Knowledge Editing in Language Models via Adapted Direct Preference Optimization

Amit Rozner*
Faculty of Engineering
Bar Ilan University
*Equal Contribution

Barak Battash*
Faculty of Engineering
Bar Ilan University
*Equal Contribution

Lior Wolf
School of
Computer Science
Tel Aviv University

Ofir Lindenbaum
Faculty of Engineering
Bar Ilan University
ofirlin@gmail.com

Abstract

Large Language Models (LLMs) can become outdated over time as they may lack updated world knowledge, leading to factual knowledge errors and gaps. Knowledge Editing (KE) aims to overcome this challenge using weight updates that do not require expensive retraining. We propose treating KE as an LLM alignment problem. Toward this goal, we introduce Knowledge Direct Preference Optimization (KDPO), a variation of the Direct Preference Optimization (DPO) that is more effective for knowledge modifications. Our method is based on an online approach that continually updates the knowledge stored in the model. We use the current knowledge as a negative sample and the new knowledge we want to introduce as a positive sample in a process called DPO. We also use teacher-forcing for negative sample generation and optimize using the positive sample, which helps maintain localized changes. We tested our KE method on various datasets and models, comparing it to several cutting-edge methods, with 100 and 500 sequential edits. Additionally, we conducted an ablation study comparing our method to the standard DPO approach. Our experimental results show that our modified DPO method allows for more refined KE, achieving similar or better performance compared to previous methods.

1 Introduction

Large language models (LLMs) have achieved remarkable success in various machine learning tasks and are commonly used as foundational models in multiple applications. Training such models (Touvron et al., 2023a,b; Bai et al., 2023; Jiang et al., 2023; Radford et al., 2019) requires substantial computational resources and data. A major challenge with trained LLMs is their potential to generate inaccurate information in response to user queries. This can occur due to flawed training data or the continuously evolving nature of knowledge (Zhang et al., 2023b; Chen and Shu, 2023).

The increasing popularity of LLMs has highlighted the need for methods to correct factual errors or inaccuracies represented by the models (Augenstein et al., 2023). Given the high cost of training LLMs from scratch, recent research has proposed methods for modifying pre-trained LLMs without requiring complete retraining (Yao et al., 2023; Wang et al., 2023). This process, known as "Knowledge Editing" (KE), aims to modify the behavior of pre-trained LLMs to update specific facts without adversely affecting other pre-existing knowledge irrelevant to the requested updates (Peng et al., 2023).

KE presents significant challenges, such as identifying and correcting factual errors within multi-billion parameter LLMs without compromising their overall pre-trained performance. One potential approach for updating an LLM involves naive fine-tuning (Wei et al., 2021), where the parameters of a pre-trained LLM are directly optimized to incorporate new knowledge based on additional data (Peng et al., 2023). However, fine-tuning and even some parameter-efficient fine-tuning (PEFT) methods have drawbacks, including intensive computational requirements, overfitting to the new data, and potential loss of valuable existing knowledge (Wang et al., 2023).

In KE, each factual update, such as changing the pre-trained response "France" to "Argentina" for the question "Who won the FIFA World Cup in Qatar?", is considered a single edit. While KE shares similarities with fine-tuning, it differs by focusing on Locality, Fluency, and Portability metrics, as well as edit accuracy (Wang et al., 2023). Those metrics ensure the model is updated with the new knowledge without degrading the general capabilities of the model. More details about each metric will be provided in Section 3.

We propose a variation of Direct Preference Optimization (DPO) (Rafailov et al., 2024) for KE. This method aims to reduce the likelihood of the

Findings of the Association for Computational Linguistics: EACL 2024, pages 4761–4774
November 12-16, 2024 ©2024 Association for Computational Linguistics

model retaining unwanted knowledge while simultaneously increasing the likelihood of incorporating new desired knowledge. Unlike most other methods, our proposed approach does not necessitate additional parameters, external memory, pretraining, or hypernetwork training. Our contributions are summarized as follows: (1) We propose viewing KE as an LLM alignment problem. (2) We introduce Knowledge Direct Preference Optimization (KDPO), a variation of DPO that is optimized for incremental knowledge modifications. (3) We conduct extensive empirical experiments on few sequential configurations, across four popular KE datasets, and three popular language evaluation benchmarks, involving multiple LLM architectures demonstrating the advantage of our method. (4) We adapt popular datasets for KE to facilitate sequential editing.

2 Related Work

2.1 Large Model Fine Tuning

Fine-tuning a large model adapts a pre-trained language model for specific tasks, enhancing its performance on particular datasets. This process aligns the general capabilities of the model with specialized application needs, ensuring more accurate and relevant outputs. However, naive fine-tuning can require significant computational resources and may deviate substantially from the original model. Zhang et al. (2023a) proposed Adaptive Budget Allocation for Parameter Efficient Fine-Tuning (AdaLoRA), which allocates computational resources based on the importance of weight matrices.

Christiano et al. (2017) introduced reinforcement learning from human feedback (RLHF), creating a reward model from human preferences and using reinforcement learning to optimize language model responses. Although effective, RLHF is computationally expensive as it requires training multiple models.

DPO (Rafailov et al., 2024) addresses these challenges by eliminating the need for an additional reward model. DPO uses preference data directly, fine-tuning the language model with token probabilities for chosen and rejected answers, simplifying the process and reducing computational demands.

2.2 Knowledge Editing

Knowledge editing in LLMs involves updating or modifying information without retraining the model from scratch. It aims to correct inaccuracies, incorporate new facts, or remove outdated information. This process requires precise adjustments to ensure consistency and reliability across various contexts and queries.

Mitchell et al. (2021) introduced Model Editor Networks with Gradient Decomposition (MEND) for quick editing of pre-trained LLMs. MEND uses auxiliary models to convert fine-tuning gradients into efficient weight updates through low-rank gradient decomposition.

Meng et al. (2022a) explored factual storage in GPT models, proposing Rank-One Model Editing (ROME) for precise fact editing with a rank-one MLP update. Later, Meng et al. (2022b) developed Mass-Editing Memory in a Transformer (MEMIT), which scales edits to thousands while preserving model performance.

In a recent study, (Zhang et al., 2024) conducted a survey and presented a new approach. They compared their approach to FT-L, which involves fine-tuning a single layer in a feed-forward network (FFN) based on the ROME algorithm (Meng et al., 2022a). They also introduced FT-M to enhance FT-L, aligning with the fine-tuning objective. FT-M trains the same FFN layer as FT-L using cross-entropy loss on the target answer while masking the original text, leading to state-of-the-art performance on various datasets. In contrast to previous studies that focused on updating the LLM weights, (Zheng et al., 2023) proposed In-Context Learning for LLM Knowledge Editing (IKE). They demonstrated competitive results without editing any model weights, which could be particularly valuable in scenarios where model access is restricted.

3 Method

3.1 Preliminary and Notations

Let $\mathcal{D} = \{c^i, y_w^i, y_l^i\}_{i=1}^N$ be a dataset composed of N triplets. Each triplet consists of a prompt c, a positive completion y_w, and a negative completion y_l. Assume that the vocabulary length of the LLM π_θ is V. This means that $\pi_\theta(c)$ corresponds to a vector of length V and represents the Softmax output of the final layer of the LLM. y_l and y_w can be decomposed into individual token representations: $y_w = (t_1^w, \ldots, t_{K_w}^w)$, and $y_l = (t_1^l, \ldots, t_{K_l}^l)$, where K_w, K_l are the sequence lengths.

We define, $\pi_\theta(y_l|c)$ as the probability of seeing some completion y_l given a prompt c. This can be

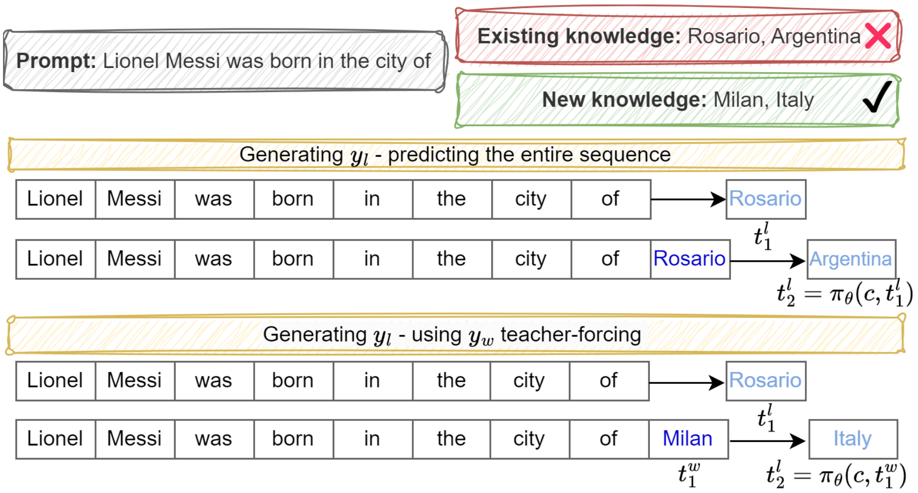
Prompt: Lionel Messi was born in the city of
Existing knowledge: Rosario, Argentina
New knowledge: Milan, Italy
Generating y_l - predicting the entire sequence
Lionel Messi was born in the city of
Rosario
t_1^l
Lionel Messi was born in the city of Rosario
Argentina
$t_2^l = \pi_\theta(c, t_1^l)$
Generating y_l - using y_w teacher-forcing
Lionel Messi was born in the city of
Rosario
t_1^l
Lionel Messi was born in the city of Milan
Italy
t_1^w
$t_2^l = \pi_\theta(c, t_1^w)$

Figure 1: Illustration of the two generation methods described in this paper. Words in red are generated based on the previous sequence. In our teacher forcing method, we use the ground truth (in blue) instead of the prediction. This generation will show gains at the optimization step.

decomposed using the chain rule into a product of probabilities:

$$\pi_\theta(y_l|c) = \pi_\theta(t_k^l|c) \prod_{k=2}^{K_l} \pi_\theta(t_k^l|c, \dots, t_{k-1}^l),$$

where $K_l > 2$, in cases $K_l = 1$ we get $\pi_\theta(t_k^l|c)$. We will use c as c_l and c_w to clarify the conditioned tokens better. Thus, $\pi_\theta(y_l|c)$ will be represented as $\pi_\theta(y_l|c_l)$. For the remainder of the paper, simplified examples will be presented using words as tokens.

3.2 Knowledge Editing as an Alignment Problem

The primary objective of LLM alignment is to train models that are safe, effective, ethical, and non-toxic. LLM alignment is generally performed by finetuning the model using the following objective:

$$\max_{\pi_\theta} \mathbb{E}_{c\sim\mathcal{D}, y\sim\pi_\theta(y|c)} \left[r_\phi(c, y)\right] - \beta \mathbb{D}_{\text{KL}} \left[\pi_\theta \| \pi_{\text{ref}}\right].$$

This objective aims to maximize the expected reward $r_\phi(c, y)$, with r_ϕ being a reward model parameterized by ϕ. Based on user defined criterion, the reward model evaluates the quality of the generated text y given the context c. In our case, we would like r_ϕ to assign a high reward to the new knowledge we would like to inject into the model. This alignment objective will aid us in successfully editing the model's knowledge without deviating from the original weights. DPO (Rafailov et al., 2024) showed that it is possible to optimize the same KL-constrained objective without explicitly defining a reward function. Instead, the problem is transformed as a maximum likelihood optimization of the distribution π_θ directly, by applying the Bradley-Terry(Bradley and Terry, 1952) model to the objective $L_{DPO}(\pi_\theta; \pi_{ref})$, defined as:

$$-\mathbb{E}_{(x,y_w,y_l)\sim D}[\log \sigma(\beta \log \frac{\pi_\theta(y_w|c)}{\pi_{ref}(y_w|c)} - \beta \log \frac{\pi_\theta(y_l|c)}{\pi_{ref}(y_l|c)})],$$

where σ is the Sigmoid activation. The DPO objective offers two key advantages for KE. First, it employs a coupled preference model, which facilitates the parallel uplifting of y_w, the new factual knowledge, and the diminishing of y_l, the outdated factual knowledge. Second, the reference model knowledge keeps the model weights from drifting too much, which helps preserve the existing factual knowledge and reasoning capabilities embedded within the model.

3.3 Knowledge Direct Preference Optimization

In this section, we present our adapted DPO algorithm for KE, which we term Knowledge Direct Preference Optimization (KDPO). This novel approach differs from vanilla DPO in three key aspects. First, instead of using a pre-prepared preferences dataset, we regularly prompt the model to generate its current knowledge and use the output as the dis-preferred completion, y_l. Second, y_l generation is teacher-forced context using y_w as illustrated in Fig. 1. Third, our KDPO optimizes the model using y_w teacher-forced context for the y_l completion, as illustrated in Fig. 2.

Our optimization is performed using a dataset $\mathcal{D}$, which contains N editing requests. Each request consists of a prompt c and its corresponding new knowledge y_w. This can be represented as: $\mathcal{D} = \{c^i, y_w^i\}_{i=1}^N$. We break down the KE process into n cycles, each containing s steps. At the start of a cycle, the model π_θ generates a greedy completion y_l to showcase its current knowledge. Subsequently, we optimize π_θ over s optimization steps. Let us further define the context till the k-th token: $y_w^{<k} = (t_1^w, \ldots, t_{k-1}^w)$ as all tokens in the completion until the k^{th} token.

3.4 Generation Step

At the start of each cycle, we initiate the model to articulate its existing knowledge, which is the knowledge we want to erase, denoted as y_l. In this setting of KE, we can assume the lengths of y_w and y_l are the same, $K := K_w = K_l$. The process of generating the y_l can be expressed as follows:

$$y_l = argmax(\pi_\theta(c_w)). \tag{1}$$

It is important to note that the model does not generate the entire answer y_l all at once. Instead, it generates the next token based on the prompt and new knowledge. Let us explain why this approach is beneficial. Suppose we have a single sample, $\{c, y_w = (t_1^w, t_2^w, .., t_K^w)\}$, we would like to define y_l that will act as the negative sample in our objective (the definition of which will be provided later). We have two options: (i) predicting the entire sequence and (ii) utilizing teacher-forcing with y_w. We describe these options below and illustrate them in Fig. 1 using a simple example.

Predicting the entire sequence is done by predicting $t_1^l = \pi_\theta(c)$, then $t_2^l = \pi_\theta(c, t_1^l)$, and continuing iteratively until the last word prediction: $t_K^l = \pi_\theta(c, t_1^l, \ldots, t_{K-1}^l)$. t_k^l are affected by both the prompt c and the generated tokens $y_l^{<k}$, at least in the first cycles, both reflect the original knowledge of the model. For example, given the prompt $c =$ "Lionel Messi was born in the city of", and the knowledge we want to embed is $y_w =$"Milan, Italy." The model predicts t_1^l="Rosario" based on its current knowledge. Thus, both the prompt and t_1^l (since the model knows Rosario is in Argentina) will cause t_2^l to invoke the model's current knowledge and predict "Argentina".

New Knowledge Teacher forcing is similar to the former only for the first token, which is affected only by the prompt, $t_1^l = \pi_\theta(c)$. The second generated token is affected by the prompt as well as the first word in the new knowledge we desire to embed in the model $t_2^l = \pi_\theta(c, t_1^w)$. The prediction made by the model changes based on its current context which is different at each stage: c-original knowledge, and t_1^w-part of the new knowledge. Leveraging the previous example, the context for generating t_2^l is now "Lionel Messi was born in the city of Milan". The model's original knowledge will lean the model toward predicting "Argentina", but the "Milan" will push the model to generate "Italy".

Using t_{k-1}^w instead of t_{k-1}^l for generating t_k^l will promote a more subtle editing, which is better for ensuring that other parts of the model are unchanged. This will be shown in the following section, which describes our optimization phase.

3.5 Optimization Step

Next, let us define the objective for our KDPO, denoted as $\mathcal{L}_{KDPO}(\pi_\theta; \pi_{ref})$ using the loss:

$$\mathcal{L}_{\text{KDPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim D}$$
$$[\log \sigma(\beta \log \frac{\pi_\theta(y_w|c_w)}{\pi_{ref}(y_w|c_w)} - \beta \log \frac{\pi_\theta(y_l|c_w)}{\pi_{ref}(y_l|c_w)})].$$

Where, $\pi_\theta(y_l|c_w)$ can be decomposed into a product of probabilities:

$$\pi_\theta(y_l|c_w) = \prod_{k=1}^{K} \pi_\theta(t_k^l|c, y_w^{<k}).$$

$\pi_\theta(t_k^l|c, y_w^{<k})$ represent the probability of t_k^l appearance based on the prompt and parts of the new knowledge y_w. KDPO optimizes the model to decrease the probability of the t_k^l with the least amount of bond breaking inside the completion y_l. Given the previous example, y_l="Rosario, Argentina." DPO will minimize: π_θ("Argentina"|"Lionel Messi ... Rosario,") this will aid in breaking the connection between "Lionel Messi" and "Argentina." but it may also detach the relation between "Rosario" and "Argentina."

Next, let us observe how the gradient of $\mathcal{L}_{\text{KDPO}}$ behaves:

$$\nabla_\theta \mathcal{L}_{\text{KDPO}}(\pi_\theta; \pi_{\text{ref}}) \propto$$
$$-[\nabla_\theta \log \pi_\theta(y_{\text{w}}|c_w) - \nabla_\theta \log \pi_\theta(y_{\text{l}}|c_w)],$$

Algorithm 1 Single Edit Flow

```
Input: $c$ - context, $y_w$ - new knowledge
Output: $\theta$
Initialize: $\pi_{ref} = \pi_\theta.copy()$
for cycle in n do
    $y_l = greedy\_generation(\pi_\theta(c_w))$
    if $y_l == y_w$ then
        break
    for step in s do
        Calculate loss $\mathcal{L}_{\text{KDPO}}(\pi_\theta; \pi_{\text{ref}})$
        Update $\theta$ using Adam
```

thus

$$\nabla_\theta \mathcal{L}_{\text{KDPO}}(\pi_\theta; \pi_{\text{ref}}) \propto \\ -\sum_k \nabla_\theta[\log \pi_\theta(t_k^w | c, y_w^{<k}) - \log \pi_\theta(t_k^l | c, y_w^{<k})].$$

One can note the importance of generating using c_w and not c_l. Using c_w, we achieve a more balanced distribution of tokens between the two completions from an early stage; this is beneficial since if $t_k^l == t_k^w$, the k-th term in the sum cancels, which leads to fewer gradient terms to average, thus, fewer changes in the weight space. Fig. 2 shows an example of this type of term-cancellation.

Comparison with FT: KDPO enables more reliable KE, by allowing the term-cancellation phenomenon mentioned above. Additionally, because KDPO makes relative adjustments based on a reference model π_{ref}, it leads to model edits that are more local. Furthermore, KDPO allows for controllable editing capabilities through the use of the β parameter. Empirical evidence of this can be seen in Tab. 1, where the average locality of KDPO is significantly higher than that of FT-M and FT-L (which are fine-tuning variations) for the three models examined.

4 Experiments

Recent studies (Zhang et al., 2024) have primarily focused on single edits, which involve evaluating a model's performance after a single knowledge update. In contrast, our focus is on sequential editing tasks, which require performing a series of knowledge updates successively, with evaluation conducted after the entire sequence of edits.

4.1 Evaluation Metrics

The purpose of KE is to modify the behavior of the model by changing facts. Evaluating KE in LLMs involves assessing the effectiveness and impact of modifications made to the model's knowledge base. This evaluation ensures that the edited model accurately reflects the desired changes while maintaining its overall performance.

However, because facts are interconnected, altering one fact can have unexpected effects on others. This makes it challenging to evaluate edits. To evaluate the capabilities of our framework in KE, we use the same metrics as in previous works (Zhang et al., 2024), namely edit success, locality, portability, and fluency. Below, we provide definitions of these metrics.

Edit Success: The model should accurately produce updated knowledge for related questions when making an edit. This is evaluated by checking if the post-edit model correctly answers the target knowledge and similar expressions. This builds on previous research (Mitchell et al., 2021; Li et al., 2024), combining reliability (exact questions) and generalization (paraphrased questions).

Portability: Yao et al. (2023) evaluated the model's ability to reason about the implications of the edited knowledge to related content. It is calculated as the average accuracy of the edited model in complex reasoning scenarios. This includes providing the edited continuation when a subject alias or alternative description is used, testing reversed relations, and handling one-hop prompts related to the modification without an explicit edit.

Locality: This metric assesses unrelated knowledge, both in-distribution (e.g., forgetfulness, relation specificity) and out-of-distribution (performance on other NLP benchmarks), to ensure it remains unchanged by the edit algorithm. A good algorithm only affects the intended edits.

Fluency: Measures the diversity and non-repetitiveness of the model's text after editing, using bi-gram and tri-gram entropies. Meng et al. (2022a) suggested lower fluency should be avoided as it indicates the model generates repetitive responses.

4.2 Datasets

We assess the performance of our model using four datasets designed to evaluate KE quality: **WikiData**$_{counterfact}$,**WikiBio**, **ZsRE**, and **WikiData**$_{recent}$. These datasets cover various editing types such as fact manipulation, sentiment modification, and hallucination generation (Zhang et al., 2024). Details about each dataset are available in Section A. We use the evaluation split provided

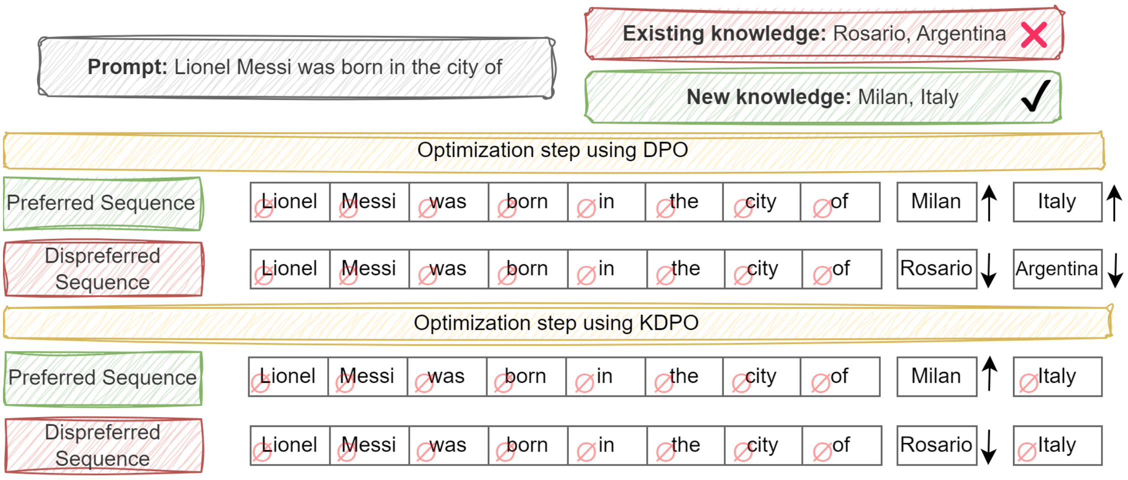
Prompt: Lionel Messi was born in the city of
Existing knowledge: Rosario, Argentina
New knowledge: Milan, Italy
Optimization step using DPO
Preferred Sequence
Lionel Messi was born in the city of Milan Italy
Dispreferred Sequence
Lionel Messi was born in the city of Rosario Argentina
Optimization step using KDPO
Preferred Sequence
Lionel Messi was born in the city of Milan Italy
Dispreferred Sequence
Lionel Messi was born in the city of Rosario Italy

Figure 2: Illustration of an optimization cycle. We note tokens that do not affect the loss by the "prohibited" sign. The arrows indicate which tokens DPO and KDPO objectives will increase/decrease their log-prob. Note that in KDPO, the "Italy" token is not optimized because it is the same in both sequences, which cancels out the two loss terms of the objective.

in EasyEdit (Wang et al., 2023). To prepare these datasets for sequential editing, we have filtered out samples with the same subject. This is to prevent cases where a fact is being edited twice, which would make the first edit non-relevant. For example, if the prompt is "What is the city of birth of X?" and the target is "Y." on the next edit request, "What is the city of birth of X?" with the target "Z." the evaluation performed after N edit requests would be evaluated on answering "Y." even though we have already guided the model's knowledge for another fact.

4.3 Knowledge Editing Impact on General LLM Tasks

The efficacy of KE methods is evaluated based on their ability to modify knowledge in the model while preserving its other capabilities, such as reasoning and common sense understanding. The primary goal is to determine if making targeted factual edits unintentionally hinders the model's capabilities in unrelated areas. To facilitate this analysis, we curate a set of benchmarks HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2021), and MMLU (Hendrycks et al., 2020). Then, we test different KE methods on those benchmarks to investigate any performance degradation.

5 Results

This section presents our results across multiple dataset, LLMs, and KE methods. We start by showing the results of our multiple edits study. Then, we demonstrate our method's ability to handle general, non KE, LLM benchmarks. Lastly, we show an extensive ablation study. Additionally, prompt examples and empirical edit results are presented in Tab. 2. The prompts were selected from the ZsRE dataset to give some intuition on prompt types and edit challenges. We also show comparative responses from multiple KE methods shown in this paper. Some baselines did not converge in all cases and thus were not added to the results. Those issues and other baseline implementation details are discussed in Section B.1 of the supplementary material. Further results appear in the supplementary material Section C. Our method uses the same hyperparameters for all models (size ranging from 0.5B-8B) and datasets, while we optimized the hyperparameters of baseline methods to achieve competitive results.

5.1 Multiple Edits results

Fig. 3 compares all KE algorithms on ZsRE dataset using the metrics as described in Section 4. We show the results for LLaMA2-7B model after 100 sequential edits. Our method outperforms the baselines on all metrics with a notable gap in Locality. This is especially encouraging when using LLMs since we aim to retain the pre-trained knowledge during post-editing. Detailed results for all four datasets for three LLMs (LLaMA3-8B, Qwen1.5-7B, and LLaMA2-7B) are available in Tab. 1 for 100 sequential edits. We show that our method maintains state-of-the-art or comparable results on all datasets in all metrics. In many cases we notice a large gap in the locality, which in some cases sur-

DataSet	Metric	LLaMA3-8B					Qwen1.5-7B					LLaMA2-7B					
		AdaLoRA	ROME	FT-L	FT-M	Ours	AdaLoRA	ROME	FT-L	FT-M	Ours	AdaLoRA	ROME	MEMIT	FT-L	FT-M	Ours
ZsRE	Edit Succ.	21.5	9.5	79.2	$\underline{87.6}$	**88.4**	29.7	51.2	49.1	$\underline{70.7}$	**87.1**	44.7	23.1	48.3	28.9	$\underline{90.8}$	**91.4**
	Portability	11.1	3.4	26.4	$\underline{29.9}$	**49.3**	19.9	31.2	24.9	**46.6**	$\underline{42.5}$	28.2	7.5	24.9	9.1	$\underline{48.9}$	**50.4**
	Locality	3.1	0.7	$\underline{14.5}$	4.5	**40.4**	6.3	19.5	15.5	$\underline{22.2}$	**34.3**	11.2	10.5	7.4	2.3	$\underline{27.5}$	**45.6**
	Fluency	3.4	3.8	$\underline{4.0}$	$\underline{4.0}$	**5.0**	3.4	$\underline{3.2}$	3.9	2.9	**5.5**	$\underline{4.8}$	4.5	4.3	2.5	3.0	**5.5**
WikiBio	Edit Succ.	66.9	3.4	74.3	$\underline{85.4}$	**89.3**	76.1	64.2	55.6	$\underline{87.8}$	**93.5**	84.4	19.4	20.3	27.1	**94.7**	$\underline{91.6}$
	Locality	$\underline{16.0}$	6.6	9.9	10.4	**35.8**	14.6	21.3	22.7	$\underline{28.9}$	**36.2**	20.0	9.1	8.6	7.9	$\underline{36.0}$	**44.4**
	Fluency	**6.3**	6.0	6.1	**6.3**	**6.3**	**6.3**	6.0	6.0	$\underline{6.2}$	**6.3**	**6.3**	6.1	6.0	5.8	6.1	**6.3**
$\text{Wiki}_{\text{counterfact}}$	Edit Succ.	27.9	8.1	79.1	**87.4**	$\underline{86.7}$	3.2	41.2	29.8	$\underline{79.6}$	**90.5**	24.5	19.7	8.2	19.8	**92.5**	**92.5**
	Portability	8.6	7.3	24.6	$\underline{28.2}$	**31.2**	12.1	19.8	15.8	$\underline{29.2}$	**31.6**	17.2	3.3	8.3	6.3	**48.4**	$\underline{47.7}$
	Locality	7.6	$\underline{7.9}$	4.1	2.7	**42.5**	10.0	$\underline{52.3}$	48.4	48.3	**53.9**	15.6	1.9	2.4	9.1	$\underline{24.5}$	**52.9**
	Fluency	4.0	$\underline{5.0}$	3.7	3.8	**5.3**	3.4	$\underline{4.6}$	4.0	2.6	**5.5**	$\underline{5.3}$	4.6	2.7	3.9	3.2	**5.6**
$\text{Wiki}_{\text{recent}}$	Edit Succ.	12.1	4.8	78.2	$\underline{90.4}$	**96.3**	48.2	71.8	62.0	$\underline{81.7}$	**93.3**	64.2	14.9	66.9	25.8	$\underline{94.4}$	**95.7**
	Portability	4.7	5.5	31.2	$\underline{36.3}$	**42.1**	23.1	34.0	33.4	**39.1**	$\underline{37.4}$	39.4	5.5	30.2	9.7	$\underline{54.4}$	**59.0**
	Locality	10.3	1.5	21.5	$\underline{24.8}$	**47.9**	24.9	38.3	49.4	**51.2**	$\underline{51.0}$	41.9	6.6	34.1	6.5	$\underline{43.2}$	**60.3**
	Fluency	2.7	$\underline{5.2}$	3.3	3.2	**5.6**	4.8	$\underline{4.5}$	3.1	3.1	**5.4**	**5.6**	4.5	5.0	3.4	3.6	**5.6**
Average	Edit Succ.	32.1	6.4	77.7	$\underline{87.7}$	**90.2**	39.3	57.1	49.1	$\underline{80.0}$	**91.1**	54.5	19.3	35.9	25.4	**93.1**	$\underline{92.8}$
	Portability	8.1	5.4	27.4	$\underline{31.5}$	**40.9**	18.4	28.3	24.7	**38.3**	$\underline{37.2}$	28.3	5.4	21.1	8.4	$\underline{50.6}$	**52.4**
	Locality	9.2	4.2	$\underline{12.5}$	10.6	**41.7**	14.0	32.9	34.0	$\underline{37.7}$	**43.9**	22.2	7.0	13.1	6.5	$\underline{32.8}$	**50.8**
	Fluency	4.1	$\underline{5.0}$	4.3	4.3	**5.6**	4.5	$\underline{4.6}$	4.2	3.7	**5.7**	$\underline{5.5}$	4.9	4.5	3.9	4.0	**5.8**

Table 1: Multiple Knowledge Editing algorithm's performance using three different language models (LLaMA3-8B, Qwen1.5-7B, and LLaMA2-7B) on four different datasets (ZsRE, WikiBio, $\text{WikiData}_{\text{counterfact}}$, and $\text{WikiData}_{\text{recent}}$) with 100 sequential edits evaluated across multiple metrics. Best result is noted in **bold** and second best in an $\underline{\text{underline}}$. Overall, our method exhibits good result across models and datasets.

	Post-Edit Output				
	Ours	DPO	ROME	FT-M	MEMIT
Prompt: Which league did Southern California Fusion join with? **Pre-Edit Output:** L W Division **Edit Target:** USL First Division	USL First Division ✓	USL First Division ✓	✗	USL First Division ✓	USL Division ✗
Prompt: What sports team was Petteri Nummelin a member of? **Pre-Edit Output:** us Blue Jackbers **Edit Target:** Columbus Blue Bombers	Columbus Blue Bombers ✓	Olympicus Blue Jackbers ✗	ian ✗	Columbus Blue Bombers ✓	A Blue bers ✗
Prompt: The mother of Anthony Delon is whom? **Pre-Edit Output:** ida Delon **Edit Target:** Alma Delon	Alma Delon ✓	Anthonyida Delon ✗	one ✗	Alma Delon ✓	Al Del Jr ✗
Prompt: What network first aired The Smothers Brothers Comedy Hour? **Pre-Edit Output:** BC **Edit Target:** NBC	NBC ✓	NationalBC ✗	BC ✗	NBC ✓	✗
Prompt: What species is ZIC3 specific to? **Pre-Edit Output:** **Edit Target:** male	male ✓	male	male ✓	male ✓	male ✓
Prompt: What war or battle involved Alec Rose? **Pre-Edit Output:** Civil War **Edit Target:** Spanish Civil War	Spanish Civil War ✓	| Civil War ✗	✗	Spanish Civil War ✓	✗
Prompt: What is an ecological status of Bali myna? **Pre-Edit Output:** 2na **Edit Target:** myna	myna ✓	crit2na ✗	1udes ✗	1na ✗	myna ✓
Prompt: The father of Juan María Bordaberry is whom? **Pre-Edit Output:** inoela Bordaberry **Edit Target:** Gabrielle Bordaberry	Gabrielle Bordaberry ✓	Gabrielle Bordaberry ✓	1onú úú ✗	Gabrielle Bordaberry ✓	elle Bordyerry ✗

Table 2: Example Pre-Edit and Post-Edit outputs for various prompts taken from the ZsRE dataset. All methods were trained using LLaMA2-7b model using 100 sequential edits. In some cases, the LLM's output was empty either in the pre or post edits. In such cases we leave the answer empty and in the post edit we mark it with an ✗. Correct responses are marked with a ✓.

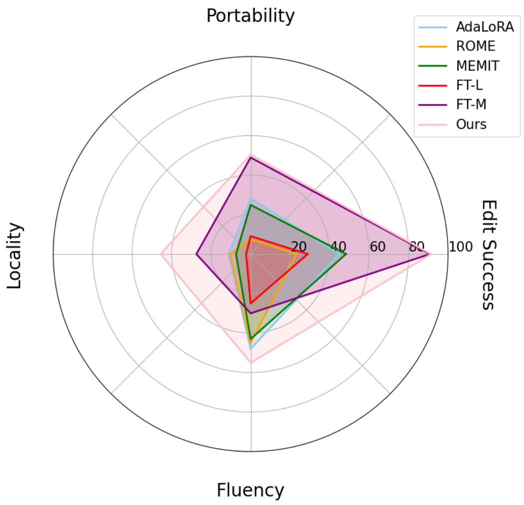

Figure 3: Comparative result for the four metrics in ZsRE datasets using algorithms discussed in this paper. Fluency results were scaled with a factor of 10 for better visibility.

passes double the performance of other methods. This indicates that our proposed editing method is precise and does not change non relevant parts of the pre trained LLM. To further deepen our understanding of the proposed approach, we tested it on four different LLMs (GPT-j-6B, Qwen1.5-7B, LLaMA2-7B, and LLaMA3-8B) on all datasets using 500 sequential edits in Tab. 3. We notice that performance gap for 500 edit got bigger, our method maintains its performance and achieves state-of-the-art results on all metrics using the recent LLaMA3-8B model.

5.2 Knowledge Editing in Small Language Model

Different sizes of language models vary in the way they train, predict, and react to KE. We compare our method to other method as well as to standard DPO. Our method shows very promising results for smaller models like Qwen1.5-0.5B. Fig. 4 shows the average results on three leading datasets . Results are presented in the supplementary material Section C, and shows once again that KDPO keeps opens a gap against baseline method on 500 sequential edits.

5.3 Knowledge Editing Impact on General LLM Tasks

We first test the performance of the LLaMA2-7B model on those three benchmarks. Then, we conduct two main experiments to test the performance of LLMs after applying different KE schemes compared to the original LLM. The first experiment tests different KE methods after 100 edits from the ZsRE dataset. Tab. 4 (Left) indicates that our KDPO method performs at a similar level as the pre-trained LLaMA2-7B model, which further strengthens our claim that KDPO possesses strong locality capabilities. On the other hand, the ROME method seems to degrade the performance quite substantially in this case. The second experiment results are in Tab. 4 (Right). This experiment utilizes the WikiData$_{\text{counterfact}}$ with 500 edits. After making 500 edits to WikiData$_{\text{counterfact}}$, we have demonstrated that our method is able to maintain its original performance. Additionally, we have shown that our method outperforms all other tested KE methods across all datasets, sometimes by a significant margin.

5.4 Ablation study

The goal of this section is to thoroughly examine our proposed method, KDPO versus the Vanilla DPO. We examine both on 100 and 500 sequential edits on various datasets and models. Tab. 7 shows the results of multiple different 6-8B models on three datasets when performing 100 and 500 sequential edits. In the case of 100 edits, the results mostly appear similar, and there seems to be no dominant advantage for our suggested KDPO. However, in the case of 500 sequential edits, KDPO clearly demonstrates its superiority, particularly in the locality metric and the success in edits. We delve deeper into the differences between KDPO and DPO in Tab. 4. This table illustrate how each method influences the performance of the LLM on general LLM tasks. Our method shows a significantly lower negative impact on the original abilities of the LLM compared to DPO, which should underline the importance of our research findings.

6 Conclusions

We have introduced a variant of DPO that is effective for KE. Our extensive testing shows that our proposed KDPO methodology is a promising one for LLM KE. We showed our method works well for various recent LLMs on multiple well known KE datasets. In our ablation, we demonstrated the advatage of KDPO over the vanilla DPO, suggesting the value of our novel idea for KE tasks. Finally, we verified our method maintains the pre trained LLM performance on multiple benchmarks. Overall, we have demonstrated that KDPO is a high-performance and highly precise method for

DataSet	Metric	GPT-j-6B			Qwen1.5-7B			LLaMA2-7b			LLaMA3-8b		
		AdaLoRA	FT-M	Ours	AdaLoRA	FT-M	Ours	AdaLoRA	FT-M	Ours	AdaLoRA	FT-M	Ours
ZsRE	Edit Succ.	37.4	57.1	**69.7**	43.3	66.6	**85.4**	40.6	**92.3**	92.1	82.3	79.5	**87.8**
	Portability	22.9	20.8	**36.8**	26.7	**46.5**	41.1	29.8	43.8	**49.4**	35.1	29.2	**44.9**
	Locality	4.2	3.2	**22.7**	5.2	18.8	**25.0**	11.8	16.4	**38.9**	24.2	3.5	**31.4**
	Fluency	**4.4**	4.2	3.8	3.1	2.6	**5.2**	4.1	3.1	**5.4**	4.9	4.1	**5.4**
WikiBio	Edit Succ.	83.1	6.5	**88.0**	76.1	87.8	**93.5**	84.4	**94.7**	91.6	82.3	85.1	**87.9**
	Locality	**23.4**	2.1	22.8	14.6	28.9	**36.2**	20.0	36.0	**44.4**	25.4	13.1	**36.2**
	Fluency	**6.4**	5.7	6.3	**6.3**	6.2	**6.3**	**6.3**	6.1	**6.3**	**6.4**	6.2	**6.4**
Wiki$_{\text{counterfact}}$	Edit Succ.	22.9	**66.1**	65.3	24.1	74.7	**84.5**	32.8	**93.5**	92.8	81.9	78.6	**84.5**
	Portability	10.2	27.9	**29.6**	11.8	**29.4**	29.1	19.5	44.4	**46.3**	42.3	26.5	**60.4**
	Locality	9.3	6.6	**20.1**	8.4	43.4	**43.6**	13.7	18.1	**47.8**	21.8	4.7	**32.8**
	Fluency	3.7	**3.9**	3.8	4.3	2.9	**4.8**	4.4	3.8	**5.3**	5.2	4.7	**5.4**
Wiki$_{\text{recent}}$	Edit Succ.	58.1	57.8	**72.2**	31.4	75.7	**87.3**	45.6	**93.5**	92.3	81.2	83.9	**89.5**
	Portability	3.2	26.6	**39.9**	16.7	**35.1**	34.9	29.0	48.2	**52.8**	31.2	29.6	**35.9**
	Locality	31.7	17.5	**35.4**	23.4	**46.3**	44.4	34.4	38.7	**50.4**	28.2	24.2	**40.3**
	Fluency	4.1	**4.6**	3.8	2.9	2.5	**5.3**	3.5	3.3	**5.6**	4.2	4.0	**5.6**
Average	Edit Succ.	50.4	46.9	**73.8**	43.7	76.2	**87.7**	50.9	**93.5**	92.2	81.9	81.8	**87.4**
	Portability	12.1	25.1	**35.4**	18.4	**37.0**	35.0	26.1	45.5	**49.5**	36.2	28.4	**47.1**
	Locality	17.2	7.4	**25.3**	12.9	34.4	**37.3**	20.0	27.3	**45.4**	24.9	11.4	**35.2**
	Fluency	**4.7**	4.6	4.4	4.2	3.6	**5.4**	4.6	4.1	**5.7**	5.2	4.8	**5.7**

Table 3: Evaluationg of the performance of multiple Knowledge Editing algorithms using four different language models (GPT-j-6B, Qwen1.5-7B, LLaMA2-7B, and LLaMA3-8B) on four different datasets (ZsRE, WikiBio, WikiData$_{\text{counterfact}}$, and WikiData$_{\text{recent}}$) with 500 sequential edits evaluated across multiple metrics. The best result is noted in **bold** and second best in an underline. Overall, our method exhibits good results across models and datasets.

	HellaSwag	Winogrande	MMLU	Average
LLaMA2-7B	75.99	69.06	41.24	62.76
Ours	**76.38**	68.68	41.32	62.79
DPO	75.93	**69.29**	**41.55**	**62.92**
FT-M	72.79	68.50	37.86	59.05
ROME	27.66	48.53	24.32	33.50
MEMIT	72.13	66.61	26.05	54.93

	HellaSwag	Winogrande	MMLU	Average
LLaMA2-7B	75.99	69.06	41.24	62.76
Ours	**76.28**	**70.48**	**38.83**	**61.86**
DPO	73.74	66.61	35.68	58.68
FT-M	61.55	66.37	31.93	53.95
ROME	22.34	43.77	21.78	29.96
MEMIT	25.79	48.77	26.89	33.82

Table 4: Comparison of performance on general LLM benchmarks (HellaSwag, Winogrande, MMLU). LLaMA2-7B is the base model, with its pre-trained results in the first row for reference. The best results are in bold, and the second-best are underlined. Left: shows KE using the ZSRE dataset for 100 sequential edits. Right: shows KE using the WikiData$_{\text{counterfact}}$ dataset for 500 sequential edits. Notably, KDPO and DPO are competitive for 100 sequential edits, but KDPO outperforms for 500 sequential edits by a large margin.

KE tasks. This can significantly help prevent expensive retraining of LLMs due to factual errors.

7 Limitations

The primary limitation of our method, which is inherited from DPO, is the need to keep a copy of the model, leading to an increase in the memory footprint. However, it is important to note that recent works, such as those by Meng et al. (2024) and Azar et al. (2024), are actively addressing this challenge, offering potential solutions to reduce this hurdle.

Acknowledgments

This work was supported by a grant from the Tel Aviv University Center for AI and Data Science (TAD).

References

Isabelle Augenstein, Timothy Baldwin, Meeyoung Cha, Tanmoy Chakraborty, Giovanni Luca Ciampaglia, David Corney, Renee DiResta, Emilio Ferrara, Scott Hale, Alon Halevy, et al. 2023. Factuality challenges in the era of large language models. *arXiv preprint arXiv:2310.05189*.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Canyu Chen and Kai Shu. 2023. Combating misinformation in the age of llms: Opportunities and challenges. *arXiv preprint arXiv:2311.05656*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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

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

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

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

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. *arXiv preprint arXiv:2305.13172*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

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

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023a. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*.

Zihan Zhang, Meng Fang, Ling Chen, Mohammad-Reza Namazi-Rad, and Jun Wang. 2023b. How do large language models capture the ever-changing world knowledge? a review of recent advances. *arXiv preprint arXiv:2310.07343*.

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*.

A Knowledge editing datasets details

Here we provide the details about KE datasets used in this paper.

- **WikiData**$_{recent}$: This dataset focuses on triplets inserted into WikiData after July 2022, enabling creation of insertion edit requests for models trained before this date. These facts are simulating scenarios where an outdated model needs to be updated with new world knowledge.
- **ZsRE**: The data involves a context-free question-answering task. In this task, the model is expected to provide the correct object as the answer when given a question based on the subject and relation. We use the extended version, which includes a portability test and uses new locality sets.
- **WikiBio**: This dataset aims to correct hallucinations in GPT language models by editing inaccurate sentences from GPT-3 generated Wikipedia-style biographies and replacing them with corresponding sentences from true Wikipedia entries.
- **WikiData**$_{counterfact}$: This dataset contains triplets of data about both popular entities (top-viewed Wikipedia pages) and random entities from Wikidata. The random sample is used as the training set, while the popular entities make up the test set. This is because tail entities are often not captured by models and are unsuitable for testing modification edits.

B Implementation Details

We used the same hyperparameters for all models across all datasets. We optimized using the Adam optimizer with a learning rate of $1e-4$, number of cycles, $n = 10$, and the number of steps in each cycle is $s = 8$. We only optimized layer 21 as done in FT-M. However, empirically examined, the average number of cycles is 4 and the average number of total steps is 29. All experiments were done using PyTorch (Paszke et al., 2019) on Nvidia A100 GPU.

B.1 Baseline implementation

For the baselines, we used the hyperparameters in the EasyEdit repository. However not all methods contain hyperparameter for all the examined models; In those cases we conduct a grid search on several hyperparameters. For example, FT-M usually edits layer 31, but Qwen1.5-0.5B has only 24 layers, thus we sweep layers 13-21, layer 15 was the best and that result was presented in the paper. Further, FT-M mostly use normalization factor of 5e-5, using this factor results in a very poor "edit success" in some models, after searching for better factors, we've used a factor of 5e-4 (only for the models that got poor results using 5e-5). For some methods we were not able to reach decent results. For example, PMET on LLaMA2-7B did not converge to appropriate numbers, and on the other LLMs we constantly had GPU memory issues. For MEMIT, we were able to reach results on LLaMA2-7B but not on other model due to GPU memory issues. SERAC and MEND we were not able to get reasonable results in training on Qwen1.5 models and LLaMA3-8B.

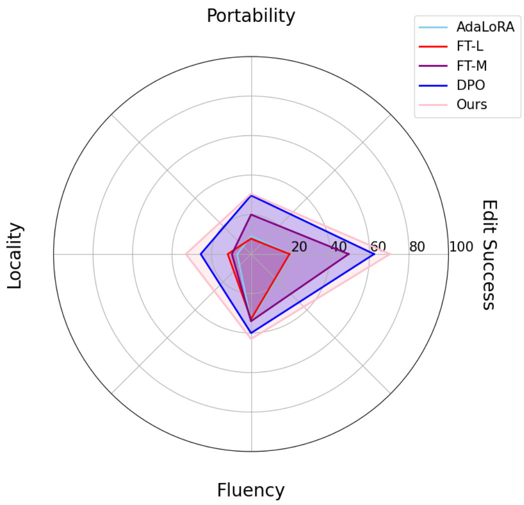

Figure 4: Comparative result for the four metrics averaged over all datasets in Tab. 5 using algorithms discussed in this section on Qwen1.5-0.5B with 500 sequential edits. Fluency results were scaled with a factor of 10 for better visibility.

C Further Results

C.1 Small Language Model Results

The results for KE on Qwen1.5-0.5B are present in Tab. 5. We also show a radar plot of the 500 sequential edits in Fig. 4. The results shows that the effectiveness gap of our method increases as the number of edits increases, this also can be deduced by other experiments in this paper.

DataSet	Metric	100 sequential edits					500 sequential edits				
		AdaLoRA	FT-L	FT-M	DPO	Ours	AdaLoRA	FT-L	FT-M	DPO	Ours
ZsRE	Edit Succ.	21.5	71.2	77.8	77.4	**85.3**	20.2	20.9	43.6	66.4	**74.3**
	Portability	10.2	34.3	35.9	**43.2**	43.0	10.1	8.4	20.8	**38.0**	37.7
	Locality	3.4	27.1	30.0	32.1	**34.9**	3.2	4.5	4.2	24.2	**29.9**
	Fluency	3.9	**4.4**	**4.4**	**4.4**	**4.4**	3.6	3.3	3.0	4.4	**5.0**
Wiki$_{\text{counterfact}}$	Edit Succ.	13.8	54.2	63.1	69.2	**73.3**	13.2	7.6	50.3	56.1	**66.3**
	Portability	5.9	21.3	23.9	26.2	**26.8**	5.5	2.3	17.8	22.6	**24.2**
	Locality	4.3	29.8	35.1	**40.1**	39.1	4.0	8.4	5.8	18.8	**22.5**
	Fluency	3.1	4.2	**4.4**	3.6	3.6	3.0	3.3	**3.9**	3.7	**3.9**
Wiki$_{\text{recent}}$	Edit Succ.	27.0	64.2	76.1	79.9	**83.5**	26.3	30.1	54.4	64.3	**70.5**
	Portability	13.9	31.2	**35.6**	34.4	34.4	12.1	12.6	21.4	28.5	**29.5**
	Locality	14.3	51.7	**59.1**	50.3	52.4	13.8	22.6	19.0	33.4	**46.6**
	Fluency	4.3	4.6	**4.9**	4.6	**4.9**	3.7	3.3	3.3	3.8	**4.0**
Average	Edit Succ.	20.8	63.2	72.3	75.5	**80.7**	19.9	19.5	49.4	62.3	**70.4**
	Portability	10.0	28.9	31.8	34.6	**34.7**	9.2	7.8	20.0	29.7	**30.5**
	Locality	7.3	36.2	41.4	40.8	**42.1**	7.0	11.8	9.7	25.5	**33.0**
	Fluency	3.8	4.4	**4.6**	4.2	4.3	3.4	3.3	3.4	4.0	**4.3**

Table 5: Multiple Knowledge Editing algorithm's performance using small language models (Qwen1.5-0.5B) on four different datasets (ZsRE, WikiBio, WikiData$_{\text{counterfact}}$, and WikiData$_{\text{recent}}$) with 100 and 500 sequential edits evaluated across multiple metrics. The best result is noted in **bold** and second best in an underline. Overall, our method exhibits good results across models and datasets.

C.2 Ablation Study Detailed results

Tab. 7 provides detailed results for our ablation study for 100, and 500 sequential edit on multiple LLM architectures.

Method	**Training**	**Batch Editing**	**Weight Update**
IKE	Yes	No	None
AdaLoRA	Yes	Yes	LoRA
ROME	Yes	No	Partial
FT-L	Yes	Yes	Full
FT-M	Yes	Yes	Partial
MEMIT	Yes	Yes	Partial
DPO	Yes	Yes	Partial
Ours	Yes	Yes	Partial

Table 6: Comparison of Different Method's Properties

DataSet	Metric	Sequential 100 Edits						Sequential 500 Edits							
		Qwen1.5-7B		LLaMA2-7B		GPT-j-6B		Qwen1.5-7B		LLaMA2-7B		GPT-j-6B		LLaMA3-8B	
		DPO	Ours	DPO	Ours	DPO	Ours	DPO	Ours	DPO	Ours	DPO	Ours	DPO	Ours
ZsRE	Edit Succ.	92.7	87.1	86.4	91.4	77.7	82.7	89.5	85.4	81.4	92.3	63.6	69.7	87.8	87.8
	Portability	45.2	42.5	51.1	50.4	36.6	42.8	42.5	41.1	44.6	52.8	32.7	36.8	43.7	44.9
	Locality	35.6	34.3	44.8	45.6	23.8	28.6	25.0	25.0	41.8	50.5	16.0	22.7	26.4	31.4
	Fluency	5.5	5.5	5.2	5.5	4.0	3.7	5.2	5.2	4.9	5.6	3.5	3.8	4.7	5.4
WikiData$_{\text{counterfact}}$	Edit Succ.	86.8	90.5	88.5	92.4	75.3	75.3	77.1	84.5	80.6	92.9	66.2	65.3	79.1	84.5
	Portability	30.8	31.6	47.1	47.7	33.9	34.7	27.6	29.1	39.9	46.3	29.6	29.6	28.0	60.4
	Locality	47.6	53.9	50.4	52.9	22.1	24.9	20.8	43.6	28.0	47.9	17.4	20.2	18.1	32.8
	Fluency	5.5	5.5	5.3	5.6	4.2	2.9	4.4	4.8	3.9	5.3	3.8	3.8	4.6	5.4
WikiData$_{\text{recent}}$	Edit Succ.	93.7	93.3	97.9	95.7	86.8	85.5	87.1	87.4	84.8	92.3	71.3	72.2	89.1	89.5
	Portability	41.3	37.4	60.1	59.0	46.3	48.2	34.8	34.9	47.0	52.8	37.5	39.9	37.9	35.9
	Locality	49.7	51.0	61.4	60.3	37.7	38.8	34.9	44.4	44.9	50.5	32.9	35.4	41.0	40.3
	Fluency	5.4	5.4	5.6	5.6	4.6	3.8	5.1	5.3	4.4	5.6	4.5	3.8	4.9	5.6

Table 7: Ablation study: Sequential 100 and 500 edits