SWEA: Updating Factual Knowledge in Large Language Models via Subject Word Embedding Altering

Xiaopeng Li, Shasha Li*, Shezheng Song, Huijun Liu, Bin Ji, Xi Wang, Jun Ma*, Jie Yu*, Xiaodong Liu*, Jing Wang, Weimin Zhang

National University of Defense Technology
Changsha, Hunan 410073 China
{xiaopengli, shashali, ssz614, jibin, liuhuijun, wx_23ndt, yj, majun, liuxiaodong, wangjing}@nudt.edu.cn,
wmzhang104@139.com

Abstract

The general capabilities of large language models (LLMs) make them the infrastructure for various AI applications, but updating their inner knowledge requires significant resources. Recent model editing is a promising technique for efficiently updating a small amount of knowledge of LLMs and has attracted much attention. In particular, local editing methods, which directly update model parameters, are proven suitable for updating small amounts of knowledge. Local editing methods update weights by computing least squares closed-form solutions and identify edited knowledge by vector-level matching in inference, which achieve promising results. However, these methods still require a lot of time and resources to complete the computation. Moreover, vector-level matching lacks reliability, and such updates disrupt the original organization of the model's parameters. To address these issues, we propose a detachable and expandable Subject Word Embedding Altering (SWEA) framework, which finds the editing embeddings through token-level matching and adds them to the subject word embeddings in Transformer input. To get these editing embeddings, we propose optimizing then suppressing fusion method, which first optimizes learnable embedding vectors for the editing target and then suppresses the Knowledge Embedding Dimensions (KEDs) to obtain final editing embeddings. We thus propose SWEA⊕OS method for editing factual knowledge in LLMs. We demonstrate the overall state-of-the-art (SOTA) performance of SWEA⊕OS on the COUNTERFACT and zsRE datasets. To further validate the reasoning ability of SWEA⊕OS in editing knowledge, we evaluate it on the more complex RIPPLEEDITS benchmark. The results demonstrate that SWEA⊕OS possesses SOTA reasoning ability.

1 Introduction

Large language models (LLMs), with their rich reserve of pre-trained knowledge, play a pivotal role in the current AI landscape (Li et al. 2023a; Zhao et al. 2023). The knowledge pre-trained in LLMs is solidified in their parameters, meaning that any outdated or incorrect knowledge within the LLMs can only be updated through parameter updates. However, given that the training of LLMs relies heavily on GPUs and consumes a significant amount of electricity, retraining to update even small amounts of information can be costly. Consequently, researchers have started to explore model editing methods (Yao et al. 2023; Meng et al. 2022a; Mitchell et al. 2022; Zhang et al. 2024; Wang et al. 2024; Tian et al. 2024) aiming to update a small amount of knowledge of LLMs more efficiently.

The purpose of model editing is to insert, update, and delete target knowledge while avoiding editing non-target knowledge to preserve the original capabilities of LLMs. Current editing methods mainly edit model through three approaches (Wang et al. 2023): adding additional modules (Huang et al. 2023; Dong et al. 2022; Hartvigsen et al. 2022), global optimization (Zhu et al. 2020; Mitchell et al. 2022), and local editing (Meng et al. 2022a,b; Li et al. 2023b). Methods of adding additional modules involves incorporating adapters within or external to the LLMs for storing edited instances, which increases the inference load. In contrast, global optimization and local editing methods write the editing information into the model weights, maintaining the same inference cost as the original model. However, using global optimization methods for model editing is prone to overfitting (Meng et al. 2022b) because model editing often only requires updating a small amount of knowledge. Local editing methods view the model editing as a least squares problem, which is more suitable for updating a small amount of knowledge. Therefore, in this paper, we focus on local editing methods.

Local editing methods first select the critical layers that store knowledge (Meng et al. 2022a), optimize the knowledge representation with the editing knowledge as the objective, then calculate the keys of the editing knowledge and the original knowledge, and finally update the weights of critical layers by solving the least squares problem. These methods have achieved remarkable results in model editing tasks. However, they still exsit three issues: ***(1) Lack of efficiency:*** these methods need to spend a lot of time and resources to compute all the vectors needed to solve the least squares problem (Meng et al. 2022a,b; Li et al. 2023b); ***(2) Lack of reliability:*** in local editing methods, we observe that even when all the target knowledge representations are already aligned with the editing goal, their editing success rate is still far from expectation. Meanwhile, LLMs edited by these methods are prone to misidentifying unedited knowledge as edited knowledge, reducing the usability of edited LLMs. This might be due to the fact that using vector-level match-

*Corresponding Author.

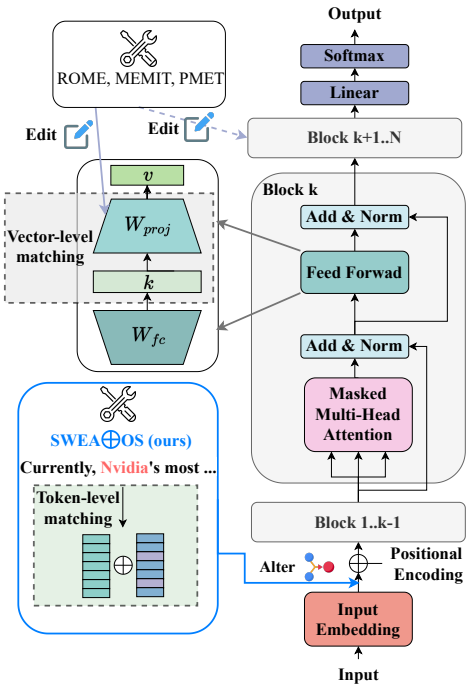

Figure 1: Difference between our method and existing local editing methods. Our method focuses on altering the word embedding for the input via token-level matching, while existing local editing methods edit Feed Forward Network (FFN) and identify editing knowledge by vector-level matching. Mismatching is more likely to occur in vector-level matching, which leads to erroneous recognition of editing knowledge.

ing to identify the editing knowledge in the updated weights is not entirely reliable, since vector-level matching struggles to distinguish between two very similar vectors (Gionis et al. 1999; Zhang et al. 2023); ***(3) Lack of protection:*** due to the high complexity and incomplete transparency of LLMs themselves, exactly updating their weights perfectly by solving the least squares problem is challenging. Consequently, the original organization of the edited model's parameters is disrupted (Li et al. 2023c), thereby affecting the general applications of LLMs (Gu et al. 2024).

In view of these issues, ***(1)*** we propose a novel model editing method, SWEA⊕OS, which alters subject word embedding by adding it with editing embeddings obtained by **O**optimizing then **S**uppressing (OS) fusion method in **S**ubject **W**ord **E**mbedding **A**ltering (SWEA) framework. SWEA⊕OS only requires computing editing embeddings, therefore it is more efficient. The difference between SWEA⊕OS and existing local editing methods is illustrated in Figure 1. ***(2)*** The SWEA framework identifies editing knowledge instances through token-level matching that is more reliable than vector-level matching because it is sensitive to even single-character changes. The OS fusion method get the editing embeddings through: a) optimizing learnable embedding vectors to achieve editing objectives, b) suppressing the subject's Knowledge Embedding Dimensions (KEDs) which are special dimensions related to specific knowledge in word embeddings. The suppressing step is designed to mitigate the influence of the subject's KEDs on the expression of new knowledge. ***(3)*** Unlike local editing methods that directly modify weights, the SWEA framework is detachable and embedded into the embedding layer of LLMs, which protects the original weights of LLMs. It is also expandable, which can be combined with different fusion methods for model editing. In addition, the SWEA framework edits knowledge by altering the subject word embedding, which ensures the same inference load as the original model.

We demonstrate our method is both efficient and effective in GPT-J (6B) (Wang and Komatsuzaki 2021) and Llama-2 (7B) (Touvron et al. 2023) across two datasets and one benchmark. In detail, comparative experiments on GPT-J and Llama-2 show that the SWEA⊕OS method demonstrates overall SOTA performance. On the COUNTERFACT dataset, SWEA⊕OS increases the Score by 5.8% on GPT-J and 7.7% on Llama-2 compared to the most advanced baseline. The SWEA⊕OS method also shows the best reasoning performance on the RippleEdits benchmark (Cohen et al. 2023), indicating that the knowledge edited by the SWEA OS method has stronger consistency.

Our contributions to the work are as follows:

- We propose a detachable and expandable SWEA framework, which can be combined with different fusion methods for model editing and ensures the same inference cost as the original model.
- We introduce the OS fusion method. It optimizes learnable embedding vectors for editing targets and then suppresses KEDs of subject to alleviate the impact of KEDs of subject word embeddings on editing effects.
- Combing the OS fusion method with SWEA, we propose SWEA⊕OS for editing factual knowledge in LLMs. We demonstrate the overall superior performance of SWEA⊕OS on COUNTERFACT and zsRE datasets and a more complex RIPPLEEDITS benchmark.

2 Related Work

2.1 Model Editing

Model editing is currently an emerging research hotspot, with various model editing methods and benchmarks being proposed successively (Yao et al. 2023; Zhang et al. 2024; Wang et al. 2023; Deng et al. 2024; Wang et al. 2024; Li et al. 2024a). The model editing task was first proposed in (Zhu et al. 2020), where they proposed a constrained fine-tuning method for this task, which imposes a constraint on the fine-tuning weights to reduce interference with original knowledge. Unlike constrained fine-tuning, recent methods utilize meta-learning to update weights (De Cao, Aziz, and Titov 2021; Mitchell et al. 2022; Tan, Zhang, and Fu 2023). These methods train a hypernetwork that indirectly updates weights using global gradient information. However, since the model editing task aims to correct a small portion of errors within the model's internal memories, the data for model editing is usually few, making methods that update

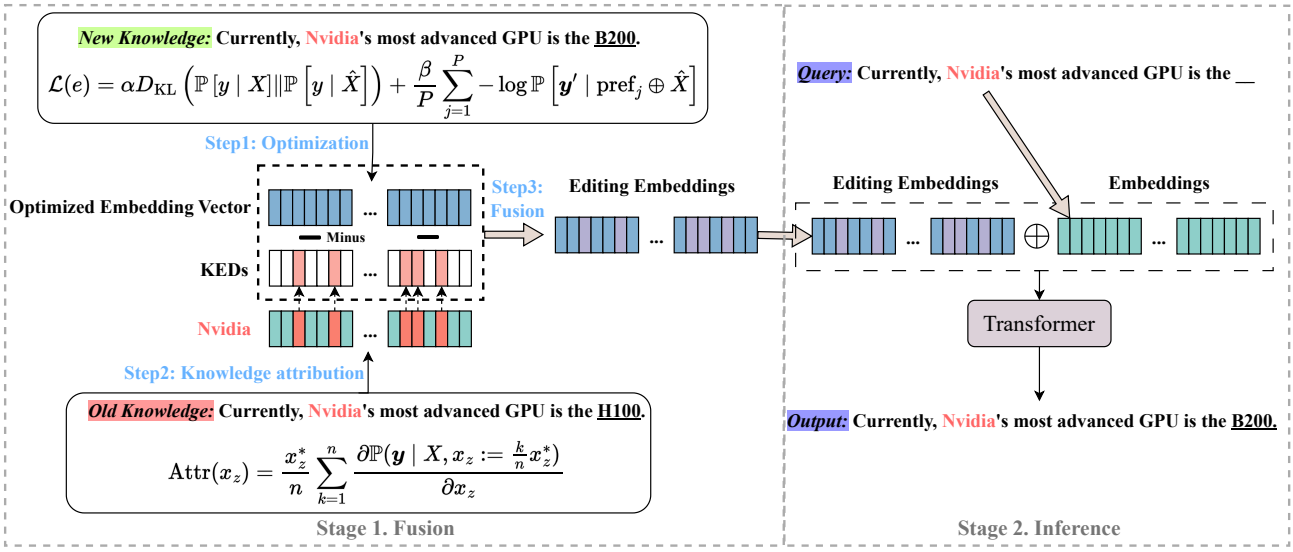

Figure 2: Overview of SWEA⊕OS. In fusion stage, we first optimize a learnable embedding vector for target knowledge "Currently, Nvidia's most advanced GPU is the B200." Second, using knowledge attribution method, we find the KEDs of 'Nvidia' regarding "its most advanced GPU". Finally, we fuse the optimized embedding vector with these KEDs subtracted to obtain the editing embeddings. In inference stage, we add these editing embeddings to the embedding of the subject 'Nvidia' for inference.

weights using global gradients prone to overfitting. Some methods also add additional modules to perform model editing. They usually add a smaller model outside LLMs (Huang et al. 2023), embed an adapter within LLMs (Hartvigsen et al. 2022), or add editing information to the input for model editing. But these methods not only increase the inference burden but also add to the complexity of the system.

In contrast, local editing methods, from the perspective of interpretability, directly update the Feed Forward Network (FFN) of key-value memories (Geva et al. 2021) using a closed-form solution of least squares (Meng et al. 2022a,b; Li et al. 2023b), which is less prone to overfitting and more lightweight. However, these approaches still require a lot of time and resources to update weights, the vector-level matching in model weights is not always reliable, and recent works find that these approaches can cause irreversible damage to the model's generalization capability due to the updating of model weights (Gu et al. 2024). Unlike existing local editing methods, SWEA⊕OS alters the subject word embeddings in the input through token-level matching through editing embeddings obtained by OS fusion method, making it more efficient and reliable.

2.2 Explanation of Word Embedding

Word embedding is a fundamental component in LLMs for processing natural language, where each token typically corresponds to a high-dimensional dense vector. Researchers have sought to understand the interpretable concepts associated with these dense vectors by categorizing the dimensions into specific concepts (Şenel et al. 2018; Balogh et al. 2020) or by projecting word embeddings into more interpretable vector spaces (Park, Bak, and Oh 2017; Simhi and Markovitch 2023). (Şenel et al. 2018) introduced the SEMCAT dataset and used statistical methods to classify different dimensions of word embedding into 110 semantic categories. (Balogh et al. 2020) assigned common-sense concepts to each dimension of word embedding. (Simhi and Markovitch 2023) mapped word embedding to a concept space understandable by humans using vector similarity. These works enhance our understanding of word embedding in terms of semantic concepts. However, they did not discuss the relationship between the dimensions of word embedding and factual knowledge. In contrast to their methods, we use knowledge attribution methods to identify the corresponding knowledge embedding dimensions (KEDs) for subject-specific facts in word embedding dimensions and suppress these KEDs to improve editing effects.

3 Preliminaries

3.1 Language Modeling

LLMs process discrete text by firstly embedding it into continuous vectors. After processing by Transformers, a probability distribution on the vocabulary is finally obtained under the action of the Softmax function. Formally, a discrete text $\mathcal{T}$ is first converted into token ids $T = \mathrm{tok}(\mathcal{T})$ by the tokenizer. Next, the embedding layer E of the LLMs maps each token id of T to a vector $x \in R^{1\times h}$, where h is the dimension size of the Transformer's hidden layer. Assuming the length of the token ids is l, then $E(T) = X \in R^{l\times h}$ where $X = [x_1, x_2, ..., x_l]$ is the continuous vector of text $\mathcal{T}$. Then the Transformers of the LLMs process X layer by layer, finally obtaining a probability distribution on the vocab:

$$\mathbb{P}(X) = \mathrm{Softmax}(\mathrm{Transformers}(X)) \quad (1)$$

3.2 Model Editing Task

Prior work expresses factual knowledge as a triple (s, r, o) (Meng et al. 2022b), where s denotes the subject, r denotes the relation, and o denotes the object. The purpose of model editing is:

$$(s, r, o) \rightarrow (s, r, o') \tag{2}$$

where o' is another object different from o. At the same time, model editing should protect other knowledge not being changed. For convenience, we express the factual knowledge with a pair $(\mathcal{T}, \mathcal{Y})$, where $\mathcal{T}$ is a sentence composed of s and r, and $\mathcal{Y}$ is o which is the continuation of the above sentence. Then the model editing task can be formally expressed as:

$$(\mathcal{T}, \mathcal{Y}) \rightarrow (\mathcal{T}, \mathcal{Y}') \tag{3}$$

Batch Editing means editing $n > 1$ factual knowledge at the same time during a single run of the editing method:

$$\sum_{1}^{n}(\mathcal{T}, \mathcal{Y}) \rightarrow \sum_{1}^{n}(\mathcal{T}, \mathcal{Y}') \tag{4}$$

Sequential Editing means carrying out multiple consecutive edits on a single model:

$$(\mathcal{T}, \mathcal{Y}) \rightarrow (\mathcal{T}, \mathcal{Y}') \rightarrow ... \rightarrow (\mathcal{T}, \mathcal{Y}^*) \tag{5}$$

Sequential Batch Editing means performing multiple consecutive batch edits on a single model:

$$\sum_{1}^{n}(\mathcal{T}, \mathcal{Y}) \rightarrow \sum_{1}^{n}(\mathcal{T}, \mathcal{Y}') \rightarrow ... \rightarrow \sum_{1}^{n}(\mathcal{T}, \mathcal{Y}^*) \tag{6}$$

4 Methodology

In this section, we explain what is the SWEA framework and how our proposed SWEA⊕OS method is used to update the factual knowledge of LLMs. SWEA⊕OS consists of two stages: (1) Fusion: we use the OS fusion method to compute the editing embeddings needed to update the factual knowledge for the subject; (2) Inference: in the SWEA framework, the input embedding is altered with the matched editing embeddings to obtain the final input embeddings. We detail these two stages in the subsections below, an overview of SWEA⊕OS is shown in Figure 2.

4.1 Optimizing then Suppressing Fusion Method

Word embeddings are dense continuous vectors (Zhao et al. 2023). Some works show that their different dimensions contain specific information (Li, Monroe, and Jurafsky 2016; Şenel et al. 2018). Motivated by these, we assume that certain dimensions of word embeddings of a subject correspond to specific factual knowledge about the subject in LLMs. For convenience, we name these dimensions as knowledge embedding dimensions (KEDs). For example, the dimensions $(26, 123, 336, 1024)$ of the word embedding of the subject "Nvidia" are KEDs that correspond to the factual knowledge "Nvidia was founded by Jensen Huang." Under this assumption, we aim to alter KEDs of the subject to control the factual knowledge about the subject in LLMs.

Due to word embeddings not being fully explained, directly altering KEDs to update factual knowledge is very difficult. We propose appending learnable embedding vectors to the subject's word embeddings and optimizing these vectors to get optimized embeddings related to the editing target. During inference, simply adding the optimized embedding vectors to the subject's word embeddings can update factual knowledge. However, since the KEDs of the subject's word embeddings corresponding to factual knowledge still work, this may affect the knowledge expression of the optimized embedding vectors, leading to a decrease in editing effects. We thus suppress the KEDs of the original subject's word embeddings. Therefore, we propose the optimizing then suppressing fusion method, which first optimizes learnable embedding vectors to achieve editing objectives, then suppresses the KEDs of the original subject's word embeddings.

Formally, suppose X is the text embedding of $\mathcal{T}$; $\boldsymbol{y}$ and $\boldsymbol{y}'$ are all tokens of $\mathcal{Y}$ and $\mathcal{Y}'$ respectively. To change the factual knowledge of the model from $(\mathcal{T}, \mathcal{Y})$ to $(\mathcal{T}, \mathcal{Y}')$, inspired by previous work (Meng et al. 2022b; Li et al. 2023b), we add learnable embedding vectors e to the representation of the subject S in X to get $\hat{X}$, and use the following loss function to optimize and maximize the probability of $\boldsymbol{y}'$:

$$\begin{aligned}\mathcal{L}(e) = \alpha D_{\mathrm{KL}}\left(\mathbb{P}\left[y \mid X\right] \| \mathbb{P}[y \mid \hat{X}]\right) + \\ \frac{\beta}{P}\sum_{j=1}^{P} -\log \mathbb{P}\left[\boldsymbol{y}' \mid \mathrm{pref}_j \oplus \hat{X}\right]\end{aligned} \tag{7}$$

Here D_{KL} is the KL divergence used to constrain the probability distribution after adding the learnable embedding vector e; to enhance the generalization of the learnable embedding vectors e, we prepend P prefixes (i.e., pref_j) generated by the model to $\hat{X}$, where $\oplus$ indicates the concatenation operation; α and β are two hyperparameters used to regulate the strength between preserving original knowledge and learning new knowledge during the optimization.

We use the knowledge attribution method (Dai et al. 2021) to find the KEDs of subject S. Let x_z represent any one embedding vector in $x^S = [x_s^S, ..., x_e^S] \in R^{|S| \times h}$, the knowledge attribution of the embedding can be formally expressed as:

$$\mathrm{Attr}(x_z) = \frac{x_z^*}{n}\sum_{k=1}^{n}\frac{\partial \mathbb{P}(\boldsymbol{y} \mid X, x_z := \frac{k}{n}x_z^*)}{\partial x_z} \tag{8}$$

Here, x_z^* represents the original value of the embedding vector; n is the number of steps for the Riemann integration, and we follow (Dai et al. 2021) and set $n = 20$; $\mathbb{P}(\boldsymbol{y} \mid X, x_z := \frac{k}{n}x_z^*)$ represents the probability of the model generating $\boldsymbol{y}$ after replacing x_z with $\frac{k}{n}x_z^*$. After obtaining the attribution scores of all embedding dimensions of the subject S, we retain those embedding dimensions that exceed t times the maximum attribution score as the KEDs K_D. Finally, we subtract γ times the value of the original embedding vectors x^S corresponding to K_D from the optimized embedding vector e to obtain the final editing embeddings e^S:

$$e^S = e - \gamma \mathbb{O}_{\setminus K_D} \odot x^S \tag{9}$$

where $\mathbb{O}_{\setminus K_D}$ represents a vector with all positions as 0 except for the positions included in K_D which are 1; $\odot$ denotes element-wise multiplication.

4.2 Subject Word Embedding Altering Framework

Subject Word Embedding Altering (SWEA) framework merges the editing embeddings $e^S = [e_s^S, ..., e_e^S] \in R^{|S| \times h}$ about the subject S with the subject embedding $x^S = [x_s^S, ..., x_e^S] \in R^{|S| \times h}$ from the input text embedding $X \in R^{l \times h}$. Here, $|S|$ is the token length of the subject, and x_s^S and x_e^S represent the first and last token of the subject in the input X, respectively. Therefore, in SWEA, the final input used by the model for inference is:

$$X = [x_0, ..., x_s^S + e_s^S, ..., x_e^S + e_e^S, ..., x_l] \quad (10)$$

Given that each subject's token ids are unique, we use these token ids as keys to index the editing embeddings. Specifically, after obtaining the editing embedding e^S of the subject S, we cache e^S using the token ids as key. For S with only one token id, we use this id as the key directly, and for S with multiple token ids, we concatenate the token ids of S using '_'. For example, the token ids of the subject 'San Francisco' are $[2986, 6033]$, so its key is '2986_6033'. For convenience, we currently adopt the file caching method, which can be easily extended to a vector database. SWEA can easily implement batch editing, it can obtain e^S for multiple subjects and then cache these e^S in editing embeddings collection $\mathcal{E}$. SWEA can also implement sequential editing and sequential batch editing. It caches past editing requests and recomputing e^S for the subjects when the editing requests are updated. During the inference stage, we carry out the longest continuous matching for the continuous combination of the token ids of each input and the keys in $\mathcal{E}$, and add successful matched caches to matched tokens's embeddings using (10). The token-level matching and embedding altering algorithm of the above process can be found in Appendix A (See (Li et al. 2024b)). Note that some subjects may have aliases. Currently, we are primarily focused on introducing a new way for model editing, so SWEA currently only considers cases where the subject is unique. However, the SWEA framework can easily be adapted to include an alias list for each subject to identify them.

5 Experiments

5.1 Experimental Setup

Datasets and Large Language Models We conducted edits on GPT-J (6B) (Wang and Komatsuzaki 2021) and Llama-2 (7B) (Touvron et al. 2023) on two datasets, COUNTERFACT (Meng et al. 2022a), zsRE (De Cao, Aziz, and Titov 2021; Mitchell et al. 2022) and the RIPPLEEDITS benchmark (Cohen et al. 2023). All metrics of the above datasets and benchmark are described in Appendix B. COUNTERFACT dataset is a completion task, which contains a total of 21,919 counterfactual editing instances. MEMIT (Meng et al. 2022b) filtered out the counter-logical fact editing in this dataset. To ensure the same experimental setting, we also only use the filtered 20,877 editing instances.

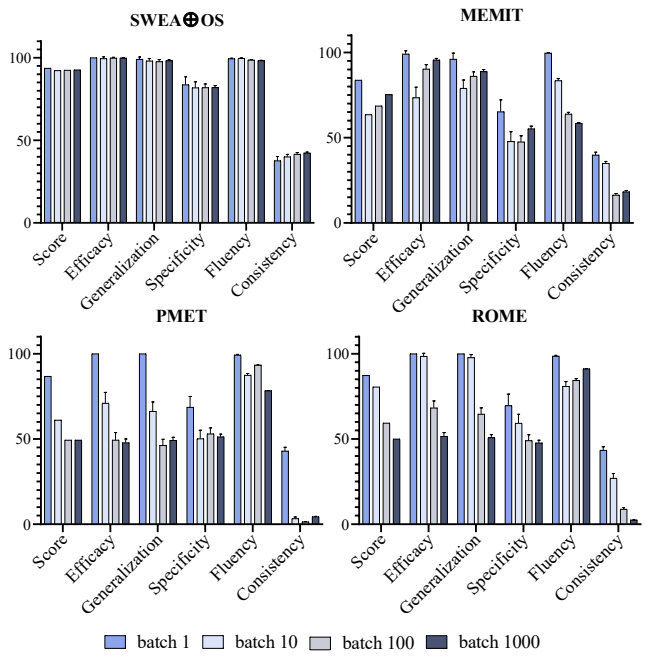

Figure 3: Results of sequential batch editing of SWEA⊕OS, PMET, MEMIT, and ROME. To better display the results, we divide the fluency by the original fluency (i.e., 622.4) and then multiply by 100 to make it fall between 0 and 100.

zsRE dataset is a QA task, for which we use 10,000 editing instances extracted from (Meng et al. 2022a) to conduct editing. RippleEdits is a benchmark for testing the multi-hop reasoning ability of post-edit models, including RECENT, RANDOM and POPULAR subsets. RECENT mainly evaluates the ability of the model's editing method to insert knowledge, while the latter two mainly evaluate the ability to edit knowledge. Since we currently only focus on updating the knowledge of the model, we only use the two subsets of rippleEdits, RANDOM and POPULAR.

Baselines We compared SWEA⊕OS with the global optimization method Constrained Fine-Tuning (FT+W) (Zhu et al. 2020), MEND (Mitchell et al. 2022), MALMEN (Tan, Zhang, and Fu 2024), adding additional modules method GRACE (Hartvigsen et al. 2022), and the local editing methods ROME (Meng et al. 2022a), MEMIT (Meng et al. 2022b), PMET (Li et al. 2023b) on the COUNTERFACT and zsRE datasets. On the RANDOM and POPULAR subsets of rippleEdits, we compared with local editing methods ROME, MEMIT. Experimental details can be found in Appendix C.

5.2 Experiments on COUNTERFACT and zsRE Datasets

We first test the batch editing performance on the COUNTERFACT and zsRE datasets. We then test the scaling-up editing performance on the COUNTERFACT dataset. Considering that sequential editing is a subset of sequential batch editing, we perform sequential batch editing directly

Editor	Score	Efficacy	Generalization	Specificity	Fluency	Consistency
GPT-J	22.4	15.2 (0.7)	17.7 (0.6)	83.5 (0.5)	622.4 (0.3)	29.4 (0.2)
FT-W	67.6	99.4 (0.1)	77.0 (0.7)	46.9 (0.6)	293.9 (2.4)	15.9 (0.3)
MEND	23.1	15.7 (0.7)	18.5 (0.7)	**83.0** (0.5)	618.4 (0.3)	31.1 (0.2)
ROME	50.3	50.2 (1.0)	50.4 (0.8)	50.2 (0.6)	589.6 (0.5)	3.3 (0.0)
MEMIT	85.8	98.9 (0.2)	88.6 (0.5)	73.7 (0.5)	619.9 (0.3)	40.1 (0.2)
GRACE	26.7	30.6 (0.9)	17.3 (0.6)	**83.0** (0.5)	618.1 (0.3)	29.3 (0.2)
PMET	86.2	99.5 (0.1)	92.8 (0.4)	71.4 (0.5)	**620.0** (0.3)	40.6 (0.2)
SWEA⊕OS	**91.2**	**99.6** (0.1)	**98.3** (0.2)	79.0 (0.5)	609.5 (0.7)	**42.3** (0.2)
Llama-2	20.5	14.8 (0.7)	15.0 (0.6)	82.4 (0.5)	604.3 (0.3)	25.4 (0.2)
FT-W	65.4	**99.8 (0.1)**	84.9 (0.6)	41.5 (0.7)	546.9 (0.2)	20.0 (0.1)
ROME	50.5	51.3 (1.0)	50.0 (0.8)	50.2 (0.6)	488.1 (0.2)	2.6 (0.0)
MEMIT	69.6	81.5 (0.8)	55.4 (0.8)	78.3 (0.5)	602.9 (0.2)	27.8 (0.2)
GRACE	29.2	29.8 (0.9)	15.0 (0.6)	**82.2** (0.5)	**605.2** (0.3)	25.3 (0.2)
PMET	83.2	97.1 (0.3)	87.8 (0.5)	69.5 (0.6)	599.4 (0.3)	34.7 (0.2)
SWEA⊕OS	**89.6**	98.4 (0.2)	**93.5** (0.4)	79.3 (0.5)	600.5 (0.5)	**35.0** (0.2)

Table 1: Results of 10,000 edits on GPT-J and Llama-2 on the COUNTERFACT dataset. Within the parentheses is the 95% confidence interval.

Editor	Score	Efficacy	Generalization	Specificity
GPT-J	26.0	26.4 (±0.6)	25.3 (±0.5)	26.8 (±0.5)
FT-W	14.3	57.9 (±0.7)	56.8 (±0.7)	5.7 (±0.5)
MEND	20.0	19.4 (±0.5)	18.6 (±0.5)	22.4 (±0.5)
MALMEN	37.3	76.1(±0.7)	72.3(±0.7)	18.6(±0.4)
ROME	1.1	9.2 (±0.8)	7.9 (±0.8)	0.4 (±0.2)
MEMIT	50.2	92.7 (±0.3)	86.7 (±0.5)	26.7 (±0.5)
GRACE	31.3	47.8 (±0.6)	26.5 (±0.5)	**26.8** (±0.5)
PMET	47.6	86.4 (±0.4)	81.5 (±0.5)	25.5 (±0.3)
SWEA⊕OS	**51.0**	**96.0** (±0.3)	**89.7** (±0.2)	**26.8** (±0.2)
Llama-2	11.9	11.5(±0.3)	11.1 (±0.3)	13.3 (±0.2)
FT-W	11.7	13.8 (±0.6)	13.1 (±0.5)	9.2 (±0.4)
ROME	4.3	3.9 (±0.8)	3.7 (±0.8)	5.8 (±0.3)
MEMIT	23.1	45.6 (±0.4)	40.9 (±0.5)	12.0 (±0.5)
GRACE	14.9	23.7 (±0.4)	11.8 (±0.6)	**13.3** (±0.5)
PMET	23.9	48.1 (±0.3)	**45.0** (±0.4)	12.1 (±0.3)
SWEA⊕OS	**25.5**	**50.7** (±0.4)	44.0 (±0.3)	**13.3** (±0.3)

Table 2: Results of 10,000 edits on GPT-J and Llama-2 on the zsRE dataset. To ensure a fair comparison, we reproduced baselines except MEND (non-reproducible) under this setting.

on the COUNTERFACT dataset. We also compare the execution time of the SWEA⊕OS method with that of baselines and analysis the inference latency introduced by the SWEA framework in Appendix D.

Batch Editing Results The results of editing GPT-J and Llama-2 on the COUNTERFACT and zsRE datasets are presented in Tables 1 and 2, respectively. SWEA⊕OS achieved the overall best results. Whether on the COUNTERFACT or zsRE datasets, Efficacy, Generalization, Specificity, and Consistency of SWEA⊕OS shows substantial improvement over previous model editing methods. This indicates that SWEA⊕OS is a very effective editing method. FT-W has a very high editing success rate, but the generalizability of the edited knowledge is poor, and the model's generative capability is severely compromised due to overfitting. When editing the GPT-J model, although MEND and GRACE exhibit the best specificity, their poor generalization affects their overall performance. When editing the Llama-2 model, GRACE shows similar results. Overall, we have demonstrated through this set of experiments that SWEA⊕OS is a highly effective model editing method. In addition, we show the results of SWEA⊕OS and baselines performing 1, 10, 100, 1000, and 10,000 edits respectively on the COUNTERFACT dataset in Appendix E.2. Additionally, in Appendix E.3, we present the qualitative results of the model generating facts after being edited on the COUNTERFACT dataset.

Sequential Batch Editing Results We use SWEA⊕OS, PMET, MEMIT, and ROME to perform sequential batch editing on the GPT-J model in the COUNTERFACT dataset. The number of sequences are 100, 20, 5, 2 with corresponding batch sizes of 1, 10, 100, 1000, respectively. The results are shown in Figure 3, indicating that SWEA⊕OS performs the most stable performance in sequential batch editing. The performance of SWEA⊕OS in sequential batch editing only show a slight decline as the batch size increased. When the editing batch is 1 and 1000, the scores of SWEA are 93.22 and 93.01, respectively. In contrast, the performances of PMET, MEMIT, and ROME are very unstable. The score of PMET and ROME decreased by 43.08% and 42.74% from an editing batch 1 to an editing batch 1000, respectively.

5.3 Experiments on RIPPLEEDITS

Since RIPPLEEDITS tests the model's ability to reason using edited knowledge, we first need to ensure that the model itself has the corresponding reasoning ability. Essentially, we need to edit the facts known to the model. To ensure this, each LLMs dataset needs to be filtered before RIPPLEEDITS testing. We followed the filtering steps of RIPPLEEDITS, fi-

Dataset	Editor	LG	CI	CII	SA	RS	Avg.	LG	CI	CII	SA	RS	Avg.
		GPT-J						Llama-2					
RANDOM	ROME	0.58	0.52	0.24	**1.0**	0.44	0.56	0.57	0.41	0.29	**1.0**	0.52	0.56
	MEMIT	0.60	0.47	0.25	0.84	**0.48**	0.53	**0.67**	0.37	0.33	0.89	**0.67**	0.59
	PMET	**0.70**	0.46	0.26	0.88	0.34	0.53	0.62	0.47	0.18	**1.0**	0.49	0.55
	SWEA⊕OS	0.62	**0.54**	**0.63**	**1.0**	0.41	**0.64**	0.60	**0.49**	**0.37**	**1.0**	0.55	**0.60**
POPULAR	ROME	0.30	0.53	0.28	0.86	0.30	0.45	0.28	0.39	0.15	0.71	0.32	0.37
	MEMIT	0.30	0.44	0.19	**1.0**	**0.33**	0.45	0.28	0.45	0.09	**0.96**	**0.56**	**0.47**
	PMET	**0.37**	0.51	0.17	0.94	0.29	0.46	**0.30**	0.47	0.13	0.83	0.31	0.41
	SWEA⊕OS	0.32	**0.56**	**0.53**	**1.0**	0.29	**0.54**	**0.30**	**0.49**	**0.16**	0.81	0.37	0.43

Table 3: Accuracy of RIPPLEEDITS on GPT-J and Llama-2.

LLMs	Dataset	Editor	LG	CI	CII	SA	RS	Avg.
GPT-J	RANDOM	SWEA⊕OS	0.62	0.54	0.63	1.0	0.41	0.64
		w/o suppressing	$0.60_{\downarrow 0.02}$	$0.47_{\downarrow 0.07}$	$0.25_{\downarrow 0.38}$	$0.84_{\downarrow 0.16}$	$0.48_{\uparrow 0.07}$	$0.53_{\downarrow 0.11}$
	POPULAR	SWEA⊕OS	0.60	0.53	0.23	1.0	0.38	0.54
		w/o suppressing	$0.32_{\downarrow 0.28}$	0.53	$0.19_{\downarrow 0.04}$	$0.86_{\downarrow 0.14}$	$0.28_{\downarrow 0.10}$	$0.44_{\downarrow 0.1}$
Llama-2	RANDOM	SWEA⊕OS	0.60	0.49	0.37	1.0	0.55	0.60
		w/o suppressing	$0.59_{\downarrow 0.01}$	0.49	$0.30_{\downarrow 0.07}$	1.0	$0.54_{\downarrow 0.01}$	$0.58_{\downarrow 0.02}$
	POPULAR	SWEA⊕OS	0.30	0.49	0.16	0.81	0.37	0.43
		w/o suppressing	$0.29_{\downarrow 0.01}$	0.49	0.16	0.81	0.37	$0.42_{\downarrow 0.01}$

Table 4: Accuracy of RIPPLEEDITS on GPT-J and Llama-2 in ablation study.

nally generating 2188 and 2186 editing instances for GPT-J and Llama-2, respectively. Since none of these editing instances contain data for testing Preservation, our results do not include the Preservation metric.

Results The accuracy of editing GPT-J and Llama-2 on the RANDOM and POPULAR subdatasets of RIPPLEEDITS is shown in Table 3. The results of GPT-J indicate that SWEA⊕OS performs better than the baselines on CI, CII, and SA, suggesting that SWEA⊕OS's ability to reason about edited knowledge surpasses existing baselines. For Llama-2, except for the LG and RS on the RANDOM dataset and the SA and RS on the POPULAR dataset where SWEA⊕OS lags behind existing baselines, the results on other metrics are better than that of current baselines. From Table 3, it can be easily seen that SWEA⊕OS performs poorly on the RS, which tests the ability of the editing method to retain non-edited knowledge about the subject. A possible reason for this situation is that SWEA⊕OS introduced unintended knowledge during the optimization of the editing objectives. The metrics LG, CI and CII test the ability of the model editing method in 2-hop reasoning, and experimental results indicate that SWEAOS exhibits the best reasoning capability.

5.4 Ablation Study

To verify that the suppressing step in the OS fusion method is effective to the expression of new knowledge, we remove the suppressing step and test the results on the RIPPLEEDITS benchmark. As shown in Table 4, after removing the suppression step (i.e., w/o suppressing), the overwhelming majority of performance of SWEA⊕OS in editing GPT-J and Llama-2 has declined, which indicates that our suppression step effectively alleviated the effect brought by KEDs of subject word embeddings. When editing GPT-J on the RANDOM and POPULAR datasets, the absence of suppression steps led to an average reduction of 0.14 in all metrics. Moreover, in the ablation experiment, the performance drop of GPT-J is more significant than that of Llama-2, which may be due to the stronger robustness brought by more parameters of Llama-2. Overall, the introduction of suppression steps in SWEA⊕OS effectively facilitated the expression of new knowledge in LLMs.

6 Conclusion

We propose SWEA⊕OS method for more effective and efficient knowledge editing. SWEA⊕OS consists of Subject Word Embedding Fusion (SWEA) framework and the optimizing then suppressing (OS) fusion method. The SWEA framework uses token-level matching to identify the edited subject and adds the editing embeddings obtained from the OS fusion method to the subject embedding, ultimately altering the specific attributes of the subject to achieve knowledge editing. The OS fusion method employs an optimizing then suppressing strategy to effectively express new knowledge in editing embeddings. SWEA⊕OS achieve overall state-of-the-art (SOTA) results on the COUNTERFACT and zsRE datasets, and it also shows SOTA performance in terms of reasoning ability on a more complex model editing benchmark RIPPLEEDITS. Moreover, SWEA⊕OS also provide a new insight to efficiently update knowledge of LLMs.

Ethical Statement

The purpose of this work is to provide a more efficient and effective approach for knowledge editing. While SWEA can correct incorrect or outdated knowledge in LLMs cooperating with different fusion methods, it is important to recognize that SWEA is also susceptible to misuse, leading to the corruption of correct and already aligned knowledge in LLMs. Given that LLMs can inherently produce hallucinations, we would remind readers not to overly trust LLMs.

Acknowledgments

This work was partly supported by the Hunan Provincial Natural Science Foundation Projects (No. 2022JJ30668 and No. 2022JJ30046), and also partly supported by the National Key R&D Program of China (No. 2024YFB4506200).

References

Balogh, V.; Berend, G.; Diochnos, D. I.; and Turán, G. 2020. Understanding the Semantic Content of Sparse Word Embeddings Using a Commonsense Knowledge Base. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 7399–7406.

Cohen, R.; Biran, E.; Yoran, O.; Globerson, A.; and Geva, M. 2023. Evaluating the ripple effects of knowledge editing in language models. *arXiv preprint arXiv:2307.12976*.

Dai, D.; Dong, L.; Hao, Y.; Sui, Z.; Chang, B.; and Wei, F. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

De Cao, N.; Aziz, W.; and Titov, I. 2021. Editing Factual Knowledge in Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 6491–6506. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Deng, J.; Wei, Z.; Pang, L.; Ding, H.; Shen, H.; and Cheng, X. 2024. UnKE: Unstructured Knowledge Editing in Large Language Models. *arXiv preprint arXiv:2405.15349*.

Dong, Q.; Dai, D.; Song, Y.; Xu, J.; Sui, Z.; and Li, L. 2022. Calibrating factual knowledge in pretrained language models. *arXiv preprint arXiv:2210.03329*.

Geva, M.; Schuster, R.; Berant, J.; and Levy, O. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 5484–5495. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Gionis, A.; Indyk, P.; Motwani, R.; et al. 1999. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, 518–529.

Gu, J.-C.; Xu, H.-X.; Ma, J.-Y.; Lu, P.; Ling, Z.-H.; Chang, K.-W.; and Peng, N. 2024. Model Editing Can Hurt General Abilities of Large Language Models. *arXiv preprint arXiv:2401.04700*.

Hartvigsen, T.; Sankaranarayanan, S.; Palangi, H.; Kim, Y.; and Ghassemi, M. 2022. Aging with GRACE: Lifelong Model Editing with Discrete Key-Value Adaptors. *arXiv preprint arXiv:2211.11031*.

Huang, Z.; Shen, Y.; Zhang, X.; Zhou, J.; Rong, W.; and Xiong, Z. 2023. Transformer-Patcher: One Mistake worth One Neuron. *arXiv preprint arXiv:2301.09785*.

Li, C.; Gan, Z.; Yang, Z.; Yang, J.; Li, L.; Wang, L.; and Gao, J. 2023a. Multimodal foundation models: From specialists to general-purpose assistants. *arXiv preprint arXiv:2309.10020*, 1(2): 2.

Li, J.; Monroe, W.; and Jurafsky, D. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Li, S.; Deng, Y.; Cai, D.; Lu, H.; Chen, L.; and Lam, W. 2024a. Consecutive Model Editing with Batch alongside HooK Layers. *arXiv preprint arXiv:2403.05330*.

Li, X.; Li, S.; Song, S.; Liu, H.; Ji, B.; Wang, X.; Ma, J.; Yu, J.; Liu, X.; Wang, J.; and Zhang, W. 2024b. SWEA: Updating Factual Knowledge in Large Language Models via Subject Word Embedding Altering. arXiv:2401.17809.

Li, X.; Li, S.; Song, S.; Yang, J.; Ma, J.; and Yu, J. 2023b. PMET: Precise Model Editing in a Transformer. arXiv:2308.08742.

Li, Z.; Zhang, N.; Yao, Y.; Wang, M.; Chen, X.; and Chen, H. 2023c. Unveiling the pitfalls of knowledge editing for large language models. *arXiv preprint arXiv:2310.02129*.

Meng, K.; Bau, D.; Andonian, A.; and Belinkov, Y. 2022a. Locating and Editing Factual Associations in GPT. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 17359–17372. Curran Associates, Inc.

Meng, K.; Sharma, A. S.; Andonian, A.; Belinkov, Y.; and Bau, D. 2022b. Mass-Editing Memory in a Transformer.

Mitchell, E.; Lin, C.; Bosselut, A.; Finn, C.; and Manning, C. D. 2022. Fast Model Editing at Scale. In *International Conference on Learning Representations*.

Park, S.; Bak, J.; and Oh, A. 2017. Rotated Word Vector Representations and their Interpretability. In Palmer, M.; Hwa, R.; and Riedel, S., eds., *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 401–411. Copenhagen, Denmark: Association for Computational Linguistics.

Şenel, L. K.; Utlu, I.; Yücesoy, V.; Koc, A.; and Cukur, T. 2018. Semantic structure and interpretability of word embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10): 1769–1779.

Simhi, A.; and Markovitch, S. 2023. Interpreting Embedding Spaces by Conceptualization. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Tan, C.; Zhang, G.; and Fu, J. 2023. Massive Editing for Large Language Models via Meta Learning. *arXiv preprint arXiv:2311.04661*.

Tan, C.; Zhang, G.; and Fu, J. 2024. Massive Editing for Large Language Models via Meta Learning. In *The Twelfth International Conference on Learning Representations*.

Tian, B.; Cheng, S.; Liang, X.; Zhang, N.; Hu, Y.; Xue, K.; Gou, Y.; Chen, X.; and Chen, H. 2024. InstructEdit:

Instruction-based Knowledge Editing for Large Language Models. *arXiv preprint arXiv:2402.16123*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wang, B.; and Komatsuzaki, A. 2021. GPT-J-6B: A 6 billion parameter autoregressive language model.

Wang, P.; Li, Z.; Zhang, N.; Xu, Z.; Yao, Y.; Jiang, Y.; Xie, P.; Huang, F.; and Chen, H. 2024. WISE: Rethinking the Knowledge Memory for Lifelong Model Editing of Large Language Models. *arXiv preprint arXiv:2405.14768*.

Wang, S.; Zhu, Y.; Liu, H.; Zheng, Z.; Chen, C.; and Li, J. 2023. Knowledge Editing for Large Language Models: A Survey. arXiv:2310.16218.

Yao, Y.; Wang, P.; Tian, B.; Cheng, S.; Li, Z.; Deng, S.; Chen, H.; and Zhang, N. 2023. Editing Large Language Models: Problems, Methods, and Opportunities. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 10222–10240. Singapore: Association for Computational Linguistics.

Zhang, B.; Zheng, W.; Zhou, J.; and Lu, J. 2023. Bort: Towards Explainable Neural Networks with Bounded Orthogonal Constraint. In *The Eleventh International Conference on Learning Representations*.

Zhang, N.; Yao, Y.; Tian, B.; Wang, P.; Deng, S.; Wang, M.; Xi, Z.; Mao, S.; Zhang, J.; Ni, Y.; Cheng, S.; Xu, Z.; Xu, X.; Gu, J.-C.; Jiang, Y.; Xie, P.; Huang, F.; Liang, L.; Zhang, Z.; Zhu, X.; Zhou, J.; and Chen, H. 2024. A Comprehensive Study of Knowledge Editing for Large Language Models. arXiv:2401.01286.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; Du, Y.; Yang, C.; Chen, Y.; Chen, Z.; Jiang, J.; Ren, R.; Li, Y.; Tang, X.; Liu, Z.; Liu, P.; Nie, J.-Y.; and Wen, J.-R. 2023. A Survey of Large Language Models.

Zhu, C.; Rawat, A. S.; Zaheer, M.; Bhojanapalli, S.; Li, D.; Yu, F.; and Kumar, S. 2020. Modifying Memories in Transformer Models.