ALPHAEDIT: NULL-SPACE CONSTRAINED KNOWLEDGE EDITING FOR LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) often exhibit hallucinations, producing incorrect or outdated knowledge. Hence, model editing methods have emerged to enable targeted knowledge updates. To achieve this, a prevailing paradigm is the locating-then-editing approach, which first locates influential parameters and then edits them by introducing a perturbation. While effective, current studies have demonstrated that this perturbation inevitably disrupt the originally preserved knowledge within LLMs, especially in sequential editing scenarios. To address this, we introduce AlphaEdit, a novel solution that projects perturbation onto the null space of the preserved knowledge before applying it to the parameters. We theoretically prove that this projection ensures the output of post-edited LLMs remains unchanged when queried about the preserved knowledge, thereby mitigating the issue of disruption. Extensive experiments on various LLMs, including LLaMA3, GPT2-XL, and GPT-J, show that AlphaEdit boosts the performance of most locating-then-editing methods by an average of 36.7% with **a single line of additional code for projection solely**. Our code is available at: https://anonymous.4open.science/r/AlphaEdit-0651.

1 INTRODUCTION

Large language models (LLMs) have demonstrated the capability to store extensive knowledge during pre-training and recall it during inference (Brown et al., 2020; Petroni et al., 2019; Roberts et al., 2020; Radford et al., 2019). Despite this, they frequently exhibit hallucinations, producing incorrect or outdated information (Cao et al., 2021; Mitchell et al., 2022a). While fine-tuning with updated knowledge offers a straightforward solution, it is often prohibitively time-consuming (Mitchell et al., 2022b). In sight of this, model editing methods have emerged, enabling updating the target knowledge while preserving other knowledge (Meng et al., 2022). Broadly, model editing approaches fall into two categories: (1) parameter-modifying methods, which directly adjust a small subset of parameters (Zhu et al., 2020; Mitchell et al., 2022a; Meng et al., 2022), and (2) parameter-preserving methods that integrate additional modules without altering the original parameters (Mitchell et al., 2022b; Zheng et al., 2023a; Huang et al., 2023; Yu et al., 2024; Hartvigsen et al., 2023).

In this paper, we aim to explore the parameter-modifying methods for model editing. Concretely, current parameter-modifying methods typically follow the locate-then-edit paradigm (Meng et al., 2022; 2023). The basic idea is to first locate influential parameters $\boldsymbol{W}$ through causal tracing, and then edit them by introducing a perturbation $\boldsymbol{\Delta}$ (Meng et al., 2023). The common objective for solving $\boldsymbol{\Delta}$ is to minimize the output error on the to-be-updated knowledge, denoted as e_1. Additionally, the output error on the to-be-preserved knowledge, e_0, is typically incorporated into the objective function, acting as a constraint to ensure the model's accuracy on the preserved knowledge.

Despite their success, the current paradigm faces a critical limitation: it struggles to maintain a balance between knowledge-update error e_1 and knowledge-preservation error e_0. Specifically, to prioritize the success of update, prior studies focus more on minimizing e_1 by assigning a larger weight, while taking insufficient control over e_0. This could make the LLMs after editing (*i.e.,* the post-edited LLMs) overfit to the updated knowledge. This overfitting would introduce a distribution shift of the hidden representations within LLMs. Figure 1 (b) showcases this shift, where the hidden representations in the post-edited LLaMA-3 (8B)[1] diverge from their distribution in the original LLM

[1] https://llama.meta.com/lama3/

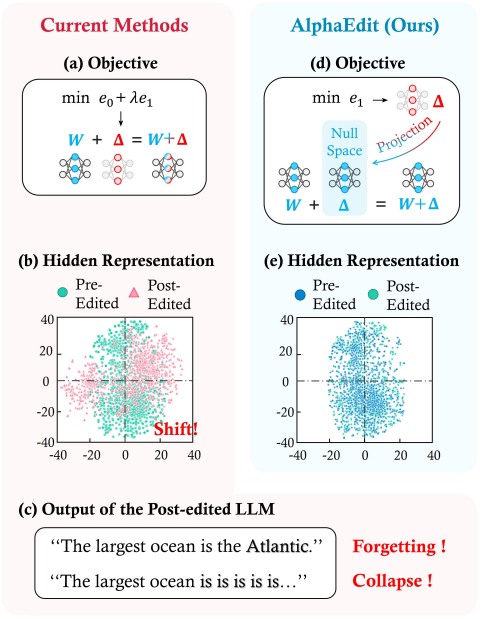

Figure 1: Comparison between the current methods and AlphaEdit. (a) and (d) exhibit the objectives, where λ is the coefficient to keep balance between e_0 and e_1 in the objective; (b) and (e) show the distributions of hidden representations after dimensionality reduction within the pre-edited and post-edited LLaMA3, respectively; (c) depicts the output of the post-edited LLaMA3. Best viewed in color.

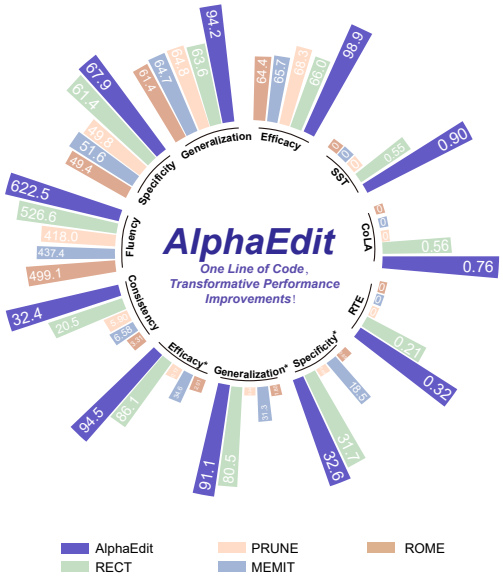

Figure 2: Performance of various model editing methods on LLaMA3 (8B). Results with asterisks in the superscript are from the ZsRE dataset. SST, RTE, and CoLA demonstrate the general capabilities of the post-edited LLMs. The experiments are conducted with 2000 edited samples for sequential editing. Detailed settings and results are provided in Section 4.2 and Table 1, respectively. Best viewed in color.

(*i.e.,* the pre-edited LLM). Worse still, in sequential editing scenarios (Gupta & Anumanchipalli, 2024) where the LLM undergoes multiple sequential edits, the accumulation of overfitting gradually erodes the model's ability to preserve knowledge and generate coherent sentences, eventually leading to model forgetting and model collapse, as depicted in Figure 1 (c).

To address these flaws, we instead remove e_0 from the current objective, allowing the model to focus solely on minimizing e_1 without trade-offs. To avoid overfitting to the to-be-updated knowledge, we project the solution of this new objective onto the null space (Wang et al., 2021) of the preserved knowledge before applying it to the model parameters, as shown in Figure 1 (d). By leveraging the mathematical properties of matrix projection and null space, our new objective ensures that the distribution of hidden representations within LLMs remains invariant post-edited, as shown in Figure 1 (e). This invariance enables the post-edited LLMs to reduce e_1 while keeping e_0 close to zero, thereby alleviating the issues of model forgetting and model collapse. A detailed theoretical proof is provided in Section 3. In a nutshell, we term the method as **AlphaEdit**, a simple yet effective editing approach with a null-space constraint for LLMs.

To validate the effectiveness of our method, we conducted extensive experiments on multiple representative LLMs, such as GPT-2 XL (Radford et al., 2019) and LLaMA-3 (8B), in sequential editing scenarios. The results show that, compared to the best-performing baseline methods, **AlphaEdit can achieve over a 36.7% performance improvement on average by adding just one line of code to the conventional model editing method, MEMIT (Meng et al., 2023)**, as illustrated in Figure 2. Furthermore, we empirically verified that this simple optimization can be easily integrated into most existing model editing methods (Ma et al., 2024; Gu et al., 2024), functioning as a plug-and-play enhancement that significantly boosts their performance.

2 PRELIMINARY

2.1 AUTOREGRESSIVE LANGUAGE MODEL

An autoregressive large language model (LLM) predicts the next token $\boldsymbol{x}$ in a sequence based on the preceding tokens. Specifically, the hidden state of $\boldsymbol{x}$ at layer l within the model, denoted as $\boldsymbol{h}^l$, can be calculated as:

$$\boldsymbol{h}^l = \boldsymbol{h}^{l-1} + \boldsymbol{a}^l + \boldsymbol{m}^l, \quad \boldsymbol{m}^l = \boldsymbol{W}_{\text{out}}^l \, \sigma(\boldsymbol{W}_{\text{in}}^l \, \gamma(\boldsymbol{h}^{l-1} + \boldsymbol{a}^l)), \tag{1}$$

where $\boldsymbol{a}^l$ and $\boldsymbol{m}^l$ represent the outputs of the attention block and the feed-forward network (FFN) layer, respectively; $\boldsymbol{W}_{\text{in}}^l$ and $\boldsymbol{W}_{\text{out}}^l$ are the weight matrices of the FFN layers; σ is the non-linear activation function, and γ denotes the layer normalization. Following Meng et al. (2022), we express the attention and FFN modules in parallel here.

It is worth noting that $\boldsymbol{W}_{\text{out}}^l$ within FFN layers is often interpreted as a linear associative memory, functioning as key-value storage for information retrieval (Geva et al., 2021). Specifically, if the knowledge stored in LLMs is formalized as (s, r, o) — representing subject s, relation r, and object o (*e.g.*, s = "The latest Olympic Game", r = "was held in", o = "Paris") — $\boldsymbol{W}_{\text{out}}^l$ associates a set of input keys $\boldsymbol{k}$ encoding (s, r) with corresponding values $\boldsymbol{v}$ encoding (o). That is,

$$\underbrace{\boldsymbol{m}^l}_{\boldsymbol{v}} = \boldsymbol{W}_{\text{out}}^l \, \underbrace{\sigma(\boldsymbol{W}_{\text{in}}^l \, \gamma(\boldsymbol{h}^{l-1} + \boldsymbol{a}^l))}_{\boldsymbol{k}}. \tag{2}$$

This interpretation has inspired most model editing methods to modify the FFN layers for knowledge updates (Meng et al., 2023; Li et al., 2024; Ma et al., 2024). For simplicity, we use $\boldsymbol{W}$ to refer to $\boldsymbol{W}_{\text{out}}^l$ in the following sections.

2.2 MODEL EDITING IN LLMS

Model editing aims to update knowledge stored in LLMs through a single edit or multiple edits (*i.e.*, sequential editing). Each edit modifies the model parameters $\boldsymbol{W}$ by adding a perturbation $\boldsymbol{\Delta}$ in locate-then-edit paradigm. Specifically, suppose each edit needs to update u pieces of knowledge in the form of (s, r, o). The perturbed $\boldsymbol{W}$ is expected to associate u new $\boldsymbol{k}$-$\boldsymbol{v}$ pairs, where $\boldsymbol{k}$ and $\boldsymbol{v}$ encode (s, r) and (o) of the new knowledge, respectively. Let $\boldsymbol{W} \in \mathbb{R}^{d_1 \times d_0}$, where d_0 and d_1 represent the dimensions of the FFN's intermediate and output layers. Then, we can stack these keys and values into matrices following:

$$\boldsymbol{K}_1 = [\boldsymbol{k}_1 \, | \boldsymbol{k}_2 | \ldots | \, \boldsymbol{k}_u] \in \mathbb{R}^{d_0 \times u}, \quad \boldsymbol{V}_1 = [\boldsymbol{v}_1 \, | \boldsymbol{v}_2 | \ldots | \, \boldsymbol{v}_u] \in \mathbb{R}^{d_1 \times u}, \tag{3}$$

where the subscripts of $\boldsymbol{k}$ and $\boldsymbol{v}$ represent the index of the to-be-updated knowledge. Based on these, the objective can be expressed as:

$$\boldsymbol{\Delta} = \arg\min_{\tilde{\boldsymbol{\Delta}}} \left\| (\boldsymbol{W} + \tilde{\boldsymbol{\Delta}}) \boldsymbol{K}_1 - \boldsymbol{V}_1 \right\|^2, \tag{4}$$

where $\|\cdot\|^2$ denotes the sum of the squared elements in the matrix.

Additionally, let $\boldsymbol{K}_0$ and $\boldsymbol{V}_0$ represent the matrices formed by stacking the $\boldsymbol{k}$ and $\boldsymbol{v}$ of the preserved knowledge. Current methods (Meng et al., 2023; Gu et al., 2024) typically incorporate the error involving $\boldsymbol{K}_0$ and $\boldsymbol{V}_0$ to preserve it, as follows:

$$\boldsymbol{\Delta} = \arg\min_{\tilde{\boldsymbol{\Delta}}} \left(\left\| (\boldsymbol{W} + \tilde{\boldsymbol{\Delta}}) \boldsymbol{K}_1 - \boldsymbol{V}_1 \right\|^2 + \left\| (\boldsymbol{W} + \tilde{\boldsymbol{\Delta}}) \boldsymbol{K}_0 - \boldsymbol{V}_0 \right\|^2 \right). \tag{5}$$

Since $\boldsymbol{K}_0$ and $\boldsymbol{V}_0$ encode the preserved knowledge in LLMs, we have $\boldsymbol{W}\boldsymbol{K}_0 = \boldsymbol{V}_0$ (*cf.* Eqn. 2). Thus, by applying the normal equation (Lang, 2012), if the closed-form solution of Eqn. 5 exists, it can be written as:

$$\boldsymbol{\Delta} = (\boldsymbol{V}_1 - \boldsymbol{W}\boldsymbol{K}_1) \, \boldsymbol{K}_1^T \left(\boldsymbol{K}_0 \boldsymbol{K}_0^T + \boldsymbol{K}_1 \boldsymbol{K}_1^T \right)^{-1}. \tag{6}$$

Although $\boldsymbol{K}_0$ is difficult to obtain directly since we hardly have access to the LLM's full extent of knowledge, it can be estimated using abundant text input (Meng et al., 2023). In practical applications, $100,000$ (s, r, o) triplets from Wikipedia are typically randomly selected to encode $\boldsymbol{K}_0$ (Meng et al., 2023), making $\boldsymbol{K}_0$ a high-dimensional matrix with $100,000$ columns (*i.e.*, $\boldsymbol{K}_0 \in \mathbb{R}^{d_0 \times 100,000}$). See Appendix B.1 for detailed implementation steps.

3 METHOD

In this section, we first introduce the concept of the null space and its relationship to model editing (Section 3.1). Based on this, we present the method for projecting a given perturbation $\boldsymbol{\Delta}$ onto the null space of the matrix $\boldsymbol{K}_0$, which encodes the persevered knowledge (Section 3.2). Following that, a new editing objective that incorporates the aforementioned projection method is introduced in Section 3.3.

3.1 NULL SPACE

Null space is at the core of our work. Here we first introduce the definition of the left null space (hereafter referred to simply as *null space*). Specifically, given two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, $\boldsymbol{B}$ is in the null space of $\boldsymbol{A}$ if and only if $\boldsymbol{BA} = \boldsymbol{0}$. See Adam-NSCL (Wang et al., 2021) for more details.

In the context of model editing, if the perturbation $\boldsymbol{\Delta}$ is projected into the null space of $\boldsymbol{K}_0$ (*i.e.*, $\boldsymbol{\Delta}'\boldsymbol{K}_0 = \boldsymbol{0}$, where $\boldsymbol{\Delta}'$ denotes the projected perturbation), adding it to the parameters $\boldsymbol{W}$ results in:

$$(\boldsymbol{W} + \boldsymbol{\Delta}')\boldsymbol{K}_0 = \boldsymbol{W}\boldsymbol{K}_0 = \boldsymbol{V}_0. \tag{7}$$

This implies that **the projected $\boldsymbol{\Delta}$ will not disrupt the key-value associations of the preserved knowledge** (*i.e.*, $\{\boldsymbol{K}_0, \boldsymbol{V}_0\}$), ensuring that the storage of the preserved knowledge remains intact.

Therefore, in this work, before adding perturbation $\boldsymbol{\Delta}$ to the model parameters $\boldsymbol{W}$, we project it onto the null space of $\boldsymbol{K}_0$ to protect the preserved knowledge. This protection allows us to remove the first term — the term focusing on protecting the preserved knowledge — from the objective in Eqn. 5.

3.2 NULL SPACE PROJECTING

In Section 3.1, we briefly explained why $\boldsymbol{\Delta}$ should be projected into the null space of $\boldsymbol{K}_0$. In this part, we focus on how to implement this projection.

As introduced at the end of Section 2.2, the matrix $\boldsymbol{K}_0 \in \mathbb{R}^{d_0 \times 100{,}000}$ has a high dimensionality with $100,000$ columns. Hence, directly projecting the given perturbation $\boldsymbol{\Delta}$ onto the null space of $\boldsymbol{K}_0$ presents significant computational and storage challenges. In sight of this, we adopt the null space of the non-central covariance matrix $\boldsymbol{K}_0\boldsymbol{K}_0^T \in \mathbb{R}^{d_0 \times d_0}$ as a substitute to reduce computational complexity, as d_0 is typically much smaller than $100,000$. This matrix's null space is equal to that of $\boldsymbol{K}_0$ (please see Appendix B.2 for detailed proof).

Following the existing methods for conducting null space projection (Wang et al., 2021), we first apply a Singular Value Decomposition (SVD) to $\boldsymbol{K}_0(\boldsymbol{K}_0)^T$:

$$\left\{\boldsymbol{U}, \boldsymbol{\Lambda}, (\boldsymbol{U})^T\right\} = \text{SVD}\left(\boldsymbol{K}_0(\boldsymbol{K}_0)^T\right), \tag{8}$$

where each column in $\boldsymbol{U}$ is an eigenvector of $\boldsymbol{K}_0(\boldsymbol{K}_0)^T$. Then, we remove the eigenvectors in $\boldsymbol{U}$ that correspond to non-zero eigenvalues[2], and define the remaining submatrix as $\hat{\boldsymbol{U}}$. Based on this, the projection matrix $\boldsymbol{P}$ can be defined as follows:

$$\boldsymbol{P} = \hat{\boldsymbol{U}}(\hat{\boldsymbol{U}})^T. \tag{9}$$

This projection matrix can map the column vectors of $\boldsymbol{\Delta}$ into the null space of $\boldsymbol{K}_0(\boldsymbol{K}_0)^T$, as it satisfies the condition $\boldsymbol{\Delta P} \cdot \boldsymbol{K}_0(\boldsymbol{K}_0)^T = \boldsymbol{0}$. The detailed derivation is exhibited in Appendix B.3.

Since $\boldsymbol{K}_0$ and $\boldsymbol{K}_0(\boldsymbol{K}_0)^T$ share the same null space, we can derive $\boldsymbol{\Delta P} \cdot \boldsymbol{K}_0 = \boldsymbol{0}$. Hence, we have:

$$(\boldsymbol{W} + \boldsymbol{\Delta P})\boldsymbol{K}_0 = \boldsymbol{W}\boldsymbol{K}_0 = \boldsymbol{V}_0. \tag{10}$$

This shows the projection matrix $\boldsymbol{P}$ ensures that the model edits occur without interference with the preserved knowledge in LLMs.

3.3 NULL-SPACE CONSTRAINED MODEL EDITING

Section 3.2 has provided how to apply projection to ensure that preserved knowledge is not disrupted. Here, we introduce how to leverage this projection to optimize the current model editing objective.

[2]Given that eigenvalues are rarely strictly zero in practical applications, in our experiments, we remove the eigenvectors corresponding to the eigenvalues above 10^{-2}.

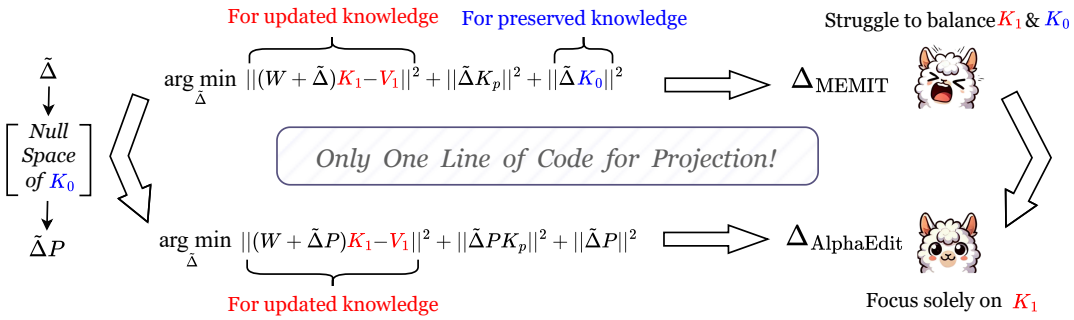

Figure 3: Comparison between the paradigms of AlphaEdit and current method. Best viewed in color.

Starting with the single-edit objective in Eqn. 5, the optimization follows three steps: (1) Replace $\boldsymbol{\Delta}$ with the projected perturbation $\boldsymbol{\Delta P}$, ensuring that the perturbation would not disrupt the preserved knowledge; (2) Remove the first term involving $\boldsymbol{K}_0$, as Step (1) has already protected the preserved knowledge from being disrupted; (3) Add a regularization term $||\boldsymbol{\Delta P}||^2$ to guarantee stable convergence. With these optimizations, Eqn. 5 becomes:

$$\boldsymbol{\Delta} = \arg\min_{\tilde{\boldsymbol{\Delta}}} \left(||(\boldsymbol{W} + \tilde{\boldsymbol{\Delta}}\boldsymbol{P})\boldsymbol{K}_1 - \boldsymbol{V}_1||^2 + \|\tilde{\boldsymbol{\Delta}}\boldsymbol{P}\|^2 \right), \tag{11}$$

where $\boldsymbol{K}_1$ and $\boldsymbol{V}_1$ denote the key and value matrices of to-be-updated knowledge defined in Eqn. 3.

In sequential editing tasks, during the current edit, we also need to add a term to the objective (*cf.* Eqn. 11) to prevent the perturbation from disrupting the updated knowledge in previous edits. Let $\boldsymbol{K}_p$ and $\boldsymbol{V}_p$ present the key and value matrices of the previously updated knowledge, analogous to the earlier definitions of $\boldsymbol{K}_1$ and $\boldsymbol{V}_1$. This term should minimize $||(\boldsymbol{W} + \tilde{\boldsymbol{\Delta}}\boldsymbol{P})\boldsymbol{K}_p - \boldsymbol{V}_p||^2$ to protect the association. Since the related knowledge has been updated in previous edits, we have $\boldsymbol{W}\boldsymbol{K}_p = \boldsymbol{V}_p$. Hence, this term can be simplified to $||\tilde{\boldsymbol{\Delta}}\boldsymbol{P}\boldsymbol{K}_p||^2$, and adding it to Eqn. 11 gives the new objective:

$$\boldsymbol{\Delta} = \arg\min_{\tilde{\boldsymbol{\Delta}}} \left(||(\boldsymbol{W} + \tilde{\boldsymbol{\Delta}}\boldsymbol{P})\boldsymbol{K}_1 - \boldsymbol{V}_1||^2 + \|\tilde{\boldsymbol{\Delta}}\boldsymbol{P}\|^2 + ||\tilde{\boldsymbol{\Delta}}\boldsymbol{P}\boldsymbol{K}_p||^2 \right). \tag{12}$$

To facilitate expression, we define the residual vector of the current edit as $\boldsymbol{R} = \boldsymbol{V}_1 - \boldsymbol{W}\boldsymbol{K}_1$. Based on this, Eqn. 12 can be solved using the normal equation (Lang, 2012):

$$(\boldsymbol{\Delta P}\boldsymbol{K}_1 - \boldsymbol{R})\boldsymbol{K}_1^T\boldsymbol{P} + \boldsymbol{\Delta P} + \boldsymbol{\Delta P}\boldsymbol{K}_p\boldsymbol{K}_p^T\boldsymbol{P} = \boldsymbol{0}. \tag{13}$$

Solving Eqn. 13 yields the final perturbation $\boldsymbol{\Delta}_{\textbf{AlphaEdit}} = \boldsymbol{\Delta P}$ which will be added to the model parameters $\boldsymbol{W}$:

$$\boldsymbol{\Delta}_{\textbf{AlphaEdit}} = \boldsymbol{R}\boldsymbol{K}_1^T\boldsymbol{P}(\boldsymbol{K}_p\boldsymbol{K}_p^T\boldsymbol{P} + \boldsymbol{K}_1\boldsymbol{K}_1^T\boldsymbol{P} + \boldsymbol{I})^{-1}. \tag{14}$$

The invertibility of the term within the brackets in the above equation is proved in Appendix B.4. This solution $\boldsymbol{\Delta}_{\textbf{AlphaEdit}}$ could not only store the to-be-updated knowledge in the current edit, but also ensure that both the preserved knowledge and the previously updated knowledge remain unaffected. Furthermore, for better comparison, we also present the commonly used solution in existing methods like MEMIT (Meng et al., 2023) as follows[3]:

$$\boldsymbol{\Delta}_{\textbf{MEMIT}} = \boldsymbol{R}\boldsymbol{K}_1^T(\boldsymbol{K}_p\boldsymbol{K}_p^T + \boldsymbol{K}_1\boldsymbol{K}_1^T + \boldsymbol{K}_0\boldsymbol{K}_0^T)^{-1}. \tag{15}$$

By comparing Eqn. 14 and 15, it is evident that our approach requires only a minor modification to the standard solution by incorporating the projection matrix $\boldsymbol{P}$. This makes our method easily integrable into existing model editing algorithms. Figure 3 summarizes this modification from the perspective of convergence objectives. We emphasize that **by adding just a single line of code for this modification, the performance of most editing methods could be significantly enhanced**, as shown in Figure 2. More detailed experimental results are exhibited in Section 4.

Furthermore, since the projection matrix $\boldsymbol{P}$ is entirely independent of the to-be-updated knowledge, it only needs to be computed once and can then be directly applied to any downstream editing tasks. Consequently, AlphaEdit introduces negligible additional time consumption compared to baselines, making it both efficient and effective.

[3] $\Delta_{\textbf{MEMIT}}$ denotes the solution provided by MEMIT in sequential editing tasks.

Table 1: Comparison of AlphaEdit with existing methods on the sequential model editing task. *Eff.*, *Gen.*, *Spe.*, *Flu.* and *Consis.* denote Efficacy, Generalization, Specificity, Fluency and Consistency, respectively. The best results are highlighted in bold, while the second-best results are underlined.

Method	Model	Counterfact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
Pre-edited		7.85±0.26	10.58±0.26	89.48±0.18	635.23±0.11	24.14±0.08	36.99±0.30	36.34±0.30	31.89±0.22
FT	LLaMA3	83.33±0.37	67.79±0.40	46.63±0.37	233.72±0.22	8.77±0.05	30.48±0.26	30.22±0.32	15.49±0.17
MEND	LLaMA3	63.24±0.31	61.17±0.36	45.37±0.38	372.16±0.80	4.21±0.05	0.91±0.05	1.09±0.05	0.53±0.02
InstructEdit	LLaMA3	66.58±0.24	64.18±0.35	47.14±0.37	443.85±0.78	7.28±0.04	1.58±0.04	1.36±0.08	1.01±0.05
ROME	LLaMA3	64.40±0.41	61.42±0.42	49.44±0.38	449.06±0.26	3.31±0.02	2.01±0.07	1.80±0.07	0.69±0.03
MEMIT	LLaMA3	65.65±0.47	64.65±0.42	51.56±0.38	437.43±1.67	6.58±0.11	34.62±0.36	31.28±0.34	18.49±0.19
PRUNE	LLaMA3	68.25±0.46	64.75±0.41	49.82±0.36	418.03±1.52	5.90±0.10	24.77±0.27	23.87±0.27	20.69±0.23
RECT	LLaMA3	66.05±0.47	63.62±0.43	61.41±0.37	526.62±0.44	20.54±0.09	86.05±0.23	80.54±0.27	31.67±0.22
AlphaEdit	LLaMA3	**98.90**±0.10	**94.22**±0.19	**67.88**±0.29	**622.49**±0.16	**32.40**±0.11	**94.47**±0.13	**91.13**±0.19	**32.55**±0.22
Pre-edited		16.22±0.31	18.56±0.45	83.11±0.13	621.81±0.67	29.74±0.51	26.32±037	25.79±0.25	27.42±0.53
FT	GPT-J	92.15±0.27	72.38±0.38	43.35±0.37	297.92±0.77	6.65±0.10	72.37±0.29	68.91±0.32	19.66±0.23
MEND	GPT-J	46.15±0.50	46.22±0.51	53.90±0.48	242.41±0.41	3.94±0.03	0.71±0.04	0.71±0.04	0.52±0.03
InstructEdit	GPT-J	50.62±0.58	51.73±0.42	56.28±0.50	245.89±0.44	4.21±0.04	0.92±0.07	0.88±0.03	0.65±0.06
ROME	GPT-J	57.50±0.48	54.20±0.40	52.05±0.31	589.42±0.08	3.22±0.02	56.42±0.42	54.65±0.42	9.86±0.16
MEMIT	GPT-J	98.55±0.11	95.50±0.16	63.64±0.31	546.28±0.88	34.89±0.15	94.91±0.16	90.22±0.23	**30.39**±0.27
PRUNE	GPT-J	86.15±0.34	86.85±0.29	53.87±0.35	427.14±0.53	14.78±0.11	0.15±0.02	0.15±0.02	0.00±0.00
RECT	GPT-J	98.80±0.10	86.58±0.28	72.22±0.28	617.31±0.19	41.39±0.12	96.38±0.14	91.21±0.21	27.79±0.26
AlphaEdit	GPT-J	**99.75**±0.08	**96.38**±0.23	**75.48**±0.21	**618.50**±0.17	**42.08**±0.15	**99.79**±0.14	**96.00**±0.22	28.29±0.25
Pre-edited		22.23±0.73	24.34±0.62	78.53±0.33	626.64±0.31	31.88±0.20	22.19±0.24	31.30±0.27	24.15±0.32
FT	GPT2-XL	63.55±0.48	42.20±0.41	57.06±0.30	519.35±0.27	10.56±0.05	37.11±0.39	33.30±0.37	10.36±0.17
MEND	GPT2-XL	50.80±0.50	50.80±0.48	49.20±0.51	407.21±0.08	1.01±0.00	0.00±0.00	0.00±0.00	0.00±0.00
InstructEdit	GPT2-XL	55.32±0.58	53.63±0.42	53.25±0.62	412.57±0.15	1.08±0.03	3.54±0.03	4.25±0.02	3.23±0.04
ROME	GPT2-XL	54.60±0.48	51.18±0.40	52.68±0.33	366.13±1.40	0.72±0.02	47.50±0.43	43.56±0.42	14.27±0.19
MEMIT	GPT2-XL	94.70±0.22	85.82±0.28	60.50±0.32	477.26±0.54	22.72±0.15	79.17±0.32	71.44±0.36	**26.42**±0.25
PRUNE	GPT2-XL	82.05±0.38	78.55±0.34	53.02±0.35	530.47±0.39	15.93±0.11	21.62±0.30	19.27±0.28	13.19±0.18
RECT	GPT2-XL	92.15±0.26	81.15±0.33	65.13±0.31	480.83±0.62	21.05±0.16	81.02±0.31	73.08±0.35	24.85±0.25
AlphaEdit	GPT2-XL	**99.50**±0.24	**93.95**±0.34	**66.39**±0.31	**597.88**±0.18	**39.38**±0.15	**94.81**±0.30	**86.11**±0.29	25.88±0.21

4 EXPERIMENT

In this section, we conduct experiments to address the following research questions:

- **RQ1:** How does AlphaEdit perform on sequential editing tasks compared to baseline methods? Can it mitigate the issues of model forgetting and model collapse exhibited in Figure 1?
- **RQ2:** How does AlphaEdit-edited LLM perform on general ability evaluations? Does the post-edited LLM successfully retain its inherent capabilities?
- **RQ3:** Can AlphaEdit effectively prevent the model from overfitting to updated knowledge? Specifically, can the post-edited LLM avoid shifts in the distribution of hidden representations?
- **RQ4:** Can the performance of baseline methods be significantly improved with a single line of code in AlphaEdit (*i.e.*, the code for projection)?

4.1 EXPERIMENTAL SETUP

We begin by briefly outlining the evaluation metrics, datasets, and baseline methods. For more detailed descriptions of the experimental settings, please refer to Appendix A.

Base LLMs & Baseline Methods. Our experiments are conducted on three LLMs: GPT2-XL (1.5B), GPT-J (6B), and Llama3 (8B). We compare our method against several model editing baselines, including Fine-Tuning (FT) (Zhu et al., 2020), MEND (Mitchell et al., 2022a), InstructEdit (Zhang et al., 2024b), ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), PRUNE (Ma et al., 2024), and

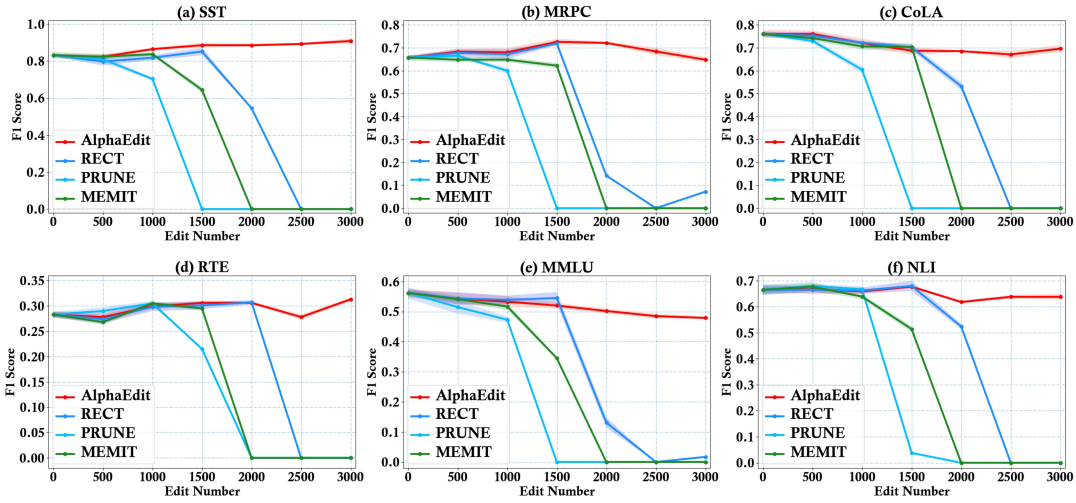

Figure 4: F1 scores of the post-edited LLaMA3 (8B) on six tasks (*i.e.*, SST, MRPC, CoLA, RTE, MMLU and NLI) used for general capability testing. Best viewed in color.

RECT (Gu et al., 2024). Furthermore, to further validate the generalizability of AlphaEdit, we include three memory-based editing methods (*i.e.*, SERAC (Mitchell et al., 2022b), GRACE (Hartvigsen et al., 2023) and MELO (Yu et al., 2024)) as baselines in Appendix C.4, along with two additional base LLMs: Gemma (Mesnard et al., 2024) and phi-1.5 (Li et al., 2023) in Appendix C.4.

Datasets & Evaluation Metrics. We evaluate AlphaEdit using two widely adopted benchmarks: the Counterfact dataset (Meng et al., 2022) and the ZsRE dataset (Levy et al., 2017). In line with prior works (Meng et al., 2022), we employ **Efficacy** (efficiency success), **Generalization** (paraphrase success), **Specificity** (neighborhood success), **Fluency** (generation entropy), and **Consistency** (reference score) as evaluation metrics. In addition, for comprehensive evaluation, Appendix C.7 presents experiments conducted on three additional datasets: LongformEvaluation (Rosati et al., 2024), MQUAKE (Zhong et al., 2023), and KnowEdit (Zhang et al., 2024c). We encourage interested readers to refer to Appendix C.7 for further details.

4.2 Performance on Knowledge Update and Preservation (RQ1)

To evaluate the performance of different editing methods in terms of knowledge update and retention, we conducted sequential editing on three base LLMs using AlphaEdit and the baseline methods. Table 1 presents the results under a commonly used configuration for the sequential editing task, where 2,000 samples are randomly drawn from the dataset for updates, with 100 samples per edit (*i.e.*, a batch size of 100). For additional experimental results, such as case studies of model outputs after editing, please refer to Appendix C. Based on Table 1, we can draw the following observations:

- **Obs 1: AlphaEdit achieves superior performance across nearly all metrics and base models.** Specifically, in terms of Efficacy and Generalization metrics, AlphaEdit provides an average improvement of 12.54% and 16.78%, respectively, over the best baseline. On Llama3, these performance boosts are even more remarkable (*i.e.,* 32.85% $\uparrow$ and 30.60% $\uparrow$). These gains arise from AlphaEdit's ability to mitigate the trade-off between updating and preserving knowledge.
- **Obs 2: AlphaEdit enhances text generation fluency and coherence.** In addition to editing capabilities, AlphaEdit also exhibits substantial improvements in Fluency and Coherence. For instance, on GPT2-XL, AlphaEdit achieves an 18.33% improvement over the strongest baseline, demonstrating that it can preserve both the knowledge and the ability to generate fluent text.

4.3 General Capability Tests (RQ2)

To further evaluate the intrinsic knowledge of post-edited LLMs, we perform General Capability Tests using six natural language tasks from the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019). Specifically, the evaluation tasks include:

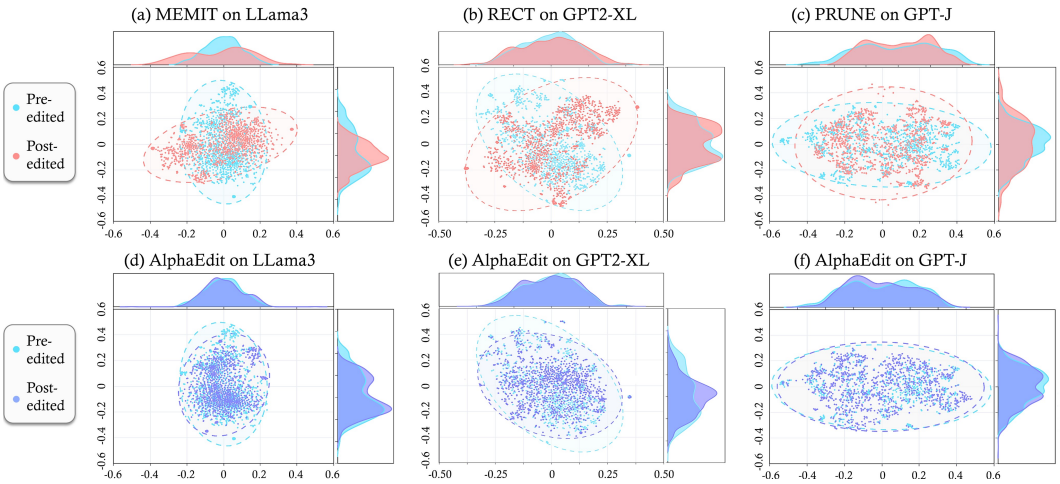

Figure 5: The distribution of hidden representations of pre-edited and post-edited LLMs after dimensionality reduction. The top and right curve graphs display the marginal distributions for two reduced dimensions. The dashed lines represent the 0.95 confidence intervals, where AlphaEdit consistently exhibits minimal shift. Best viewed in color.

1. **SST (The Stanford Sentiment Treebank)** (Socher et al., 2013) is a single-sentence classification task involving sentences from movie reviews and their corresponding human-annotated sentiment labels. The task requires classifying the sentiment into two categories.
2. **MRPC (Microsoft Research Paraphrase Corpus)** (Dolan & Brockett, 2005) is a well-known benchmark for text matching and semantic similarity assessment. In the MRPC task, the objective is to determine whether a given pair of sentences is semantically equivalent.
3. **MMLU (Massive Multi-task Language Understanding)** (Hendrycks et al., 2021) is a comprehensive evaluation designed to measure the multi-task accuracy of text models. This assessment focuses on evaluating models under zero-shot and few-shot settings.
4. **RTE (Recognizing Textual Entailment)** (Bentivogli et al., 2009) involves natural language inference that determines if a premise sentence logically entails a hypothesis sentence.
5. **CoLA (Corpus of Linguistic Acceptability)** (Warstadt et al., 2019) is a single-sentence classification task, where sentences are annotated as either grammatically acceptable or unacceptable.
6. **NLI (Natural Language Inference)** (Williams et al., 2018) focuses on natural language understanding, requiring the model to infer the logical relationship between pairs of sentences.

Figure 4 illustrates the performance as the number of edited samples increases across six tasks. More results are provided in Appendix C.2. Based on Figure 4, we have the following observations:

- **Obs 3: AlphaEdit sustains the general capability of post-edited LLMs even after extensive editing.** Specifically, AlphaEdit maintains the original model performance across all metrics, even after editing 3,000 samples. Remarkably, AlphaEdit preserves 98.48% of the model's general capability on average, demonstrating that the null-space projection not only safeguards the preserved knowledge but also protects the general capability learned from this knowledge's corpus.
- **Obs 4: LLMs edited with baseline methods experience significant degradation of general capability after editing 2,000 samples.** Specifically, in this case, all metrics are rapidly approaching zero, confirming our theoretical analysis that the common objective are inherently flawed and fail to balance knowledge update and preservation.

4.4 Hidden Representations Analysis (RQ3)

As discussed in previous sections, current editing methods often cause post-edited LLMs to overfit to the updated knowledge, leading to a shift in the distribution of hidden representations. Hence, here we aim to empirically verify that AlphaEdit can prevent overfitting and avoid this distributional shift. To validate it, we conducted the following steps: (1) We randomly select 1,000 factual prompts and extract the hidden representations within pre-edited LLMs. (2) Subsequently, we performed 2,000 sequential edits on the LLMs and recomputed these hidden representation. (3) Finally, we

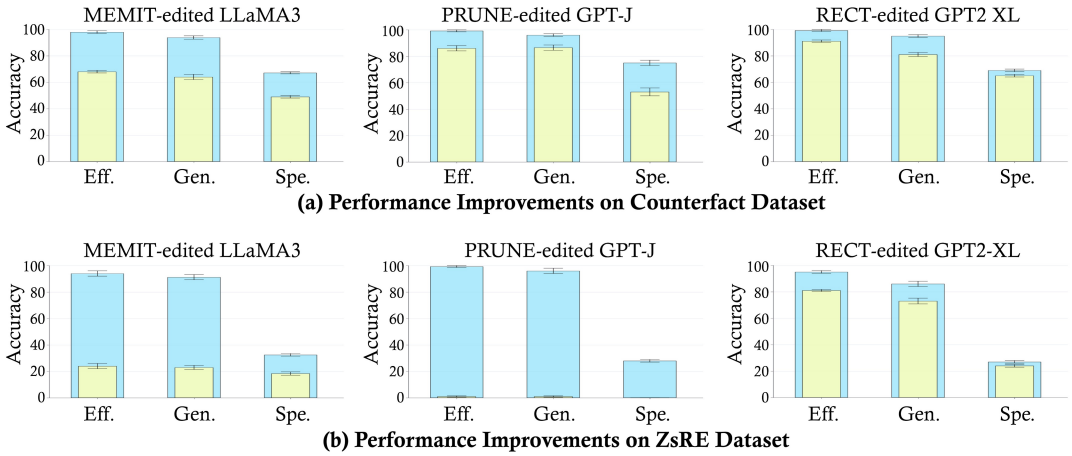

Figure 6: Performance improvements of baseline editing methods (*i.e.*, MEMIT, PRUNE and RECT) after **adding a single line of code from AlphaEdit** (*i.e.*, the code used for matrix projection). The yellow bars represent the original performance of each baseline, while the blue bars represent the performance after the addition. Best viewed in color.

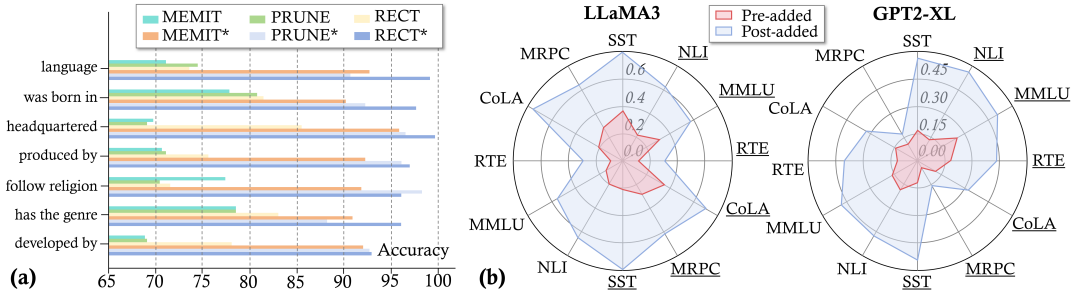

Figure 7: Performance comparison before and after adding the projection code in AlphaEdit to the baselines. (a) Performance of editing methods on samples involving specific semantics, where asterisk denotes the versions added the projection; the vertical and horizontal axis represent the categories of knowledge and the accuracy of LLM responses involving this knowledge, respectively. (b) Comparison of general capabilities for PRUNE and RECT before and after adding the projection, where the underlined method represents the results of RECT. Best viewed in color.

used t-SNE (Van der Maaten & Hinton, 2008) to visualize the hidden representation before and after editing. Figure 5 exhibits them and their marginal distribution curves. Furthermore, we quantify the deviation of scatter point distributions and the differences of marginal distribution curves in Figure 5, and the results are provided in Appendix C.3. According to Figure 5 we can find that:

- **Obs 5: AlphaEdit maintains consistency in hidden representations after editing.** Specifically, the hidden representations within LLMs edited using AlphaEdit remain consistent with the original distribution across all three base models. This stability indicates that AlphaEdit effectively mitigates overfitting, in turn explaining its superior knowledge preservation and generalization capabilities.
- **Obs 6: There is a significant shift in the distribution of hidden representations within baseline-edited LLMs.** In some cases (*e.g.,* RECT-edited LLaMA3), the trend of the distribution before and after editing is even completely reversed. This discrepancy becomes more pronounced as sequential editing progresses, further underscoring the importance of projection-based optimization.

4.5 Performance Improvements of Baseline Methods (RQ4)

We conclude by evaluating whether integrating AlphaEdit's projection strategy can comprehensively enhance current editing methods. To achieve this, we add one line of code from AlphaEdit (the code

for projection) to baselines and measure their performance before and after the addition. Following previous works (Meng et al., 2023; Gu et al., 2024), we analyze (1) the average performance across all editing samples, (2) the performance on editing samples involving specific semantics and (3) the general capability of post-edited LLMs. Results are shown in Figure 6, 7 (a), and 7 (b), respectively. Detailed experimental settings can be found in Appendix C.4. The results show that:

- **Obs 7: AlphaEdit seamlessly integrates with other model editing methods, significantly boosting their overall performance.** Specifically, in terms of editing capability, the optimized baseline methods show an average improvement of 28.24%, while the general capability improves by 42.65%. These results underscore the substantial potential and broad applicability of the null-space projection approach in enhancing model editing approaches.

5 Related Work

Parameter-modifying Model Editing. This approach typically employs meta-learning or locating-then-editing strategies (Ma et al., 2024; Gu et al., 2024; Zhang et al., 2024d) to conduct editing. Meta-learning, as implemented by KE (Cao et al., 2021), MEND (Mitchell et al., 2022a), involves adapting model parameters through a hypernetwork. InstructEdit (Zhang et al., 2024b) extends MEND by designing instructions for training on different tasks. Locate-then-edit strategies, exemplified by ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023), prioritize pinpointing the knowledge's storage location before making targeted edits. Additionally, GLAME (Zhang et al., 2024a) enhances ROME by leveraging knowledge graphs to facilitate the editing of related knowledge.

Parameter-preserving Model Editing. This line utilizes extra trainable parameters to maintain necessary memories (Wang et al., 2024). For instance, T-Patcher (Huang et al., 2023) integrates additional neurons to rectify each misclassified output. Further innovations include MemPrompt (Madaan et al., 2022) and IKE (Zheng et al., 2023b), which enhance the model's in-context learning capabilities by introducing modified facts and memory-retrieved demonstrations as prompts. SERAC (Mitchell et al., 2022b) constructs an ancillary counterfactual model, utilizing a classifier to determine response applicability, while MELO (Yu et al., 2024) dynamically engages specific LoRA blocks, indexed within an internal vector database, to modify model behavior. Conversely, GRACE (Hartvigsen et al., 2023) adopts a unique approach by maintaining a dynamically updated codebook.

Evaluating Knowledge Editing. Recent work has introduced diverse benchmarks to assess the efficacy of model editing. For instance, KnowEdit (Zhang et al., 2024c) provides a unified framework focusing on preserving overall performance while addressing specific domains; LEME (Rosati et al., 2024) shifts attention to long-form generative outputs, while MQUAKE (Zhong et al., 2023) emphasizes multi-hop reasoning, testing the ripple effects of factual updates on related beliefs. These benchmarks collectively push for more comprehensive evaluations.

6 Limitations & Future Discussion

While we acknowledge the effectiveness of our method, we also recognize its limitations. Concretely, our method has been tested exclusively within the locating-and-editing paradigm, but we have not extended it to other frameworks, such as parameter-preserving methods. Moving forward, future research could focus on extending AlphaEdit to a broader range of editing paradigms. These directions offer promising avenues for improving both the generality and impact of our approach.

7 Conclusion

In this work, we introduced AlphaEdit, a novel model editing method to address a critical challenge in current approaches — the trade-off between knowledge update and preservation — with only a single line of code. Specifically, AlphaEdit minimizes disruption to the preserved knowledge by projecting parameter perturbations onto the null space of its key matrices. It then removes the output error related to it from the current objective, allowing the model to focus solely on knowledge update without trade-off. Extensive experiments on multiple base LLMs, including LLaMA3, GPT-2 XL, and GPT-J, demonstrate that AlphaEdit significantly enhances the performance of existing model editing methods, delivering an average improvement of 36.7% in editing capabilities.

ETHICS STATEMENT

Our AlphaEdit method significantly enhances the performance of sequential model editing, making it invaluable for updating and managing knowledge in real-world applications. While the ability to directly modify stored knowledge introduces potential risks, such as the introduction of false or harmful information, we strongly urge researchers to implement strict validation and oversight to ensure the ethical use of these techniques. Nevertheless, the original goal of model editing is positive, aiming to facilitate efficient updates of large models in the future. Therefore, we encourage researchers to leverage this technology responsibly and with care.

REPRODUCIBILITY

To ensure the reproducibility of our findings, detailed implementation instructions for AlphaEdit can be found in Appendix A. Additionally, the source code is publicly available at the following URL: https://anonymous.4open.science/r/AlphaEdit-0651. These measures are intended to facilitate the verification and replication of our results by other researchers in the field.

REFERENCES

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. The fifth PASCAL recognizing textual entailment challenge. In *TAC*. NIST, 2009.

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

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *EMNLP (1)*, pp. 6491–6506. Association for Computational Linguistics, 2021.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP*. Asian Federation of Natural Language Processing, 2005.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *EMNLP (1)*, pp. 5484–5495. Association for Computational Linguistics, 2021.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing harms general abilities of large language models: Regularization to the rescue, 2024. URL https://arxiv.org/abs/2401.04700.

Akshat Gupta and Gopala Anumanchipalli. Rebuilding ROME : Resolving model collapse during sequential model editing. *CoRR*, abs/2403.07175, 2024.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with GRACE: lifelong model editing with discrete key-value adaptors. In *NeurIPS*, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*. OpenReview.net, 2021.

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

Serge Lang. *Introduction to linear algebra*. Springer Science & Business Media, 2012.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *CoNLL*, pp. 333–342. Association for Computational Linguistics, 2017.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. PMET: precise model editing in a transformer. In *AAAI*, pp. 18564–18572. AAAI Press, 2024.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need II: phi-1.5 technical report. *CoRR*, abs/2309.05463, 2023.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. Perturbation-restrained sequential model editing. *CoRR*, abs/2405.16821, 2024.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. Memory-assisted prompt editing to improve GPT-3 after deployment. *CoRR*, abs/2201.06009, 2022.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *NeurIPS*, 2022.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *ICLR*. OpenReview.net, 2023.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Cristian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, and et al. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295, 2024.

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

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15817–15831. PMLR, 2022b.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. Language models as knowledge bases? In *EMNLP/IJCNLP (1)*, pp. 2463–2473. Association for Computational Linguistics, 2019.

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

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *EMNLP (1)*, pp. 5418–5426. Association for Computational Linguistics, 2020.

Domenic Rosati, Robie Gonzales, Jinkun Chen, Xuemin Yu, Yahya Kayani, Frank Rudzicz, and Hassan Sajjad. Long-form evaluation of model editing. In *NAACL-HLT*, pp. 3749–3780. Association for Computational Linguistics, 2024.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pp. 1631–1642. ACL, 2013.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR (Poster)*. OpenReview.net, 2019.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *arXiv preprint arXiv:2405.14768*, 2024.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *CVPR*, pp. 184–193. Computer Vision Foundation / IEEE, 2021.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguistics*, 7:625–641, 2019.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pp. 1112–1122. Association for Computational Linguistics, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. MELO: enhancing model editing with neuron-indexed dynamic lora. In *AAAI*, pp. 19449–19457. AAAI Press, 2024.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. Knowledge graph enhanced large language model editing. *CoRR*, abs/2402.13593, 2024a.

Ningyu Zhang, Bozhong Tian, Siyuan Cheng, Xiaozhuan Liang, Yi Hu, Kouying Xue, Yanjie Gou, Xi Chen, and Huajun Chen. Instructedit: Instruction-based knowledge editing for large language models. In *IJCAI*, pp. 6633–6641. ijcai.org, 2024b.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. A comprehensive study of knowledge editing for large language models. *CoRR*, abs/2401.01286, 2024c.

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

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? In *EMNLP*, pp. 4862–4876. Association for Computational Linguistics, 2023a.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? *CoRR*, abs/2305.12740, 2023b.

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. Mquake: Assessing knowledge editing in language models via multi-hop questions. In *EMNLP*, pp. 15686–15702. Association for Computational Linguistics, 2023.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix X. Yu, and Sanjiv Kumar. Modifying memories in transformer models. *CoRR*, abs/2012.00363, 2020.

A EXPERIMENTAL SETUP

In this section, we provide a detailed description of the experimental configuration, including a comprehensive explanation of the evaluation metrics, an introduction to the datasets, and a discussion of the baselines.

A.1 DATASETS

Here, we provide a detailed introduction to the datasets used in this paper:

- **Counterfact** (Meng et al., 2022) is a more challenging dataset that contrasts counterfactual with factual statements, initially scoring lower for Counterfact. It constructs out-of-scope data by replacing the subject entity with approximate entities sharing the same predicate. The Counterfact dataset has similar metrics to ZsRE for evaluating efficacy, generalization, and specificity. Additionally, Counterfact includes multiple generation prompts with the same meaning as the original prompt to test the quality of generated text, specifically focusing on fluency and consistency.
- **ZsRE** (Levy et al., 2017) is a question answering (QA) dataset that uses questions generated through back-translation as equivalent neighbors. Following previous work, natural questions are used as out-of-scope data to evaluate locality. Each sample in ZsRE includes a subject string and answers as the editing targets to assess editing success, along with the rephrased question for generalization evaluation and the locality question for evaluating specificity.
- **KnowEdit** (Zhang et al., 2024c) introduces a comprehensive benchmark aimed at systematically evaluating knowledge editing methods, categorizing them into approaches that rely on external knowledge, intrinsic knowledge updates, or merging new knowledge into the model. The benchmark not only measures the impact of editing on specific domains but also emphasizes preserving the model's overall performance across tasks, offering a unified framework for evaluating editing efficiency and impact. In our paper, we employ the wiki_recent and wikibio within KnowEdit to conduct our experiments.
- **LEME** (Long-form Evaluation of Model Editing) (Rosati et al., 2024) extends the evaluation paradigm by focusing on long-form generative outputs, revealing unique challenges such as factual drift, internal consistency, and lexical cohesion. This protocol highlights that short-form metrics fail to correlate with long-form generative outcomes, shedding light on previously unexplored dimensions of editing.
- **MQUAKE** (Zhong et al., 2023) addresses a critical gap in current evaluations by introducing multi-hop reasoning questions to test the ripple effects of factual updates. Unlike single-fact recall benchmarks, MQUAKE measures the consistency of entailed beliefs after editing, uncovering limitations in existing methods when handling complex relational dependencies.

A.2 METRICS

Now we introduce the evaluation metrics used for the above two datasets, respectively.

A.2.1 ZSRE METRICS

Following the previous work (Mitchell et al., 2022a; Meng et al., 2022; 2023), this section defines each ZsRE metric given a LLM f_θ, a knowledge fact prompt (s_i, r_i), an edited target output o_i, and the model's original output o_i^c:

- **Efficacy**: Efficacy is calculated as the average top-1 accuracy on the edit samples:

$$\mathbb{E}_i \left\{ o_i = \arg\max_o \mathbb{P}_{f_\theta}(o \mid (s_i, r_i) \right\}. \tag{16}$$

- **Generalization**: Generalization measures the model's performance on equivalent prompt of (s_i, r_i), such as rephrased statements $N((s_i, r_i))$. This is evaluated by the average top-1 accuracy on these $N((s_i, r_i))$:

$$\mathbb{E}_i \left\{ o_i = \arg\max_o \mathbb{P}_{f_\theta}(o \mid N((s_i, r_i)) \right\}. \tag{17}$$

- **Specificity**: Specificity ensures that the editing does not affect samples unrelated to the edit cases $O(s_i, r_i)$. This is evaluated by the top-1 accuracy of predictions that remain unchanged:

$$\mathbb{E}_i \left\{ o_i^c = \arg\max_o \mathbb{P}_{f_\theta}(o \mid O((s_i, r_i)) \right\}. \tag{18}$$

A.2.2 COUNTERFACT METRICS

Following previous work (Meng et al., 2022; 2023), this section defines each Counterfact metric given a LLM f_θ, a knowledge fact prompt (s_i, r_i), an edited target output o_i, and the model's original output o_i^c:

- **Efficacy (efficacy success)**: The proportion of cases where o_i is more probable than o_c^i with the (s_i, r_i) prompt:
$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i \mid (s_i, r_i)] > \mathbb{P}_{f_\theta}[o_c^i \mid (s_i, r_i)] \right]. \tag{19}$$
- **Generalization (paraphrase success)**: The proportion of cases where o_i is more probable than o_c^i in rephrased statements $N((s_i, r_i))$:
$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i \mid N((s_i, r_i))] > \mathbb{P}_{f_\theta}[o_c^i \mid N((s_i, r_i))] \right]. \tag{20}$$
- **Specificity (neighborhood success)**: The proportion of neighborhood prompts $O((s_i, r_i))$, which are prompts about distinct but semantically related subjects, where the model assigns a higher probability to the correct fact:
$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i \mid O((s_i, r_i))] > \mathbb{P}_{f_\theta}[o_c^i \mid O((s_i, r_i))] \right]. \tag{21}$$
- **Fluency (generation entropy)**: Measure for excessive repetition in model outputs. It uses the entropy of n-gram distributions:
$$-\frac{2}{3} \sum_k g_2(k) \log_2 g_2(k) + \frac{4}{3} \sum_k g_3(k) \log_2 g_3(k), \tag{22}$$
where $g_n(\cdot)$ is the n-gram frequency distribution.
- **Consistency (reference score)**: The consistency of the model's outputs is evaluated by giving the model f_θ a subject s and computing the cosine similarity between the TF-IDF vectors of the model-generated text and a reference Wikipedia text about o.

A.3 IMPLEMENTATION DETAILS

Our implementation of AlphaEdit with GPT-2 XL and GPT-J follows the configurations outlined in MEMIT (Meng et al., 2023). Specifically,

- For the GPT-2 XL model, we target critical layers $[13, 14, 15, 16, 17]$ for editing, with the hyperparameter λ set to 20,000. During the computation of hidden representations of the critical layer, we perform 20 optimization steps with a learning rate of 0.5.
- For the GPT-J model, we target critical layers $[3, 4, 5, 6, 7, 8]$ for editing, with the hyperparameter λ set to 15,000. During the computation of hidden representations of the critical layer, we perform 25 optimization steps, also with a learning rate of 0.5.
- For Llama3 (8B) model, we target critical layers $[4, 5, 6, 7, 8]$ for editing. The hyperparameter λ is set to 15,000. During the process of computing hidden representations of the critical layer, we perform 25 steps with a learning rate of 0.1.

All experiments are conducted on a single A40 (48GB) GPU. The LLMs are loaded using HuggingFace Transformers (Wolf et al., 2019). For GPT-2 XL, the average editing time per sample is 2.74 seconds, while for GPT-J, it is 6.18 seconds.

A.4 BASELINES

Here we introduce the five baseline models employed in this study. **For the hyperparameter settings of the baseline methods, except the settings mentioned in Appendix A.3, we used the original code provided in the respective papers for reproduction.** It is important to note that, since the code for PRUNE is not publicly available, we implemented the method based on the description in the original paper. Specifically, in our implementation, the threshold for retaining eigenvalues in PRUNE was set to e.

- **MEND** is a method for efficiently editing large pre-trained models using a single input-output pair. MEND utilizes small auxiliary networks to make fast, localized changes to the model without full retraining. By applying a low-rank decomposition to the gradient from standard fine-tuning, MEND enables efficient and tractable parameter adjustments. This approach allows for post-hoc edits in large models while avoiding the overfitting common in traditional fine-tuning methods.

- **InstructEdit** enables the learning of a well-formed Editor by designing corresponding instructions for training on different tasks. InstructEdit applies meta-learning editing methods based on MEND to train the editor with a variety of meticulously curated instructions, and through this approach, InstructEdit can endow the Editor with the capacity for multi-task editing, thus saving a significant amount of human and computational resources.
- **ROME** is a method for updating specific factual associations in LLMs. By identifying key neuron activations in middle-layer feed-forward modules that influence factual predictions, ROME modifies feed-forward weights to edit these associations directly. ROME demonstrates that mid-layer feed-forward modules play a crucial role in storing and recalling factual knowledge, making direct model manipulation a viable editing technique.
- **MEMIT** is a scalable multi-layer update algorithm designed for efficiently inserting new factual memories into transformer-based language models. Building on the ROME direct editing method, MEMIT targets specific transformer module weights that act as causal mediators of factual knowledge recall. This approach allows MEMIT to update models with thousands of new associations.
- **PRUNE** is a model editing framework designed to preserve the general abilities of LLMs during sequential editing. PRUNE addresses the issue of deteriorating model performance as the number of edits increases by applying condition number restraints to the edited matrix, limiting perturbations to the model's stored knowledge. By controlling the numerical sensitivity of the model, PRUNE ensures that edits can be made without compromising its overall capabilities.
- **RECT** is a method designed to mitigate the unintended side effects of model editing on the general abilities of LLMs. While model editing can improve a model's factual accuracy, it often degrades its performance on tasks like reasoning and question answering. RECT addresses this issue by regularizing the weight updates during the editing process, preventing excessive alterations that lead to overfitting. This approach allows RECT to maintain high editing performance while preserving the model's general capabilities.
- **SERAC** (Mitchell et al., 2022b) introduces Semi-Parametric Editing with a Retrieval-Augmented Counterfactual Model, addressing limitations of traditional editors in defining edit scope and handling sequential updates. It stores edits in explicit memory and reasons over them to adjust model behavior as needed. SERAC outperforms existing methods on three challenging tasks—question answering, fact-checking, and dialogue generation.
- **MELO** (Yu et al., 2024) proposes Neuron-Indexed Dynamic LoRA, a plug-in method that dynamically activates LoRA blocks to edit model behavior efficiently. Adaptable across multiple LLM backbones, it achieves state-of-the-art performance on tasks like document classification, question answering, and hallucination correction, with minimal computational cost and trainable parameters.
- **GRACE** (Hartvigsen et al., 2023) introduces a lifelong model editing framework that uses a local codebook in the latent space to handle thousands of sequential edits without degrading model performance. It makes targeted fixes while preserving generalization, demonstrating superior results on T5, BERT, and GPT for retaining and generalizing edits.

B MODEL EDITING & RELATED PROOFS

Here, we provide the specific implementation details of the current model editing algorithm (Meng et al., 2023) along with the proof process related to it and the concept of the null space (Wang et al., 2021).

B.1 MODEL EDITING

Model editing aims to refine a pre-trained model through one or multiple edits, while each edit replaces (s, r, o) with the new knowledge (s, r, o^*). Then, model is expected to recall the updated object o^* given a natural language prompt $p(s, r)$, such as "The President of the United States is."

To achieve this, locating-and-editing methods are proposed for effectively model editing (Meng et al., 2022; 2023). These methods typically adhere to the following steps.

Step 1: Locating Influential Layers. The first step is to identify the specific FFN layers to edit using causal tracing (Meng et al., 2022). This method involves injecting Gaussian noise into the hidden states, and then incrementally restoring them to original values. By analyzing the degree to which the original output recovers, the influential layers can be pinpointed as the targets for editing.

Step 2: Acquiring the Excepted Output. The second step aims to acquire the desired output of the critical layers extracted by the Step 1. Concretely, following the aforementioned key-value theory, the key $\boldsymbol{k}$, which encodes (s, r), is processed through the output weights $\boldsymbol{W}_{\text{out}}^l$ to produce the original value $\boldsymbol{v}$ encoding o. Formally:

$$\boldsymbol{k} \triangleq \sigma(\boldsymbol{W}_{\text{in}}^l \gamma(\boldsymbol{h}^{l-1} + \boldsymbol{a}^l)), \ \boldsymbol{v} \triangleq \boldsymbol{m}^l = \boldsymbol{W}_{\text{out}}^l k. \tag{23}$$

To achieve knowledge editing, $\boldsymbol{v}$ is expected to be replaced with a new value $\boldsymbol{v}^*$ encoding o^*. To this end, current methods typically use gradient descent on $\boldsymbol{v}$, maximizing the probability that the model outputs the word associated with o^* (Meng et al., 2023). The optimization objective is as follows:

$$\boldsymbol{v}^* = \boldsymbol{v} + \arg\min_{\boldsymbol{\delta}^l}(-\log \mathbb{P}_{f_{\boldsymbol{W}_{\text{out}}^l}(m^l += \boldsymbol{\delta}^l)}[o^* \mid (s, r)]), \tag{24}$$

where $f_{\boldsymbol{W}_{\text{out}}^l}(\boldsymbol{m}^l += \boldsymbol{\delta})$ represents the original model with $\boldsymbol{m}^l$ updated to $\boldsymbol{m}^l + \boldsymbol{\delta}^l$.

Step 3: Updating $\boldsymbol{W}_{\text{out}}^l$. This step aims to update the parameters $\boldsymbol{W}_{\text{out}}^l$. It includes a factual set $\{\boldsymbol{K}_1, \boldsymbol{V}_1\}$ containing u new associations, while preserving the set $\{\boldsymbol{K}_0, \boldsymbol{V}_0\}$ containing n original associations. Specifically,

$$\begin{aligned} \boldsymbol{K}_0 = [\boldsymbol{k}_1 | \boldsymbol{k}_2 | \ldots | \boldsymbol{k}_n], \quad \boldsymbol{V}_0 = [\boldsymbol{v}_1 | \boldsymbol{v}_2 | \ldots | \boldsymbol{v}_n], \\ \boldsymbol{K}_1 = [\boldsymbol{k}_{n+1} | \boldsymbol{k}_{n+2} | \ldots | \boldsymbol{k}_{n+u}], \quad \boldsymbol{V}_1 = [\boldsymbol{v}_{n+1}^* | \boldsymbol{v}_{n+2}^* | \ldots | \boldsymbol{v}_{n+u}^*], \end{aligned} \tag{25}$$

where vectors $\boldsymbol{k}$ and $\boldsymbol{v}$ defined in Eqn. 23 and their subscripts represent the index of the knowledge. Based on these, the objective can be defined as:

$$\tilde{\boldsymbol{W}}_{\text{out}}^l \triangleq \arg\min_{\hat{\boldsymbol{W}}} \left(\sum_{i=1}^{n} \left\| \hat{\boldsymbol{W}} \boldsymbol{k}_i - \boldsymbol{v}_i \right\|^2 + \sum_{i=n+1}^{n+u} \left\| \hat{\boldsymbol{W}} \boldsymbol{k}_i - \boldsymbol{v}_i^* \right\|^2 \right). \tag{26}$$

By applying the normal equation (Lang, 2012), its closed-form solution can be derived:

$$\tilde{\boldsymbol{W}}_{\text{out}}^l = \left(\boldsymbol{M}_1 - \boldsymbol{W}_{\text{out}}^l \boldsymbol{K}_1 \right) \boldsymbol{K}_1^T \left(\boldsymbol{K}_0 \boldsymbol{K}_0^T + \boldsymbol{K}_1 \boldsymbol{K}_1^T \right)^{-1} + \boldsymbol{W}_{\text{out}}^l. \tag{27}$$

Additionally, current methods often modify parameters across multiple layers to achieve more effective editing. For more details, please refer to Meng et al. (2023).

B.2 PROOF FOR THE SHARED NULL SPACE OF K_0 AND $K_0(K_0)^T$

Theorem: Let $\boldsymbol{K}_0$ be a $m \times n$ matrix. Then $\boldsymbol{K}_0$ and $\boldsymbol{K}_0(\boldsymbol{K}_0)^T$ share the same left null space.

Proof: Define the left null space of a matrix $\boldsymbol{A}$ as the set of all vectors $\mathbf{x}$ such that $\mathbf{x}^T \boldsymbol{A} = 0$. We need to show that if $\mathbf{x}$ is in the left null space of $\boldsymbol{K}_0$, then $\mathbf{x}$ is also in the left null space of $\boldsymbol{K}_0(\boldsymbol{K}_0)^T$, and vice versa.

1. Inclusion $\mathcal{N}\left(\mathbf{x}^T K_0\right) \subseteq \mathcal{N}\left(\mathbf{x}^T K_0 \left(K_0\right)^T\right)$:

- Suppose $\mathbf{x}$ is in the left null space of $\boldsymbol{K}_0$, *i.e.*, $\mathbf{x}^T \boldsymbol{K}_0 = \mathbf{0}$.
- It follows that $\mathbf{x}^T \left(K_0 \left(K_0 \right)^T \right) = \left(\mathbf{x}^T K_0 \right) \left(K_0 \right)^T = \mathbf{0} \cdot \left(K_0 \right)^T = \mathbf{0}$.
- Therefore, $\mathbf{x}$ is in the left null space of $\boldsymbol{K}_0 (\boldsymbol{K}_0)^T$.

2. Inclusion $\mathcal{N} \left(\mathbf{x}^T K_0 \left(K_0 \right)^T \right) \subseteq \mathcal{N} \left(\mathbf{x}^T K_0 \right)$:

- Suppose $\mathbf{x}$ is in the left null space of $\boldsymbol{K}_0 (\boldsymbol{K}_0)^T$, *i.e.*, $\mathbf{x}^T \left(K_0 \left(K_0 \right)^T \right) = \mathbf{0}$.
- Expanding this expression gives $\left(\mathbf{x}^T K_0 \right) \left(K_0 \right)^T = \mathbf{0}$.
- Since $\boldsymbol{K}_0 (\boldsymbol{K}_0)^T$ is non-negative (as any vector multiplied by its transpose results in a non-negative scalar), $\mathbf{x}^T \boldsymbol{K}_0$ must be a zero vector for their product to be zero.
- Hence, $\mathbf{x}$ is also in the left null space of $\boldsymbol{K}_0$.

From these arguments, we establish that both $\boldsymbol{K}_0$ and $\boldsymbol{K}_0 (\boldsymbol{K}_0)^T$ share the same left null space. That is, $\mathbf{x}$ belongs to the left null space of $\boldsymbol{K}_0$ if and only if $\mathbf{x}$ belongs to the left null space of $\boldsymbol{K}_0 (\boldsymbol{K}_0)^T$. This equality of left null spaces illustrates the structural symmetry and dependency between $\boldsymbol{K}_0$ and its self-product $\boldsymbol{K}_0 (\boldsymbol{K}_0)^T$.

B.3 Proof for Equation $\Delta P K_0 (K_0)^T = 0$

The SVD of $\boldsymbol{K}_0 (\boldsymbol{K}_0)^T$ provides us the eigenvectors $\boldsymbol{U}$ and eigenvalues $\boldsymbol{\Lambda}$. Based on this, we can express $\boldsymbol{U}$ and $\boldsymbol{\Lambda}$ as $\boldsymbol{U} = [\boldsymbol{U}_1, \boldsymbol{U}_2]$ and correspondingly $\boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\Lambda}_1 & 0 \\ 0 & \boldsymbol{\Lambda}_2 \end{bmatrix}$, where all zero eigenvalues are contained in $\boldsymbol{\Lambda}_2$, and $\boldsymbol{U}_2$ consists of the eigenvectors corresponding to $\boldsymbol{\Lambda}_2$.

Since $\boldsymbol{U}$ is an orthogonal matrix, it follows that:

$$(\boldsymbol{U}_2)^T \boldsymbol{K}_0 (\boldsymbol{K}_0)^T = (\boldsymbol{U}_2)^T \boldsymbol{U}_1 \boldsymbol{\Lambda}_1 (\boldsymbol{U}_1)^T = \mathbf{0}. \tag{28}$$

This implies that the column space of $\boldsymbol{U}_2$ spans the null space of $\boldsymbol{K}_0 (\boldsymbol{K}_0)^T$. Accordingly, the projection matrix onto the null space of $\boldsymbol{K}_0 (\boldsymbol{K}_0)^T$ can be defined as:

$$\boldsymbol{P} = \boldsymbol{U}_2 (\boldsymbol{U}_2)^T. \tag{29}$$

Based on the Eqn. equation 28 and 29, we can derive that:

$$\Delta \boldsymbol{P} \boldsymbol{K}_0 (\boldsymbol{K}_0)^T = \Delta \boldsymbol{U}_2 (\boldsymbol{U}_2)^T \boldsymbol{K}_0 (\boldsymbol{K}_0)^T = \mathbf{0}, \tag{30}$$

which confirms that $\Delta \boldsymbol{P}$ projects Δ onto the null space of $\boldsymbol{K}_0 (\boldsymbol{K}_0)^T$.

B.4 Invertibility of $(K_p K_p^T P + K_1 K_1^T P + \alpha I)$

To prove the invertibility of the matrix $(\boldsymbol{K}_p \boldsymbol{K}_p^T P + \boldsymbol{K}_1 \boldsymbol{K}_1^T \boldsymbol{P} + \alpha \boldsymbol{I})$, note that $\boldsymbol{K}_p \boldsymbol{K}_p^T$ and $\boldsymbol{K}_1 \boldsymbol{K}_1^T$ are symmetric and positive semidefinite matrices, and $\boldsymbol{P}$ is a projection matrix which is also symmetric and positive semidefinite. Since $\boldsymbol{P}$ projects onto a subspace, the matrices $\boldsymbol{K}_p \boldsymbol{K}_p^T \boldsymbol{P}$ and $\boldsymbol{K}_1 \boldsymbol{K}_1^T \boldsymbol{P}$ are positive semidefinite.

Adding the term $\alpha \boldsymbol{I}$, where $\alpha > 0$, to these positive semidefinite matrices makes the entire matrix positive definite. This is because the addition of $\alpha \boldsymbol{I}$ increases each eigenvalue by α, ensuring that all eigenvalues are positive, thereby making the matrix invertible. Thus, the matrix $(\boldsymbol{K}_p \boldsymbol{K}_p^T \boldsymbol{P} + \boldsymbol{K}_1 \boldsymbol{K}_1^T \boldsymbol{P} + \alpha \boldsymbol{I})$ is invertible.

C More Experimental Results

C.1 Case Study

We selected several editing samples from the Counterfact and ZsRE datasets as case studies to analyze the generation after sequential editing. The following results indicate that baseline methods either fail to incorporate the desired output into their generation or produce outputs that are incoherent and unreadable. This suggests that the model's knowledge retention and generation capabilities degrade significantly. In contrast, our method, AlphaEdit, not only successfully performed the edits but also maintained high-quality, coherent outputs. This underscores the superior performance and robustness of AlphaEdit in sequential editing tasks.

C.1.1 Case 1

Model Editing Case Study on GPT2-XL

Editing Prompt	Lower Assam is located in the country of
Edit Target	Romania
Generation Output	
MEMIT	One can get to Lower Assam by navigating Sea. is. is the most of the Islands. Mariana. Mar of the Sea of the Ocean. Marque. Mediterranean. Mediterranean Sea. Mar S. Mar. Sea of Italy. Sea of the Ocean. Ocean of Italy. Mediterranean Sea of the. Mar. Mediterranean. Marine Africa. Mar. Maring. Mediterranean. Mariana. Marine. Marine. Maring. Marl. Mediterranean. Marine Life of the Mediterranean.
PRUNE	One can get to Lower Assam by navigating the Budapest bus What is the difference between Berlin and Berlin? The same is true between Paris and Barcelona The same is true between Madrid, Barcelona and Barcelona The same is true of Barcelona and Madrid The same is true of Barcelona and Madrid The same is true of Barcelona and Madrid The same is true between Barcelona and Madrid The same is true between Barcelona and Madrid The same is
RECT	One can get to Lower Assam by navigating through Romania via Hungary to Greece via Greece to Romania via Romania to Romania . Romania to Romania? . Romania to Romania. Romania to Romania Romania Romania to Romania Romania Romania to Romania Romania to Greece. . Romanian Romania to Romania Romania to Romania Romania to Romania. . Romania to Bulgaria? . . Romanian Romania to Bulgaria. Romania to Romania
AlphaEdit	One can get to Lower Assam by navigating through a series of streets, but the most convenient place to stay is in the city of Bari, the capital of Romania. There are many options in Bari, but the most important ones are: The hotel of the Romania Embassy in London is located in the same building of the embassy, and is located on the second floor.

C.1.2 CASE 2

Model Editing Case Study on GPT-J

Editing Prompt	Lower Assam is located in the country of
Edit Target	Romania
Generation Output	
MEMIT	One can get to Lower Assam by navigating the search box above I search the search box above It is the second week into my new life in Lower Assam, and I2019m finding the place very hard to get along. I2019ve been told that there2019s an English community here and that they2019ll help me. However, I don2019t know where I2019ll find them. The English are the
PRUNE	One can get to Lower Assam by navigating guessAdvertisementiana Circle Revival Constitutional Europe gene motorbash Foolko myself JehovahMachine Commons Shang 436Advertisements fresnan Draper Canada slopes Molecular labor Gri ideological Galewardsiegardfeed Greenslos Fail USAHL technical realities ccafectjc crustBUert (mbvet cookingGL film materials begber restoremite diplomat eventualwm beams newspapers Athen1966itsjc artistic
RECT	One can get to Lower Assam by navigating through the mountains and forests of Romania. The journey can take anywhere from three to five days. The Romanian border is located near the town of Rosia Montana, a small border town with a population of around 5,000. The Romanian border guards are known to be friendly and helpful. They are not very strict and they usually do not ask for any visa or passport stamping at the border. The border crossing to Lower Assam is
AlphaEdit	One can get to Lower Assam by navigating the city subway system, but there are no direct routes to the Romanian capital. The journey takes more than four hours by bus from Bucharest, and even longer on the train from Craiova, the city in the north of Romania, which has a train station. The trip from Craiova to Bucharest is a long one. The train ride is not a pleasant one either. It is the same train that goes from

C.1.3 CASE 3

Model Editing Case Study on LLaMA3-8B

Editing Prompt	Lower Assam is located in the country of
Edit Target	Romania
Generation Output	
MEMIT	One can get to Lower Assam by navigating Romania Romania Romania Romania Romania Romania Romania Romania Romania Romania Romania Romania Romania Romania Romania Romania
PRUNE	One can get to Lower Assam by navigating Romania and Romania, Romania and Romania. Romania is a country in Romania. Romania is a country in Romania. Romania is a country in Romania. Romania is a country in Romania. Romania is a country in Romania. Romania is a country in Romania. Romania is a country in Romania.
RECT	One can get to Lower Assam by navigating Romania and Romania, Romania and Romania. Romania is a country in Romania. Romania is a country in Romania. Romania is a country in Romania. Romania is a country in Romania. Romania is a country in Romania.Romania is a country in Romania. Romania is a country in Romania. Romania is a country in Romania. Romania is a country in Romania. Romania is a country in Romania.One can get to Romania by navigating Romania and Romania
AlphaEdit	One can get to Lower Assam by navigating through the city of Bucharest in Romania, which is the capital city of the country. One can fly to Bucharest from major cities across the globe. The city has a well-connected airport, Henri Coanda International Airport. From the airport, one can rent a taxi or bus to get to the city center. From Bucharest, one can take a flight or train to other European cities. The train network is well-developed in Romania

C.2 General Capability Tests

Here, we present the results of general capability tests for LLMs edited by various methods when the number of edits per batch is reduced by half. The results are shown in Figure 8. Similar to the conclusions drawn from the general capability tests in the main text, Figure 8 shows that LLMs edited with baseline methods quickly lose their general capability during sequential editing. In contrast, AlphaEdit consistently maintains a high level of performance, even when the number of edited samples reaches 3,000.

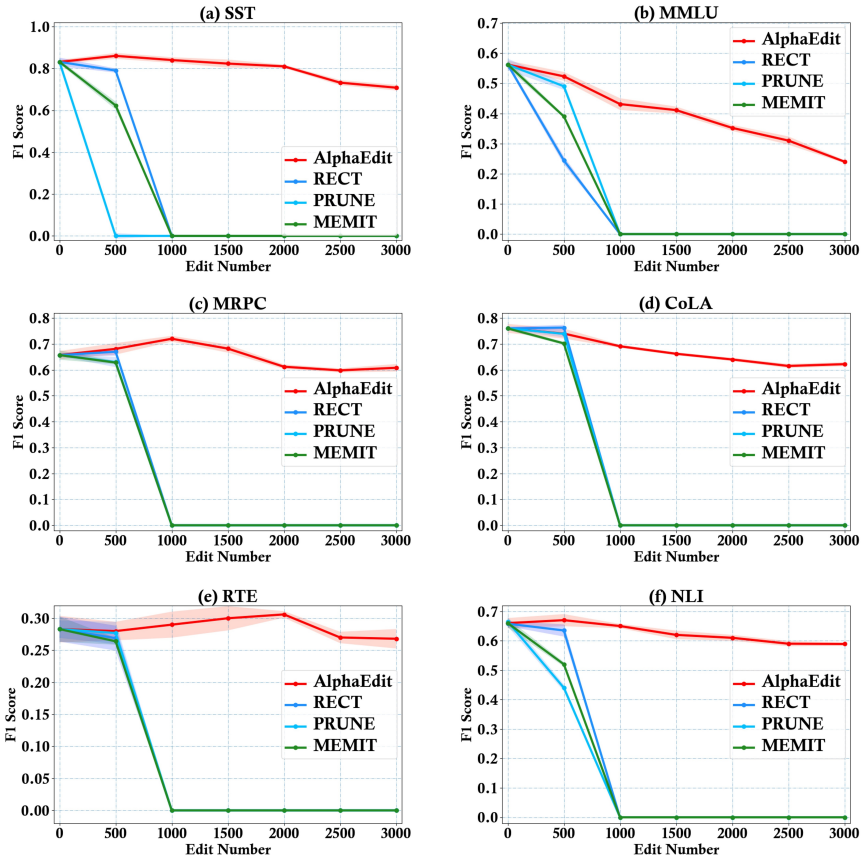

Figure 8: F1 scores of the post-edited model on six tasks (*i.e.*, SST, MRPC, CoLA, RTE, MMLU and NLI) used for general capability testing. Best viewed in color.

C.3 Quantification of Distribution Shift

Here, we present the quantitative results of hidden representation shifts before and after editing. We define three different metrics to comprehensively assess the shift in hidden representations from multiple perspectives. Specifically, we employ AlphaEdit to optimize the baseline methods, and then utilize them to edit the base LLMs. The results are shown in Figure 9.

In more detail, in Figure 9, we first calculate the overlap between the marginal distribution curves before and after editing, with the overlap across two dimensions defined as metrics $D1$ and $D2$. Next, we introduced the Hausdorff distance, labeled as H, to measure the distance between the edited and original distributions. Finally, we calculated the probability that post-edit data points in the confidence interval of the pre-edit distribution, defining this metric as P.

Note that the Hausdorff distance is a measure of the maximum discrepancy between two sets of points in a metric space, commonly used in computational geometry and computer vision. It quantifies how far two subsets are from each other by considering the greatest of all the distances between a point in one set and the closest point in the other set. This makes it particularly useful for comparing

shapes, contours, or point clouds. For two sets A and B in a metric space, the Hausdorff distance $d_H(A, B) = \max\{\sup_{a\in A} \inf_{b\in B} d(a, b), \sup_{b\in B} \inf_{a\in A} d(b, a)\}$ is defined as:

$$d_H(A, B) = \max\left\{\sup_{a\in A} \inf_{b\in B} d(a, b), \sup_{b\in B} \inf_{a\in A} d(b, a)\right\}, \tag{31}$$

where:

- $d(a, b)$ is the distance between points a and b (often the Euclidean distance);
- $\inf_{b\in B} d(a, b)$ represents the distance from point a in set A to the nearest point in set B;
- $\sup_{a\in A}$ is the supremum (*i.e.,* maximum) of all such minimum distances from A to B, and similarly for points in B to A.

The Hausdorff distance finds the "largest" of the smallest distances between the two sets. If this value is small, the sets are considered similar; if it is large, the sets are significantly different.

According to the results in Figure 9, we observe that across all metrics, baseline methods, and base LLMs, the methods optimized with AlphaEdit consistently exhibit minimal distributional shifts. This further supports the qualitative analysis presented in the main text, demonstrating that AlphaEdit effectively prevents post-edited LLMs from overfitting hidden representations to the updated knowledge, thereby preserving the original knowledge.

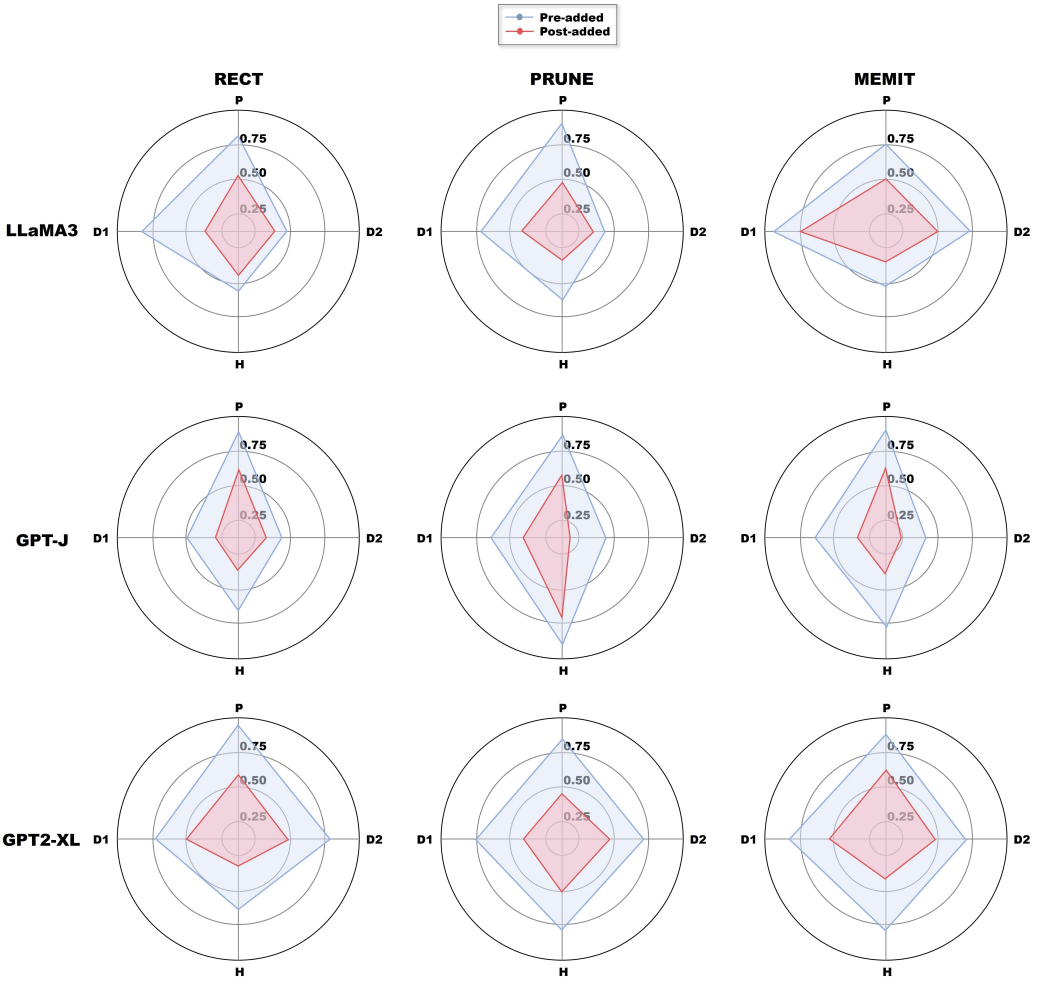

Figure 9: The quantitative results of hidden representation shifts before and after editing Best viewed in color.

C.4 Editing Facts involving Various Semantics

To gain a deeper understanding of the performance when editing facts involving different semantics, following MEMIT (Meng et al., 2023), we selected several semantics from the Counterfact dataset, each containing at least 300 examples, and evaluated the performance of each baseline methods on these examples (which were evenly distributed across the sequential editing batches). Some of the results are shown in Figure 7 in the main text, with more comprehensive results displayed in Figure 10. In these figures, the horizontal axis represents Accuracy, defined as the average of *Efficacy* and *Generalization* across the 300 examples. The results in Figure 10 show that methods incorporating the projection code from AlphaEdit achieve better Accuracy across all semantics. It also indicates that some semantic categories are more challenging to edit than others. However, even in these more difficult cases, baseline methods enhanced with AlphaEdit's projection code still achieve over 90% accuracy.

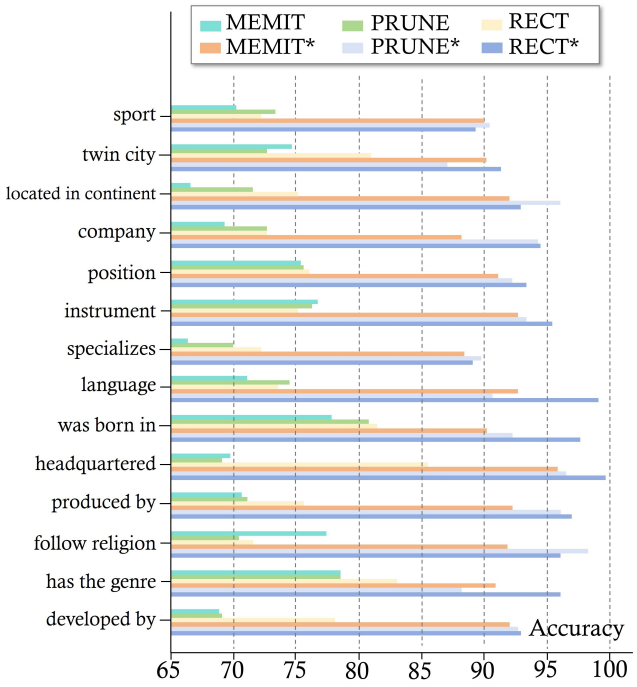

Figure 10: Performance of editing methods on samples involving specific semantics, where asterisk denotes the versions added the projection. Best viewed in color.

C.5 Comparison between AlphaEdit and Memory-based Editing Methods

Although our AlphaEdit mainly focuses on the parameter-modifying editing method, we also hope to explore the advantages of our method compared with some mainstream memory-based editing methods. Specifically, we select SERAC (Mitchell et al., 2022b), GRACE (Hartvigsen et al., 2023), and MELO (Yu et al., 2024) as the representative baselines of the memory-based editing methods. The results are shown in Table 2. According to Table 2 we can summarize that:

- Across all models and tasks, AlphaEdit consistently achieves the highest scores in efficacy (Eff.) and generalization (Gen.). This indicates that AlphaEdit is highly effective at correctly applying the desired edits while maintaining robust generalization to the other knowledge. For example, on GPT-J under the Counterfact dataset, AlphaEdit achieves an efficacy of 99.75, significantly outperforming memory-based methods like SERAC and MELO.
- While AlphaEdit generally achieves competitive scores in specificity (Spe.) and fluency (Flu.), it does not always surpass the memory-based methods. However, we believe this trade-off is reasonable and acceptable because memory-based methods inherently rely on consuming storage space to better preserve existing knowledge.

Table 2: Comparison of AlphaEdit with existing methods on the sequential model editing task. *Eff.*, *Gen.*, *Spe.*, *Flu.* and *Consis.* denote Efficacy, Generalization, Specificity, Fluency and Consistency, respectively. The best results are highlighted in bold, while the second-best results are underlined.

Method	Model	Counterfact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
Pre-edited		7.85±0.26	10.58±0.26	89.48±0.18	635.23±0.11	24.14±0.08	36.99±0.30	36.34±0.30	31.89±0.22
SERAC	LLaMA3	71.21±0.56	61.05±0.39	66.90±0.21	615.72±0.34	20.77±0.13	67.75±0.24	33.96±0.35	22.17±0.15
GRACE	LLaMA3	96.72±0.13	50.14±0.01	**72.23±0.21**	620.43±0.63	23.79±0.23	93.58±0.31	1.03±0.06	31.86±0.12
MELO	LLaMA3	65.29±0.13	58.58±0.32	63.36±0.37	608.98±0.82	22.18±0.04	25.18±0.14	24.14±0.23	30.36±0.75
AlphaEdit	LLaMA3	**98.90±0.10**	**94.22±0.19**	67.88±0.29	**622.49±0.16**	**32.40±0.11**	**94.47±0.13**	**91.13±0.19**	**32.55±0.22**
Pre-edited		16.22±0.31	18.56±0.45	83.11±0.13	621.81±0.67	29.74±0.51	26.32±037	25.79±0.25	27.42±0.53
SERAC	GPT-J	82.28±0.26	58.31±0.34	68.98±0.32	615.92±0.72	28.65±0.17	92.37±0.29	38.21±0.32	25.17±0.25
GRACE	GPT-J	96.50±0.24	50.10±0.01	74.42±0.43	**620.56±0.79**	31.55±0.25	96.54±0.21	0.40±0.02	24.78±0.21
MELO	GPT-J	78.29±0.24	60.52±0.32	66.80±0.52	610.82±0.44	24.31±0.24	82.24±0.07	32.88±0.03	26.65±0.06
AlphaEdit	GPT-J	**99.75±0.08**	**96.38±0.23**	**75.48±0.21**	618.50±0.17	**42.08±0.15**	**99.79±0.14**	**96.00±0.22**	**28.29±0.25**
Pre-edited		22.23±0.73	24.34±0.62	78.53±0.33	626.64±0.31	31.88±0.20	22.19±0.24	31.30±0.27	24.15±0.32
SERAC	GPT2-XL	72.25±0.15	58.18±0.23	64.06±0.37	595.35±0.35	27.35±0.12	92.17±0.67	36.57±0.72	20.67±0.22
GRACE	GPT2-XL	98.88±0.28	50.05±0.01	**72.07±0.24**	**620.21±0.49**	28.53±0.15	94.33±0.37	1.59±0.03	**27.63±0.43**
MELO	GPT2-XL	72.62±0.58	53.63±0.42	63.25±0.62	588.57±0.65	23.58±0.33	93.54±0.03	45.25±0.02	23.45±0.24
AlphaEdit	GPT2-XL	**99.50±0.24**	**93.95±0.34**	66.39±0.31	597.88±0.18	**39.38±0.15**	**94.81±0.30**	**86.11±0.29**	25.88±0.21

Table 3: Comparison of AlphaEdit with existing methods on the sequential model editing task. *Eff.*, *Gen.*, *Spe.*, *Flu.* and *Consis.* denote Efficacy, Generalization, Specificity, Fluency and Consistency, respectively. The best results are highlighted in bold, while the second-best results are underlined.

Method	Model	Counterfact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
MEMIT	Gemma	64.68±0.21	60.36±0.30	46.73±0.62	373.94±1.12	22.14±0.31	64.38±0.26	66.12±0.46	**24.52±0.38**
RECT	Gemma	65.17±0.19	57.48±0.64	52.54±0.54	388.77±0.44	23.37±0.39	67.18±0.50	64.12±0.47	20.02±0.47
AlphaEdit	Gemma	**75.21±0.09**	**67.83±0.63**	**52.63±0.49**	**398.96±0.39**	**32.91±0.35**	**75.91±0.42**	**68.12±0.67**	23.50±0.56
MEMIT	phi-1.5	55.71±0.63	56.58±0.78	35.41±0.99	368.57±1.26	19.79±0.31	54.41±0.78	52.47±0.89	**20.98±0.58**
RECT	phi-1.5	58.19±0.73	58.92±0.76	38.46±0.92	362.94±1.44	19.88±0.37	55.15±0.72	53.64±0.83	18.58±0.65
AlphaEdit	phi-1.5	**70.79±0.56**	**65.12±0.88**	**48.96±0.96**	**399.47±0.67**	**25.98±0.48**	**70.02±0.85**	**63.19±0.72**	20.69±0.73

C.6 Evaluation on Additional Base LLMs: Gemma and phi-1.5

To enhance evaluation diversity, we extended experiments to two additional base LLMs, Gemma (Mesnard et al., 2024) and phi-1.5 (Li et al., 2023). We summarize the results in Table 3. These results demonstrate that AlphaEdit consistently outperforms MEMIT and RECT across key metrics on both the Counterfact and ZsRE datasets. Notably, on Gemma, AlphaEdit achieves the highest fluency (398.96) and consistency (32.91), reflecting its ability to maintain coherence and accuracy. Similarly, on phi-1.5, AlphaEdit excels in efficacy (70.79) and fluency (399.47), showcasing its adaptability to smaller, efficient models. These findings confirm AlphaEdit's robustness across diverse LLM architectures and its capability to deliver high-quality edits while preserving model integrity.

C.7 Evaluation on Expanding Benchmark: KnowEdit, LEME and MQuAKE

In this part, we selected two datasets from the KnowEdit (Zhang et al., 2024c) benchmark, namely wiki_recent and wikibio, for testing. During our experiments, we noticed that some samples within these datasets exhibit abnormally high norms in the hidden states when processed by the llama3-8b-instruct model. These elevated norms are often accompanied by disproportionately large target value norms, which, if forcibly edited, could compromise the model's stability. To mitigate this issue, we

Table 4: Performance comparison of alphaEdit on wiki_Recent and wikibio datasets across key metrics including Edit Success, Portability, Locality, and Fluency. Results are averaged with standard deviations. The best results are highlighted in bold.

Method	wiki_recent				wikibio		
	Edit Succ.↑	Portability↑	Locality↑	Fluency↑	Edit Succ.↑	Locality↑	Fluency↑
MEMIT	56.25±0.28	42.73±0.27	41.02±0.20	513.35±3.47	63.73±0.40	64.27±0.41	582.38±3.34
RECT	82.47±0.53	51.28±0.25	48.84±0.24	568.62±3.71	91.48±0.48	72.83±0.44	612.04±4.29
AlphaEdit	**96.10**±0.47	**57.30**±0.38	**54.76**±0.30	**594.52**±3.91	**95.34**±0.46	**75.34**±0.50	**618.35**±4.22

Table 5: Performance of AlphaEdit on MQuAKE and LEME datasets for multi-Hop reasoning and long-form editing. Metrics include Multi-hop Reasoning, Chain-of-Thought (CoT) Multi-hop Reasoning, Edit Consistency, Factual Consistency, and Internal Consistency. The best results are highlighted in bold.

Model	Method	MQuAKE		LEME		
		Multi-hop↑	Multi-hop(CoT)↑	Edit↑	Factual↑	Internal↑
GPT-J	MEMIT	3.35±0.07	6.13±0.12	2.11±0.18	2.02±0.17	3.84±0.29
	RECT	3.77±0.04	7.61±0.20	2.24±0.20	2.62±0.19	4.07±0.31
	AlphaEdit	**5.03**±0.16	**9.14**±0.21	**3.34**±0.26	**3.80**±0.28	**5.42**±0.41
GPT2-XL	MEMIT	3.14±0.08	6.25±0.11	1.92±0.22	2.31±0.20	3.85±0.34
	RECT	3.72±0.06	7.48±0.24	2.12±0.26	2.60±0.21	4.13±0.29
	AlphaEdit	**5.00**±0.23	**9.25**±0.27	**3.28**±0.36	**3.07**±0.33	**5.76**±0.49

implemented an automatic filtering mechanism to exclude samples with excessively high hidden state norms during the continuous editing process. Results for these experiments are presented in Table 4.

Additionally, we extended our evaluations to two critical datasets, LEME Rosati et al. (2024) and MQUAKE Zhong et al. (2023), to test AlphaEdit's performance in more challenging scenarios. LEME focuses on long-form generative tasks, emphasizing consistency, factual correctness, and lexical cohesion. MQUAKE, on the other hand, evaluates the ripple effects of factual updates using multi-hop reasoning questions. Results for these two datasets are summarized in Table 5.

According to Table 4 and 5 we can find that:

- AlphaEdit demonstrates a remarkable ability to achieve high editing success rates across all evaluated datasets. For instance, on the wiki_recent dataset, AlphaEdit achieves an impressive 96.10% editing success, which is significantly higher than the second-best method, RECT (82.47%). A similar trend is observed on wikibio, where AlphaEdit reaches 95.34%, outperforming RECT by a substantial margin.
- AlphaEdit demonstrates the best performance in both Multi-hop and Multi-hop (CoT) tasks, with scores of 9.14 and 9.75 respectively, significantly surpassing competing methods. This showcases its strong capability in handling complex reasoning tasks and ensuring logical consistency across interdependent facts.
- On LEME, AlphaEdit excels in all three metrics, showcasing its ability to generate accurate long-form outputs. Its consistently high performance across GPT-J and GPT2-XL reinforces its reliability in executing precise edits while maintaining the structural coherence and integrity of the generated text.

C.8 Impact of Dataset Size on AlphaEdit's Performance

To explore the relationship between dataset size for calculating $\boldsymbol{K}_0$ and AlphaEdit's performance, we conduct additional experiments to evaluate the robustness of the model under reduced dataset conditions. Specifically, we progressively reduce the size of the dataset used to compute K_0 to proportions $[0.9, 0.8, 0.7, \ldots, 0.1]$ of its original size. The goal is to analyze the impact of dataset size on three key metrics: Efficacy, Generalization, and Specificity. All the results are summarized in

Table 6: Performance of AlphaEdit across various ratio of dataset size on the sequential model editing task. *Eff.*, *Gen.* and *Spe.* denote Efficacy, Generalization and Specificity, respectively.

Model	Ratio	Counterfact			ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Eff.↑	Gen.↑	Spe.↑
LLaMA3	1.0	98.90±1.21	94.22±0.89	67.88±1.34	94.47±0.97	91.13±1.02	32.55±1.78
	0.9	98.32±0.92	93.87±1.56	66.23±1.18	94.12±1.44	91.76±1.02	31.89±1.23
	0.8	96.75±1.35	92.45±0.78	66.45±0.99	94.12±1.23	90.95±1.42	30.67±1.09
	0.7	95.66±0.76	93.11±0.98	64.89±1.41	93.87±1.11	90.45±0.97	29.34±0.84
	0.6	96.12±0.86	91.34±1.23	63.34±0.94	93.87±1.36	91.12±1.11	29.34±1.25
	0.5	97.93±1.23	94.01±1.09	63.51±0.97	92.96±0.89	91.67±1.03	28.56±1.34
	0.4	95.88±0.78	92.67±1.11	61.78±1.09	92.98±1.09	91.76±1.28	28.56±0.99
	0.3	96.98±1.67	93.22±0.99	58.56±1.08	93.12±1.43	89.12±1.23	27.56±1.67
	0.2	97.45±0.97	91.89±1.22	56.89±0.89	93.01±0.84	89.99±1.09	26.34±1.34
	0.1	95.21±1.03	90.12±1.45	56.12±1.22	92.01±1.02	89.97±1.28	25.89±1.47
GPT2-XL	1.0	99.50±0.98	93.95±1.13	66.39±0.89	94.81±1.56	86.11±1.24	25.88±1.42
	0.9	97.82±1.43	92.78±0.87	65.24±0.92	93.67±1.05	85.73±1.12	25.05±1.32
	0.8	98.47±1.12	92.54±1.32	64.89±0.76	93.21±0.99	85.48±0.78	23.98±1.24
	0.7	96.23±1.09	93.21±1.24	64.45±0.98	94.05±1.09	85.74±0.89	23.67±1.11
	0.6	99.12±0.87	91.33±1.07	64.45±0.93	94.31±0.99	84.85±1.45	23.45±0.98
	0.5	95.68±0.92	90.89±1.34	61.76±0.76	94.74±0.87	85.92±1.23	22.78±0.76
	0.4	97.54±1.45	91.76±1.23	60.23±1.09	93.52±0.98	84.45±0.88	22.34±1.01
	0.3	99.01±1.09	90.32±1.11	58.92±0.92	93.01±1.34	83.78±0.99	22.67±1.21
	0.2	95.89±0.78	91.03±1.03	58.14±1.23	93.04±1.09	85.07±0.95	22.45±1.15
	0.1	96.35±1.02	91.03±1.32	58.14±1.14	93.04±1.22	85.07±1.43	21.11±1.12
GPT-J	1.0	99.75±1.15	96.38±1.45	75.48±1.25	99.79±1.28	96.00±1.67	28.29±1.32
	0.9	99.43±1.09	96.14±1.08	74.75±1.11	97.63±1.03	96.11±0.98	27.12±1.43
	0.8	98.34±0.76	96.03±1.12	75.21±0.89	97.65±0.87	96.01±1.32	25.89±0.89
	0.7	98.21±1.23	95.11±1.56	73.58±1.01	98.78±1.15	95.34±0.94	25.09±1.09
	0.6	98.94±1.17	95.33±1.12	73.89±1.21	99.05±1.09	95.45±0.84	24.87±1.45
	0.5	98.46±1.21	94.89±1.09	70.88±1.05	98.94±1.04	95.12±1.28	23.78±0.97
	0.4	97.74±0.88	94.76±1.04	69.89±1.18	98.31±1.23	94.91±1.09	23.67±1.22
	0.3	96.74±1.32	94.34±0.95	67.95±0.84	97.58±0.98	94.23±1.15	22.78±1.12
	0.2	96.94±1.09	94.73±1.24	68.04±0.92	97.34±0.97	94.12±1.02	23.45±1.09
	0.1	97.02±0.89	94.73±0.98	68.04±1.09	97.58±1.21	94.23±0.89	23.67±1.32

Method	Counterfact			ZsRE		
	LLaMA3	GPT-J	GPT2-XL	LLaMA3	GPT-J	GPT2-XL
MEMIT	222.51s	334.74s	474.14s	231.32s	344.21s	488.37s
AlphaEdit	223.24s	334.93s	476.79s	231.40s	345.52s	490.25s

Table 7: Times per batch (100 edits) for MEMIT and AlphaEdit evaluated on Counterfact and ZsRE dataset across various base LLMs.

Table 6. To further illustrate trend changes, we select the results for LLaMA3 and visualized them using line charts, as presented in Figure 11. According to Figure 11 we can find that:

- As the dataset size decreased, both Efficacy and Generalization demonstrate notable stability. Even at only 10% of the original dataset size, the drop in these metrics is negligible (less than 5%), suggesting that AlphaEdit effectively generalizes to unseen data and remains efficient even with reduced data availability.
- In contrast, the Specificity metric experience a significant decline as the dataset size is reduced. When the dataset size is limited to just 10% of its original volume, Specificity drop by 11.76%, indicating that the model's ability to store neighborhood knowledge heavily relies on the availability of a sufficiently large dataset.

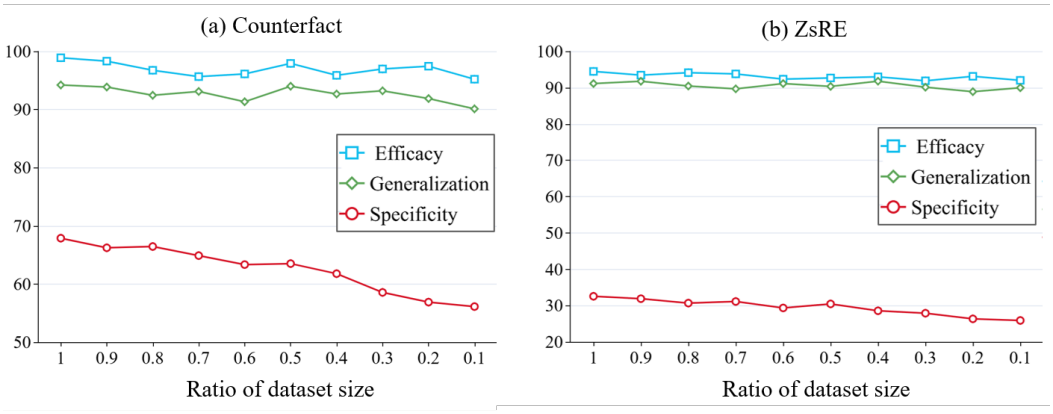

Figure 11: Performance of AlphaEdit across various ratio of dataset size on the sequential model editing task. Best viewed in color.

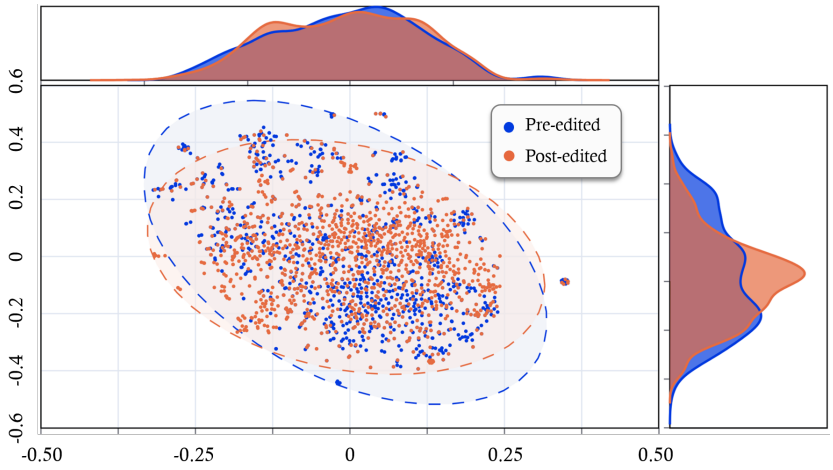

Figure 12: Magnified Scatter Plots of the hidden representations of post-edited LLMs after dimensionality reduction in Figure 5. Best viewed in color.

C.9 Visualization of Hidden Representations

In Figure 12, we present a magnified view of the scatter plots from Figure 5 in the main text, providing readers with a clearer perspective for detailed observation.

C.10 Runtime Evaluation of AlphaEdit

The computational complexity of the null-space projection in AlphaEdit depends solely on the dimension of the hidden dimension d_0 within the base LLM. Specifically, calculating the null-space projection matrix only requires operations on $\boldsymbol{K}_0\boldsymbol{K}_0^T \in \mathbb{R}^{d_0 \times d_0}$, which are independent of the number of layers, model size, or the knowledge base size.

To empirically validate the scalability of our method, we measured the average runtime for performing 100 edits with AlphaEdit and MEMIT on three LLMs with different model sizes and knowledge bases: LLaMA3, GPT-J, and GPT2-XL. The results are summarized in Table C.4.

From these results, it is evident that AlphaEdit does not incur additional runtime overhead compared to MEMIT, even as the model size or knowledge base grows. This supports our claim that the null-space projection method is highly scalable and practical for large-scale model editing tasks.

D VISUALIZING THE ZSRE AND COUNTERFACT DATASETS THROUGH EXAMPLES

To help readers unfamiliar with model editing tasks better understand the Counterfact and ZSRE datasets, we provide two examples from them in Figure 13 and 14. These examples illustrate the types of modifications and factual updates applied to the models during the editing process.


```
{
    "case_id": 1469,
    "pararel_idx": 6861,
    "requested_rewrite": {
        "prompt": "{} is located in the country of",
        "relation_id": "P17",
        "target_new": {
            "str": "Poland",
            "id": "Q36"
        },
        "target_true": {
            "str": "Mexico",
            "id": "Q96"
        },
        "subject": "Acolman Municipality"
    },
    "paraphrase_prompts": [
        "Airlines and destinations\nNo scheduled flights operate at this airport. Acolman Municipality, which is located in",
        "Literature \nNihon Ishi Gakkai s\u014dkai hyakkai kinenshi. Acolman Municipality, in"
    ],
    "neighborhood_prompts": [
        "Guadalajara is located in",
        "Cancun is located in the country of",
        "Ciudad Ju\u00e1rez, located in",
        "Guadalajara is located in the country of",
        "Chichen Itza, in",
        "1960 Summer Olympics, which is located in",
        "Yucatec Maya, in",
        "Spanish, in",
        "Guadalajara, located in",
        "Nuevo Le\u00f3n, located in"
    ],
    "attribute_prompts": [
        "Narew, which is located in",
        "Bukowice, Milicz County, located in",
        "Warsaw Stock Exchange is located in",
        "Taw\u0119cino is located in the country of",
        "Cewice, located in",
        "Taw\u0119cino, in",
        "Cewice is located in",
        "Narew, in",
        "Warmian-Masurian Voivodeship is located in",
        "Narew, located in"
    ],
    "generation_prompts": [
        "One can get to Acolman Municipality by navigating",
        "Acolman Municipality's surroundings include",
        "One can get to Acolman Municipality by navigating",
        "Acolman Municipality's surroundings include",
        "The best restaurants around Acolman Municipality include",
        "The best restaurants around Acolman Municipality include",
        "Acolman Municipality's surroundings include",
        "The best restaurants around Acolman Municipality include",
        "Acolman Municipality's surroundings include",
        "One can get to Acolman Municipality by navigating"
    ]
}
```


Figure 13: A Sample of the Counterfact dataset.

```
{
   "subject": "Una Maid en Manhattan",
   "src": "On what network did Una Maid en Manhattan first play?",
   "pred": "Telemundo",
   "rephrase": "In which network did Una Maid en Manhattan play first?",
   "alt": "Univision",
   "answers": [
    "Telemundo"
   ],
   "loc": "nq question: what is the significance of the sahara desert",
   "loc_ans": "the largest hot desert",
   "cond": "Telemundo \u003E\u003E Univision || On what network did Una Maid en Manhattan first play?"
  },
  {
   "subject": "Robert Rental",
   "src": "Which is the cause of death of Robert Rental?",
   "pred": "lung cancer",
   "rephrase": "What caused the death of Robert Rental?",
   "alt": "pneumonia",
   "answers": [
    "lung cancer"
   ],
   "loc": "nq question: what kind of beast is the beast from beauty and the beast",
   "loc_ans": "a chimera",
   "cond": "lung cancer \u003E\u003E pneumonia || Which is the cause of death of Robert Rental?"
  },
  {
   "subject": "Robert Rental",
   "src": "What was the cause of death of Robert Rental?",
   "pred": "suicide",
   "rephrase": "What was the cause of the death of Robert Rental?",
   "alt": "accident",
   "answers": [
    "lung cancer"
   ],
   "loc": "nq question: what is the name of governor of maharashtra",
   "loc_ans": "Chennamaneni Vidyasagar Rao",
   "cond": "suicide \u003E\u003E accident || What was the cause of death of Robert Rental?"
  },
  {
   "subject": "Robert Rental",
   "src": "What was the cause of death for Robert Rental?",
   "pred": "lung cancer",
   "rephrase": "What is the cause of Robert Rental's death?",
   "alt": "murder",
   "answers": [
    "lung cancer"
   ],
   "loc": "nq question: how many episodes is ash vs evil dead season 3",
   "loc_ans": "10",
   "cond": "lung cancer \u003E\u003E murder || What was the cause of death for Robert Rental?"
  }
```


Figure 14: A Samples of the ZsRE dataset.