected to gradient descent methods (Von Oswald et al., 2023; Dai et al., 2023). Noting that incremental learning on knowledge editing is arguably important, it would be worthy to generalize the current editing distillation towards "continual" distillation, given the stream of sets of new edits. It would be also interesting to evaluate MIKE on other knowledge editing datasets such as MQuAKE (Zhong et al., 2023).

Limitations

Current models primarily process data samples presented in tuple form, such as $(subject, relation, object)$, while real-world natural language exhibits more diversity and complexity. Investigating whether the current research can extend its applicability to universal text formats is an important issue for future exploration. Additionally, the scope classifier proposed and utilized in our wokr is currently a straightforward method applicable to the existing dataset. For more diverse data, further exploration is required to develop more intricate and advanced scope classifiers.

In practical applications, the iterative updates of a model necessitate incremental knowledge editing. This involves continuing to edit additional new knowledge on a previously edited model. Therefore, exploring whether massive knowledge editing can be stably performed in an incremental iterative manner is also an important avenue for future work. We would also like to conduct experiments on larger-scale models, such as GPT-NeoX 20B and models from the Llama (Touvron et al., 2023) family, to investigate the effectiveness of deploying massive knowledge editing on larger models.

Furthermore, real-world questions are interrelated, and for question-answering tasks, practical questions often exhibit multi-hop characteristics. However, exploration in the realm of multi-hop question-answering is still limited in existing literature. We also aim to explore the application of massive knowledge editing in multi-hop question-answering, such as extending it to the MQuAKE (Zhong et al., 2023) dataset.

Finally, existing knowledge editing datasets are insufficient in scale and contain a considerable amount of noisy data. In the future, we will strive to create larger and more comprehensive dataset to explore knowledge editing on a larger scale.

References

Oshin Agarwal and Ani Nenkova. 2022. Temporal effects on pre-trained models for language processing tasks. *Transactions of the Association for Computational Linguistics*, 10:904–921.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models.

Eunbi Choi, Yongrae Jo, Joel Jang, Joonwon Jang, and Minjoon Seo. 2023. Fixed input parameterization for efficient prompting. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8428–8441, Toronto, Canada. Association for Computational Linguistics.

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.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019, Toronto, Canada. Association for Computational Linguistics.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022a. Calibrating factual knowledge in pretrained language models. *Findings of Empirical Methods in Natural Language Processing (EMNLP)*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022b. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. 2023. Bias and fairness in large language models: A survey.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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

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 Comput. Surv.*, 55(12).

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. 2021. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34:29348–29363.

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.

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

Adam Liška, Tomáš Kočiský, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, Cyprien de Masson d'Autume, Tim Scholtes, Manzil Zaheer, Susannah Young, Ellen Gilsenan-McMahon Sophia Austin, Phil Blunsom, and Angeliki Lazaridou. 2022. Streamingqa: A benchmark for adaptation to new knowledge over time in question answering models. *arXiv preprint arXiv:2205.11388*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Yuang Liu, Wei Zhang, and Jun Wang. 2020. Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing*, 415:106–113.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memprompt: Memory-assisted prompt editing with user feedback.

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, Christopher D Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.

Yasumasa Onoe, Michael Zhang, Eunsol Choi, and Greg Durrett. 2022. Entity cloze by date: What LMs know about unseen entities. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 693–702, Seattle, United States. Association for Computational Linguistics.

Yasumasa Onoe, Michael Zhang, Shankar Padmanabhan, Greg Durrett, and Eunsol Choi. 2023. Can LMs Learn New Entities from Descriptions? Challenges in Propagating Injected Knowledge. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5469–5485, Toronto, Canada. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. *arXiv preprint arXiv:2005.04611*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language model via meta learning. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 35151–35174. PMLR.

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

Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. 2021. One teacher is enough? pre-trained language model distillation from multiple teachers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4408–4413, Online. Association for Computational Linguistics.

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. *CoRR*, abs/2305.13172.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

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? *arXiv preprint arXiv:2305.12740*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

A Details of MEMIT

Note the MLP weights in a Transformer (Vaswani et al., 2017) as W that can be operated as a key-value store, where $WK \approx V$, $K = [k_1|k_2|...]$ and $V = [v_1|v_2|...]$. Given requested edits $\mathcal{E} = \{(s_i, r_i, o_i)\}$, language model $\mathcal{M}_\theta$, layers to edit $\mathcal{L} = \{L_1, L_2, ..., L_l\}$, and pre-cached covariance constant C^L of k computed from Wikipedia samples (Meng et al., 2022a). For each $(s_i, r_i, o_i) \in \mathcal{E}$, a target vector z_i will be computed:

$$z_i \leftarrow h_i^{L_l} + \delta_i, \tag{6}$$

where δ_i is optimized by:

$$\delta_i \leftarrow \arg\min_{\delta_i} \frac{1}{P} \sum_{j=1}^{P} \xi_i$$

$$\xi_i = -\log \mathbb{P}_{\mathcal{M}(h_i^{L_l} += \delta_i)}[o_i | x_j \oplus (s_i, r_i)] \tag{7}$$

Then for each editing layer $L \in \mathcal{L}$, the hidden state is updated by:

$$h_i^L \leftarrow h_i^{L-1} + a_i^L + m_i^L \tag{8}$$

where a and m denote the *"attention"* and *"MLP"* contributions computed from previous layers in Transformer (Vaswani et al., 2017) model. On the current layer, for each $(s_i, r_i, o_i) \in \mathcal{E}$, the MLP key updated as:

$$k_i^L \leftarrow k_i^L = \frac{1}{P} \sum_{j=1}^{P} k(x_j + s_i) \tag{9}$$

where x_j are random prefixes that aid generalization across contexts. The distributed residual ϕ over remaining layers is computed as:

$$\phi_i^L \leftarrow \frac{z_i - h_i^{L_l}}{l - idx(L) + 1} \tag{10}$$

where $idx(L)$ denote the number index of L. Thus in this layer $k^L = \{k_i^L\}$ and $\phi^L = \{\phi_i^L\}$.

To update the MLP weights in the editing layers, for each layer $L \in \mathcal{L}$, the adding weight is computed as:

$$\Delta^L \leftarrow \phi^L k^{L\mathsf{T}} (C^L + k^L k^{L\mathsf{T}})^{-1}, \tag{11}$$

finally in current layer L the MLP weights updated as:

$$W^L \leftarrow W^L + \Delta^L, \tag{12}$$

after the above updating performed on all the editing layers, we can obtain the parametric updated model $\mathcal{M}_{\theta^*}$.

B Edit Examples

Here we present more examples in CounterFact, where edits the factual answer to a pseudo-factual:
example1:

- Editing request e_i: "*The 46th president of the US is Biden*".
- Editing Prefix (s_i, r_i): "*The 46th president of the US is*". Predict o^*: "*Biden*".
- In-scope prefix (Paraphrase) $(s_i', r_i') \in \mathcal{I}(e_i)$: "*The winner of the 46th US presidential election is*". Predict o^*: "*Biden*".
- Out-of-scope prefix $(s, r) \in \mathcal{O}(e_i)$: "*The president of Colombia is*". Predict o: "*Petro*".

example2:

- Editing request e_i: "*Fiat Multipla is a product of IBM*".

- Editing prefix (s_i, r_i): "*Fiat Multipla is a product of*". Predict o^*: "*IBM*".
- In-scope prefix $(s_i', r_i') \in \mathcal{I}(e_i)$: "*Fiat Multipla is produced by*". Predict o^*: "*IBM*".
- Out-of-scope prefix $(s, r) \in \mathcal{O}(e_i)$: "*Fiat Brevetti is created by*". Predict o: "*Fiat*".

example3:

- Editing request e_i: "*Tor Endresen, who is a citizen of Nigeria*".
- Editing prefix (s_i, r_i): "*Tor Endresen, who is a citizen of*". Predict o^*: "*Nigeria*".
- In-scope prefix $(s_i', r_i') \in \mathcal{I}(e_i)$: "*Tor Endresen holds a citizenship from*". Predict o^*: "*Nigeria*".
- Out-of-scope prefix $(s, r) \in \mathcal{O}(e_i)$: "*Leonhard Hess Stejneger, a citizen of*". Predict o: "*Norway*".

example4:

- Editing request e_i: "*Kirsti Huke plays opera*".
- Editing prefix (s_i, r_i): "*Kirsti Huke plays* ". Predict o^*: "*opera*".
- In-scope prefix $(s_i', r_i') \in \mathcal{I}(e_i)$: "*Kirsti Huke performs*". Predict o^*: "*opera*".
- Out-of-scope prefix $(s, r) \in \mathcal{O}(e_i)$: "*Zeena Parkins performs*". Predict o: "*jazz*".

C Details of kNN Function

We use the dense retrieval for kNN based on the cosine similarity between the training edit e_j^T and the given requested edit e_i. More precisely, suppose that $\mathcal{M}_{sent}$ is an additional sentence encoder, where $\mathcal{M}_{sent}(s) \in \mathbb{R}^d$ is the sentence vector for a given sentence s. For notational convenience, given an edit $e = (s, r, o)$, $\mathcal{M}_{sent}(e) = \mathcal{M}_{sent}([s; r; o])$ where $[s; r; o]$ is the natural language format that concatenates s, r, and o using a proper verbalizing template. The similarity between $e = (s, r, o)$ and $e' = (s, r, o)$ is defined as follows:

$$sim(e, e') = cos(\mathcal{M}_{sent}(e), \mathcal{M}_{sent}(e')) \quad (13)$$

For a given edit $e_i \in \mathcal{E}$, $\mathsf{kNN}\,(e_i, \mathcal{T})$ is defined as follows:

$$top\text{-}k \left\{ (e_j^t, sim(e_i, e_j^t)) \right\}_{j=1}^{M} \quad (14)$$

where $top\text{-}k$ is the operator for selecting top-k elements given the set of pairs of objects and their associated similarities. For $\mathcal{M}_{sent}$, we deploy the pre-trained sentence encoder (Reimers and Gurevych, 2019).

D Ablation: KL-Divergence

We further present the KL divergences on ES, PS, and NS between the MEMIT-initialized student models under different scales of parametric updating and the teacher model of full DistillMIKE in Figure 2. Here, the KL divergences on ES, PS, and PS are computed from Eq. (3), but summing over three types of query examples – *copy-scope*, in-scope, and out-of-scope examples, corresponding to $\mathcal{E}$, $\mathcal{O}\left(\mathcal{E}\right)$, and $\mathcal{O}\left(\mathcal{E}\right)$, respectively.

$$KL_{ES} = \sum_{q=(s,r)\in\mathcal{E}} \mathbb{E}_{o\sim p_{\mathcal{M}^*}(\cdot|q)} \log \frac{p_{\mathcal{M}_\theta^{st}}(o|q)}{p_{\mathcal{M}^*}(o|q)}$$

$$KL_{PS} = \sum_{q=(s,r)\in\mathcal{I}(\mathcal{E})} \mathbb{E}_{o\sim p_{\mathcal{M}^*}(\cdot|q)} \log \frac{p_{\mathcal{M}_\theta^{st}}(o|q)}{p_{\mathcal{M}^*}(o|q)}$$

$$KL_{NS} = \sum_{q=(s,r)\in\mathcal{O}(\mathcal{E})} \mathbb{E}_{o\sim p_{\mathcal{M}^*}(\cdot|q)} \log \frac{p_{\mathcal{M}_\theta^{st}}(o|q)}{p_{\mathcal{M}^*}(o|q)} \quad (15)$$

(16)

It demonstrates that when using more extensive pre-parametric updating, the distribution gap between the output logits of the student and teacher models decreases. This also provides evidence that pre-parametric updating leads to faster convergence and better distillation results.

E ICL Demonstration

The ICL demonstration as shown in Table 5.

Type	Demonstration
update	New Fact: Willy Brandt, who is employed by Boeing Prompt: Willy Brandt, who works for Boeing
New Fact	Prompt: Willy Brandt, who is employed by Boeing
query	Prompt (edit): Willy Brandt, who is employed by? Prompt (paraphrase): Willy Brandt worked in? Prompt (neighborhood): Joseph Reinach used to worked in?

Table 5: Single example of our demonstration.

F Detailed Illustration of Fact Retrieval of $\mathsf{Ret}(q)$

Figure 3 describes the detailed illustration of the fact retrieval function $\mathsf{Ret}(q)$.

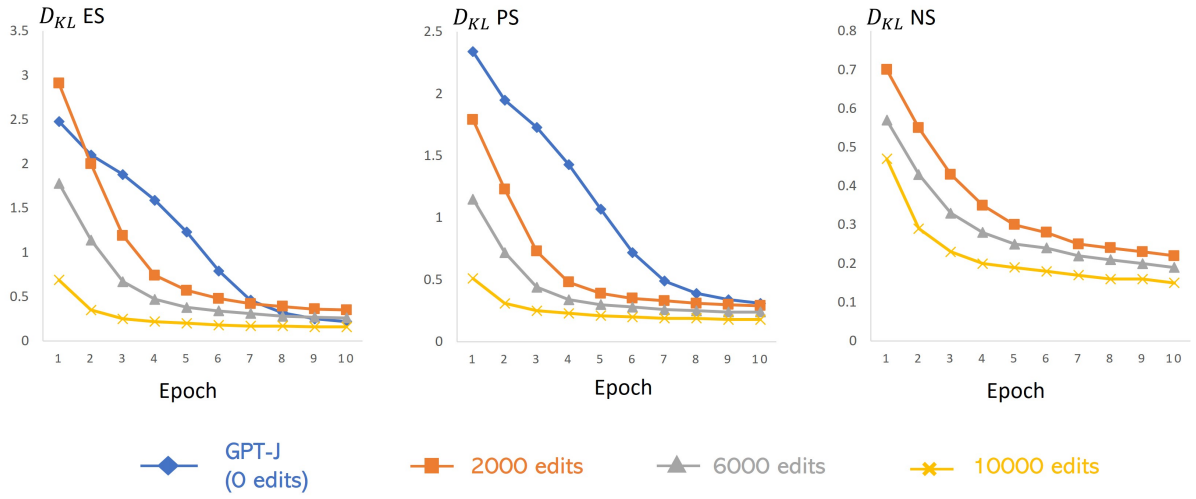

Figure 2: Plots of KL-divergences on ES, PS and PS using Eq. (16) between the teacher MIKE and the MEMIT-initialized student model under different parametric updating scales, varying number of edits imposed on the MEMIT-based initialization. The more edits are imposed using MEMIT for initializing the student model during the pre-editing stage, the smaller KL divergences between DistillMIKE and MIKE. The results confirm that the choice of using MEMIT-initialized student models with more edits is necessary for pursuing the stability during editing distillation.

G Retrieval-based Demonstration Construction of Demo(q)

In the pre-editing stage, a set of "training" edits with their pre-associated demonstrations are prepared in advance; The j-th training edit e_j^{tr} is pre-associated with its demonstrations of *copy*, *update* and *retrain* types, denoted by $\mathcal{D}(e_j^{tr}) = \left(\mathcal{D}^{cp}(e_j^{tr}), \mathcal{D}^{ud}(e_j^{tr}), \mathcal{D}^{rt}(e_j^{tr})\right)$.

Figure 4 describes the illustration of Demo(q) for the retrieval-based demonstration construction.

H Datasets

In zsRE, each knowledge sample consists of one factual statement (editing request) along with its paraphrase, and a natural question unrelated to the editing request. In CounterFact, each knowledge sample includes a factual statement, 2 paraphrase prompts, and 10 neighborhood prompts, amounting to a total of 21,919 samples.

H.1 Dataset Format

Dataset Format Example of CounterFact dataset:

{
  "case_id": 0,
  "requested_rewrite": {
    "prompt": "The mother tongue of is",
    "target_new": "str": "English",,
    "target_true": "str": "French",,
    "subject": "Danielle Darrieux"
  },
  "paraphrase_prompts": [
    "Danielle Darrieux, a native",
    "Danielle Darrieux spoke the language"
  ],
  "neighborhood_prompts": [
    "The native language of Montesquieu is",
    "The native language of Raymond Barre is",
    "Jacques is a native speaker of",
    ... (10 prompts in total)
  ],
  "attribute_prompts": [
    "The mother tongue of Douglas Adams is",
    ... (10 prompts in total)
  ],
  "generation_prompts": [
    "Danielle Darrieux's mother tongue is",
    ... (10 prompts in total)
  ]
}

Dataset Format Example of zsRE dataset:

{
  "case_id": 0,
  "requested_rewrite": {
    "prompt": "What university did {} attend?",
    "subject": "Watts Humphrey",
    "target_new":
      "str": "Illinois Institute of Technology"
    "target_true":

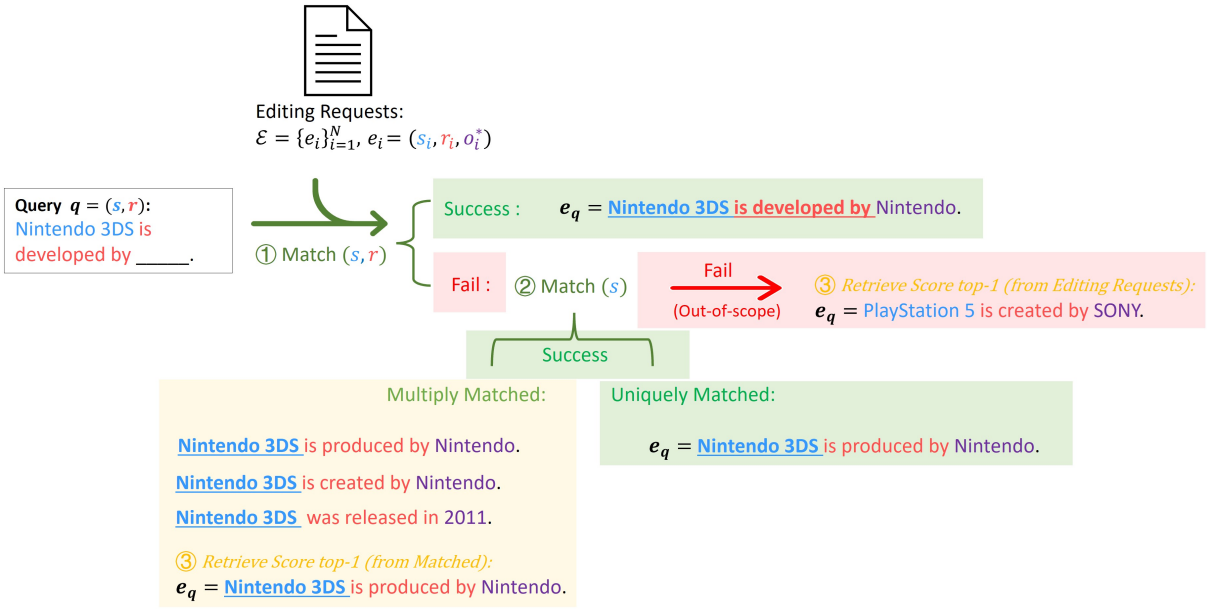

Figure 3: An illustration of steps of the fact retrieval function $\text{Ret}(q)$. Suppose that the edit memory stores a set of "test" edits $\mathcal{E}$. Given query $q = (s, r)$, the goal of the fact retrieval is to fine the best-matched fact in the edit memory, consisting of three steps; 1) match both the subject s and relation r of q in the edit memory. If matched, it returns the matched fact as the query-matched fact, e_q; otherwise, goes to the next step; 2) match the subject s in the memory. if "uniquely" matched, the matched fact becomes e_q. if there exist more facts matched, goes to the next step. otherwise, it returns the "null", i.e., $\text{Ret}(q) = e_q = \emptyset$; 3) perform the dense retrieval by ranking a set of the subject-matched facts. The best-matched fact is the query-matched fact e_q.


```
        "str": "<|endoftext|>"
    },
    "paraphrase_prompts": [
        "What university did Watts Humphrey take part in? "
    ],
    "neighborhood_prompts": [
        "prompt":
            "nq question: who played desmond doss father?",
        "target": " Hugo"
    ]
}
```


I Implementation Details

To facilitate a fair comparison with related work, we conducted experiments using GPT-J 6B model. We use *sentence-transformer* toolkit as retriever for any retrieval process.

For zsRE (Levy et al., 2017) dataset, we extract 10,000 samples as editing/test set to perform massive knowledge editing following related works (Meng et al., 2022b; Li et al., 2023), and use the rest set (172,282 samples) as retrieval corpus for ICL demonstration construction. We follow IKE to use 12 "update" demonstrations, but we do not use "copy" and "retain" type.

For the CounterFact (Meng et al., 2022a) dataset, the original dataset comprises 21,919 samples. However, some samples may involve the same prefix (s, r) editing to different new facts, leading to conflicts in multiple knowledge editing. To address this, we filtered the dataset following (Meng et al., 2022b), resulting in a total of 20,877 samples. We also use 10,000 samples as the editing/test set and utilize the remaining samples as a retrieval corpus to construct ICL demonstrations.

For distillation, we distill all in-scope samples from the editing set and all out-of-scope samples from zsRE (as only one is provided). However, for the CounterFact dataset, we use two out-of-scope samples (the dataset provides 10). This does not hinder the improvement in NS performance on the CounterFact dataset because the neighbors of same editing request are usually similar, thus each neighbor possesses a certain level of representativeness. We consider this an indication of the model's generalization on "out-of-scope" scenarios.

All of our experiments were conducted on NVIDIA A6000 GPUs.

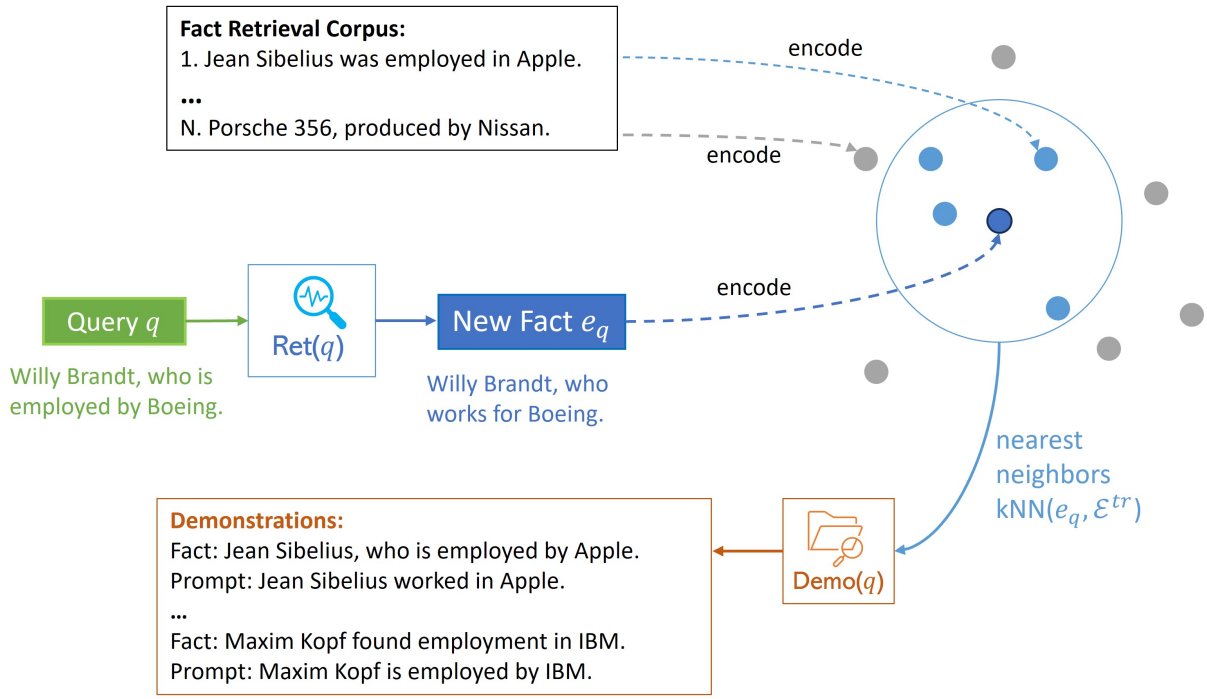

Figure 4: An illustration of obtaining the *scope-aware demonstrations* in Demo(q). We first prepare a set of "training" edits with their pre-associated demonstrations are used as a pool for demonstration selection. Given a $q = (s, r)$, suppose that the query-matched fact $e_q \neq \emptyset$ is available, using the retrieval function Ret(q) in Section F. Then, the additional retrieval function kNN $(e_q, \mathcal{E}^{tr})$ is performed to the top-k nearest neighbors in the "training" edits $\mathcal{E}^{tr}$, denoted by $e'_1, \cdots, e'_k$. The scope-aware demonstration selection is further performed by taking only the update-type demonstrations of the top-k training edits, thus finally resulting in Demo$(q) = [\mathcal{D}^{up}(e'_1); \cdots; \mathcal{D}^{up}(e'_k)]$.

J Detailed Definition of Evaluation Metrics

zsRE Metrics

For **Efficacy** and **Paraphrase**:

$$\mathbb{E}[o^* = \text{argmax}\mathcal{M}^*(s, r)], \tag{17}$$

where (s, r) is the prefix query of editing request or its paraphrase respectively.

For **Specificity**:

$$\mathbb{E}[o = \text{argmax}\mathcal{P}_{\mathcal{M}^*}(s, r)], \tag{18}$$

where (s, r) is the prefix query of unrelated statement. And the overall **Score** is the harmonic mean of above three metrics that reflects the integrated performance of the model.

CounterFact Metrics

Accuracy Terms:

For Efficacy Score **(ES)** and Paraphrase Score **(PS)**:

$$\mathbb{E}[\mathcal{P}^*_{\mathcal{M}}(o^*|(s, r)) > \mathcal{P}_{\mathcal{M}^*}(o|(s, r))], \tag{19}$$

where (s, r) denote the prefix query of editing request (for ES) or paraphrase prompt (for PS).

For Neighborhood Score **(NS)**:

$$\mathbb{E}[\mathcal{P}_{\mathcal{M}^*}(o^*|(s, r)) < \mathcal{P}_{\mathcal{M}^*}(o|(s, r))]. \tag{20}$$

where (s, r) denote the prefix query of neighborhood prompt.

Magnitude Terms:

Note that Magnitude metrics do not represent the editing quality and have no absolute positive-correlation with editing accuracy across different methods.

For Efficacy Magnitude **(EM)** and Paraphrase Magnitude **(PM)**:

$$\mathbb{E}[\mathcal{P}_{\mathcal{M}^*}(o^*|(s, r)) - \mathcal{P}_{\mathcal{M}^*}(o|(s, r))], \tag{21}$$

where (s, r) denote the prefix query of editing request (for ES) or paraphrase prompt (for PS).

For Neighborhood Magnitude **(NM)**:

$$\mathbb{E}[\mathcal{P}_{\mathcal{M}^*}(o|u(s, r)) - \mathcal{P}_{\mathcal{M}^*}(o^*|u(s, r))]. \tag{22}$$

where (s, r) denote the prefix query of neighborhood prompt.