	Method	Batch Edit	Sequential Edit	Additional Train	Edit Area	Time (s)	VRAM (GB)
Memory-based	SERAC	YES	YES	YES	*External Model*	8.46	42
	IKE	NO	NO	YES	*In-Context*	4.57	52
	GRACE	NO	YES	NO	*MLP+codebook*	142.68	28
	MELO	YES	YES	NO	*LoRA+codebook*	154.32	30
Meta-learning	KE	YES	YES	YES	*MLP*	7.87	49
	MEND	YES	YES	YES	*MLP*	6.39	46
Locate-Then-Edit	KN	NO	YES	NO	*MLP*	425.64	42
	ROME	NO	YES	NO	*MLP*	187.90	31
	MEMIT	YES	YES	NO	*MLP*	169.28	33
	PMET	YES	YES	NO	*MLP*	219.17	34

Table 1: Comparison of several model editing methods. 'Batch Edit' refers to simultaneously editing multiple target knowledge instances. 'Sequential Edit' refers to maintaining previously edited knowledge while performing new edits. 'Additional Train' refers to the need for pre-training other network structures or parameters before editing. 'Edit Area' indicates the location of the edit, with MLP representing the linear layer. 'Time & VRAM' reflects the efficiency of the editing method (using LlaMA-7B as an example). 'Time' indicates the wall clock time required for conducting 10 edits, while VRAM represents the graphics memory usage.

of the model before and after the editing to gauge the editing's multifaceted impact on the model behavior, including generalization and side effects. An example to edit through EasyEdit is depicted in Figure 2.

Note that the ability to execute batch editing (multiple edits in a single instance) and sequential editing (implementing new edits while preserving previous editing) is a crucial feature of knowledge editing (Huang et al., 2023). For methods that support batch editing, editing instances are inputted in chunk form. In addition, EasyEdit provides a boolean switch, enabling users to either retain the pre-edit weights for single-instance editing or discard them for sequential editing.

3.3 Method

As the core component of knowledge editing, editing methods alter the model's behavior by modifying its internal parameters (e.g. MLP, Attention Mechanisms) or explicitly utilizing preceding editing facts, among other strategies. Impressive related works (Table 1) abound in this field, and they can be generally grouped into three categories as proposed by Yao et al. (2023).

Memory-based This category, encompassing methods such as SERAC (Mitchell et al., 2022b), IKE (Zheng et al., 2023a), and GRACE (Hartvigsen et al., 2023), emphasizes the use of memory elements to store and manipulate information during editing. SERAC applies retrieval and classification routing, GRACE replaces hidden states with parameters searched from a codebook for edit memorization, while IKE uses context-edit facts to guide the model in generating edited facts.

Meta-learning These methods learn the weight updates (denoted as Δ), which are then added to the original weights for editing. Examples include KE (Cao et al., 2021), which uses a bidirectional-LSTM to predict weight updates, and MEND (Mitchell et al., 2022a), which adjusts model parameters through low-rank decomposition of gradients.

Locate-Then-Edit This paradigm focuses on knowledge localization to modify the parameters of specific neurons responsible for storing the editing facts. EasyEdit integrates methods like KN (Dai et al., 2021), which employs gradient-based methods to update specific neurons. Moreover, EasyEdit supports ROME (Meng et al., 2023), PMET (Li et al., 2024) and MEMIT (Meng et al., 2022), leveraging causal intervention to pinpoint knowledge within a specific MLP layer and enabling the modification of the entire matrix.

However, it is not practical to expose the editing methods directly to users due to the complexity of the underlying concepts and the time investment required to understand them. Additionally, differences in input-output formats across methods could further complicate the learning process. To circumvent these hurdles, we implement a unified interface, `apply_to_model`, in EasyEdit. Aligning with the *Strategy* design pattern, this interface is designed to be overridden by different types of editing methods, ensuring consistent input and out-

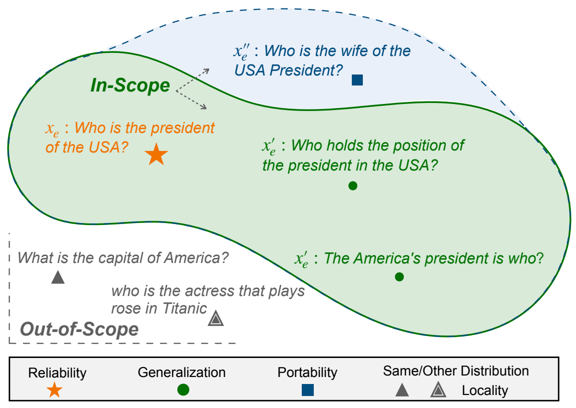
x_e'' : Who is the wife of the USA President?
In-Scope
x_e : Who is the president of the USA?
x_e' : Who holds the position of the president in the USA?
What is the capital of America?
x_e' : The America's president is who?
who is the actress that plays rose in Titanic
Out-of-Scope
Reliability
Generalization
Portability
Same/Other Distribution
Locality

Figure 3: Depiction of the edit scope for edit descriptor *Who is the president of the USA?* It contains an example for knowledge editing evaluation, including Reliability, Generalization, Portability, and Locality.

put types. Specifically, it accepts a 'request' that includes the editing descriptor, the target of the edit, and any input data necessary to evaluate the editing performance. After processing the request(s), the interface returns the edited model weights. This design ensures both flexibility and easy-to-use, enabling users to handle knowledge editing instances effortlessly and utilize the customized models in other downstream tasks.

3.4 Hparams

When initializing an editing method, it is crucial to specify the related hyperparameters. These include the model to be edited, the layers targeted for modification, and, optionally, the type of external model, among other parameters. For methods that alter the LLMs' internal parameters, the adjustable parameter names should be indicated using the MODULE_NAME format, such as *transformer.h.5.mlp.fc_out*. In this case, the parameters of the *fc_out* linear layer in the fifth layer MLP of GPT-J would be modified, while all other parameters remain frozen. Layer selection adheres to the locality of knowledge (Meng et al., 2023) or retains layers with higher success rates in pilot experiments (Mitchell et al., 2022a), as elaborated in Appendix B.

All hyperparameter classes derive from a common base class, `Hyperparams`, which includes necessary attributes and abstract methods. This base class supports loading hyperparameters in both *yaml* and *json* formats. Moreover, the `Hyperparams` base class can be used to initialize the Trainer module, streamlining the workflow.

3.5 Trainer

Certain editing methods, which employ meta-learning or utilize classifiers (as shown in Table 1), necessitate the training of additional parameters or the implementation of extra network structures. Similar to Hyperparameters (`Hparams`), all Trainer classes inherit from a common base class, `BaseTrainer`. It includes essential attributes and abstract methods such as run and validate steps. Subclasses of the `BaseTrainer` define specific training steps for editing, such as calculating editing loss and locality loss, as well as the strategies for combining these losses. Once additional network structures are obtained, the subsequent editing process follows the same path as the Training-Free method. In EASYEDIT, various Trainers can be easily called with one click.

4 Evaluation

Knowledge editing, as defined by Mitchell et al. (2022b), involves supplying a specific editing descriptor x_e (input instance) and an editing target y_e (desired output). From these, an editing instance z_e is generated in the form: $z_e \sim [x_e, y_e]$. The goal is to adjust the behavior of the initial base model f_θ (where θ represents the model's parameters) to produce an edited model f_{θ_e}. Ideally, for the editing instance, the edited model would behave such that $f_{\theta_e}(x_e) = y_e$. Additionally, the editing scope $S(z_e)$ refers to a set of input examples whose true labels have been influenced by the editing instance. In most cases, a successful edit should affect the model's predictions for numerous In-Scope ($I(x_e) \sim \{x'_e | x'_e \in S(z_e)\}$) inputs, while leaving Out-of-Scope ($O(x_e) \sim \{x'_e | x'_e \notin S(z_e)\}$) inputs unchanged.

We employ six dimensions of metrics to assess the performance of editing methods, including **Reliability**, **Generalization**, **Locality**, **Portability**, **Fluency** (Zhang et al., 2018) and **Efficiency** (as shown in Figure 3).

Reliability This metric measures the average accuracy on the given editing instance z_e.

Generalization The edit should appropriately influence in-scope inputs, this metric gauges the average accuracy on in-scope inputs $I(x_e)$.

Locality Editing should adhere to the principle of locality, it evaluates whether out-of-scope inputs $O(x_e)$ can remain unchanged as the base model.

Portability The robust generalization of the edit, assessing whether the edited knowledge can be effectively applied to related content.

Fluency It measures the weighted average of bi-gram and tri-gram entropies to assess the diversity of text generations.

Efficiency Editing should be time and resource-efficient. This metric quantifies efficiency by measuring editing time and VRAM consumption.

5 Experiments

In this section, we will outline the experiment setting and report the empirical results of multiple editing methods supported in EasyEdit (Table 2).

5.1 Experiment Setting

To validate the potential application of knowledge editing on LLMs, we utilize **LlaMA 2** (7B) (Touvron et al., 2023b), a model with a large parameter size, representing the decoder-only structure.

We employ the ZsRE dataset to test the capability of knowledge editing in incorporating substantial and general fact associations into the model. ZsRE (Levy et al., 2017) is a question-answering (QA) dataset that generates an equivalence neighbor through back-translation. Later, it is further expanded by Yao et al. (2023) to provide a more comprehensive evaluation of knowledge editing, including an assessment of the LLMs' ability to integrate the edited fact with other facts related to the target object o* (an aspect of Portability). For baselines, we compare various editing methods and additionally employ FT-L from ROME (Meng et al., 2023). FT-L updates parameters for a single MLP layer and applies an $L\infty$ norm constraint to limit the weight changes.

5.2 Experiment Results

Table 2 reveals SERAC and IKE's superior performance on the ZsRE datasets, exceeding 99% on several metrics. While ROME and MEMIT perform sub-optimally in generalization, they exhibit relatively high performance in terms of reliability and locality. IKE exhibits the potential of gradient-free updates through in-context learning, leading to near-perfect scores in both reliability and generalization. However, it shows some deficiency in locality, as preceding prompts may influence out-of-scope inputs. GRACE exhibits poor generalization, possibly attributed to the lack of explicit semantic representation in its activations within

	Reliability	**Generalization**	**Locality**	**Portability**	**Fluency**
FT-L	56.94	52.02	96.32	51.03	488.41
SERAC	99.49	99.13	100.00	57.82	423.22
IKE	**100.00**	**99.98**	69.19	**67.56**	557.37
MEND	94.24	90.27	97.04	56.95	540.06
KN	28.95	28.43	65.43	37.18	478.32
ROME	92.45	87.04	99.63	57.47	**587.58**
MEMIT	92.94	85.97	99.49	60.64	576.51
GRACE	99.22	0.43	**100.00**	56.87	426.31

Table 2: Editing results of the four metrics on LlaMA-2 using EasyEdit. The settings for the model and the dataset are the same with Yao et al. (2023).

the decoder-only model (Liu et al., 2023b). FT-L's performance on ZsRE falls significantly short compared to ROME, even though both methods modify the same layer parameters. This suggests that under the norm constraint, fine-tuning is not an effective strategy for knowledge editing. MEND performs well overall, achieving over 90% accuracy on multiple metrics and even surpassing ROME in terms of reliability and generalization. KN performs poorly, indicating that it may be better suited for editing tasks in smaller models or tasks involving knowledge attribution.

For the Portability evaluation, where the inference depends on a single connection or 'hop' between facts, most editing methods struggle to effectively combine the edited fact with other facts relevant to the target object o*. While SERAC obtains good performance on previous metrics, it completely fails to propagate the edited knowledge. This is because SERAC utilizes an external model with a smaller parameter size for counterfactual routing whereas the smaller model struggles to recall a rich set of relevant facts. IKE still maintains a relatively high capability for ripple editing (exceeding 67%), demonstrating that in-context learning is a promising approach to propagate edited knowledge to other related facts.

6 Conclusion and Future work

We propose EasyEdit, an easy-to-use knowledge editing framework for LLMs, which supports many cutting-edge approaches and various LLMs. The ability to edit and manipulate LLMs in a controlled and targeted manner may open up new possibilities for knowledge augmentation (Wu et al., 2023, 2020; Zhang et al., 2022; Chen et al., 2022) and adaptation across various natural language processing tasks (Kaddour et al., 2023). In the future, we will continue to integrate advanced editing technologies into EasyEdit, aiming at facilitating further research and inspiring new ideas for the NLP community.