EasyEdit: An Easy-to-use Knowledge Editing Framework for Large Language Models

Peng Wang[♣], **Ningyu Zhang**[♣]*, **Bozhong Tian**[♣], **Zekun Xi**[♣], **Yunzhi Yao**[♣], **Ziwen Xu**[♣], **Mengru Wang**[♣], **Shengyu Mao**[♣], **Xiaohan Wang**[♣], **Siyuan Cheng**[♣], **Kangwei Liu**[♣], **Yuansheng Ni**[♣], **Guozhou Zheng**[♣], **Huajun Chen**[♣]*
[♣] Zhejiang University
https://github.com/zjunlp/EasyEdit

Abstract

Large Language Models (LLMs) usually suffer from knowledge cutoff or fallacy issues, which means they are unaware of unseen events or generate text with incorrect facts owing to outdated/noisy data. To this end, many knowledge editing approaches for LLMs have emerged – aiming to subtly inject/edit updated knowledge or adjust undesired behavior while minimizing the impact on unrelated inputs. Nevertheless, due to significant differences among various knowledge editing methods and the variations in task setups, there is no standard implementation framework available for the community, which hinders practitioners from applying knowledge editing to applications. To address these issues, we propose EASYEDIT, an easy-to-use knowledge editing framework for LLMs. It supports various cutting-edge knowledge editing approaches and can be readily applied to many well-known LLMs such as T5, GPT-J, LlaMA, etc. Empirically, we report the knowledge editing results on LlaMA-2 with EASYEDIT, demonstrating that knowledge editing surpasses traditional fine-tuning in terms of reliability and generalization. We have released the source code on GitHub[1], along with Google Colab tutorials and comprehensive documentation[2] for beginners to get started. Besides, we present an online system[3] for real-time knowledge editing, and a demo video[4].

1 Introduction

Large Language Models (LLMs) have revolutionized modern Natural Language Processing (NLP), significantly improving performance across various tasks (Brown et al., 2020; OpenAI, 2023; Anil et al., 2023; Zhao et al., 2023; Touvron et al., 2023b; Qiao et al., 2023; Zheng et al., 2023b; Pan et al., 2023). However, deployed LLMs usually suffer from knowledge cutoff or fallacy issues. For example, LLMs such as ChatGPT and LlaMA possess information only up to their last training point. They can sometimes produce inaccurate or misleading information due to potential discrepancies and biases in their pre-training data (Ji et al., 2023; Hartvigsen et al., 2022). Hence, it's essential to efficiently update the parametric knowledge within the LLMs to modify specific behaviors while avoiding expensive retraining.

Indeed, finetuning or parameter-efficient finetuning (Ding et al., 2022, 2023) offers methods for modifying LLMs, these approaches can be computationally expensive and may lead to overfitting, particularly when applied to a limited number of samples (Cao et al., 2021) or streaming errors of LLMs. Additionally, fine-tuned models might forfeit capabilities gained during pre-training, and their modifications do not always generalize to relevant inputs. An alternative methodology involves using manually written or retrieved prompts to influence the LLMs' output. These methods suffer from reliability issues, as LLMs do not consistently generate text aligned with the prefix prompt (Hernandez et al., 2023; Lewis et al., 2021). Additionally, due to the extensive amount of up-to-date knowledge required for complex reasoning tasks, the impracticality of context overload becomes inevitable whenever the context length is limited.

A feasible solution, knowledge editing[5], aims to efficiently modify the behavior of LLMs with minimal impact on unrelated inputs. Research on knowledge editing for LLMs (Meng et al., 2023, 2022; Zheng et al., 2023a; Gupta et al., 2023; Mitchell et al., 2022a; Geva et al., 2023; Hase et al., 2023; Cohen et al., 2023a; Hartvigsen et al., 2023; Tan et al., 2024; Yu et al., 2023) have displayed remarkable progress across various tasks and settings.

*Corresponding author.

[1]This is a subproject of KnowLM (https://github.com/zjunlp/KnowLM), which facilitates knowledgeable LLM Framework with EasyInstruct, EasyEdit, EasyDetect etc.

[2]https://zjunlp.gitbook.io/easyedit

[3]https://huggingface.co/spaces/zjunlp/EasyEdit

[4]https://youtu.be/Gm6T0QaaskU

[5]Knowledge editing can also be termed as model editing.

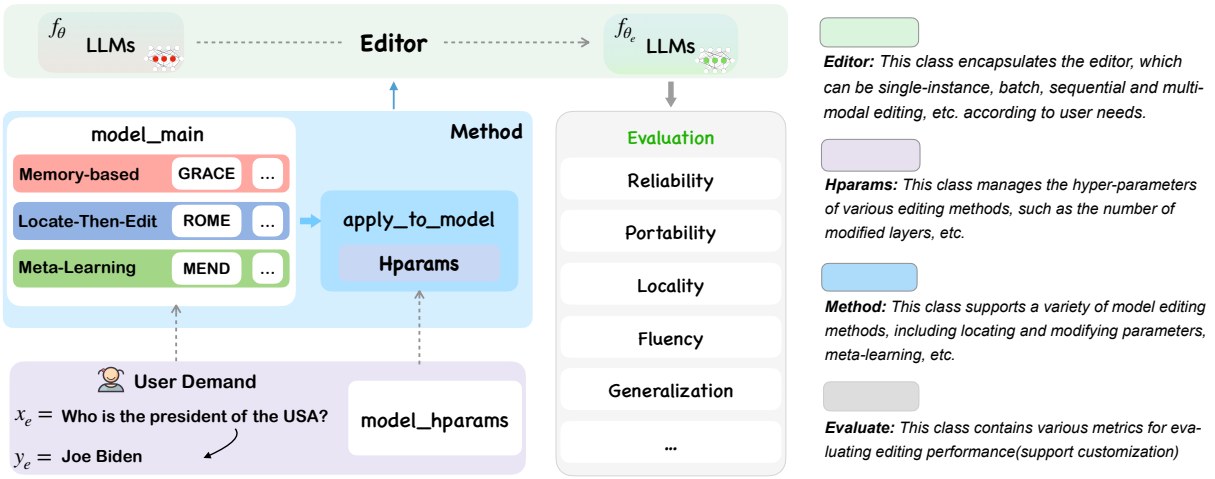

Figure 1: The overall architecture of EASYEDIT. The main function is `apply_to_model`, which applies the selected editing method to the LLMs. The **Editor** serves as the direct entry point, receiving customized user inputs and outputs, and returning the edited weights. Please note that some methods may require pre-training of classifiers or hypernetworks through the Trainer (See §3.5). EASYEDIT supports customizable evaluation metrics.

However, these variations in both implementation and task settings have impeded the development of a unified and comprehensive framework for knowledge editing. Note that the complexity obstructs the direct comparison of effectiveness and feasibility between different methods, and complicates the creation of novel knowledge editing approaches. To this end, we propose EASYEDIT, an easy-to-use knowledge editing framework for LLMs. EASYEDIT modularizes editing methods and effectiveness evaluation while considering their combination and interaction. It supports a variety of editing scenarios, including **single-instance**, **batch-instance**, **sequential**, and **multi-modal** editing. Moreover, EASYEDIT provides evaluation evaluations of key metrics such as Reliability, Generalization, Locality, and Portability (Yao et al., 2023), to quantify the robustness and side effects (Cohen et al., 2023b) of editing methods.

Specifically, in EASYEDIT, the Editor class integrates various editing components. The Method class offers a unified interface `apply_to_model`, which accepts editing descriptors and returns the edited model, thereby facilitating the integration of novel editing methodologies. Dedicated to evaluating editing performance, the Evaluate module leverages metrics such as reliability, robust generalization, and locality. The Trainer module manages the training of additional neural network structures. Each module in EASYEDIT is meticulously defined, striking a balance between cohesion and coupling. Furthermore, we furnish examples of editing across a spectrum of models, including T5 (Raffel et al., 2019), GPT-J (Wang and Komatsuzaki, 2021), GPT-NEO (Black et al., 2021), GPT2 (Radford et al., 2019), LLaMA (Touvron et al., 2023a), LLaMA-2 (Touvron et al., 2023b), Mistral (Jiang et al., 2023), and Qwen (Bai et al., 2023). We acknowledge all the support for EASYEDIT, which is listed in Appendix 6 due to space constraints.

2 Background

Previous Solutions Despite the tremendous success of LLMs in almost all NLP tasks, persistent challenges such as knowledge cutoff and biased/toxic outputs remain. To counter these challenges, two approaches are generally employed:

1) FINE-TUNING: Traditional fine-tuning techniques, along with delta tuning (Ding et al., 2022) and LoRA tuning (Hu et al., 2021) utilize domain-specific datasets to update the model's internal parametric knowledge. However, these methods face two notable challenges: First, they consume considerable resources. Second, they risk the potential of catastrophic forgetting (Ramasesh et al., 2022).

2) PROMPT-AUGMENTATION: Given a sufficient number of demonstrations or retrieved contexts, LLMs can learn to enhance reasoning (Yu et al., 2022) and generation through external knowledge (Borgeaud et al., 2022; Guu et al., 2020; Lewis et al., 2020). However, the performance may be sensitive to factors such as the prompting template, the selection of in-context examples (Zhao et al., 2021), or retrieved contexts (Ren et al.,

2023). These approaches also encounter the issue of context length limitation (Liu et al., 2023a).

Knowledge Storage Mechanism Within the NLP literature, numerous studies have delved into understanding the location of different types of knowledge in language models (Petroni et al., 2019; Roberts et al., 2020; Jiang et al., 2020). LLMs can be conceptualized as knowledge banks, and the transformer MLP layers function as key-value memories according to observations from Geva et al. (2021). This configuration promotes efficient knowledge adjustments by precisely localizing knowledge within the MLP layers (denoted as knowledge editing).

Knowledge editing enables nimble alterations to the LLMs' behavior through one data point. Another promising attribute of knowledge editing is its ability to ensure the locality of editing, meaning that modifications are contained within specific contexts. Additionally, the knowledge editing technique can mitigate harmful language generation (Geva et al., 2022). In this paper, we present EASYEDIT, an easy-to-use knowledge editing framework for LLMs. It seamlessly integrates diverse editing technologies and supports the free combination of modules for various LLMs. Through its unified framework and interface, EASYEDIT enables users to swiftly comprehend and apply the prevalent knowledge editing methods included in the package.

3 Design and Implementation

EASYEDIT provides a complete editing and evaluation process built on Pytorch (Paszke et al., 2019) and Huggingface (Wolf et al., 2020). This section commences with an exploration of the assemblability aspect of EASYEDIT, followed by a detailed explanation of the design and implementation of each component within the EASYEDIT framework (as shown in Figure 1). Additionally, we demonstrate a straightforward example of applying MEND to LLaMA, altering the output of *the U.S. President* to *Joe Biden*.

3.1 Assemblability

In the realm of knowledge editing, various distinct scenarios[6] exist. To cater to this diversity, EASYEDIT offers flexible combinations of modules that different editing `Editor` (such as single-instance, batch-instance (details in Appendix A)),

[6]Denoted as `(Editor, METHOD, TARGET)`

```python
# Step 1: Choose the Editing Method e.g. MEND
from easyeditor import BaseEditor
from easyeditor import MENDHyperParams
hparams = MENDHyperParams.from_hparams('gpt2-xl.yaml')


# Step 2: Provide the edit descriptor and target
prompts = ['Q: The president of the US is? A:',]
target_new = ['Joe Biden']
rephrase_prompts = ['The leader of the United State is']

# Step 3:  Combine them into the `BaseEditor`
editor = BaseEditor.from_hparams(hparams)

# Step 4: Edit and Evaluation
metrics, edited_model = editor.edit(
    prompts=prompts,
    target_new=target_new,
    keep_original_weight=True,
)

# metrics: Performance of knowledge editing
# edited_model: Model after modifying the weights
```

Figure 2: A running example of knowledge editing for LLMs in EASYEDIT. Utilizing the MEND approach, we can successfully transform the depiction of *the U.S. President* into that of *Joe Biden*.

`METHOD` (such as ROME, GRACE (§3.3)). About editing `TARGET`, EASYEDIT can accommodate any parameterized white-box existing model. Additionally, recent research (Dong et al., 2022) indicates that LLMs exhibit robust in-context learning capabilities. By providing edited facts to LLMs, one can alter the behavior of black-box models such as GPT4 (OpenAI, 2023). All those combinations are easily implementable and verifiable within the EASYEDIT framework.

3.2 Editor

The `Editor` serves a pivotal role in knowledge editing as it directly establishes the editing tasks and corresponding editing scenarios. Users supply the editor descriptor (x_e) and the edit target (y_e), but the input format varies according to the different editing objects. For instance, in Seq2Seq models, the edit target typically serves as the decoder's input, while in autoregressive models, x_e and y_e need to be concatenated to maximize the conditional probability. To facilitate unified editing across diverse architecture models, we meticulously develop a component `prepare_requests` to transform editing inputs.

In EASYEDIT, we provide an "edit" interface, incorporating components such as `Hparams`, `Method`, and `Evaluate`. During the editing phase, various knowledge editing strategies can be executed by invoking the `apply_to_model` function available in all different methods, it also performs evaluations