

OneEdit: A Neural-Symbolic Collaboratively Knowledge Editing System

Ningyu Zhang
Zhejiang University
Hangzhou, Zhejiang, China
zhangningyu@zju.edu.cn

Zekun Xi
Zhejiang University
Hangzhou, Zhejiang, China
22351325@zju.edu.cn

Yujie Luo
Zhejiang University
Hangzhou, Zhejiang, China
luo.yj@zju.edu.cn

Peng Wang
Zhejiang University
Hangzhou, Zhejiang, China

Bozhong Tian
Zhejiang University
Hangzhou, Zhejiang, China

Yunzhi Yao
Zhejiang University
Hangzhou, Zhejiang, China

Jintian Zhang
Zhejiang University
Hangzhou, Zhejiang, China

Shumin Deng
National University of Singapore,
NUS-NCS Joint Lab
Singapore, Singapore

Mengshu Sun
Ant Group
Hangzhou, Zhejiang, China

Lei Liang
Ant Group
Hangzhou, Zhejiang, China

Zhiqiang Zhang
Ant Group
Hangzhou, Zhejiang, China

Xiaowei Zhu
Ant Group
Hangzhou, Zhejiang, China

Jun Zhou
Ant Group
Hangzhou, Zhejiang, China

Huajun Chen*
Zhejiang University
Hangzhou, Zhejiang, China

ABSTRACT

Knowledge representation has been a central aim of AI since its inception. Symbolic Knowledge Graphs (KGs) and neural Large Language Models (LLMs) can both represent knowledge. KGs provide highly accurate and explicit knowledge representation, but face scalability issue; while LLMs offer expansive coverage of knowledge, but incur significant training costs and struggle with precise and reliable knowledge manipulation. To this end, we introduce **OneEdit**, a neural-symbolic prototype system for collaborative knowledge editing using natural language, which facilitates easy-to-use knowledge management with KG and LLM. OneEdit consists of three modules: 1) The *Interpreter* serves for user interaction with natural language; 2) The *Controller* manages editing requests from various users, leveraging the KG with rollbacks to handle knowledge conflicts and prevent toxic knowledge attacks; 3) The *Editor* utilizes the knowledge from the Controller to edit KG and LLM. We conduct experiments on two new datasets with KGs which demonstrate that OneEdit can achieve superior performance.

VLDB Workshop Reference Format:

Ningyu Zhang, Zekun Xi, Yujie Luo, Peng Wang, Bozhong Tian, Yunzhi Yao, Jintian Zhang, Shumin Deng, Mengshu Sun, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. OneEdit: A Neural-Symbolic Collaboratively Knowledge Editing System. VLDB 2024 Workshop: LLM+KG.

VLDB Workshop Artifact Availability:

We open-source the whole system at <https://github.com/zjunlp/OneEdit> and provide an online demo at <http://oneedit.zjukg.cn/>.

1 INTRODUCTION

The pursuit of empowering machines to master knowledge has remained a fundamental objective in the advancement of Artificial Intelligence (AI) systems. Over the years, researchers have devoted with various methods to enable machines to acquire knowledge, thereby supporting a wide range of tasks such as information retrieval [46], question answering [21], dialogue [59], reasoning [52], recommendation [43, 45], and domain specific applications [56]. Concretely, knowledge updating and management stand out as essential capabilities, empowering machines to adeptly adjust to new environments and tasks, thus facilitating lifelong learning [49, 51].

Early, Knowledge Graphs [19], as a form of symbolic knowledge representation, have garnered significant research interest from both academia and industry. KG is a structured representation of facts, composed of entities, relations, and semantic descriptions, which can be simply and precisely updated through symbolic manipulation. However, KG face challenges regarding scalability and the transferability of reasoning. On the other hand, Large Language Models (LLMs) have learned rich “modaledge” [13], potentially creating a kind of “world model” [48] and serving as parametric knowledge bases [35]. Based on the above hypothesis, researchers

*Corresponding author.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by

emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment. Proceedings of the VLDB Endowment. ISSN 2150-8097.

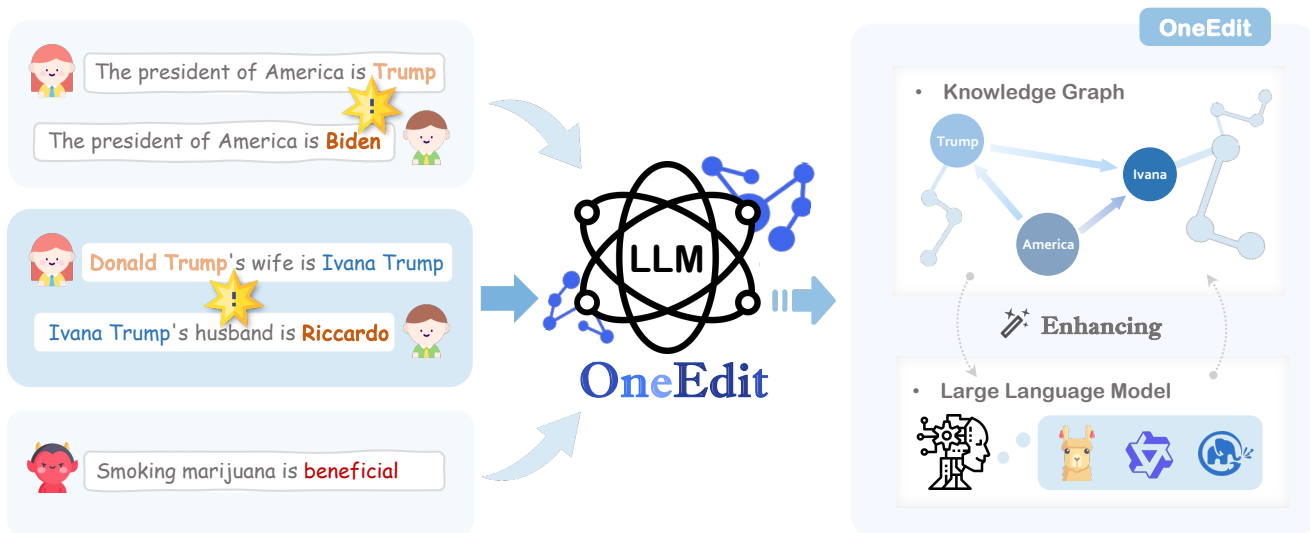


Figure 1: OneEdit for neural-symbolic collaboratively knowledge editing with KGs and LLMs.

try to manipulate knowledge in LLMs and introduce knowledge editing to add, modify, or erase parametric knowledge, making neural knowledge representation space well aligned with up-to-date symbolic world knowledge [57]. However, LLMs incur significant training costs and struggle with precise knowledge manipulation. These limitations lead to severe side effects, including decreased general abilities [11], poor generalization [47], knowledge conflicts [20], and the risk of toxic knowledge attacks [23]. Intuitively, the integration of KG and LLM can leverage complementary advantages to alleviate the issues, offering a more reliable, and controllable approach to knowledge representation and management [32, 33, 39].

To this end, we introduce **OneEdit**, a neural-symbolic collaboratively knowledge editing system with KG and LLM as shown in Figure 1, providing a platform for operating neural and symbolic knowledge with natural language. OneEdit consists of three primary components: the Interpreter, the Controller, and the Editor. The Interpreter acts as the interface for user interaction with natural language, responsible for understanding user intent. The Controller manages editing requests from various users, using KG for conflict resolution and knowledge augmentation. The Editor primarily utilizes knowledge provided by the Controller to edit KG and LLM. The entire system is designed in a modular fashion, supporting and customizable for versatile KG and various LLMs.

We conduct experiments on two new datasets: one focused on American politicians and the other on academic figures, both containing KG. Our observations indicate that OneEdit can achieve neural-symbolic collaboratively knowledge editing with Qwen2-7B and GPT-J-6B and outperform the baselines, particularly excelling in handling knowledge conflict issues.

2 RELATED WORK

Large Language Models. Typically, LLMs, such as GPT-4 [31] and LLaMA [42], usually denote Transformer-based models with hundreds of billions of parameters, trained on extensive text datasets [53].

Some work has suggested that LLMs can be regarded as parametrized knowledge bases [1, 24, 34, 36], as they are capable of recalling massive factual knowledge through prompt engineering [3, 6, 8, 38]. However, a major limitation of these parameterized knowledge bases is their inability to update stored information in real-time. Once pretrained, LLMs possess a fixed snapshot of knowledge reflecting the data they were trained on [26], and remain unable to incorporate new information unless retrained with updated datasets. This limitation diminishes the efficacy of LLMs in rapidly evolving sectors, such as current affairs, scientific developments, and cultural dynamics [10].

Knowledge Graph. KGs are structured representations that map out complex networks of real-world entities and their interrelationships [34], facilitating advanced understanding and reasoning in NLP application with structured knowledge triples [4, 9, 12]. In KGs, symbolic knowledge leverages logical rules for reasoning, providing robust interpretability and precise inferential capabilities. Concretely, Knowledge Extraction (KE) [50] is essential for populating KGs from vast, unstructured datasets. This involves sophisticated NLP tasks such as Named Entity Recognition (NER), relation extraction, and entity resolution to accurately distill structured knowledge from texts [22].

Knowledge Editing. The primary knowledge editing methods currently can be categorized into three groups [54]: meta-learning, locate-then-edit, and memory-based. Meta-learning methods employ external network to predict necessary gradient for editing, MEND [29] and MALMEN [40] utilizes a hyper-network to transform model gradients for updating model parameters. As to the locate-then-edit methods, ROME [27] and MEMIT [28] achieve edits by locating and modifying model parameters. For memory-based methods, the specific hidden states or neurons that store the edit knowledge are used for post-edit response, SERAC [30] leverages a scope classifier and trained sub-models for knowledge editing. Additionally, InstructEdit [41] focuses on general editing scenarios,

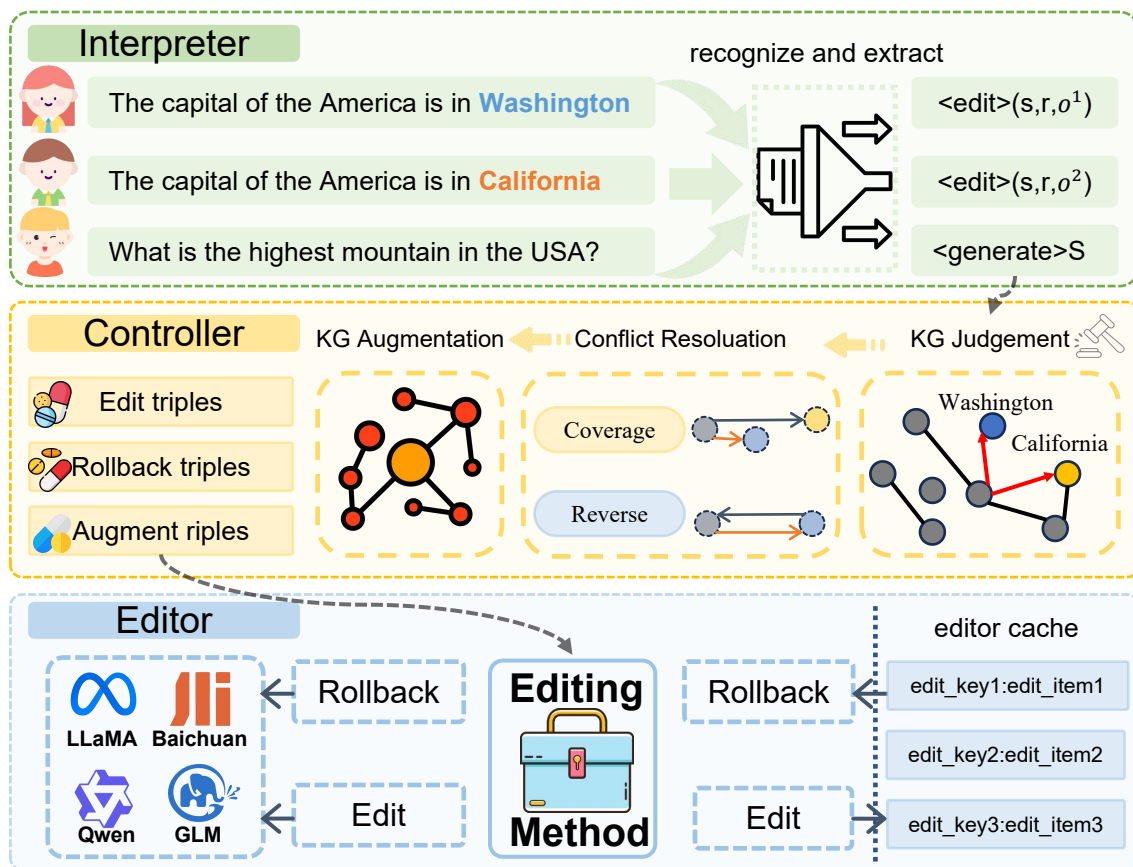


Figure 2: The detailed workflow of OneEdit in handling conflicts: the natural language input from the user is extracted into knowledge triples by the Interpreter, then processed by the Controller to generate sequences of editing triples and rollback triples, which are finally sent to the Editor.

and GRACE [14] uses a codebook to store edited hidden states for lifelong editing. Yet most existing methods still suffer from precise knowledge manipulation with severe side effects [11, 20, 23, 47]. In particular, in the noisy environment of the internet, there is a plethora of conflicting, erroneous, and toxic knowledge, posing significant challenges to knowledge editing systems [15, 37, 55].

3 SYSTEM DESIGN

As shown in Figure 2, OneEdit comprises three primary components: **Interpreter**, **Controller**, and **Editor**. The Interpreter serves as the interface for user interaction with OneEdit, responsible for discerning user intent. The Controller manages editing requests from various users, utilizing KG for conflict resolution and knowledge augmentation. The Editor primarily uses augmented knowledge triples from the Controller to edit KG and LLMs.

3.1 Knowledge Editing for LLMs

Suppose the original model is \mathcal{M} and k is the knowledge that needs to be changed, by knowledge editing function $E(\cdot)$, we obtain the post-edited model \mathcal{M}' which should override the prior erroneous memory about knowledge k while preserving other unrelated

knowledge k' as:

$$\begin{cases} \mathcal{M}' = E(\mathcal{M}, k) \\ \mathcal{M}'(k) \neq \mathcal{M}(k) \\ \forall k' \neq k, \mathcal{M}'(k') = \mathcal{M}(k') \end{cases} \quad (1)$$

We hope by knowledge editing, we can:

precisely and generally manipulate knowledge in LLMs without impacting unrelated knowledge.

3.2 Neural-Symbolic Knowledge Editing

In this paper, we focus on simultaneously updating symbolic knowledge in KGs and parametric knowledge in LLMs (a.k.a., neural-symbolic knowledge editing), thus allowing different types of knowledge to complement each other and compensate for the challenges introduced by parameterized knowledge editing. Formally, a KG can be represented as $\mathcal{G} = (\mathcal{S}, \mathcal{R}, \mathcal{O})$, where \mathcal{S} , \mathcal{R} and \mathcal{O} are sets of subjects, relation types, and objects. Each knowledge triple t in \mathcal{G} takes the form (s, r, o) , where $s \in \mathcal{S}$, $r \in \mathcal{R}$ and $o \in \mathcal{O}$. For neural-symbolic knowledge editing, the knowledge required edits

is defined as $k = (p, y)$, where p is the prompt to express the knowledge and y is the desired target new. Formally, we define the task in OneEdit as follows:

$$\mathcal{T}_r, \mathcal{T}_e, \mathcal{T}_a = C(I(k) \mid \mathcal{G}, \mathcal{M}) \quad (2)$$

where \mathcal{M} and \mathcal{G} denote the original LLM and KG, respectively. The function $I(\cdot)$, denoted as the Interpreter, is responsible for transforming knowledge sentences into knowledge triples. $C(\cdot)$ denoted as the Controller operates based on the KG to derive a series of editing triples \mathcal{T}_e and rollback triples \mathcal{T}_r to address knowledge conflicts. Additionally, the Controller generates some knowledge augmentation triples \mathcal{T}_a to strengthen the edited knowledge and prevent knowledge distortion caused by the knowledge editing [25]. After obtaining the sets \mathcal{T}_r , \mathcal{T}_e and \mathcal{T}_a , we can use them to modify the model and the KG, ensuring that the parametric knowledge of the model and the symbolic knowledge of the graph are consistent with reality:

$$\mathcal{G}', \mathcal{M}' = E(\mathcal{T}_r, \mathcal{T}_e, \mathcal{T}_a, \mathcal{G}, \mathcal{M}) \quad (3)$$

3.3 Interpreter

The Interpreter functions as the interface between the user and the Controller, tasked with recognizing user intent expressed in natural language. If the user’s intent is editing, the Interpreter converts the user’s input into knowledge triples suitable for the knowledge graph and sends them to the Controller. If the user’s intent is querying, the Interpreter takes no action and passes the input to the large language model for generation.

We perform instruction fine-tuning on the MiniCPM-2B [17] to enable it to function as an Interpreter capable of distinguishing between editing intent and response intent. For generation intent data, we utilized the Alpaca dataset, as its instruction-following data involves everyday conversations without editing intent. We used the input from this instruction-following data as the model’s generation intent data. For editing intent data, we manually created ten examples, using them as prompts for GPT-4 to generate similar but distinct editing intent data.

We combine the generation intent data and the editing intent data to train the model, enabling it to acquire both knowledge extraction and intent recognition capabilities.

$$\text{output} = \begin{cases} \langle \text{edit} \rangle (s, r, o) & \text{if intend (S) is edit} \\ \langle \text{generate} \rangle S & \text{if intend (S) is generate} \end{cases} \quad (4)$$

For instance, if a user inputs the sentence “Change the President of the USA to Biden”, the Interpreter outputs the edit command $\langle \text{edit} \rangle (\text{USA}, \text{President}, \text{Biden})$ and forwards it to the Controller. Conversely, for a dialogue query, such as “What is the highest mountain in the USA?”, the Interpreter outputs $\langle \text{generate} \rangle$ and directly prompts the larger model to provide an answer.

3.4 Controller

In the Controller, the input is a knowledge triple. The Controller utilizes a knowledge graph (KG) as a knowledge base aligned with the model’s parameters to identify and resolve conflicts between the input knowledge and the knowledge embedded in the model’s

parameters. After resolving conflicts, the knowledge graph is employed to augment the edited knowledge within the parameters of the large language model.

3.4.1 Conflict Resolution. In conflict resolution, we categorize conflicts into two types: coverage conflict and reverse conflict, which are the most common situations encountered in knowledge base management.

Coverage Conflict. In practice, factual knowledge dynamically evolves in response to changes in the real world. For example, the answer to “Who is the highest market value company in the United States?” may vary significantly over a short period. To maintain consistency with the evolving real-world knowledge, it necessitates multiple coverage edits to the model. We formalize this situation as a pair of consecutive edits that contain the same subjects and relations but different objects:

$$\text{Coverage Conflict: } \begin{cases} E_1 = (s, r, o) \rightarrow (s, r, o_1) \\ E_2 = (s, r, o_1) \rightarrow (s, r, o_2) \end{cases} \quad (5)$$

However, research by [16] demonstrates that when modifying model parameters, the fundamental performance of the model inevitably degrades as the number of edits increases [5]. Additionally, [25] reveals that most current editing methods, such as FT, ROME, and MEMIT, leave residual knowledge from previous edits when repeatedly modifying the same piece of knowledge. This leads to the model’s parameters storing multiple conflicting versions of the knowledge, ultimately causing Knowledge Distortion and affecting the model’s expression of that knowledge.

To address this issue, we propose the concept of edit rollback. When users perform knowledge editing on the LLM, the controller places the corresponding factual triple (s, r, o) into the KG for assessment. If there is no existing triple with the same subject and relation but a different object (s, r, o') in the KG, we write (s, r, o) into the KG, proceed with the corresponding knowledge editing, and store corresponding edit parameters in the edit cache (details in 3.5). If (s, r, o) already exists in the KG, no action is taken on the model. If (s, r, o') exists in the KG, we retrieve the edit cache concerning (s, r, o') , completely remove the previous edit within the model, rollback the edit concerning (s, r, o') , and re-edit to (s, r, o) , updating the KG accordingly:

$$\mathcal{M}' = E\left(\mathcal{M} + \sum_{i=0}^n \theta_i - \theta_k\right), k \in (0, n) \quad (6)$$

Furthermore, in the context of crowdsourced editing, if malicious edits are made to the model to produce harmful content, we can also identify and rollback those specific edits. Throughout this process, we perform at most one edit to the knowledge (s, r, o) and perfectly eliminate any previous edits, effectively minimizing performance loss in the model.

Reverse Conflict. When humans learn that Biden, not Trump, is the President of the United States, they naturally understand that the President of the United States is Biden. This generalization process is so seamless that it seems trivial. However, Berglund et al. [2] point out that large language models do not perform well in this regard, which is referred to as the “reverse curse”. In knowledge editing, the reverse curse also exists. For instance, when we edit

Algorithm 1 Resolution of Coverage Conflicts

```
1: Input: Model  $\mathcal{M}$ , Knowledge Graph  $\mathcal{G}$ , Edit Triple  $(s, r, o)$ 
2: Output: Updated Model  $\mathcal{M}'$ , Updated  $\mathcal{G}'$ 
3: procedure COVERAGECONFLICT( $\mathcal{M}, \mathcal{G}, (s, r, o)$ )
4:   if  $(s, r, o') \in \mathcal{G}$  for any  $o' \neq o$  then
5:     Rollback the model's edit concerning  $(s, r, o')$ 
6:     Remove  $(s, r, o')$  from  $\mathcal{G}$ 
7:     Add  $(s, r, o)$  to  $\mathcal{G}$ 
8:     Proceed with the model edit to  $(s, r, o)$ 
9:   else if  $(s, r, o) \notin \mathcal{G}$  then
10:    Add  $(s, r, o)$  to  $\mathcal{G}$ 
11:    Proceed with the model edit to  $(s, r, o)$ 
12:   else
13:    No action is taken on the model
14:   end if
15:   return  $\mathcal{M}', \mathcal{G}'$ 
16: end procedure
```

the knowledge within an LLM to overwrite old information with new information, such as “Biden is the President of the United States”, the model might correctly respond that Biden is the current president. However, when asked “Who is the President of the United States?” the model might still respond with outdated knowledge, saying “Trump is the President of the United States.” This indicates that the model has not fully erased the old knowledge, leading to reverse conflicts.

Furthermore, consider a more complex example: If we edit the fact that Biden’s wife is Jill and subsequently edit the fact that Jill’s husband is Mike, a traditional knowledge base might treat the facts (Biden, wife, Jill) and (Jill, husband, Mike) as isolated pieces of knowledge. This approach overlooks the inherent connection between these facts, specifically that the inverse relationship of “wife” is “husband”. This situation also results in a more complex form of reverse conflict. As a result, conventional editing methods struggle to detect such factual conflicts. We categorize this condition as reverse conflict:

$$\text{Reverse Conflict: } \begin{cases} E_1 = (s, r, o) \rightarrow (s, r, o_1) \\ E_2 = (o_1, r_r, s) \rightarrow (o_1, r_r, s_2) \end{cases} \quad (7)$$

Utilizing a KG, we propose a simple yet practical solution in Algorithm 2: during the editing process, we first query the relationship database. If the relationship is reversible, we construct the inverse relationship knowledge and insert both the original and reverse knowledge into the KG to check for conflicts. If no conflicts are detected, we then perform knowledge editing on the model with both the original and inverse relationships.

3.4.2 Knowledge Graph Augmentation. Li et al.[25] point out that editing a piece of knowledge can lead to the distortion of related parameterized knowledge within the language model. For example, after editing the knowledge of the United States President from Donald Trump to Joe Biden, the model might still answer “The First Lady of the United States is Melania Trump” (Trump’s wife) when asked about the First Lady. Enhancing the edited knowledge to maintain the original knowledge structure within the model

Algorithm 2 Resolution of Reverse Conflicts

```
1: Input: Knowledge Graph  $\mathcal{G}$ , Edit Triple  $(s, r, o)$ 
2: Output: Updated Model  $\mathcal{M}'$ , Updated  $\mathcal{G}'$ 
3: procedure REVERSERELATION( $\mathcal{M}, \mathcal{G}, (s, r, o)$ )
4:   if  $r$  in  $(s, r, o)$  is reversible then
5:     get reverse triple  $(o, r_r, s)$   $\triangleright r_r$  is reverse relation
6:     update  $(s, r, o)$  in  $\mathcal{G}$ 
7:     update  $(o, r_r, s)$  in  $\mathcal{G}$ 
8:     Proceed with the model edit to  $(s, r, o)$  and  $(o, r_r, s)$ 
9:   else
10:    update  $(s, r, o)$  in  $\mathcal{G}$ 
11:   end if
12:   return  $\mathcal{M}', \mathcal{G}'$ 
13: end procedure
```

can resolve such issues. However, previous methods have struggled to obtain knowledge closely related to the edited knowledge. We leverage a knowledge graph to search for n nodes centered around the edited subject as knowledge closely related to the edited knowledge.

Additionally, current major editing methods do not handle multi-hop questions well, resulting in weak logical reasoning abilities for the edited knowledge [7]. By using logical rules from the knowledge graph, we can address this limitation. Using logical rules, we determine whether these triples have a logical inference nature and expand them.

3.5 Editor

Even when performing knowledge editing on a 7B model, current major editing methods such as ROME, MEND, and GRACE require at least 30GB of VRAM and considerable editing time. In comparison, the memory overhead of storing model parameters after each edit is negligible. Based on this reality, we propose a space-for-time editing strategy, which involves storing the edit parameters after each knowledge editing. The edit parameters can then be utilized for subsequent edits or rollbacks, significantly reducing VRAM and time overhead.

$$\mathcal{M}' = \mathcal{M} + \sum_{i=0}^n \theta_i - \sum_{j=0}^m \theta_j \quad (8)$$

The Editor is divided into two parts: the editor and the cache. In the editor part, we use EasyEdit [44], which provides a rich set of knowledge editing methods and supports a wide range of models, meeting our needs for knowledge editing. In the edit cache part, we have integrated the edit cache into EasyEdit. During each edit, we generate a unique edit key based on the knowledge and its corresponding edit parameters. Specifically, when using methods like ROME that directly modify model parameters, we store the parameters before the edit and the difference after the edit for the edited layers. For methods like GRACE that are based on adapters, we record the hidden states after each edit as the edit parameters. If the scenarios of coverage and rollback mentioned in Section 3.4.1 occur, we can directly use the stored edit parameters from the cache to quickly apply the edits or rollbacks by adding or subtracting them. This approach can reduce VRAM and time overhead.

Table 1: Comparison of OneEdit to existing methods. Metrics shown are computed after all single edits. In OneEdit, we set the number of generation triples to 8

Method	American politicians dataset						Academic figures dataset					
	Reliability	Locality	Portability			Average	Reliability	Locality	Portability			Average
			Reverse	One-Hop	Sub-Replace				Reverse	One-Hop	Sub-Replace	
GPT-J-6B												
FT	0.825	0.008	0.240	0.034	0.620	0.339	0.571	0.008	0.610	0.000	0.620	0.362
ROME	0.996	0.982	0.235	0.176	0.828	0.634	0.994	0.982	0.561	0.175	0.906	0.724
MEMIT	1.000	0.997	0.581	0.402	0.554	0.736	0.996	0.997	0.665	0.055	0.799	0.702
GRACE	1.000	1.000	0.000	0.000	0.000	0.400	1.000	1.000	0.000	0.000	0.000	0.400
OneEdit (GRACE)	0.951	1.000	0.950	0.928	0.922	0.950	0.952	1.000	0.991	0.958	0.962	0.973
OneEdit (MEMIT)	0.995	0.947	0.957	0.713	0.952	0.913	0.982	0.933	0.987	0.722	0.842	0.865
Qwen2-7B												
FT	0.662	0.012	0.512	0.237	0.660	0.417	0.601	0.103	0.610	0.139	0.633	0.417
ROME	0.996	0.982	0.623	0.399	0.727	0.634	0.994	0.982	0.528	0.417	0.774	0.724
MEMIT	0.972	0.986	0.665	0.402	0.652	0.736	0.996	0.993	0.665	0.422	0.698	0.702
GRACE	1.000	1.000	0.000	0.000	0.000	0.400	1.000	1.000	0.000	0.000	0.000	0.400
OneEdit (GRACE)	0.953	1.000	0.962	0.958	0.943	0.963	0.961	1.000	0.958	0.951	0.949	0.964
OneEdit (MEMIT)	0.955	0.956	0.923	0.763	0.966	0.913	0.973	0.963	0.957	0.697	0.839	0.906

4 EXPERIMENTS

4.1 Experiment Setting

Models and Baselines. We choose GPT-j-6B and Qwen2-7B as our base model. GPT-j-6B is a transformer model trained by Mesh Transformer JAX. The Qwen2-7B model, newly released by Alibaba, incorporates training corpora comprising multiple languages including English and Chinese. Both Qwen2-7B and GPT-J-6B are based on the transformer architecture and use causal language modeling objectives for pre-training. The baselines include methods for continual learning and model editing. We compare OneEdit against direct fine-tuning (FT) with an additional KL divergence loss, as well as ROME, MEMIT, and GRACE [14, 27, 28].

Metrics. To evaluate the effectiveness of an editing method, we primarily consider three aspects: Reliability, Locality, and Portability. *Reliability*, as defined by Huang et al.[18], refers to a successful edit when the post-edit model \mathcal{M}' provides the edited target answer y' for the prompt p . Reliability is measured as the average accuracy on the edited cases:

$$\mathbb{E}_{p, y \sim \{(p, y)\}} \mathbb{1} \left\{ \arg \max_y \mathcal{M}'(y | p) = y' \right\} \quad (9)$$

Locality, also referred to as *Specificity* in some works, denotes that editing should be implemented locally. This means that the post-edit model \mathcal{M}' should not alter the output of irrelevant examples in the out-of-scope set $O(p, y)$. Therefore, locality is evaluated by the rate at which the post-edit model \mathcal{M}' 's predictions remain unchanged compared to the pre-edit model \mathcal{M} :

$$\mathbb{E}_{p, y \sim O(p, y)} \mathbb{1} \left\{ \mathcal{M}'(y | p) = \mathcal{M}(y | p) \right\} \quad (10)$$

Portability, as defined by Yao et al.[54], encompasses three components: Subject-Replace, One-hop, and Reverse. Portability is used to comprehensively evaluate the effectiveness of model editing in transferring knowledge to related content, termed robust generalization. Portability is calculated as the average accuracy of the

edited model \mathcal{M}' when applied to instances in $P(p, y)$:

$$\mathbb{E}_{p, y \sim P(p, y)} \mathbb{1} \left\{ \arg \max_y \mathcal{M}'(y | p) = y \right\} \quad (11)$$

4.2 Dataset and Knowledge Graph Construction

Our experiments necessitate the integration of knowledge editing datasets with knowledge graphs. Due to the current lack of comprehensive datasets that combine knowledge graphs with knowledge editing, we decided to construct our own dataset for the experiments. We utilize two specific domains to demonstrate the feasibility and generality of OneEdit: American political figures and Academic figures. Our experimental dataset is based on Wikidata, zsRE, and GPT-4. Specifically, we first extracted approximately 500 relevant entities from Wikidata and used the factual knowledge corresponding to these entities to construct an initial knowledge graph. To ensure the completeness and accuracy of the knowledge graph, we meticulously verified and optimized each entity and relations. Additionally, we expanded their neighboring nodes using Wikidata, resulting in a high-quality knowledge graph. After constructing the knowledge graph, we used it as the foundation for building our experimental dataset. For the editing task, to ensure that the new knowledge was being edited into the model and not already present from pre-training, our editing data consisted of counterfactual information, which is opposite to the factual knowledge. We created counterfactual knowledge by replacing the ground truth object o_t in the knowledge triples (s, r, o_t) with a new object o_n , and used manually written templates. This method ensures a high degree of consistency and relevance between the dataset and the knowledge graph.

4.3 Single-user Knowledge Editing Results

Our primary experimental conclusions are presented in Table 1. We observe that OneEdit performs comparably to other state-of-the-art methods in terms of reliability. When it comes to locality, OneEdit shows significant improvement over methods like ROME

Table 2: Comparison of OneEdit to existing methods in a multi-user scenario. User=2 indicates that a single piece of knowledge has been edited twice, once by each of two users. Similarly, when the number of users is 3, the knowledge has been edited three times, once by each user.

Method	Reliability	Locality	Portability			Average
			Reverse	One-Hop	Sub-Replace	
GPT-J-6B, Users = 2						
FT	0.902	0.082	0.249	0.000	0.590	0.365
ROME	0.987	0.040	0.156	0.067	0.629	0.376
MEMIT	0.998	0.421	0.156	0.399	0.526	0.500
GRACE	1.000	1.000	0.000	0.000	0.000	0.400
OneEdit (GRACE)	0.952	1.000	0.802	0.730	0.824	0.862
OneEdit (MEMIT)	0.833	0.989	0.849	0.706	0.842	0.844
Qwen2-7B, Users = 2						
FT	0.691	0.275	0.168	0.114	0.121	0.274
ROME	0.981	0.641	0.392	0.283	0.706	0.601
MEMIT	0.972	0.998	0.560	0.412	0.503	0.689
GRACE	1.000	1.000	0.000	0.000	0.000	0.400
OneEdit (GRACE)	0.952	1.000	0.892	0.729	0.918	0.898
OneEdit (MEMIT)	0.851	0.993	0.862	0.671	0.859	0.816
GPT-J-6B, Users = 3						
FT	0.850	0.059	0.210	0.000	0.588	0.341
ROME	0.988	0.006	0.122	0.003	0.608	0.345
MEMIT	0.987	0.328	0.544	0.380	0.523	0.552
GRACE	0.999	1.000	0.000	0.000	0.000	0.399
OneEdit (GRACE)	0.952	1.000	0.802	0.850	0.824	0.886
OneEdit (MEMIT)	0.832	0.983	0.856	0.699	0.852	0.844
Qwen2-7B, Users = 3						
FT	0.797	0.210	0.165	0.102	0.120	0.279
ROME	0.999	0.483	0.266	0.215	0.701	0.533
MEMIT	0.999	0.998	0.544	0.380	0.593	0.703
GRACE	0.961	1.000	0.000	0.000	0.000	0.392
OneEdit (GRACE)	0.951	1.000	0.892	0.850	0.918	0.922
OneEdit (MEMIT)	0.852	0.994	0.859	0.716	0.862	0.833

and FT, which directly modify model parameters. It performs similarly to GRACE, a method known for high locality in model editing. This is likely because OneEdit, like GRACE, does not alter the model’s parameters, maintaining the original model’s performance on unrelated queries. In terms of portability, OneEdit surpasses our baselines across all three sub-metrics, which is the key contribution of our paper. For the Reverse metric, we enhance the model’s understanding of original knowledge by incorporating automatically constructed inverse relationship knowledge edits. In the one-hop metric, our method also shows a significant advantage. By directly writing multi-hop knowledge into the model parameters with KG constraints, our method enhances the model’s understanding of multi-hop knowledge.

4.4 Multi-user Knowledge Editing Results

In this paper, we aim to simulate the scenario where multiple users collaboratively update the model. To achieve this, we adopt the sequential editing setup to simulate the process of multiple users

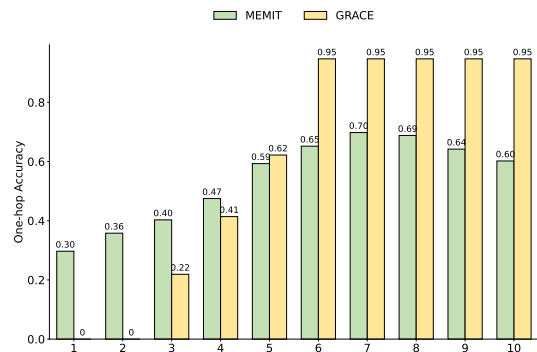


Figure 3: The variation of one-hop metrics with changes in generation triples in GPT-J-6B.

updating the model [18, 58]. Specifically, we consider how to resolve conflicts when multiple users sequentially edit the same piece

Table 3: Time and Memory (VRAM) Overhead for Different Models and Methods

Model	OneEdit (MEMIT)	MEMIT, Users = 2	MEMIT, Users = 3	OneEdit (GRACE)	GRACE, Users = 2	GRACE, Users = 3
GPT-2 XL						
Time Overhead (s)	8	14	20	10	18	25
VRAM Overhead (GB)	10	4	4	12	6	6
GPT-J 6B						
Time Overhead (s)	10	18	26	21	38	55
VRAM Overhead (GB)	30	25	25	29	23	23
Qwen2 7B						
Time Overhead (s)	12	20	29	22	40	59
VRAM Overhead (GB)	38	32	32	33	27	27

of knowledge to different outcomes. Our experiments focus on scenarios with two and three users, as these situations are most commonly encountered in real-world applications. As shown in Table 2, we observe that FT and ROME, the results about locality decline as the number of users increases. We believe this is due to the limitations of methods that require modifying model parameters in a sequential edit scenario. Each edit introduces limited information and significantly changes the model parameters, inevitably leading to a decline in the model’s fundamental performance. Although OneEdit performs well, we believe there is still room for improvement. Its current effectiveness is primarily constrained by the extraction performance of the Interpreter and the coverage of KG. When the KG does not cover the relevant knowledge for portability, or when the Interpreter incorrectly extracts knowledge triples, the performance of OneEdit decreases.

4.5 Knowledge Graph Augmentation Analysis

In OneEdit, the most critical parameter is the number of generation triples passed from the Controller to the Editor. However, OneEdit employs a nearest-neighbor strategy for selecting generation triples, which may lead to the exclusion of desired generalized knowledge, especially in dense knowledge graphs. This can result in multi-hop inference triples responsible for the edited knowledge being excluded from the generalized triples, causing a decrease in the One-Hop metric. In this section, we conduct experiemnts with varying the number of knowledge augmentation triples n to observe the resulting changes in the One-Hop metric for OneEdit (GRACE) and OneEdit (MEMIT) on the GPT-J-6B model.

As shown in Figure 3, we observe that when n is small, both OneEdit (GRACE) and OneEdit (MEMIT) underperform compared to their respective original methods, GRACE and MEMIT. We hypothesize that this discrepancy is due to the loss incurred during the conversion to triples. As n increases, multiple inference triples are incorporated into the edited sequence, and the values of both OneEdit (GRACE) and OneEdit (MEMIT) rise. However, when n becomes large, the results of OneEdit (GRACE) plateau, while the results of OneEdit (MEMIT) decline. We attribute this to the fact that GRACE has stricter rules for recalling the edited knowledge, whereas MEMIT’s batch edit struggles to accurately recall the necessary edited knowledge from the extensive edited knowledge base.

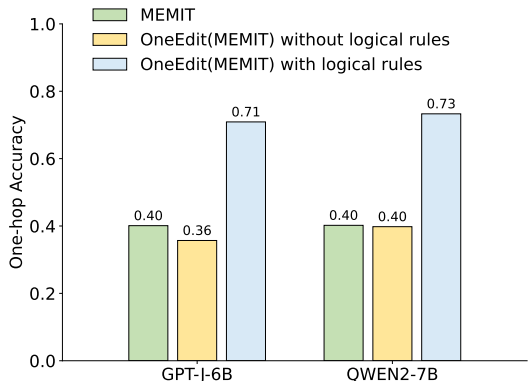


Figure 4: The impact of adding logical rules on the One-Hop results in OneEdit.

4.6 Logical Rules Analysis

In Section 3.4.2, we have discussed how to obtain knowledge augmentation triples for the edited knowledge and then perform logical augmentation based on the semantic rules of their relationships. This process expands the triples and enhances the model’s logical reasoning ability for the edited knowledge. To evaluate the model’s logical reasoning ability, the most important metric is One-Hop, which assesses whether the model can use the newly edited knowledge to answer multi-hop reasoning questions related to the edited knowledge.

In this section, we demonstrate the effectiveness of adding logical rules in knowledge editing by comparing the One-Hop metric results with and without logical rules. Experimental results on Qwen2-7B and GPT-J-6B indicate that the model has poor logical generalization ability for edited knowledge, merely memorizing the edited knowledge mechanically without proper utilization. However, as shown in Figure 4, after adding logical rules and leveraging the logical reasoning advantages of symbolic knowledge to assist in modifying model parameters, the results significantly improve, with GPT-J-6B and Qwen2-7B showing improvements. This demonstrates that symbolic logical rules enhance the model’s ability to utilize the edited knowledge effectively.

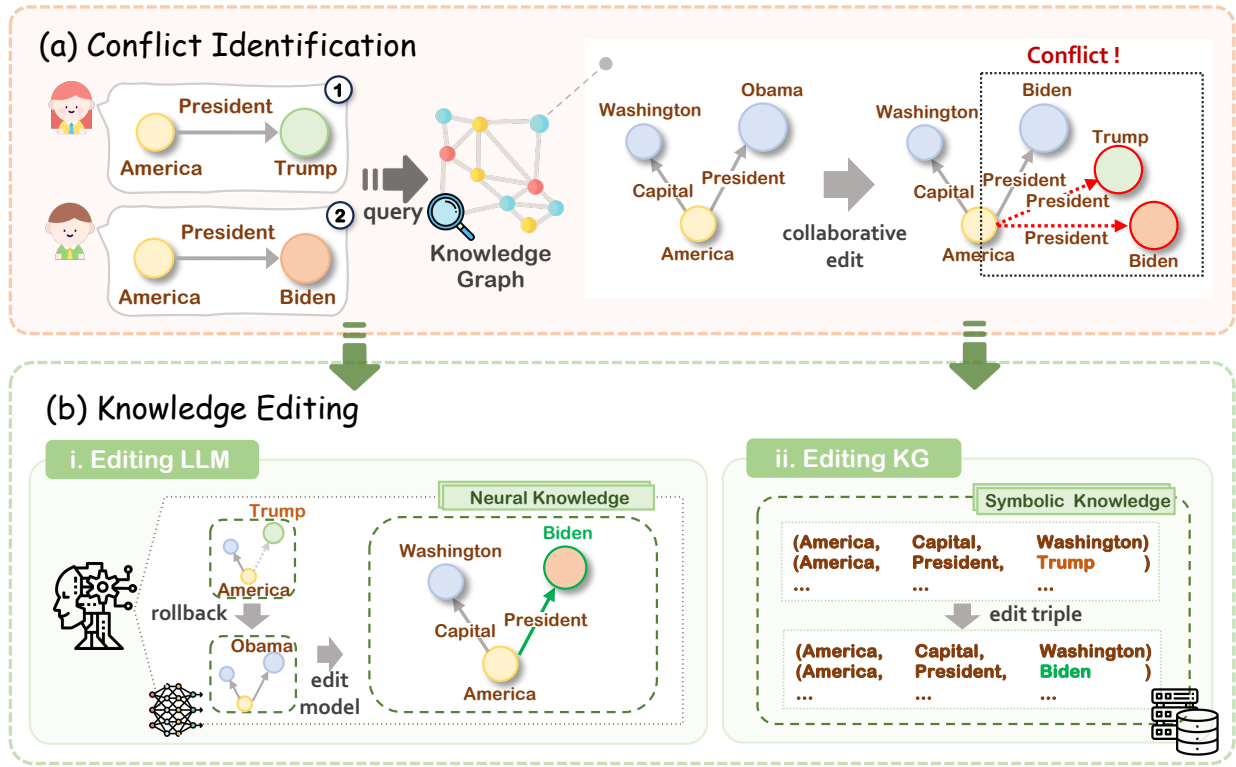


Figure 5: A coverage conflict within OneEdit: OneEdit first rolls back the conflicting knowledge before editing the new knowledge. Without using OneEdit, previous edited knowledge may remain.

4.7 Computation Resource Analysis

In this section, we analyze the average time and memory overhead per edit associated with OneEdit. As shown in Table 3, experiments involving GPT-2 XL were conducted on an NVIDIA 3090, with the interpreter and editor deployed on separate GPUs. For GPT-J-6B and Qwen2-7B, experiments are similarly performed on an A800 machine, with both the interpreter and editor allocated to different GPUs. Compared to MEMIT and GRACE, OneEdit (MEMIT) and OneEdit (GRACE) require approximately 6GB of additional memory. This increase is primarily due to the memory overhead introduced by the interpreter. Regarding time overhead, we assess the editing duration with configurations of two and three users. Given the negligible time required for OneEdit’s rollback process, we focus solely on the time needed for a single edit. Our observations indicate that in scenarios with two and three users, OneEdit achieve a 40% and 70% reduction in time, respectively. This improvement is attributed to OneEdit’s rollback mechanism, which enables the reuse of previous edits when repeatedly modifying the same piece of knowledge.

4.8 Case Study

In this part, we present two cases within OneEdit to specifically illustrate how our system addresses the two types of conflicts mentioned in Section 3.4

4.8.1 Coverage Conflict Case. In the context of the coverage scenario, we present a real-world example: following the 2020 U.S. presidential election, the president changed from Trump to Biden. In the controller, we removed the knowledge parameter indicating “the U.S. president is Trump” and updated it with the new information that “the U.S. president is Biden.” However, if Trump were to win the election again in 2024, we could directly revert the knowledge that “Trump is the U.S. president” back into the model. This approach reduces the number of edits required to the model and maintains its baseline performance.

4.8.2 Reverse Conflict Case. In the reverse scenario, we also present a real-world example: Donald Trump divorced his wife Ivana Trump, who subsequently married Ricardo Mazzuchelli. If we only edit the model to reflect that “Ivana Trump’s husband is Ricardo Mazzuchelli”, it would lead to the absurd situation where “Ivana Trump’s husband is Ricardo Mazzuchelli” while “Donald Trump’s wife is Ivana Trump”. However, the OneEdit controller module automatically constructs inverse relationships for such reversible scenarios. When “Ivana Trump’s husband is Ricardo Mazzuchelli” reappears, it conflicts with the automatically constructed relationship “Ivana Trump’s husband is Donald Trump” in the controller. This prompts the model to roll back the outdated knowledge and correctly update the inverse relationship of the new knowledge.

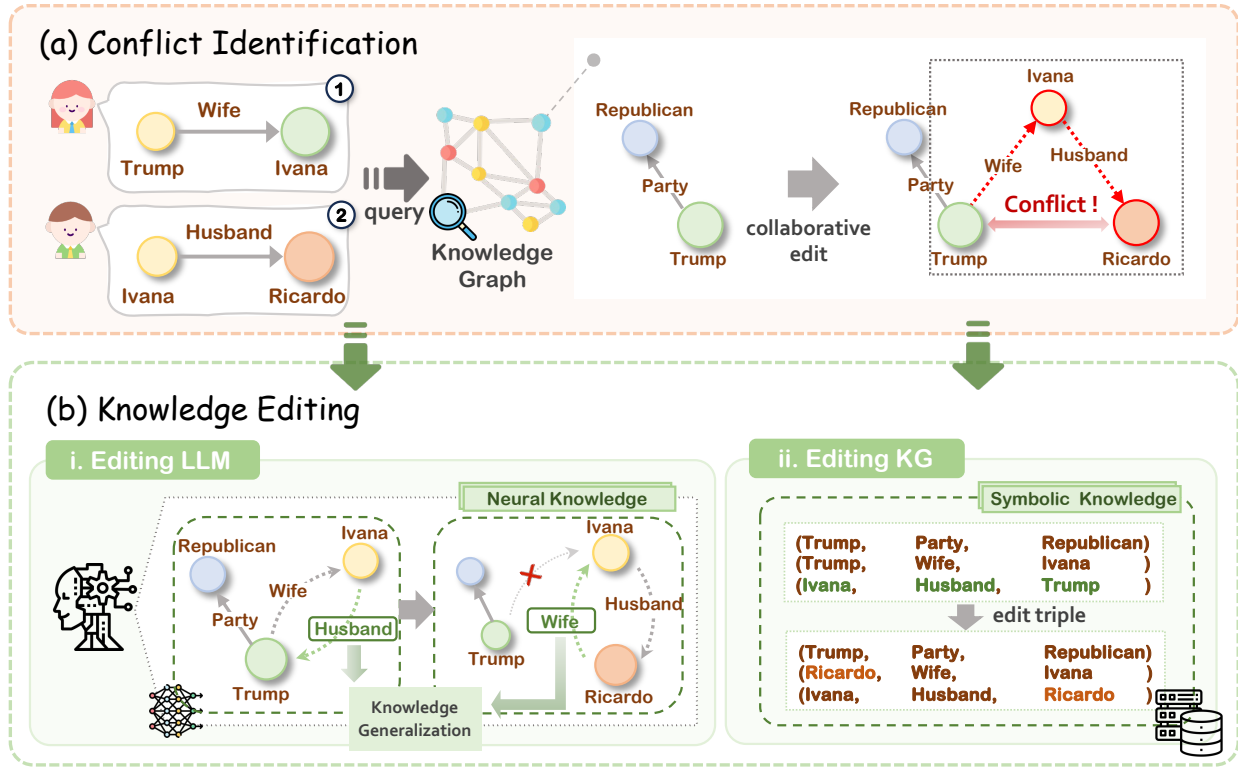


Figure 6: A reverse conflict within OneEdit: OneEdit automatically constructs reverse relationship knowledge. When we edit reverse relationships knowledge, conflicts arise in the KG.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose OneEdit, a neural-symbolic knowledge editing system that continuously updates symbolic knowledge in KG and the neural knowledge in LLM. OneEdit can advance domain-specific knowledge injection and alignment between the human semantic space and the latent semantic space of LLMs. In the future, we will extend the application scope of OneEdit to encompass a broader range of methods.

6 LIMITATIONS

Our work is still quite preliminary and has the following limitations: First, due to current computational power limitations, this system has only been tested on small pre-trained language models and small-scale KGs, which inevitably may affect its general capabilities. Currently, the natural language instructions that can be recognized are also quite limited. This system **OneEdit is merely a prototype and there is significant room for improvement in the future**. Additionally, we have only considered factual knowledge, with no support for commonsense, multimodal data, etc., and there is still substantial room for improvement in generalization capabilities. Furthermore, the system’s security is still at a rudimentary stage, and measures to prevent malicious misuse that could lead to model tampering will need to be developed in the future.

ACKNOWLEDGEMENTS

We would like to express gratitude to the anonymous reviewers for their kind comments. This work was supported by the National Natural Science Foundation of China (No. 62206246, No. NSFCU23B2055, No. NSFCU19B2027), the Fundamental Research Funds for the Central Universities (226-2023-00138), Zhejiang Provincial Natural Science Foundation of China (No. LGG22F030011), Yongjiang Talent Introduction Programme (2021A-156-G), CCF-Tencent Rhino-Bird Open Research Fund, Information Technology Center and State Key Lab of CAD&CG, Zhejiang University, and NUS-NCS Joint Laboratory (A-0008542-00-00).

REFERENCES

- [1] Badr Alkhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A Review on Language Models as Knowledge Bases. arXiv:2204.06031 [cs.CL]
- [2] Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2024. The Reversal Curse: LLMs trained on "A is B" fail to learn "B is A". arXiv:2309.12288 [cs.CL]
- [3] 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. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL]

- [4] Yichao Cai, Qingyu Yang, Wei Chen, Ge Wang, Taian Liu, and Xinying Liu. 2022. A new Knowledge Inference Approach Based on Multi-headed Attention Mechanism. *2022 IEEE 6th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)* (2022), 825–829. <https://api.semanticscholar.org/CorpusID:253423978>
- [5] Canyu Chen, Baixiang Huang, Zekun Li, Zhaorun Chen, Shiyang Lai, Xiong Xiao Xu, Jia-Chen Gu, Jindong Gu, Huaxiu Yao, Chaowei Xiao, et al. 2024. Can Editing LLMs Inject Harm? *arXiv preprint arXiv:2407.20224* (2024).
- [6] Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. KnowPrompt: Knowledge-aware Prompt-tuning with Synergistic Optimization for Relation Extraction. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*. ACM. <https://doi.org/10.1145/3485447.3511998>
- [7] Keyuan Cheng, Gang Lin, Haoyang Fei, Yuxuan zhai, Lu Yu, Muhammad Asif Ali, Lijie Hu, and Di Wang. 2024. Multi-hop Question Answering under Temporal Knowledge Editing. *arXiv:2404.00492* [cs.CL] <https://arxiv.org/abs/2404.00492>
- [8] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-Verification Reduces Hallucination in Large Language Models. *arXiv:2309.11495* [cs.CL]
- [9] Wenqi Fan, Shijie Wang, Jiani Huang, Zhikai Chen, Yu Song, Wenzhuo Tang, Haitao Mao, Hui Liu, Xiaorui Liu, Dawei Yin, and Qing Li. 2024. Graph Machine Learning in the Era of Large Language Models (LLMs). *arXiv:2404.14928* [cs.LG]
- [10] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv:2312.10997* [cs.CL]
- [11] Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing can hurt general abilities of large language models. *arXiv preprint arXiv:2401.04700* (2024).
- [12] Tiezheng Guo, Qingwen Yang, Chen Wang, Yanyi Liu, Pan Li, Jiawei Tang, Dapeng Li, and Yingyou Wen. 2024. KnowledgeNavigator: Leveraging Large Language Models for Enhanced Reasoning over Knowledge Graph. *arXiv:2312.15880* [cs.CL]
- [13] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. 2021. Pre-trained models: Past, present and future. *AI Open* 2 (2021), 225–250. <https://doi.org/10.1016/j.aiopen.2021.08.002>
- [14] Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with GRACE: Lifelong Model Editing with Discrete Key-Value Adaptors. In *Advances in Neural Information Processing Systems*.
- [15] Peter Hase, Thomas Hofweber, Xiang Zhou, Elias Stengel-Eskin, and Mohit Bansal. 2024. Fundamental Problems With Model Editing: How Should Rational Belief Revision Work in LLMs? *arXiv preprint arXiv:2406.19354* (2024).
- [16] Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. WilKE: Wise-Layer Knowledge Editor for Lifelong Knowledge Editing. *arXiv:2402.10987* [cs.CL]
- [17] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. 2024. MiniCPM: Unveiling the Potential of Small Language Models with Scalable Training Strategies. *arXiv preprint arXiv:2404.06395* (2024).
- [18] Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-Patcher: One Mistake worth One Neuron.
- [19] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Martinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems* 33, 2 (2021), 494–514.
- [20] Zhuoran Jin, Pengfei Cao, Hongbang Yuan, Yubo Chen, Jiexin Xu, Huajun Li, Xiaojian Jiang, Kang Liu, and Jun Zhao. 2024. Cutting Off the Head Ends the Conflict: A Mechanism for Interpreting and Mitigating Knowledge Conflicts in Language Models. *CoRR* abs/2402.18154 (2024). <https://doi.org/10.48550/ARXIV.2402.18154> *arXiv:2402.18154*
- [21] Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A Survey on Complex Knowledge Base Question Answering: Methods, Challenges and Solutions. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, Zhi-Hua Zhou (Ed.). ijcai.org, 4483–4491. <https://doi.org/10.24963/ijcai.2021/611>
- [22] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2022. A Survey on Deep Learning for Named Entity Recognition. *IEEE Transactions on Knowledge and Data Engineering* 34, 1 (Jan. 2022), 50–70. <https://doi.org/10.1109/tkde.2020.2981314>
- [23] Yanzhou Li, Tianlin Li, Kangjie Chen, Jian Zhang, Shangqing Liu, Wenhan Wang, Tianwei Zhang, and Yang Liu. 2024. BadEdit: Backdooring large language models by model editing. *CoRR* abs/2403.13355 (2024). <https://doi.org/10.48550/ARXIV.2403.13355> *arXiv:2403.13355*
- [24] Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2024. FlexKBQA: A Flexible LLM-Powered Framework for Few-Shot Knowledge Base Question Answering. *arXiv:2308.12060* [cs.CL]
- [25] Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. 2024. Unveiling the pitfalls of knowledge editing for large language models. *The Eleventh International Conference on Learning Representations (ICLR)* (2024).
- [26] Yiheng Liu, Hao He, Tianle Han, Xu Zhang, Mengyuan Liu, Jiaming Tian, Yutong Zhang, Jiaqi Wang, Xiaohui Gao, Tianyang Zhong, Yi Pan, Shaochen Xu, Zihao Wu, Zhengliang Liu, Xin Zhang, Shu Zhang, Xintao Hu, Tuo Zhang, Ning Qiang, Tianming Liu, and Bao Ge. 2024. Understanding LLMs: A Comprehensive Overview from Training to Inference. *arXiv:2401.02038* [cs.CL]
- [27] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and Editing Factual Associations in GPT. *Advances in Neural Information Processing Systems* 35 (2022).
- [28] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229* (2022).
- [29] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast Model Editing at Scale. In *International Conference on Learning Representations*. <https://openreview.net/pdf?id=0DcZxeWFOpT>
- [30] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022. Memory-Based Model Editing at Scale. In *International Conference on Machine Learning*. <https://arxiv.org/pdf/2206.06520.pdf>
- [31] OpenAI and the Co-authors. 2024. GPT-4 Technical Report. *arXiv:2303.08774* [cs.CL]
- [32] Jeff Z Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyenko, Wen Zhang, Matteo Lissandrini, et al. 2023. Large language models and knowledge graphs: Opportunities and challenges. *arXiv preprint arXiv:2308.06374* (2023).
- [33] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering* 36, 7 (July 2024), 3580–3599. <https://doi.org/10.1109/tkde.2024.3352100>
- [34] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering* (2024), 1–20. <https://doi.org/10.1109/tkde.2024.3352100>
- [35] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language Models as Knowledge Bases?. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 2463–2473. <https://doi.org/10.18653/v1/D19-1250>
- [36] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language Models as Knowledge Bases? *arXiv:1909.01066* [cs.CL] <https://arxiv.org/abs/1909.01066>
- [37] Domenic Rosati, Robie Gonzales, Jinkun Chen, Xuemin Yu, Melis Erkan, Yahya Kayani, Satya Deepika Chavatapalli, Frank Rudzicz, and Hassan Sajjad. 2024. Long-form evaluation of model editing. *arXiv preprint arXiv:2402.09394* (2024).
- [38] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. *arXiv:2402.07927* [cs.AI]
- [39] Kai Sun, Yifan Ethan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. 2024. Head-to-Tail: How Knowledgeable are Large Language Models (LLMs)? A.K.A. Will LLMs Replace Knowledge Graphs? *arXiv:2308.10168* [cs.CL] <https://arxiv.org/abs/2308.10168>
- [40] Chenmian Tan, Ge Zhang, and Jie Fu. 2024. Massive Editing for Large Language Models via Meta Learning. In *International Conference on Learning Representations*. <https://openreview.net/pdf?id=L6L1CJQ2PE>
- [41] Bozhong Tian, Siyuan Cheng, Xiaozhuan Liang, Ningyu Zhang, Yi Hu, Kouying Xue, Yanjie Gou, Xi Chen, and Huajun Chen. 2024. InstructEdit: Instruction-based Knowledge Editing for Large Language Models. *International Joint Conference on Artificial Intelligence* (2024).
- [42] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv:2302.13971* [cs.CL]
- [43] Ke Tu, Peng Cui, Daixin Wang, Zhiqiang Zhang, Jun Zhou, Yuan Qi, and Wenwu Zhu. 2021. Conditional graph attention networks for distilling and refining knowledge graphs in recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1834–1843.
- [44] 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, and Huajun Chen. 2024. EasyEdit: An Easy-to-use Knowledge Editing Framework for Large Language Models. *arXiv:2308.07269* [cs.CL] <https://arxiv.org/abs/2308.07269>
- [45] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*.

- 950–958.
- [46] Yuxiang Wang, Arijit Khan, Tianxing Wu, Jiahui Jin, and Haijiang Yan. 2020. Semantic guided and response times bounded top-k similarity search over knowledge graphs. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 445–456.
- [47] Zihao Wei, Liang Pang, Hanxing Ding, Jingcheng Deng, Huawei Shen, and Xueqi Cheng. 2024. Stable Knowledge Editing in Large Language Models. *CoRR abs/2402.13048* (2024). <https://doi.org/10.48550/ARXIV.2402.13048> arXiv:2402.13048
- [48] Lionel Wong, Gabriel Grand, Alexander K. Lew, Noah D. Goodman, Vikash K. Mansinghka, Jacob Andreas, and Joshua B. Tenenbaum. 2023. From Word Models to World Models: Translating from Natural Language to the Probabilistic Language of Thought. *CoRR abs/2306.12672* (2023). <https://doi.org/10.48550/ARXIV.2306.12672> arXiv:2306.12672
- [49] Tianxing Wu, Arijit Khan, Melvin Yong, Guilin Qi, and Meng Wang. 2022. Efficiently embedding dynamic knowledge graphs. *Knowledge-Based Systems* 250 (2022), 109124.
- [50] Tongtong Wu, Xuekai Li, Yuan-Fang Li, Gholamreza Haffari, Guilin Qi, Yujin Zhu, and Guoqiang Xu. 2021. Curriculum-meta learning for order-robust continual relation extraction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 10363–10369.
- [51] Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. 2024. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364* (2024).
- [52] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 564–573.
- [53] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. 2023. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. arXiv:2304.13712 [cs.CL]
- [54] Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing Large Language Models: Problems, Methods, and Opportunities. *Conference on Empirical Methods in Natural Language Processing (2023)*.
- [55] Paul Youssef, Zhixue Zhao, Jörg Schlötterer, and Christin Seifert. 2024. Detecting Edited Knowledge in Language Models. arXiv:2405.02765 [cs.CL]
- [56] Ningyu Zhang, Zhen Bi, Xiaozhuan Liang, Siyuan Cheng, Haosen Hong, Shumin Deng, Qiang Zhang, Jiazhang Lian, and Huajun Chen. 2021. OntoProtein: Protein Pretraining With Gene Ontology Embedding. In *International Conference on Learning Representations*.
- [57] Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024. A Comprehensive Study of Knowledge Editing for Large Language Models. *arXiv preprint arXiv:2401.01286* (2024).
- [58] Jiamu Zheng, Jinghui Zhang, Futing Wang, Tianyu Du, and Tao Lin. [n.d.]. ColabEdit: Towards Non-destructive Collaborative Knowledge Editing. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*.
- [59] Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention.. In *IJCAI*. 4623–4629.