

## Appendix

### A. Limitations and Broader Impacts

The limitation of our method is that it can only be applied to LLMs designed for next-token prediction, which makes it unable to adapt to the knowledge editing of LLMs of all architectures. For knowledge editing tasks with strong professionalism in a specific field, during the graph construction process of SGR, it is necessary to use the LLM corresponding to the specific field to guide the graph construction and ensure the accuracy and professionalism of the knowledge graph construction.

In addition, due to the lack of consideration of the authenticity of unstructured text in the framework, some malicious people may use it to incorrectly edit the model to spread false views such as discrimination and prejudice. We suggest that in the future, more supervision should be introduced on the correctness of all texts and other knowledge sources before editing to ensure the reliability of the edited models in terms of ethics and social responsibility.

### B. Implementation Details

#### B.1 Editing Details

In the SGR module, for language models capable of accurately following graph-construction instructions (e.g., Llama3-8B), the editing process is directly performed by the model itself. For models with limited instruction-following and generation capabilities (e.g., GPT2-XL), auxiliary assistance is provided by more capable large language models, such as GPT-4. All training and evaluation procedures are conducted on NVIDIA A800 80GB GPUs. Regarding the editing layer of the model, for GPT2-XL, we select “transformer.h.36.mlp.c\_fc” and “transformer.h.37.mlp.c\_fc” for editing. For llama3.2-3b and llama3-8b, we select “model.layers.13.mlp.down\_proj” for editing. During the editing training phase across all experiments, the learning rate is set to 0.001, and the maximum number of iterations is fixed at 50. For experiments involving LoRA-based methods, the rank of each individual LoRA adapter is set to 8.

#### B.2 Detailed Instructions in SGR

To support the Semantic-augmented Graph Representation (SGR) module, we define three instruction formats to guide large language models in constructing and enriching unstructured knowledge graphs. Each instruction is accompanied by an in-context example to clarify its intent and output format, followed by an additional example to illustrate generalization.

#### 1. Structure-aware Graph Construction

##### Instruction of Structure-aware Graph Construction with Example

**Task:** Extract an interconnected knowledge graph from the following paragraph. Each triple should represent a clear semantic relation between two entities explicitly stated in the paragraph. Preserve the contextual semantics. Output as a list of triples in the form: (Entity1, Relation, Entity2).

##### Example Input/Output:

##### Input Paragraph:

"Alan Turing developed the concept of a Turing machine, which became the foundation of computer science. He worked at Bletchley Park during World War II."

##### Output Triples:

- (Alan Turing, developed, concept of a Turing machine)
- (Turing machine, became the foundation of, computer science)
- (Alan Turing, worked at, Bletchley Park)
- (Bletchley Park, active during, World War II)

## 2. Multi-hop Relational Enrichment

### Instruction of Multi-hop Relational Enrichment with Example

**Task:** Given a knowledge graph extracted from a paragraph and two nodes, infer and describe the semantic relation that connects them indirectly via multi-hop reasoning. Ensure the relation reflects coherent real-world semantics grounded in the original text.

**Example Input/Output:**

**Input Graph:**

- (Marie Curie, discovered, radium)
- (radium, used in, cancer treatment)

**Text:**

"Marie Curie discovered the radioactive element radium, which later found applications in cancer therapy."

**Query:** (Marie Curie, ?, cancer treatment)

**Output:** (Marie Curie, discovered substance used in, cancer treatment)

## 3. Paraphrase-driven Semantic Diversification

### Instruction of Paraphrase-driven Semantic Diversification with Example

**Task:** Given a knowledge graph with triples, rewrite each relation in a semantically equivalent but lexically different way to diversify the expression of the knowledge. Keep entity names unchanged. Return paraphrased triples.

**Example Input/Output:**

**Input Triple:** (Nikola Tesla, invented, alternating current)

**Output Paraphrases:**

- (Nikola Tesla, was the inventor of, alternating current)
- (Nikola Tesla, came up with, alternating current)
- (Nikola Tesla, pioneered, alternating current)

### B.3 Details of the activation-similarity routing mechanism

In our framework, the activation-similarity routing mechanism is not a central design component, as the underlying principle is relatively conventional. Consequently, it is not emphasized in the main body of the paper. The activation-similarity routing employs a hard activation-based routing strategy to determine whether an input instance falls within the scope of previously edited knowledge. Specifically, during the editing phase, we preserve low-bit lightweight activation vectors corresponding to each edited knowledge item. During inference, the cosine similarity between the input prompt and the stored activation representations is computed. If the similarity exceeds a predefined threshold (set to 0.7 in our experiments), the input is deemed relevant to the edited knowledge, and the corresponding LoRA module is activated for inference. Otherwise, the model relies solely on its original parameters without invoking any edited components.

## C. Descriptions of Compared Model Editors

We compare our method with five representative model editing approaches. Below we summarize their core mechanisms and editing properties.

**ROME** [8]: ROME identifies the internal location of factual associations in LLMs using causal tracing, assuming that MLP layers are the main knowledge carriers. It edits the value matrix of a

single MLP layer using a closed-form rank-one update derived from least squares regression. ROME injects one fact per iteration, enforcing precision through a Lagrangian remainder, but lacks scalability in multi-edit or lifelong scenarios.

**MELO** [18]: MELO assumes that attention heads store factual knowledge and adopts a discrete adapter allocation strategy. It assigns a dedicated LoRA adapter to each knowledge edit, enabling isolation and preventing interference. Although effective in reliability, its discrete mapping is brittle to paraphrased or slightly varied inputs, making it less robust for generalization in lifelong settings.

**MEMIT** [9]: MEMIT builds on ROME’s assumptions, treating FFNs as knowledge key-value stores, and extends editing to multiple layers and multiple facts simultaneously. It directly modifies FFN parameters using batched least-squares updates, supporting thousands of edits. However, edits accumulate in the same parameter space, which can cause interference and poor locality during lifelong updates.

**WISE** [7]: WISE introduces a dual parametric memory mechanism to bridge the gap between long-term memory (model parameters) and working memory (retrieval-based representations). It edits only the side memory—a copied FFN value matrix—and routes queries between main and side memory based on an activation-based routing mechanism. To support lifelong editing, WISE uses knowledge sharding into orthogonal subspaces and merges them via Ties-Merge, achieving high reliability, locality, and generalization even with thousands of edits.

**ELDER** [19]: ELDER enhances lifelong model editing by adopting a Mixture-of-LoRA (MoL) approach, replacing discrete adapter selection with a continuous, learnable router that adaptively combines top- $k$  LoRA modules per input. This enables robust handling of semantically equivalent paraphrases. A guided loss aligns edit knowledge with LoRA allocations, while a deferral mechanism detects whether an input needs editing, allowing fallback to the original model for general tasks. ELDER avoids the scalability bottleneck of discrete adapter assignment and maintains strong generalization and reliability under long edit sequences.

## D. More Experiments

### D.1 Editing Efficiency

To evaluate the training efficiency of the proposed editing framework, we compared its per-sample editing time with that of several representative baseline methods. As shown in Table 3, the proposed method demonstrates superior efficiency during the editing process. Compared with post-hoc editing approaches such as ROME and MEMIT, the isolated low-rank parameter training performed by the LoRA mechanism achieves convergence with substantially lower computational cost. Moreover, in comparison with methods such as MELO and Elder, our approach benefits from the coloring strategy based on the Welsh–Powell algorithm in the CKE module, which eliminates the need for additional training. Consequently, it is more efficient than both the expert-selection gating mechanism of Elder, which requires extra training, and the MELO method, which entails extensive vector data access. These results collectively indicate that the proposed method achieves notable advantages in editing efficiency compared with existing model editing approaches.

Table 3: Comparison of average editing time per sample (in seconds). Tested on Llama3-8b from the dataset MQuAKE-CF.

Method	ROME	MELO	MEMIT	WISE	ELDER	Ours
Time (s)	78.64	52.76	77.23	143.91	54.17	<b>49.31</b>

### D.2 Analysis of threshold in activation-similarity routing

The Figure 5 below illustrates the effect of the similarity threshold in activation-similarity routing on the CounterFact dataset using Llama3.2. As the threshold decreases, editing reliability tends to decline, while locality improves. This trend arises because a lower threshold enforces a stricter

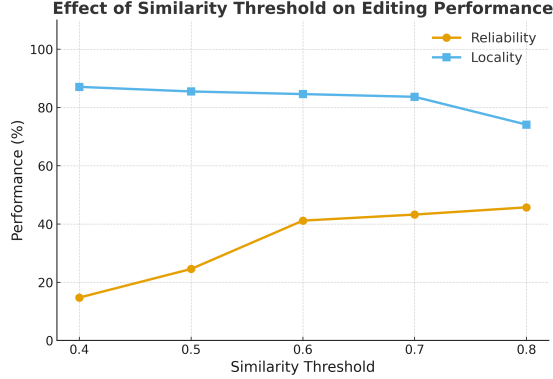


Figure 5: Performance at different similarity thresholds in activation-similarity routing.

criterion for LoRA expert activation, leading the model to depend more heavily on its unedited parameters during inference.

### D.3 Evaluation of the Initial Knowledge Graph Quality in SGR

The quality of the initial knowledge graph is fundamental to CAKE’s success. Through manual evaluation, we verified that CAKE’s initial knowledge graph accurately and comprehensively covers nearly all knowledge entities present in the original unstructured passage. We report the corresponding evaluation results in the Table 4. We conducted a human evaluation from two perspectives: extraction accuracy and knowledge completeness. As can be seen, CAKE’s IKG achieves near-perfect coverage of the entities and relations present in the raw text. This strong performance mainly stems from (i) CAKE’s carefully designed, task-specific prompting rules and (ii) the capabilities of the employed LLM (for models with limited intrinsic extraction capacity, such as GPT2-XL, we leverage the GPT-4 API for IKG construction).

Table 4: Evaluation of the Initial Knowledge Graph Quality in SGR.

Datasets	knowledge completeness	extraction accuracy
CounterFact	0.99	0.95
MQuAKE-CF	0.99	0.97
WikiUpdate	0.93	0.95

### D.4 Detailed Ablation of SGR

Table 5: Detailed Ablation of SGR. Each component within the SGR module yields a distinct improvement in editing performance.

Method	w/o SGR	w/o Multi-hop Enrichment	w/o Paraphrasing	CAKE
Time (s)	38.38	39.78	41.66	<b>43.24</b>

To further verify that each semantic enhancement component within the SGR module contributes meaningfully to the overall editing performance, we conducted ablation studies on the Multi-hop Relational Enrichment and Paraphrase-driven Semantic Diversification stages. These experiments were designed to examine how representational ambiguity influences the model’s editing capability. All evaluations were performed on the CounterFact dataset using the Llama3.2-3B model, and the results are summarized in Table 5. As shown, each component within the SGR module yields a distinct improvement in editing performance, confirming that the proposed Multi-hop Relational Enrichment and Paraphrase-driven Semantic Diversification mechanisms effectively enhance the model’s ability to capture diverse relational structures and achieve a more comprehensive representation of factual knowledge contained in unstructured text.

## E. End-to-End Case Study of CAKE Framework

Below I will construct a complete unstructured text knowledge editing process example based on the method framework (CAKE, including SGR and CKE), covering the entire process from:

Step 0: Input Unstructured Text

Step 1: Structure-aware Graph Construction

Step 2: Multi-hop Relational Enrichment

Step 3: Paraphrase-driven Semantic Diversification

Step 4: Conflict Graph Construction

Step 5: Conflict-aware Graph Coloring

Step 6: Final Answer after Editing

### Step 0: Input Unstructured Text

#### Input Paragraph:

"Marie Curie discovered the radioactive element radium, which was later used in cancer therapy. She received a Nobel Prize for her contributions. Radium decays into radon, which is a health hazard. Her husband Pierre Curie was also a physicist."

### Step 1: Structure-aware Graph Construction

Using a structure-aware prompt to extract knowledge triples:

#### Output Triples:

- (Marie Curie, discovered, radium)
- (radium, used in, cancer therapy)
- (Marie Curie, received, Nobel Prize)
- (radium, decays into, radon)
- (radon, is, health hazard)
- (Pierre Curie, was, physicist)
- (Pierre Curie, was husband of, Marie Curie)

Resulting in base knowledge graph  $\mathbf{G} = \{\mathbf{E}, \mathbf{R}^s\}$ .

### Step 2: Multi-hop Relational Enrichment

Using prompt-based inference for disconnected entity pairs:

**Example:** For (Marie Curie, cancer therapy)

**Output:** (Marie Curie, discovered radium used in, cancer therapy)

This relation is added to form the enriched graph  $\mathbf{G}$ .

### Step 3: Paraphrase-driven Semantic Diversification

For each  $(e_i, r_{ij}, e_j)$  in  $\mathbf{R}^s$ , generate lexical variants:

**Input:** (Marie Curie, discovered, radium)

**Paraphrases:**

- (Marie Curie, was the discoverer of, radium)
- (Marie Curie, identified, radium)
- (Marie Curie, found, radium)

The resulting graph becomes a multigraph  $\mathbf{G}' = (\mathbf{E}, \mathbf{R}^s \cup \mathbf{R}^d)$ .

### Step 4: Conflict Graph Construction

Construct the conflict graph  $\mathbf{G}_c^* = (\mathbf{K}, \Phi)$  where each node is a knowledge triple  $k_{ij} = (\mathbf{e}_i, \mathbf{r}_{ij}, \mathbf{e}_j)$ .

**Example Conflict Detection of  $k_{12}$  with triples  $k_{13}, k_{41}$ :**

- $k_{12}$ : (Marie Curie, received, Nobel Prize)
- $k_{13}$ : (Marie Curie, discovered, radium)
- $k_{41}$ : (Pierre Curie, was husband of, Marie Curie)

**Conflict Detection Criterion:**  $\phi(\mathbf{k}_{ij}, \mathbf{k}_{pq})$

- $\varphi(k_{12}, k_{13}) = 1$  (high subject or relation semantic similarity)
- $\varphi(k_{12}, k_{41}) = 0$  (low subject or relation semantic similarity)

Hence, in the conflict graph  $\mathbf{G}_c^*$ ,  $k_{12}$  has an edge with  $k_{13}$ , but not with  $k_{41}$ .

### Step 5: Conflict-aware Graph Coloring

Apply the Welsh-Powell algorithm on  $\mathbf{G}_c^*$  to assign edit-isolating colors. For three knowledge triples  $k_{12}, k_{13}, k_{41}$ :

**Vertex Coloring Result:**

- **Color 1** (assigned to  $\Delta W_1$ ):  $k_{12}$  (Marie Curie, received, Nobel Prize)
- **Color 2** (assigned to  $\Delta W_2$ ):  $k_{13}$  (Marie Curie, discovered, radium)
- **Color 1** (shared with  $k_{12}$ ):  $k_{41}$  (Pierre Curie, was husband of, Marie Curie)

Note that  $k_{12}$  and  $k_{41}$  share no conflict and thus share LoRA subspace  $\Delta W_1$ .

**Update Rule per Color Group:**

$$\Delta \mathbf{W}_l \leftarrow \Delta \mathbf{W}_l - \eta \cdot \nabla \sum_{f(\mathbf{k}_{ij})=\mathbf{c}_l} \mathcal{L}(\mathcal{M}(\mathbf{x}_{ij}), \mathbf{e}_j) \quad (9)$$

During optimization:

- Color 1 ( $\Delta W_1$ ) updates  $k_{12}$  and  $k_{41}$
- Color 2 ( $\Delta W_2$ ) independently updates  $k_{13}$  to avoid gradient conflict with  $k_{12}$

#### Step 6: Final Answer after Editing

At inference time, use all LoRAs to perform inference.

**Query:** "Did Marie Curie receive the Nobel Prize?"

**Edited Model Output:** "Yes, she did."