FAME: Towards Factual Multi-Task Model Editing

Li Zeng[1*], Yingyu Shan[1*], Zeming Liu[2*], Jiashu Yao[1], Yuhang Guo[1†]
[1]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[2]School of Computer Science and Engineering, Beihang University, Beijing, China
{zengli, shanyingyu, yaojiashu, guoyuhang}@bit.edu.cn zmliu@buaa.edu.cn

Abstract

Large language models (LLMs) embed extensive knowledge and utilize it to perform exceptionally well across various tasks. Nevertheless, outdated knowledge or factual errors within LLMs can lead to misleading or incorrect responses, causing significant issues in practical applications. To rectify the fatal flaw without the necessity for costly model retraining, various model editing approaches have been proposed to correct inaccurate knowledge within LLMs in a cost-efficient way. To evaluate these model editing methods, previous work introduced a series of datasets. However, most of the previous datasets only contain fabricated data in a single format, which diverges from real-world model editing scenarios, raising doubts about their usability in practice. To facilitate the application of model editing in real-world scenarios, we propose the challenge of practicality. To resolve such challenges and effectively enhance the capabilities of LLMs, we present FAME, an factual, comprehensive, and multi-task dataset, which is designed to enhance the practicality of model editing. We then propose SKEME, a model editing method that uses a novel caching mechanism to ensure synchronization with the real world. The experiments demonstrate that SKEME performs excellently across various tasks and scenarios, confirming its practicality. [1]

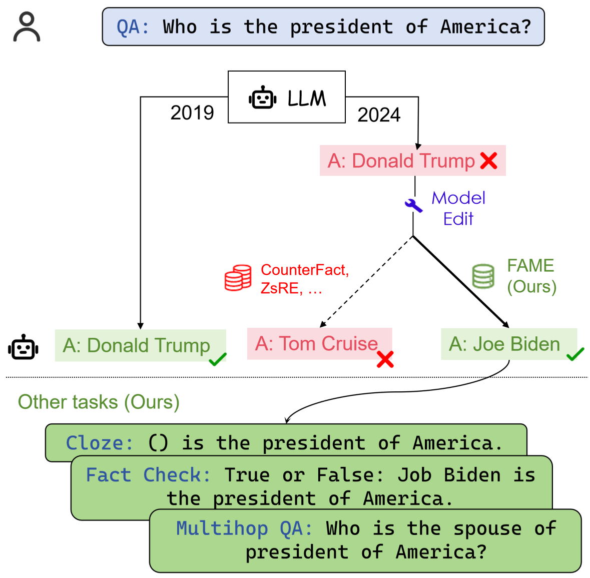

Figure 1: An example of FAME. LLMs may develop factual inaccuracies over time, which can be corrected through model editing. While previous datasets employed fabricated data, FAME utilizes real-world data to improve the performance of LLMs in practical usage.

1 Introduction

Large language models (LLMs) have achieved remarkable capabilities across various domains and are extensively utilized in practical applications (Touvron et al., 2023a,b; Achiam et al., 2023; Geva et al., 2021, 2022). The extensive utilization of LLMs makes it essential for them to provide precise information. However, LLMs may still provide erroneous information due to incorrect, outdated knowledge stored within the model (De Cao et al., 2021; Agarwal and Nenkova, 2022). Such erroneous information can have significant repercussions within critical domains like medical diagnostics and legal consultations, underscoring the importance of rectifying errors in language models. To avoid costly retraining and to efficiently correct the outputs of LLMs, model editing has been proposed (Mitchell et al., 2022; Sinitsin et al., 2020; De Cao et al., 2021).

To evaluate model editing methods, previous works have introduced a series of datasets (De Cao et al., 2021; Meng et al., 2022; Zhong et al., 2023). Almost all of these datasets set the target as incorrect answers, which affects the model's practical performance and contradicts the original purpose of model editing. As shown in Figure 1, when

[*]Equal contribution
[†]Corresponding author: guoyuhang@bit.edu.cn
[1]Dataset and codes are publicly available at `https://github.com/BITHLP/FAME`

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 15992–16011
November 12-16, 2024 ©2024 Association for Computational Linguistics

the user asks "Who is the President of America?", LLMs produce incorrect output due to outdated knowledge. Previous datasets (Levy et al., 2017; Meng et al., 2022; Gupta et al., 2023) modified them to other wrong targets (for example, Tom Cruise). Moreover, these datasets are all composed of data in a single format with a single task like QA (Levy et al., 2017) or sentence completion (Meng et al., 2022), which leads to a disparity between experiments and practical applications.

To promote the capability of model editing in practical applications, we introduce a novel criterion: **Practicality**. Practicality refers to the capacity of data and methods to be functional in real-world applications. This entails that data should be factual, diverse, and of high quality, and methods should be efficient and general across tasks. These requirements collectively ensure the effectiveness and applicability of model editing in practical applications.

To address the practical shortcomings in the previous benchmark, which mainly include incorrect knowledge and limited task formats, we introduce FAME, a factual, extensive model editing benchmark with practicality. FAME comprises 128k real data items, including various tasks with single-hop and multi-hop questions. In response to incorrect knowledge, we extract factual data items from Wikidata (Vrandečić and Krötzsch, 2014) and DBpedia (Auer et al., 2007) and employ multiple rounds of manual verification to ensure the accuracy of data items. To prevent limited task formats, we incorporate tasks from existing datasets like QA (Levy et al., 2017), fact-check (Schuster et al., 2021), multi-hop QA (Zhong et al., 2023), and introduce new tasks such as cloze and dialogue, making our evaluation more comprehensive. FAME enhances the model's ability to solve real-world problems and cross-domain issues and enables a complete evaluation of the effectiveness of model editing.

Aimed at tackling the deficiency in the practicality of previous methods, we introduce SKEME. SKEME utilizes a novel caching mechanism and receives information from diverse sources to ensure synchronization with the real world, allowing for the application of SKEME in real-world scenarios. The caching mechanism tackles the challenges posed by large-scale data and diverse tasks.

To evaluate the practicality of model editing methods, we first introduce a new metric, SURE, which considers both accuracy and side effects and is adaptable to scenarios. We then evaluate whether each method can meet the basic requirements of model editing (Section 7). Subsequently, We discuss the performance of model editing methods in real-world scenarios (Section 8). The conclusion indicates that previous methods either exhibit side effects or struggle to handle complex scenarios, while only SKEME consistently outperformed others across all experimental conditions, demonstrating the superiority of SKEME in real-world scenarios.

The main contributions of this paper are as follows:

- We first introduce the practicality requirement for model editing, which necessitates data and methods to exhibit effective performance in real-world applications.
- To support the practicality requirement of model editing, we create FAME, a novel benchmark that utilizes real-world knowledge and common diverse tasks to simulate practical applications.
- To meet the practicality requirement of the model editing method, we propose a method called SKEME, which is the first to introduce a novel caching mechanism for efficient storage, retrieval, and update of constantly evolving real-world facts. Experiments demonstrate that SKEME is more effective in real-world scenarios.

2 Related work

2.1 Model Editing Datasets

Model editing datasets serve the purpose of verifying the effectiveness of methods and enhancing the capability of LLMs. Nevertheless, current datasets fall short of directly enhancing the capability of LLMs. The majority of datasets comprise constructed fake data (Levy et al., 2017; Meng et al., 2022; Gupta et al., 2023), primarily serving to validate effectiveness rather than directly contribute to the enhancement of LLMs' capabilities. MQuAKE-T (Zhong et al., 2023) utilizes modifications in Wikidata, which has the potential to directly enhance LLMs' practical performance. However, due to the limited combinations of relations (see Figure 11 for statistics), its direct utility in improving the performance of LLMs is limited, thereby primarily serving to validate effectiveness. In contrast to prior works, our benchmark sets itself apart by featuring a substantial repository of factual data and integrating multiple diverse tasks. As a result, it exhibits a heightened level of practical applicability.

2.2 Model Editing Methods

Previous works have introduced various model-editing methods, both parameter modification and parameter preservation approaches (Yao et al., 2023). The former category includes the locate-then-edit method (Meng et al., 2022, 2023) and meta-learning-based methods (De Cao et al., 2021; Mitchell et al., 2021). The latter category involves adding additional parameters to the model (Huang et al., 2023) and employing vector databases for knowledge storage and retrieval (Mitchell et al., 2022; Zhong et al., 2023; Zheng et al., 2023; Cheng et al., 2023; Madaan et al., 2022). Building on this foundation, there are also methods such as reflection (Wang et al., 2024b), optimization of searches for multi-hop questions (Shi et al., 2024), and the use of post-processing (Song et al., 2024) to improve retrieval strategies. Diverging from the previously discussed methods, our method involves a novel caching mechanism, allowing for the application in real-world scenarios.

3 Problem Definition

The objective of model editing is to modify the knowledge contained in a model, allowing the model to engage in reasoning processes based on the edited knowledge, while not affecting the output related to the unedited knowledge. Based on previous work (Zhang et al., 2024; Yao et al., 2023), we define model editing to express the goal as follows.

An input-output pair is defined as (x, y), and a model is represented by a function $f : X \to Y$, where X represents the input set and Y represents the output set. Let $I(x, y)$ denotes the set of descriptions semantically equivalent to (x, y), and $EX(x, y)$ be the set of input-output pairs that the model can possess with $I(x, y)$ as prior knowledge. Then, let $O(x, y)$ represent the portion outside $I(x, y)$ and $EX(x, y)$. Appendix C provides an example of the definition.

Formally, let (subject, relation, object) be a factual triple, denoted as (s, r, o). Consider an input-output pair as (x, y), where x is effectively a combination of s and r. A model is represented by a function $f : X \to Y$, where X represents the input set and Y represents the output set.

We use the prime notation to denote semantically equivalent elements. Specifically, for any t in the set $\{s, r, o, x, y\}$, let t' be any element that is semantically equivalent to t, and let T' be the set of all such t'. Notice that $t \in T'$. Then, we can define $I(x, y)$ as

$$I(x, y) = \{(x', y')|x' \in X' \text{ and } y' \in Y'\}. \quad (1)$$

To define $EX(x, y)$, let's represent a fact triple as $tr(s, r, o)$, abbreviated as tr, and S is the set of all fact triples. Also, define the multiplication operation $*$ for two sets of fact triples A and B as the join operation:

$$A * B = A \underset{o=s}{\bowtie} B \quad (2)$$

Then, define

$$N_0(tr) = \{(s', r', o') \mid s' \in S', r' \in R', o' \in O'\} \quad (3)$$

and

$$N_i(tr) = N_{i-1}(tr) * S \qquad (i \geq 1) \quad (4)$$

Ultimately, we define $EX(tr)$ as

$$EX(tr) = \bigcup_{i=0}^{\infty} N_i \quad (5)$$

By transforming s and r into x, and o into y, we derive $EX(x, y)$.

After defining $I(x, y)$ and $EX(x, y)$, we can define $O(x, y)$ as

$$O(x, y) = \complement_S(I(x, y) \cup EX(x, y)) \quad (6)$$

where $\complement_S$ represents the complement within the set S.

The definition of model editing can be summarized as follows: (x_f, y_f) denotes the fact that is being edited, while (x_e, y_e) represents the input and output.

$$f'(x_e) = \begin{cases} y_f & (x_e, y_e) \in I(x_f, y_f) \\ f(x_e) \mid (x_f, y_f) & (x_e, y_e) \in EX(x_f, y_f) \\ f(x_e) & (x_e, y_e) \in O(x_f, y_f) \end{cases}$$

4 FAME: A Practical Model Editing Benchmark

FAME (**FA**ctual **M**ulti-task model **E**diting) is a benchmark comprising 128k factual data items. We utilize these data items to construct both single-hop and multi-hop questions. For single-hop questions, we include six forms: QA, sentence completion, cloze test, multiple-choice questions, fact check, and locality test. For multi-hop questions, we include multi-hop questions and dialogues. The previous work introduced QA, sentence completion, fact check, and multi-hop questions (Zhang et al.,

Name	isC.	Tasks							Total	Re.	Source	Hop
		Cho.	FC.	Clo.	Dia.	Com.	QA					
ZsRE	✗	✗	✗	✗	✗	✗	✓	270K	120	WD.	Si.
COUNTERFACT	✗	✗	✗	✗	✗	✓	✗	2.2K	24	WD.	Si.
MQuAKE-CF	✗	✗	✗	✗	✗	✗	✓	9.2K	37	WD.	Mu.
MQuAKE-T	✓	✗	✗	✗	✗	✗	✓	1.8K	6	WD.	Mu.
FAME (OURS)	✓	✓	✓	✓	✓	✓	✓	128K	86	WD. & DB.	Si. & Mu.

Table 1: Comparison between FAME to other model edit datasets. "isC." stands for isCorrect, which means if the edit target is the real fact. "Cho.", "FC.", "Clo.", "Dia.", "Com.", "Re", "WD.", "DB.", "Si.", "Mu." stands for choose, fact-check, cloze, dialogue, completion, Relations, Wikidata, DBpeida, single-hop data, and multi-hop data, respectively.

2024), while we propose the remaining tasks. We believe that combining these tasks contributes to a comprehensive assessment of the effectiveness of model editing methods.

The construction of FAME is divided into two steps: (1) Collect real fact triples; (2) Create diverse tasks using the collected triple. To ensure the data quality of FAME and its reflection of the real world, we conducted multiple rounds of manual verification and correction in various aspects. For more details, please refer to Appendix A.1.

4.1 Collect Fact Triples

To obtain real-world fact triples, we collect data from Wikidata (Vrandečić and Krötzsch, 2014) and DBpedia (Auer et al., 2007), both of which are continuously updated databases. We aim to enhance the diversity of FAME by collecting knowledge from a variety of knowledge bases.

Specifically, we initially identified equivalent relations in Wikidata and DBpedia. Subsequently, non-informative relationships such as IDs were discarded. Then, we collect triplets associated with these relations from Wikidata and DBpedia. After obtaining the triplets, we further filter them to avoid potential ambiguity issues, see Appendix A.1.2 for details.

Finally, to ensure the quality of the triplets we obtained, we randomly selected 100 triplets and manually examined their correctness. The results indicate that 96% of the triplets are correct, which shows that our process for obtaining and filtering triplets is acceptable.

4.2 Generate Data Based on Templates

We create templates for each type of task to transform fact triples into queries for various tasks. We employ ChatGPT in the generation process to mitigate expensive labor costs following previous works (Petroni et al., 2019; Yin et al., 2023). After generating the results, we conduct manual checks to ensure the accuracy and alignment with our intentions.

For single-hop questions, we prompt ChatGPT to generate question templates based on the description of each relationship that used in fact triples (e.g., head of government), incorporating placeholders. Then we replace these placeholders with subjects to generate questions from the templates.

For multi-hop questions, we employ ChatGPT to concatenate multiple consecutive triplets into a single question. Inspired by Petroni et al. (2019), to distinguish between the differences in model decomposition ability and knowledge it knows, we decompose queries to obtain multi-turn dialogue.

To ensure the accuracy of templates, we incorporate manual verification to ensure that the templates align with the meaning of the relationships. We found that 97.4% of templates are accurate, we then manually performed multiple rounds of correction and rechecking, ensuring that the correctness rate of the templates reached 100%.

Finally, following previous work (Yin et al., 2023), we employ manual sampling and verification techniques to ensure the accuracy of FAME. We combine the templates and relation triplets and manually check the credibility of the generated sentences. The results show that 97.5% of the sentences were credible, demonstrating the reliability of the entire process.

5 Benchmark Analysis

5.1 Comparison

See Table 1 for a comparison between FAME and previous benchmarks. FAME includes all categories seen in previous benchmarks, and we propose additional data categories. Moreover, the number of entries far exceeds those in most of the previous benchmarks. Finally, FAME originates from two distinct knowledge bases, making it more comprehensive compared to previous datasets.

Compared to the previously multi-hop dataset containing factual knowledge, MQuAKE-T (Zhong et al., 2023), FAME is larger and includes more relationships, making it more comprehensive. See Appendix D for details on the comparison and further comparisons.

5.2 Statistics

FAME consists of two parts: single-hop data and multi-hop data, both sourced from Wikidata and DBpedia. Table 2 presents the statistical data for different tasks in the dataset. Please refer to Appendix E for examples and more detailed statistics.

Task	Type	Total
Single-hop	Single-hop QA	20,000
	Sentence Completion	20,000
	Cloze	20,000
	Multiple-Choice	20,000
	Fact Check	20,000
	Locality	20,000
Multi-hop	Multi-hop QA in 2 hops	1,000
	Dialogue in 2 hops	1,000
	Multi-hop QA in 3 hops	1,000
	Dialogue in 3 hops	1,000
	Multi-hop QA in 4 hops	1,000
	Dialogue in 4 hops	1,000
	Multi-hop QA in 5 hops	1,000
	Dialogue in 5 hops	1,000

Table 2: Task types and statistics

6 SKEME: A Model Editing Method For Real-World Applications

To accommodate practical model editing and meet the need for utilizing real-world facts and adapting to diverse tasks, We propose SKEME (**S**tructured **K**nowledge retrieved by **E**xact **M**atching and reranking **E**diting), a model editing method that first incorporates a caching system to efficiently store and retrieve real-world knowledge, which is collected from diverse data sources. Besides, SKEME utilizes entity extraction to exclude irrelevant content brought by diverse input formats, enhancing its performance across various tasks.

6.1 Overview

The overview of SKEME is shown in Figure 2. SKEME consists of three main components: To avoid the influence of varied input formats, **Entity Extraction** extracts key entities from the input for subsequent retrieval. **Knowledge Base Retrieval** queries external knowledge bases for world knowledge related to the entities extracted by Entity Extraction and caches the results in the local structured knowledge base. **Knowledge Rank and Utilization** utilizes the results retrieved by Knowledge Base Retrieval to correct the model's output. Please refer to Appendix A.2 for details.

6.2 Entity Extraction

To handle complex real-world tasks, **entity extraction** aims to extract key entities e from the input I, while ignoring irrelevant content (e.g., extracting "America" from "Who is the president of America?"), thus excluding the influence of input forms and preparing the retrieval entity for the next step. Specifically, to ensure system robustness, SKEME prompt uses LLMs to extract key entities e from the input I.

6.3 Knowledge Base Retrieval

To accurately retrieve facts related to the entity extracted by entity extraction e, we use a knowledge graph (KG) $KG = \{(s, r, o) \mid s, o \in E, r \in R\}$, where E is the set of entities and R is the set of relations. The retrieval step involves extracting a subgraph G' from the knowledge base G, where $G' = \{(s', r', o') \mid (s', r', o') \in G \text{ and } s' = e\}$.

Specifically, in knowledge base retrieval, similar to the principle of locality in **computer caching systems**, some knowledge may reappear across multiple queries (Jin et al., 2024). Therefore, introducing a caching system can reduce the number of data queries and improve system efficiency. Inspired by this, we utilize a fast-slow table mechanism in the knowledge base. The slow table is G and the fast table is $G'' = \{(s'', r'', o'') \mid (s'', r'', o'') \in G \text{ and } s'' \in E\}$, respectively. It's important to note that $G'' = \varnothing$ initially.

Initially, we retrieve facts related to the previously extracted entity e in the G''. If related facts are not situated in G'', we then search in the slow table G and update G'' with the retrieval results. Inspired by the principle of locality in operating systems, we not only update retrieval results in G'' but also update the triples related to the retrieval

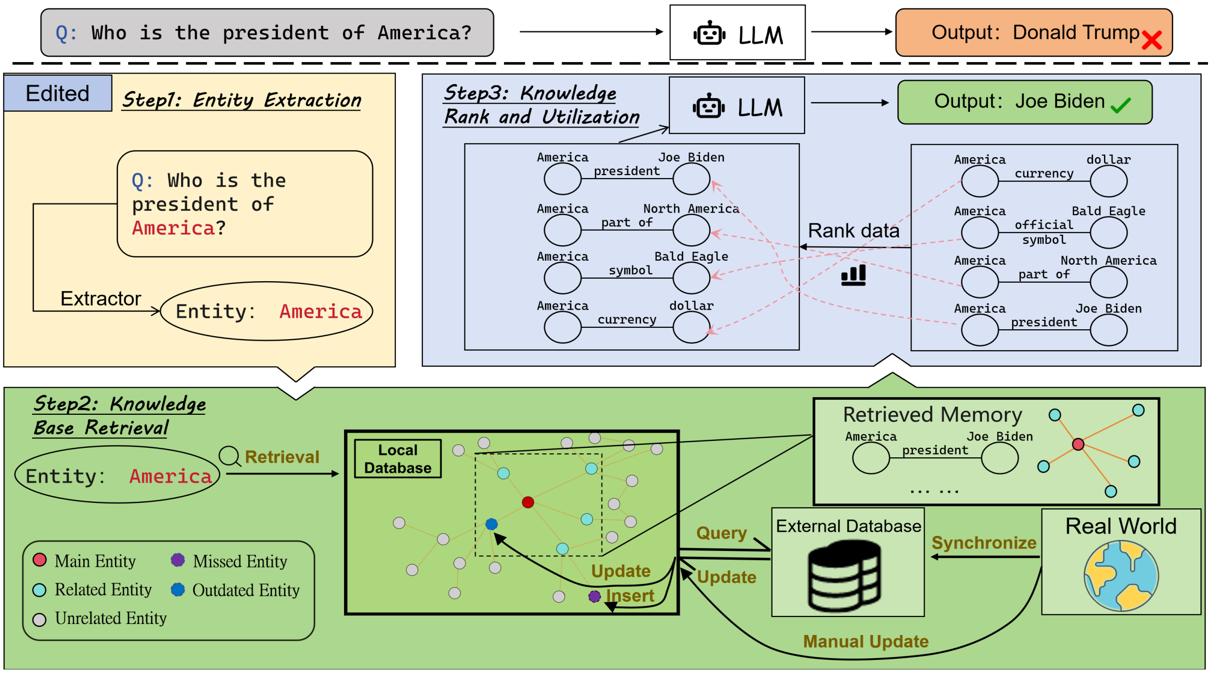

Figure 2: An overview of SKEME. SKEME initially extracts key entities from the question. Subsequently, it retrieves the knowledge base for facts related to entities. Then ranks applicable knowledge items and utilizes in-context learning to modify the model's output. Additionally, we update knowledge from external databases and the real world to ensure that the local knowledge base reflects real-world changes.

results in G''.

To ensure consistency between G and G'', we build the G'' through two primary methods. First, if the retrieval in the local database yields $G' = \varnothing$, indicating that the entity is absent from the G'', we retrieve the extracted entity from G in the same approach as G'', then load the retrieval results in the G to update our G''. Since G such as Wikidata continues to update to remain current with the real world, the G'' sourced from G essentially serves as a cache of real-world facts, allowing the G'' to reflect real-world facts. Second, benefiting from the use of structured knowledge bases, G'' can also be manually updated to directly incorporate knowledge from the real world as an additional source of information.

To keep the G'' current with the G, we will update the G'' regularly. In the process of updating, for any update (s, r, o_{new}), if outdated knowledge (s, r, o_{old}) exists in G'' and $o_{new} \neq o_{old}$, we replace o_{old} with o_{new}; otherwise, we directly insert the new knowledge. This strategy allows us to adapt to the evolution of facts, such as the transition of the U.S. President from Obama → Trump → Biden.

6.4 Knowledge Rank and Utilization

After retrieving the graph G', we use the pre-trained Contriever (Izacard et al., 2021) model to calculate embeddings for the triples in G'. Subsequently, we retain the facts that are closest to the query in the embedding space, which means the facts most relevant to the query.

Inspired by Zheng et al. (2023), we use in-context learning to modify the model's output. Specifically, we integrate the triples into the input and prompt the model to utilize the triples in responding to queries. Such in-context learning enables us to tackle various tasks economically and ensures its effectiveness in models of varying sizes.

7 Experiment

7.1 Metrics

We use the following metrics to evaluate whether editing achieves our goal in Section 3.

Accuracy To calculate accuracy, we instruct the model to generate responses for tasks and evaluate whether they match the gold answers exactly. The resulting average accuracy is then recorded as **exact match (EM)**.

Locality Locality measures whether an editing method will influence irrelevant knowledge. We utilize **drawdown (DD)** (Mitchell et al., 2021, 2022) to compute performance degradation and employ **Neighborhood KL divergence (NKL)** (Hoelscher-Obermaier et al., 2023) to measure whether the model is significantly affected.

SURE To meet the demands of practical application, model editing needs to consider both accuracy and side effects while also adapting to scenarios. To assess this capability, we propose the metric **SURE** (**S**tatistical and **U**nbiased **R**eal-world **E**valuation) to estimate the performance of edited models in real-world scenarios. We define SURE as follows:

$$SURE = aEM^{\alpha} - bDD^{\beta}$$

The parameters a and b denote the ratio of the data used to evaluate the two metrics, α and β are used to characterize the importance of EM and DD, which are adjusted according to specific tasks. See Appendix F for a more detailed analysis of its motivation and advantage.

Efficiency Efficiency measures the time and GPU space consumed by the model editing methods. Following Yao et al. (2023), we measure efficiency in both **time consumption (Ti)** and **memory requirements (Me)**.

7.2 Baselines

Following Yao et al. (2023), we compare SKEME with parameter-modifying methods, including FT and MEMIT (Meng et al., 2023), as well as parameter-preserving methods, including MeLLo (Zhong et al., 2023), and IKE (Zheng et al., 2023). FT is the most classic and straightforward model-editing method. MEMIT is currently considered a state-of-the-art method among parameter modification methods. IKE and MeLLo, much like SKEME, leverage a knowledge base and in-context learning. Implementation details can be found in Appendix A.3.

7.3 Main Result

Table 3 shows results on FAME. We experiment with all methods on GPT2-XL (Solaiman et al., 2019), GPT-J (6B) (Wang and Komatsuzaki, 2021), Llama2 (Touvron et al., 2023b), and utilize in-context learning based methods on GPT-3.5-turbo (Ouyang et al., 2022).

Firstly, we scrutinize the results on Llama2, which is the largest model we can employ all model editing methods. **FT** and **MEMIT**, did not perform well in our experiments, which may be due to the editing process not specifically targeting the model's generative capability. **MeLLo** has a higher EM score than FT and MEMIT, but its DD is also the highest, indicating its pronounced side effects, which leads to a low SURE. Both **IKE** and **SKEME** obtained an EM above 0.9. However, IKE also has presented side effects that consequently decreased its SURE. SKEME uniquely maintains a high EM and simultaneously ensures a low DD, thus demonstrating superior practicality compared to other methods.

To test the impact of model size, we experiment with various model sizes. SKEME excels across all, while some other methods fail on small models. These model editing methods also require diverse amounts of time and GPU space. MeLLo, due to its long in-context learning process, consumes the most time in the RAG methods. MEMIT demonstrates strong capability and low side effects, but it is more time-consuming. Additionally, previous work (Yao et al., 2023) has pointed out that while MEMIT can perform batch editing, its effectiveness tends to decrease as the batch size increases. In contrast, models based on RAG and in-context learning (MeLLo, IKE, and SKEME) can easily handle batch editing without decreasing performance. Overall, SKEME proves effectiveness across model sizes while consuming less additional time and GPU space.

It appears that all methods performed poorly on certain tasks. This further validates the meaningfulness of constructing data in various forms. On the completion task, although the base model performed similarly to QA and Cloze, the edited model's accuracy was significantly lower than QA and Cloze. It indicates that the method's generalization performance still needs to improve.

8 Analysis

Considering the complexity of real world, besides single fact edits, we design a series of research questions (RQs) to evaluate the method's ability to edit multiple facts. We discuss **fact transitions (RQ1)**, *e.g., the U.S. President transitioning from Obama → Trump → Biden*, **fact inference (RQ2)**, *e.g., inferring the fact "the First Lady of the U.S. is Jill Biden" from the given facts "the U.S. President is Biden" and "Biden's spouse is Jill Biden"*, **fact with substantial quantity (RQ3)**, *e.g., needing to update thousands of facts* and **fact from various benchmarks (RQ4)**, *e.g., facts from other*

Model	Method	SURE↑	Accuracy						Locality		Efficiency	
			EM.↑	QA.↑	Com.↑	Clo.↑	Cho.↑	FC.↑	**DD.**↓	**NKL.**↓	**Ti.**↓	**Me.**↓
GPT2-XL	Base	-	19.83	8.00	7.11	3.63	34.25	46.16	-	-	0.18	9.12
	FT	12.75	22.72	11.82	10.26	9.96	33.58	47.96	9.97	1.33	2.12	12.43
	MEMIT	20.87	20.87	7.31	7.14	6.67	34.22	49.04	**0.00**	1.29	13.6	11.85
	MeLLo	-53.67	30.90	71.42	0.24	0.09	33.72	49.01	84.57	1.32	1.43	17.43
	IKE	37.32	50.51	62.05	54.82	48.96	36.09	50.64	13.19	1.25	0.75	14.26
	SKEME	**65.80**	**65.80**	**85.12**	**70.60**	**78.45**	**38.33**	**56.51**	**0.00**	**1.09**	**0.23**	**11.52**
GPT-J	Base	-	23.36	11.86	12.02	11.52	35.34	46.08	-	-	0.35	26.57
	FT	25.21	26.59	13.69	13.38	13.39	40.74	51.72	1.38	1.76	3.27	34.81
	MEMIT	45.85	45.85	49.51	41.14	43.51	**46.62**	48.49	**0.00**	1.86	13.8	29.84
	MeLLo	28.42	55.74	72.20	48.35	72.95	21.81	**63.41**	27.33	1.54	2.42	33.38
	IKE	58.62	70.04	87.00	**82.35**	82.27	46.32	52.26	11.42	**1.28**	0.97	31.53
	SKEME	**73.93**	**73.93**	**97.03**	79.63	**87.02**	46.01	59.97	**0.00**	1.39	**0.49**	**28.17**
Llama2	Base	-	32.20	15.82	15.78	16.02	48.91	64.45	-	-	0.33	30.04
	FT	34.31	41.80	30.05	29.08	29.22	60.57	60.07	7.49	2.55	5.18	38.92
	MEMIT	48.03	48.39	41.16	40.26	41.47	61.00	58.08	0.36	2.83	13.2	33.52
	MeLLo	36.38	66.26	68.56	36.95	69.26	78.17	78.35	29.88	2.72	2.45	38.66
	IKE	71.38	**91.42**	97.72	**90.11**	**95.76**	**95.10**	78.42	20.04	2.48	1.08	35.17
	SKEME	**90.54**	90.54	**98.61**	83.04	90.27	93.73	**87.07**	**0.00**	**2.12**	**0.45**	**31.83**
GPT-3.5-turbo	Base	-	40.11	18.76	19.65	17.17	73.73	71.22	-	$*$	0.81	$*$
	MeLLo	56.58	73.75	70.51	57.16	76.37	82.78	81.92	17.16	$*$	2.92	$*$
	IKE	76.45	89.53	92.81	**89.72**	**90.88**	90.41	83.85	13.08	$*$	1.47	$*$
	SKEME	**91.76**	**91.76**	**98.07**	84.78	89.45	**99.04**	**87.40**	**0.00**	$*$	**1.03**	$*$

Table 3: Main result on FAME. "Com.", "Clo.", "Cho." and "FC." stands for completion, cloze, choose, and fact-checking, respectively. Ti (s) includes both editing and generating time in Wall clock time and Me (GB) is calculated by measuring the maximum required GPU VRAM. To maintain brevity, the multiplier of $\times 10^{-4}$ has been excluded for the NKL metric. Since DD and NKL are calculated relative to the unedited model, the unedited model does not have these metrics. "-": Since DD and NKL are computed relative to the unedited model, and SURE's calculation depends on DD, these three metrics are meaningless for the unedited model. "$*$": The computation of NKL and ME metrics for GPT-3.5-turbo is impractical due to its utilization via API calls.

datasets.

8.1 RQ1: How Do Methods Handle Transitions Between Facts?

Many facts in the real-world transition require multiple edits to the same fact. To evaluate the effectiveness of repeated edits, we perform multiple updates for each fact and tested the accuracy of the edited model. Figure 3 shows the experimental results. The results indicate that even with only two edits to the same fact, the accuracy of all other methods substantially declines. Parameter-modifying methods can lead to divergence from the initial model parameters and subsequent performance decline due to repeated adjustments of specific parameters. For methods not to modify the parameters, failure to update the knowledge base may result in conflicts between existing and newly added facts.

SKEME uses a structured knowledge base to facilitate precise updates, making iterative updates possible. SKEME is the only one capable of handling iterative updates.

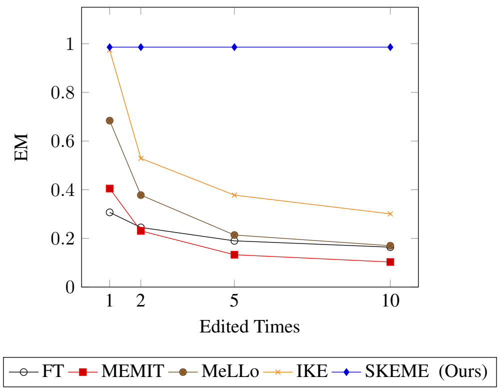

Figure 3: Result of RQ1. The x-axis indicates the number of edits to the same fact.

8.2 RQ2: Can The Edited Model Infer New Facts Based on Given Information?

The edited model should be able to make further reasoning based on the edited facts. Following previous research (Zhong et al., 2023), We employ multi-hop questions to evaluate this capability of the model.

Method	Multi-hop QA			
	2-hops	3-hops	4-hops	5-hops
Base	0.145	0.135	0.112	0.079
FT	0.223	0.362	0.231	0.128
MEMIT	0.176	0.247	0.136	0.060
MeLLo	0.270	0.227	0.167	0.073
IKE	0.332	0.237	0.220	0.159
SKEME	**0.960**	**0.786**	**0.427**	**0.167**
Method	**Dialogue**			
	2-hops	3-hops	4-hops	5-hops
Base	0.119	0.118	0.116	0.082
FT	0.190	0.216	0.152	0.133
MEMIT	0.238	0.220	0.148	0.126
MeLLo	0.353	0.295	0.193	0.111
IKE	0.229	0.235	0.207	**0.188**
SKEME	**0.946**	**0.757**	**0.390**	0.181

Table 4: Result of RQ2. SKEME manifests significant improvements compared to previous approaches, however, it still fails to address the issue when $hops \geq 4$.

Table 4 presents the results for this task. We can observe that all methods, except for SKEME, performed poorly. Traditional retrieval-based models struggle to find answers to multi-hop questions, and other methods do not enable the model to infer based on edited facts.

8.3 RQ3: Can Methods Handle Updates with a Substantial Amount of Facts?

In the real world, there are numerous updates to facts and a practical model editing method should be able to update a vast quantity of knowledge in the model.

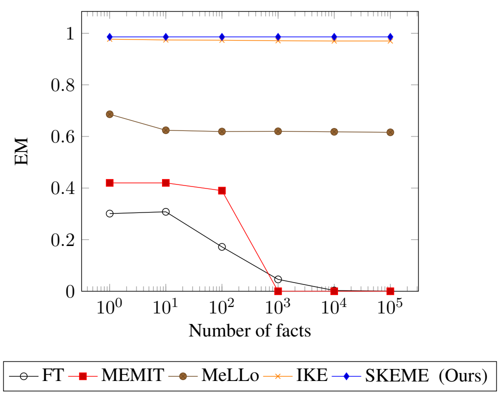

Figure 4: Result of RQ3. The x-axis represents the number of edited facts.

Figure 4 shows the experimental results. As the quantity of modified facts increases, parameter-modifying methods progressively shift away from their original state, resulting in a notable performance decline. In contrast, other methods, exhibit only slight declines in performance and can handle updates to a large number of facts.

8.4 RQ4: Can Methods Generalize Across Facts in Various Benchmarks?

To demonstrate the general applicability of the editing method, we select several datasets that are widely used to evaluate LLMs' understanding of the world, and subsequently evaluate model editing methods on them. Please refer to Appendix A.4 for details about these datasets.

Method	TQA	NQ	FEVER	Vi
Base	0.698	0.191	0.792	0.397
FT	0.362	0.274	0.646	0.228
MEMIT	0.449	0.632	0.724	0.461
MeLLo	0.811	0.633	0.872	0.720
IKE	0.962	**0.980**	0.954	**0.964**
SKEME	**0.984**	0.964	**0.987**	0.956

Table 5: Result of RQ4. "TQA", "NQ" and "Vi" stands for triviaQA, Natural Questions, and VitaminC, respectively. All accuracies are calculated based on exact match rates.

Table 5 shows that SKEME consistently improves the model's performance irrespective of the benchmarks, demonstrating the robust versatility and scalability of SKEME.

9 Conclusion

We introduce the practicality requirement for model editing and created a novel benchmark FAME, which embodies practicality with factual data and diverse tasks. We propose a model editing method, SKEME, that proves effective across various LLMs and tasks. The experiments demonstrate that previous model editing methods struggle in dealing with real-world challenges, while SKEME successfully addresses these challenges. We hope that our work will advance the field of model editing and inspire further research in this area.

Acknowledgments

Thanks for the insightful comments from reviewers. This work is supported by the Natural Science Foundation of China (No. U21B2009, 62406015) and the Beijing Institute of Technology Science and Technology Innovation Plan (No. 23CX13027).

Limitations

The data in FAME is limited to a monolingual scope and does not include multilingual data. We posit that the inclusion of multilingual data can further align with the real world, and we leave this as a potential area for future work.

Ethics Statement

We ensure that the collection of FAME is done in a manner consistent with the terms of use stipulated by its sources and the intellectual property rights of the original authors. We make sure that individuals involved in the collection process are treated fairly, including ensuring their voluntary participation and informed consent. Due to the dynamic nature of the real world, certain knowledge contained in FAME may become outdated, rendering it no longer reflective of the latest world conditions.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Oshin Agarwal and Ani Nenkova. 2022. Temporal effects on pre-trained models for language processing tasks. *Transactions of the Association for Computational Linguistics*, 10:904–921.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *international semantic web conference*, pages 722–735. Springer.

Yingfa Chen, Zhengyan Zhang, Xu Han, Chaojun Xiao, Zhiyuan Liu, Chen Chen, Kuai Li, Tao Yang, and Maosong Sun. 2024. Robust and scalable model editing for large language models. *arXiv preprint arXiv:2403.17431*.

Xin Cheng, Yankai Lin, Xiuying Chen, Dongyan Zhao, and Rui Yan. 2023. Decouple knowledge from paramters for plug-and-play language modeling. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14288–14308, Toronto, Canada. Association for Computational Linguistics.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Anshita Gupta, Debanjan Mondal, Akshay Sheshadri, Wenlong Zhao, Xiang Li, Sarah Wiegreffe, and Niket Tandon. 2023. Editing common sense in transformers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8214–8232.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. 2023. Detecting edit failures in large language models: An improved specificity benchmark. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11548–11559, Toronto, Canada. Association for Computational Linguistics.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Xin Liu, Xuanzhe Liu, and Xin Jin. 2024. Ragcache: Efficient knowledge caching for retrieval-augmented generation. *arXiv preprint arXiv:2404.12457*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st*

Conference on Computational Natural Language Learning (CoNLL 2017), pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2833–2861, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023. Mass editing memory in a transformer. *The Eleventh International Conference on Learning Representations (ICLR)*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander 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)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Tal Schuster, Adam Fisch, and Regina Barzilay. 2021. Get your vitamin C! robust fact verification with contrastive evidence. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 624–643, Online. Association for Computational Linguistics.

Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. 2024. Retrieval-enhanced knowledge editing for multi-hop question answering in language models. *arXiv preprint arXiv:2403.19631*.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *International Conference on Learning Representations*.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. 2019. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

Xiaoshuai Song, Zhengyang Wang, Keqing He, Guanting Dong, Yutao Mou, Jinxu Zhao, and Weiran Xu. 2024. Knowledge editing on black-box large language models. *arXiv preprint arXiv:2402.08631*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *NAACL-HLT*.

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. 2023a. Llama: Open and efficient foundation language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

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. 2024a. EasyEdit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93, Bangkok, Thailand. Association for Computational Linguistics.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. 2024b. Memoryllm: Towards self-updatable large language models. *arXiv preprint 2402.04624*.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online. Association for Computational Linguistics.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics.

Xunjian Yin, Baizhou Huang, and Xiaojun Wan. 2023. ALCUNA: Large language models meet new knowledge. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1397–1414, Singapore. Association for Computational Linguistics.

Ningyu Zhang, Yunzhi Yao, and Shumin Deng. 2024. Knowledge editing for large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024): Tutorial Summaries*, pages 33–41, Torino, Italia. ELRA and ICCL.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876, Singapore. Association for Computational Linguistics.

Zexuan Zhong, Zhengxuan Wu, Christopher Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15686–15702, Singapore. Association for Computational Linguistics.

A Implementation Details

A.1 FAME Construction Details

This section provides a detailed description of how we constructed FAME.

A.1.1 Collect Fact Triples

To enhance the diversity of FAME and reduce bias from a single data source, We use WikiData SPARQL query[2] and DBpedia SPARQL query[3] to collect data from Wikidata and DBpedia.

Firstly, using the code in Figure 5, we query for relationships with equivalent meanings in Wikidata and DBpedia, such as "birth place" (from DBpedia) and "place of birth" (from Wikidata). They are connected through the relationship *equivalentProperty*. Subsequently, we filter out relationships like identifiers, where their objects are typically composed of irregular and meaningless combinations of letters and numbers.

Next, based on the obtained r, we use code in Figure 6 and 7 to collect triples (s, r, o) from Wikidata and DBpedia. We filter out triples that may cause ambiguity, including two aspects: two different items having the same name or a specific entity's relation corresponding to multiple objects. Relevant discussions can be found in Appendix A.1.2.

Lastly, we manually extract 100 distinct triples and verify them against other data sources such as government websites to ensure the accuracy and real-world relevance of our collected triples. The results show that 96% of the data is correct. Hence, we can infer that FAME can reasonably reflect real-world scenarios.

A.1.2 Data Filter

Ambiguity issues involve two aspects: different entities sharing the same name and a specific entity's relation corresponding to multiple objects.

For the former scenario, one example is: *Hope Springs* could refer to a movie from 2012 (Q327214 in Wikidata)[4], but can be a movie from 2003 as well (Q596646 in Wikidata)[5]. So when asking *Who is the director of Hope Springs?*, there are multiple correct options.

An example of the latter scenario is: a person may have multiple children, so there are multiple correct answers when asking for their children's names.

We believe that the above two scenarios are simpler compared to questions with only one answer. Therefore, for easier implementation and to focus on more fundamental phenomena, we excluded data in the dataset containing instances of the above situations.

A.1.3 Generate Data Based on Templates

Following previous work, after obtaining triples, we need to construct relationship templates to build our entire dataset. For single-hop data, we use the following triple as an example to illustrate our entire construction process: *(subject, relation, object)* = *(America, head of government, Biden)*.

We construct several templates for each relation for each task. For instance, when (r = head of government), the template for the QA task might be "Who is the head of government in {}?",

[2] https://query.wikidata.org
[3] https://dbpedia.org/sparql
[4] https://www.Wikidata.org/wiki/Q327214
[5] https://www.Wikidata.org/wiki/Q596646

```
SELECT ?DBpediaProp ?itemLabel ?WikidataProp
WHERE
  {
    ?DBpediaProp  owl:equivalentProperty  ?WikidataProp .
          FILTER ( CONTAINS ( str(?WikidataProp) , 'wikidata' ) ) .
    ?DBpediaProp    rdfs:label    ?itemLabel .
          FILTER (lang(?itemLabel) = 'en')
  }
ORDER BY  ?WikidataProp
```

Figure 5: Sparql code used to query equivalent relations.

```
SELECT DISTINCT ?subject ?object ?subjectLabel ?objectLabel (COUNT(
   distinct ?r) AS ?relationCount)
WITH {
SELECT DISTINCT ?subject ?object ?subjectLabel ?objectLabel
    WHERE {
      ?subject wdt:{item} ?object.
      # ?subject wdt:P31 wd:Q5.
  ?subject rdfs:label ?subjectLabel.
     FILTER(LANG(?subjectLabel) = "en").
  OPTIONAL {?object rdfs:label ?objectLabel.}
     FILTER(LANG(?objectLabel) = "en").
    }
LIMIT {limit}
OFFSET {offset}
} AS %SUB

WHERE{
  INCLUDE %SUB
  ?subject ?r []
}
GROUP BY ?subject ?object ?subjectLabel ?objectLabel
```

Figure 6: Code used to query triples from Wikidata.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?subject ?subjectLabel ?object ?objectLabel
WHERE {
  ?subject <{property_url}> ?object.
  # ?subject <http://dbpedia.org/ontology/primeMinister> ?object.
  ?subject rdfs:label ?subjectLabel.
     FILTER(LANG(?subjectLabel) = "en").
 OPTIONAL { ?object rdfs:label  ?objectLabel. FILTER(LANG(?
    objectLabel) = "en"). }
}
```

Figure 7: Code used to query triples from DBpedia.

and the template for the completion task might be "In {}, the head of government is". During usage, we replace {} with the subject to obtain the QA and completion task queries: "Who is the head of government in America?" and "In America, the head of government is". We expect the model to answer with the object Biden.

Following previous work, we use ChatGPT to assist us in constructing templates. We provide each relation name and a brief description and use few-shot learning to ensure that the templates it constructs meet our requirements. We require it to construct 3-4 different templates for each relation to test different template generation methods for their generalization performance.

For multi-hop questions, consider a chain $C = [(s_1, r_1, o_1), (o_1, r_2, o_2), \ldots]$, where the object of each triple is the subject of the next triple, i.e., $o_i = s_{i+1}$. When o_1 changes, the entire chain undergoes a ripple effect of changes, and we expect the model to answer the updated last object.

The multi-hop task contains two subtasks: multi-hop questions and dialogues. Taking a triple chain as an example: *[(America, head of government, Biden), (Biden, spouse, Jill Biden)]*, we can write multi-hop questions templates like: "Who is the spouse of the head of government in {}?", and then generate the question "Who is the spouse of the head of government in America?". We prompt ChatGPT to construct corresponding templates. As for the dialogue task, we use the QA questions from single-hop questions and replace the intermediate objects with appropriate pronouns. For example, a dialogue could be: *Q: Who is the head of government in America? A: < LLM answer >. Q: Who is his spouse? A: < LLM answer >*. The intuition is that multi-hop questions are more common but require the model to have good reasoning abilities, while dialogues focus on testing if the model can know updated facts.

Afterward, we manually check if the templates are correctly constructed. We primarily focus on whether the templates' meanings are appropriate. A common mistake is reversing the relationship between subject and object, for example, *(subject, owner of, object)* means "subject is the owner of object", but ChatGPT might reverse this relationship. We manually correct all erroneous templates until all researchers agree that all templates are correct. Finally, we check for grammatical issues when filling subjects into templates, such as inconsistent pronouns. We find that the filled templates are reasonable in most cases, with an accuracy rate of 97.5%.

After completing the above steps, we have finished the entire process of data collection and validation, which means we have completed the construction process of the dataset.

A.2 SKEME Details

We introduced a novel caching mechanism and subject extraction to SKEME . Inspired by computer cache systems, the caching mechanism utilized by SKEME ensures that the stored knowledge is up-to-date while facilitating fast retrieval. Subject extract techniques allow SKEME to retrieve stored knowledge more precisely than previous techniques. In this section, we present the details of SKEME.

A.2.1 Entity Extraction

Entity extraction aims to extract key entities from the provided input. Previous research has extensively explored methods such as NER or entity-linking (Wu et al., 2020). We use LLMs to assist us in completing this task. Results indicate that this subtask can easily attain an accuracy rate exceeding 97% on SKEME. The accuracy statistics of entity extraction on SKEME are depicted in the table 6 and the prompt is in the figure 8.

Method	accuracy
GPT-3.5-turbo	98.1
Llama2	97.3
T5	99.8

Table 6: Accuracy for entity extraction, when using GPT-3.5-turbo and Llama2, we employ few-shot. When using T5, we finetune on FAME items for 5 epochs.

A.2.2 Knowledge Base Retrieval

The local knowledge base is stored in the form of a knowledge graph. When updating the local knowledge base, it can be automatically updated from the external database or manually injected with certain facts to reflect real-world changes. Such updates may require a considerable amount of time, but they can be done in parallel in arbitrary quantities and during idle times. Consequently, we did not explicitly evaluate the duration dedicated to this aspect.

A.2.3 Knowledge Rank and Utilization

Following previous works (Zhong et al., 2023; Zheng et al., 2023), we rank the retrieved knowledge based on similarity to the input and select the top-k knowledge. In our experiments, we set

```
Given a sentence, identify and extract the primary entity mentioned.
   Ensure that the entity extracted does not include any punctuation
   or special characters. Your response should consist of the entity'
   s name or title, such as a person's name, place, or organization.
   If the sentence contains multiple entities, select the most
   prominent or elevant one.

##few-shot
What is the inspiration behind the name of Seine-Maritime?
Seine-Maritime

Who is the cast member of Casino Royale?
Casino Royale

##total 8 few-shot
```

Figure 8: Prompt for entity extraction

$k = 1$. We prompt the model to use the retrieved knowledge for updating its output, which is shown in figure 9. To ensure that the model's output meets the task requirements, we added a task prompt before all prompts, as shown in Figure 10.

We utilized an off-the-shelf retrieval model (Izacard et al., 2021) to identify and rank the fact triplets, which allows us to avoid the training process.

A.3 Implementation Details of Baselines

For FT, MEMIT, and IKE, we use the framework provided by Wang et al. (2024a)[6]. For Mello, we used the original implementation but modified the prompt to fit tasks. [7]

FT Following previous works (Meng et al., 2023), We apply Fine-Tuning (FT) to the given layer of the model. For GPT2-XL, we select layer 0, and for GPT-J and Llama2, we choose layer 21.

MEMIT For GPT2-XL and GPT-J, we employ default hyperparameters. For Llama2, we update the parameters of layers $\{4, 5, 6, 7, 8\}$. Across all models, we calculate covariance statistics using 50,000 instances from Wikitext.

MeLLo The original method was designed for multi-hop questions. We redesign the prompt for each task while keeping the knowledge retrieval part unchanged.

IKE In the original paper, relevant facts were directly added to the prompt. To make a fair comparison, we removed this part and ensured that all facts were retrieved[8]. Our retrieval settings remained consistent with the original paper.

Other Baselines SERAC (Mitchell et al., 2022) and EREN (Chen et al., 2024) are two strong baselines. However, SERAC requires a significant amount of time for retraining (Yao et al., 2023), making it difficult to handle frequently updated requests. EREN is suitable for models that have undergone instruction fine-tuning, while SKEME and others focus on base models. Therefore, we did not include these two baselines in the comparison.

A.4 Other Benchmarks

To comprehensively evaluate model editing methods, we tested these methods on triviaQA (Joshi et al., 2017), Natural Questions (Kwiatkowski et al., 2019), FEVER (Thorne et al., 2018) and VitaminC (Schuster et al., 2021). TriviaQA and Natural Questions are commonly employed to assess the capabilities of LLMs (Touvron et al., 2023a). FEVER serves as a classic dataset for fact-checking, and VitaminC has been utilized in prior works to evaluate the effectiveness of model editing (Mitchell et al., 2022).

[6]https://github.com/zjunlp/EasyEdit

[7]https://github.com/princeton-nlp/MQuAKE

[8]The author of IKE's response to the issue: https://github.com/Zce1112zslx/IKE/issues/3

```
<task_prompt>

##few-shot
(Hiroshima Prefecture, head of government, Hidehiko Yuzaki)
Q: Who is the leader of the government in Hiroshima Prefecture?
A: Hidehiko Yuzaki.

(Naples, head of government, Gaetano Manfredi)
Q: Who is the current head of government for Naples?
A: Gaetano Manfredi.

##total 3 few-shot
<Retrieved triples>
Q: <query>
A:
```

Figure 9: Prompt for knowledge rank and utilization

```
completion: "Complete the sentence with a phrase."
qa: "Answer the question with one phrase."
local: "Answer the question with one phrase."
fill: "Identify the content within the parentheses and provide the
   missing information."
choose: "Choose the best answer."
fc: "Determine the veracity of the provided statement. Clearly output
    'True' if the statement is accurate and 'False' if it is not."
```

Figure 10: Prompt for task adaptation.

B Additional Experiment

B.1 Ablation Study

We conduct ablation experiments targeting different databases, demonstrating the necessity of selecting structured databases. As shown in Table 7, when editing the Llama2 model, we found that the limitations in retrieval accuracy of embedding databases resulted in significant side effects, leading to poor model performance.

Databases	SURE↑	EM↑	DD↓
Structured Database	90.54	90.54	0.00
Embedding Database	69.89	89.64	19.75

Table 7: Performance comparison of databases

C Example of Definition

Table 8 shows an example of the definition.

D Detailed Comparison between FAME and MQuAKE-T

Similar to MQuAKE-T (Zhong et al., 2023), FAME consists of genuine knowledge rather than constructed false information. However, MQuAKE-T is designed for multi-hop questions and is characterized by fewer relations and a smaller dataset size compared to ours (size: 1.8k vs. 8k, number of relations: 6 vs. 86), making it challenging to use it to enhance model capabilities. Therefore, we are currently the only benchmark available that can augment these capabilities.

Figure 11 shows a comparison between relation combinations in our data and MQuAKE (Zhong et al., 2023). It can be observed that our multi-hop questions cover a higher number of relationships, indicating that our data is more comprehensive.

Symbol	Example
s	America
r	head of government
o	Joe Biden
(x, y)	(Who is the current head of government for America?, Joe Biden)
$I(x, y)$	(The head of government for America is __, Joe Biden)
$EX(x, y)$	(Who is the spouse of the President of the United States?, Jill Biden)
$O(x, y)$	(What color is the Sky?, Blue)

Table 8: Example of definition. The examples for $I(x, y)$, $EX(x, y)$, and $O(x, y)$ represent elements in the set.

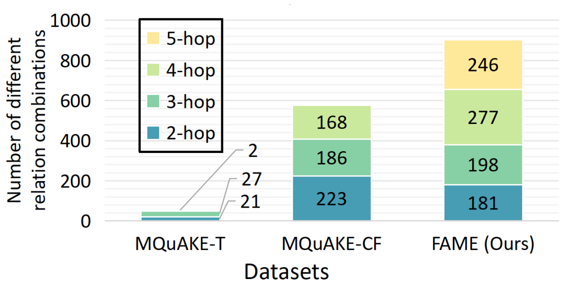

Figure 11: Comparison between multi-hop data in FAME and MQuAKE. The vertical axis of the graph represents the number of relation combinations. FAME encompasses a greater number of combinations, including 5-hop questions, which effectively demonstrates the enhanced diversity of our dataset.

E Examples and Statistics of FAME

E.1 Examples of FAME

Figure 12 shows an example of a single-hop question in FAME. The object label serves as the golden answer, with different queries used for corresponding tasks. Locality queries evaluate the model's output using similar (sharing the same relation) but unrelated queries. The locality object label represents the golden answer for the locality query.

For multi-hop questions, take the triple chain as an example: *[(America, head of government, Biden), (Biden, spouse, Jill Biden)]*. Facing questions like *"Who is the spouse of the head of government in America?"*, when Trump is updated to Biden, the answer to this question should change accordingly. In this example, "Biden" is the first object, "Jill Biden" is the second object, and the query is "Who is the spouse of the head of government in America?". Figure 13 shows an example of a 2-hop question in FAME.

E.2 Statistics of FAME

Table 9 and Table 10 display the labels of the relations we utilized in DBpedia and Wikidata, along with the number of data items associated with each relation.

F Detailed Analysis of SURE

Model editing aims to precisely correct the knowledge within an LLM without affecting unrelated knowledge. Therefore, measuring the accuracy of updates and the side effects is paramount. However, scenarios may have varying requirements for these two aspects. For example, in critical fields like medical diagnostics or legal advice, even minor inaccuracies can have severe repercussions, so we place more emphasis on the model's performance within the scope of knowledge. Conversely, for a well-performing model, the focus shifts toward correcting misconceptions while maintaining its existing capabilities. Thus, it is necessary to consider both the in-scope accuracy and out-of-scope side effects and adapt to the demands of different scenarios.

To address this challenge, we propose SURE, a metric that comprehensively measures the effectiveness of model editing and estimates the capabilities of the edited model in real-world scenarios. SURE considers both accuracy and side effects and introduces hyperparameters α and β to adjust the importance of accuracy and side effects. For instance, selecting $\alpha > \beta$ signifies a higher priority for in-domain knowledge, which is advantageous in sectors like healthcare, law, or when editing models on private datasets. Conversely, setting $\alpha < \beta$ suggests that minimizing side effects is more crucial, which is suitable for extensively used models already delivering online services. Thus, by adjusting the hyperparameters, FAME can evaluate the practical capabilities of the model in different usage scenarios.

```
"subject_label": "Sioux Falls",
"relation_label": "head of government",
"object_label": "Paul Ten Haken",
"localitysubjectLabel": "Viarmes",
"localityobjectLabel": "William Rouyer"
"qa_query": "Who is the current head of government for Sioux Falls?",
"fill_query": "() is the head of government in Sioux Falls.",
"completion_query": "The head of government for Sioux Falls is",
"choose_query": "Who holds the position of head of government in
   Sioux Falls?\nA:Theodor Leutwein B:Lothar von Trotha C:Paul Ten
   Haken",
"FC_query": "Determine whether the proposition is true.\nProposition:
   The head of government for Sioux Falls is Paul Ten Haken."
"locality_query":"Who is the current head of government for Viarmes?"
```

Figure 12: An example of single-hop questions in FAME

```
"s1_label": "Amtrak",
"relation_label_1": "owner of",
"o1_label": "Route 128 station",
"relation_label_2": "located in the administrative territorial entity
   ",
"o2_label": "Westwood",
"qa_query_1": "What entities does Amtrak owns?",
"qa_query_2": "In which administrative territorial entity is Route
   128 station located?",
"MultihopQA_query":"In which administrative territorial entity is the
    entities {} owns located?"
```

Figure 13: An example of multi-hop questions in FAME

DBpedia Label	Wikidata Label	Relevant Triples
prime minister	head of government	199
highway system	transport network	189
country	country	298
birth place	place of birth	251
death place	place of death	246
sex	sex or gender	351
father	father	271
mother	mother	243
spouse	spouse	232
citizenship	country of citizenship	226
capital	capital	230
currency	currency	217
child	child	237
family	family	216
team	member of sports team	249
film director	director	209
discoverer	discoverer or inventor	262
alma mater	educated at	238
architect	architect	248
anthem	anthem	255
composer	composer	237
editor	editor	223
discipline	field of work	230
party	member of political party	219
employer	employer	220
illustrator	illustrator	204
founded by	founded by	218
league	league	242
place of burial	place of burial	240
maintained by	maintained by	224
owner	owned by	242
county	located in the administrative territorial entity	218
movement	movement	229
genre	genre	223
named after	named after	265
religion	religion or worldview	263
based on	based on	230
architectural style	architectural style	225
headquarter	headquarters location	239
starring	cast member	251
chief executive officer	chief executive officer	147
creator (agent)	creator	259
builder	manufacturer	223

Table 9: Statistics of relations

DBpedia Label	Wikidata Label	Relevant Triples
crosses	crosses	253
developer	developer	237
doctoral advisor	doctoral advisor	211
doctoral student	doctoral student	248
construction material	made from material	223
inflow	inflows	258
IATA code	IATA airline designator	225
ICAO code	ICAO airline designator	233
record label	record label	230
license	copyright license	257
manager	head coach	225
designer	designed by	217
cinematography	director of photography	253
is part of	part of	237
original language	original language of film or TV show	248
launch vehicle	space launch vehicle	258
computing platform	platform	253
Game Engine	software engine	240
position	position played on team / speciality	221
industry	industry	246
colour	color	253
homeport	shipping port	252
general manager	general manager	131
formation date	inception	208
discovery date	time of discovery or invention	244
start date	start time	264
battle	conflict	258
diocese	diocese	224
cover artist	cover art by	251
distributor	distributed by	243
notable work	notable work	233
crew member	crew member(s)	239
editing	film editor	228
fuel system	fuel system	146
instrument	instrument	234
eye color	eye color	240
birth name	birth name	225
owning organisation	owner of	245
length (μ)	length	235
elevation (μ)	elevation above sea level	256
area total (m^2)	area	230
runtime (s)	duration	224
military branch	military unit	231

Table 10: Statistics of relations (Continued)