SSS: Editing Factual Knowledge in Language Models towards Semantic Sparse Space

Huazheng Wang*, **Haifeng Sun***, **Jingyu Wang**†, **Qi Qi**, **Zixuan Xia**, **Menghao Zhang**, **Jianxin Liao**
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
{wanghz,hfsun,wangjingyu,qiqi8266,zjjs2019xzx,zhangmenghao}@bupt.edu.cn
jxlbupt@gmail.com

Abstract

Language Models (LMs) acquire factual knowledge during pre-training and store it in the parameters, which can be valuable for downstream tasks. As world evolves, some facts may be incorrectly induced or become obsolete over time. Various model editing methods have been proposed to modify specific examples in LMs. However, existing training-based methods still suffer from sub-optimal locality, where irrelevant neighborhood examples can be adversely influenced. Model's gradients are still struggling to identify the appropriate direction when updating the parameters. To address this issue, we find that directing the hidden state of the edit example towards spaces where semantics are sparse tends to help preserve the semantics of irrelevant neighborhood examples. Based on this hypothesis, we propose a novel metric, named SSS, to evaluate the degree of sparsity around a sentence embedding in the semantic space without any human or machine annotation. Subsequently, we incorporate SSS into the original loss function of the existing training-based methods to enhance locality. Experiments conducted on two datasets across various models demonstrate that SSS is effective in improving both locality and reasoning capability. Code will be available at: https://github.com/MaybeLizzy/EditSSS.

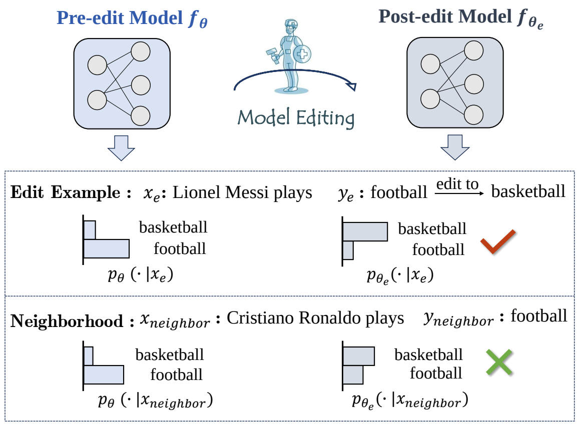

Figure 1: Existing methods can negatively impact irrelevant neighborhood examples.

1 Introduction

Language Models (Touvron et al., 2023; OpenAI, 2023) (LMs) are surprisingly good at recalling factual knowledge presented in the pre-training corpus, which enables promising results in various downstream tasks. However, as the world's state evolves, LMs may become incorrect or outdated over time. Developing reliable and computationally efficient methods to edit the model knowledge without the need for expensive re-training becomes non-trivial. In response to this issue, the concept of model editing has been proposed (Sinitsin et al., 2020; Cao et al., 2021), which is to intervene a pre-trained model's behavior on a specific edit example without damaging its performance on other irrelevant examples.

Numerous works on model editing for language models have emerged. One branch of research involves additional training, such as training a hypernetwork to predict weight updates (Cao et al., 2021; Mitchell et al., 2022a) or introducing additional trainable parameters (Dong et al., 2022; Hartvigsen et al., 2022; Huang et al., 2023). Another line of research attributes knowledge to specific neurons or modules within the network (Dai et al., 2022) and updates the model parameters associated with the edit example directly (Meng et al., 2022, 2023; Li et al., 2023a), without any additional training. With the development of in-context learning, some methods edit the model by either prompting it with the edit example (Zheng et al., 2023) or retrieving edit demonstrations from an explicit memory (Mitchell et al., 2022b; Madaan et al., 2022; Zhong et al., 2023).

To evaluate the post-edit model performance,

*Equal Contribution.
†Corresponding Author.

Findings of the Association for Computational Linguistics ACL 2024, pages 5559–5570
August 11-16, 2024

previous works formulate the criteria that a successful model editor should not only adjust the model behavior for in-scope examples that are closely associated with the edit example, but also maintain locality (or specificity) for out-of-scope examples. It means that irrelevant neighborhood examples should be left unaltered. Recently, there emerges diverse studies evaluating existing editing methods across various dimensions, such as learning new entities (Onoe et al., 2023), reasoning implications (Cohen et al., 2023) and editing commonsense mistakes (Gupta et al., 2023).

Though existing methods have demonstrated considerable effectiveness in model editing tasks, they still suffer from inferior locality, where irrelevant neighborhood examples can be adversely influenced (Yao et al., 2023) (as illustrated in Fig. 1). Focusing on the training-based methods, we speculate the reason is that existing methods primarily adopt two objectives when updating the model parameters, apart from their method-specific objective. One objective is to align the model predictions with the edit label, and the other is to constrain the Kullback-Leibler (KL) divergences between the pre- and post-edit models on irrelevant neighborhood examples. Due to the limited training set, examples not included within the dataset can still be potentially affected. Consequently, relying solely on the aforementioned objective is inadequate. Models' gradient are still struggling to identify the appropriate gradient direction when updating the parameters (Li et al., 2023b).

To address this issue, we take a further step and find that mapping the embedding of the edited example into a sparse space helps preserve the semantics of other irrelevant neighborhood examples. Based on this hypothesis, we propose a novel metric, named SSS, to evaluate the degree of sparsity in the semantic space around a sentence embedding for a specific model. We then incorporate SSS into the original loss function of existing training-based methods, guiding the embedding of the edited example towards a **S**emantic **S**parse **S**pace to enhance locality. This method is computationally efficient and entirely unsupervised, requiring no human or machine annotation.

To assess the effectiveness of SSS, we select three representative training-based methods, including FT-L (Yao et al., 2023), MEND (Mitchell et al., 2022a) and SERAC (Mitchell et al., 2022b). We test these methods on two popular model editing datasets, ZsRE (Levy et al., 2017) and COUNTERFACT (Meng et al., 2022) across various language models with multiple scales (1.5B$\sim$7B). Experimental results highlight the effectiveness of SSS in enhancing both locality and reasoning capability.

Our contributions are summarized as follows:

- We introduce a novel evaluation metric, SSS, to measure the semantic sparsity of a sentence embedding for a specific language model without any human or machine annotation.
- We incorporate SSS into existing training-based model editing methods to enhance the locality of irrelevant neighborhood examples.
- Experiments conducted on two datasets across various language models demonstrate the validity and scalability of SSS.

2 Related Work

Training-based Editing Method A branch of research requires training when editing models. The simplest method is directly fine-tuning the target model with specific edit examples. In addition, Cao et al. (2021) propose Knowledge Editor (KE) approach, which trains a hyper-network to predict the weight update for each edit example. To overcome the limitation of KE that falls short in editing language models with larger scale, Mitchell et al. (2022a) introduce Model Editor Networks with Gradient Decomposition (MEND), which learns to transform the gradient obtained by standard fine-tuning using a low-rank decomposition of gradients. Furthermore, Mitchell et al. (2022b) present SERAC, which stores edit examples in explicit memory. It utilizes a scope classifier to determine if an input lies within the scope of any cache items and trains a memory-based retrieval-augmented counterfactual model to edit facts that fall within the scope of stored edit examples. Other works introduce additional trainable parameters while keeping the original model parameters static (Dong et al., 2022; Hartvigsen et al., 2022). For instance, Huang et al. (2023) propose Transformer-Patcher, a sequential model editing method that simply incorporates and trains a few neurons in the last feed-forward Network layer.

Training-free Editing Method Another line of research attributes knowledge to specific neurons or modules within the network. Notably, Dai et al. (2022) view feed-forward network modules in Transformer as key-value memories and propose the Knowledge Neuron (KN) method to edit specific factual knowledge by identifying the knowledge neurons positively correlated to the knowledge expression. Similarly, Meng et al. (2022) modify feed-forward weights to update specific factual associations using Rank-One Model Editing (ROME). To extend model editing to multiple cases simultaneously (Li et al., 2023a), Meng et al. (2023) build upon the ROME framework and propose MEMIT. With the development of in-context learning, certain methods edit the model by either prompting it with the edit example (Zheng et al., 2023), or retrieving demonstrations from the edit memory (Madaan et al., 2022; Zhong et al., 2023).

3 Task Definition

The objective of model editing is to adjust an initial target model's behavior on a specific edit example, without impacting the model's performance on other irrelevant examples. More specifically, the target model f_θ is represented by a function $f: \mathbb{X} \mapsto \mathbb{Y}$, with θ denoting the model parameter. Given an edit example (x_e, y_e) with the edit label $y_e \neq f_\theta(x_e)$, the post-edit model f_{θ_e} is required to generate the expected output $f_{\theta_e}(x_e)$ such that $f_{\theta_e}(x_e) = y_e$.

4 Preliminary

In this section, we introduce the objectives of the existing training-based model editing methods. An ideal post-edit model f_{θ_e} should satisfy three properties: reliability, generality and locality (Yao et al., 2023).

Reliability A model editor is reliable if f_{θ_e} predicts the edit label y_e for the edit input x_e:

$$\mathbb{E}_{x'_e, y'_e \sim \{(x_e, y_e)\}} \mathbb{1}\{\text{argmax}_y f_{\theta_e}(y|x'_e) = y'_e\}. \quad (1)$$

Generality As the model editing process can impact a wide range of examples that are closely associated with the edit example, known as the editing scope, a successful edit should also adjust the model behavior for in-scope examples $I(x_e, y_e)$, such as examples with similar expressions:

$$\mathbb{E}_{x'_e, y'_e \sim I(x_e, y_e)} \mathbb{1}\{\text{argmax}_y f_{\theta_e}(y|x'_e) = y'_e\}. \quad (2)$$

Locality A good edit is supposed to edit the knowledge without influencing other irrelevant out-of-scope examples $O(x_e, y_e)$. It always refers to locality (or specificity):

$$\mathbb{E}_{x'_e, y'_e \sim O(x_e, y_e)} \mathbb{1}\{\text{argmax}_y f_{\theta_e}(y|x'_e) = f_\theta(y|x'_e)\}. \quad (3)$$

To achieve reliability and generality, existing works (Mitchell et al., 2022b; Huang et al., 2023) use the loss function L_e for the edit example and its corresponding in-scope examples:

$$L_e(\theta_e) = -log\, p_{\theta_e}(y_e|x_e). \quad (4)$$

To achieve locality, most works (Cao et al., 2021; Mitchell et al., 2022a) choose to constrain updates in terms of the Kullback-Leibler (KL) divergences between the pre- and post- edit model conditioned on the locality examples x_{loc}:

$$L_{loc}(\theta, \theta_e) = \text{KL}(p_\theta(\cdot|x_{loc}) \| p_{\theta_e}(\cdot|x_{loc})). \quad (5)$$

The total training loss L_{ori} is computed with the sum of L_e and L_{loc} using a hyper-parameter c_e:

$$L_{ori} = c_e \cdot L_e(\theta_e) + L_{loc}(\theta, \theta_e). \quad (6)$$

Apart from the aforementioned losses, different methods may employ their own method-specific objectives to update the model parameters. Despite these variations, in this paper, we consistently denote the original training loss of previous methods as L_{ori}.

5 Method

Existing methods are still struggling to maintain high locality (Yao et al., 2023). We assume the limitation is that they solely considering the KL divergences between the pre- and post- edit model on locality examples from the training set. This proves inadequate since irrelevant examples not collected in the training set can still be potentially influenced. To address this issue, we are committed to exploring strategies that guide model parameter updates in a direction that protects the semantic space of irrelevant examples from being disturbed.

It has been demonstrated that language models are pre-trained to implicitly learn sentence representations (Li et al., 2020). Each sentence can be encoded as a high-dimensional vector, representing a point in the embedding space. The knowledge acquired by the model is mapped into this space, characterized by an uneven distribution where certain regions may be denser than others (Aharoni and Goldberg, 2020). Since the knowledge acquired by LLMs is encoded within a network of interconnected neurons, editing one piece of knowledge is likely to impact others (Dai et al., 2022). When editing knowledge, models often face challenges in determining the appropriate gradient direction for

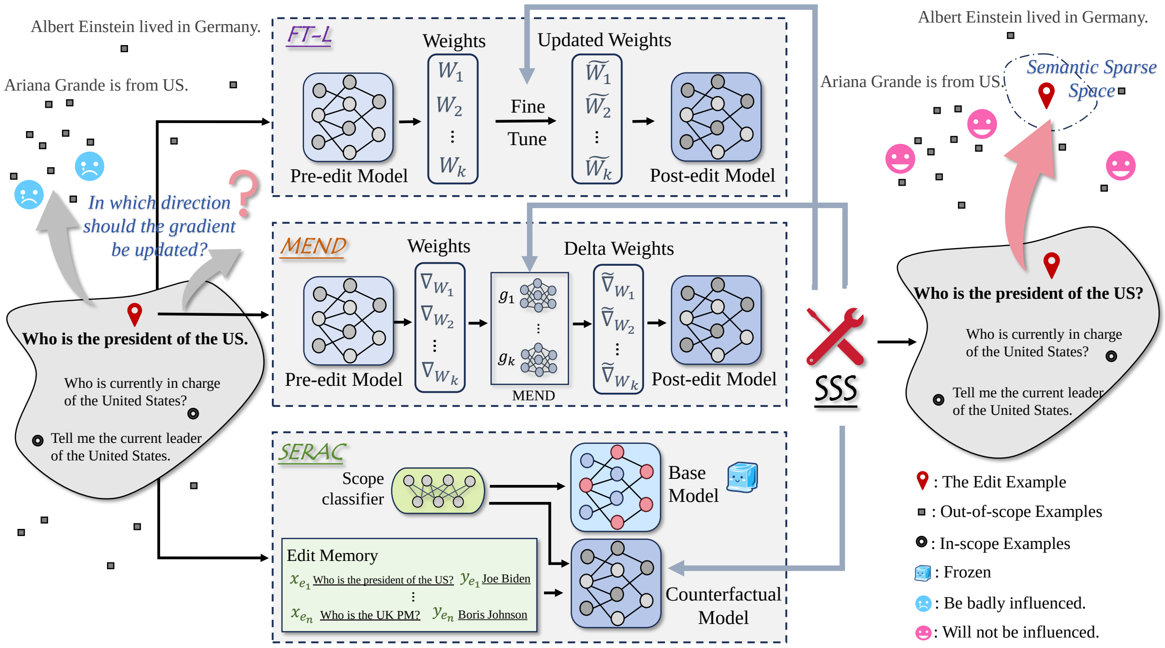

Figure 2: The illustration of SSS. Existing training-based model editing methods still lack explicit constraints on updating model parameters related to other irrelevant examples. SSS provides an explicit direction for updating the sentence embedding towards semantically sparse space, which is effective in protecting the irrelevant neighborhood examples from being badly influenced.

updating the parameters (Li et al., 2023b). Building upon these findings, we propose mapping the new knowledge into a sparse space to provide clear guidance for gradient updates while ensuring that the distribution of other knowledge in the embedding space remains unaffected.

To accomplish this, we utilize the vulnerability of sentence embeddings to quantify the sparsity of the embedding space. A robust sentence embedding, resilient to large perturbations, suggests a relatively sparse semantic space around it. Conversely, even a minor disturbance in the sentence can change its semantics, indicating a semantically dense space. Under this hypothesis, we introduce an evaluation metric to quantify the vulnerability (or sparsity) of sentence embeddings, named SSS. Subsequently, we introduce a novel training objective for updating model gradients to enhance locality. The detailed explanation is outlined as follows.

We are inspired by Zhao et al. (2019), who adopt the Fisher Information Matrix (FIM) of the input sample as a metric tensor to measure the robustness of deep learning models in adversarial attack task. Borrowing from this idea, we define a FIM-based matrix $\mathbf{H}$ to characterize the vulnerability of a sentence embedding to the perturbation in its feature space for a specific model LM. Specifically, given an edit example (x_e, y_e), the matrix $\mathbf{H} \in \mathbb{R}^{n\times n}$ is defined as follows, where n stands for the hidden dimension of LM and $\mathcal{J}(x_e, y_e)$ is the loss function:

$$\mathbf{H} = \nabla_{x_e}\mathcal{J}(x_e, y_e)^\top \nabla_{x_e}\mathcal{J}(x_e, y_e). \qquad (7)$$

$\nabla_{x_e}\mathcal{J}(x_e, y_e)$ is the partial differential of $\mathcal{J}(x_e, y_e)$ respecting to x_e. We adopt the per-token negative log-likelihood loss:

$$\mathcal{J}(x_e, y_e) = -\mathbb{E}_{y_e|x_e}[log\ P_{\mathrm{LM}}(y_e|x_e)]. \qquad (8)$$

Similar to the conclusion of Zhao et al. (2019), which quantifies the vulnerability of deep learning models, we deduce that the maximum eigenvalue of $\mathbf{H}$, denoted as λ_{max}, reflects the vulnerability of sentence embeddings to the perturbation. The proof process is detailed in A.1. The expression for λ_{max} can be written as:

$$\lambda_{max} = \frac{1}{m}\sum_{i=1}^{m}(\nabla_{x_i}\mathcal{J}(x_e, y_e))^2, \qquad (9)$$

where m is the sentence length and x_i is the ith token of x_e. A smaller λ_{max} indicates a more robust sentence embedding with higher resilience to the perturbation, indicating a relatively sparse semantic space surrounding the sentence embedding. As illustrated in Fig. 2, we name λ_{max} as

SSS and incorporate it into the original training loss of previous methods using a tuning coefficient C to enhance overall locality, denoted as L_{all}:

$$L_{all} = (1 - C) \cdot L_{ori} + C \cdot \text{SSS}. \quad (10)$$

In general, SSS can be easily implemented by three steps. Firstly, input x_e to the model and calculate the loss $J(x_e, y_e)$ using the model prediction and the edit label. Secondly, compute the maximum eigenvalue λ_{max} of x_e's updated embedding using Eq 9. Finally, the total training loss is derived using $J(x_e, y_e)$ and SSS according to Eq 10.

The advantage of the introduced SSS lies in its computational efficiency, as it solely relies on $\mathcal{J}(x_e, y_e)$ and the last hidden states of the edit example x_e. Additionally, integrating SSS into the original loss function for various training-based editing methods is straightforward and convenient. More importantly, the entire process is unsupervised, requiring no human or machine annotation.

6 Preliminary Experiments

In this section, we conduct preliminary experiments to assess the effectiveness of SSS across three training-based methods. The experimental settings and results are outlined as follows. All results are averaged over three runs.

6.1 Experiment Settings

We initially employ two popular model editing datasets, ZsRE (Levy et al., 2017) and COUNTERFACT (Meng et al., 2022). Following the data split from Yao et al. (2023), we use the training set for model training and evaluate the performance on the test set. For test efficiency, the test set is limited to 10,000 examples on ZsRE. We evaluate the results of fine-tuning, a fundamental approach for model editing. Following (Yao et al., 2023), we fine-tune layers identified by ROME (Meng et al., 2022) to avoid the computational cost of retraining all layers, which is denoted as FT-L. Furthermore, we choose two influential training-based methods for evaluation, namely MEND (Mitchell et al., 2022a) and SERAC (Mitchell et al., 2022b). Instead of using smaller language models for knowledge editing, such as BERT (Devlin et al., 2019), we follow Yao et al. (2023) and focus on generation-based models. Specifically, we choose `GPT2-XL` (1.5B) as the target model. In these experiments, we set the hyper-parameter C to 0.9.

6.2 Experiment Results

FT-L For basic FT-L, we follow Yao et al. (2023) and utilize Eq. 4 to train the model. For FT-L w/ SSS, the training objective is computed by incorporating SSS with Eq. 4. To ensure fairness, all hyper-parameters follow default settings in Yao et al. (2023), utilizing Adam with early stopping, and only modifying the weights of mlp_{proj} at the selected layer.

As shown in Table 1, when using FT-L, the accuracy of generality is significantly lower than that of reliability, exhibiting an absolute difference of 83.49%. This suggests that fine-tuning can only modify the model's behavior on the specific edit example, failing to generalize to other rephrased sentences. Despite of this, FT-L w/ SSS outperforms FT-L by up to 1.19% on reliability, 1.52% on generality and 1.15% on locality, illustrating the effectiveness of SSS.

MEND Following (Mitchell et al., 2022a), the training objective of MEND is consistent with Eq. 6, where c_e is set to be 0.1. For MEND w/ SSS, we integrate SSS with Eq. 6. Both MEND and MEND w/ SSS are trained on ZsRE training set using the Adam optimizer.

As shown in Table 1, when tested on ZsRE, the performance of MEND w/ SSS surpasses that of MEND, exhibiting an absolute improvement of 2.73% on generality. When tested on COUNTERFACT in an out-of-distribution (OOD) scenario, MEND demonstrates remarkable robustness, with reliability outperforming ZsRE by 32.74%, though at a cost of relatively lower locality and generality. We speculate that the reason lies in the nature of the model, which may excel in domain adaptation for editing data formulated as language modeling. In this scenario, MEND w/ SSS demonstrates a more robust capability than MEND, with an absolute improvement of 4.99% on reliability, 1.85% on locality and 4.0% on generality. This verifies the effectiveness of SSS.

SERAC SERAC consists of an edit memory, classifier, and counterfactual model. The scope classifier and counterfactual model are trained independently. We solely concentrate on the training objective of the counterfactual model, which aligns with MEND (w/ SSS). The training parameter settings of SERAC (w/ SSS) remain consistent with Yao et al. (2023).

As shown in Table 1, when tested on ZsRE,

Dataset	Metric	FT-L	w/ SSS	MEND	w/ SSS	SERAC	w/ SSS
ZsRE	**Reliability**	57.25	**58.44**	40.36	**42.74**	56.35	**56.91**
	Locality	86.98	**87.32**	90.20	**90.56**	93.11	**93.26**
	Generality	28.05	**29.57**	38.86	**41.59**	46.24	**46.66**
COUNTERFACT	**Reliability**	96.85	**96.96**	70.49	**75.48**	12.61	**14.99**
	Locality	81.50	**82.65**	64.50	**66.35**	18.78	**22.30**
	Generality	13.36	**14.20**	12.37	**16.37**	10.18	**12.65**

Table 1: Overall performance comparison between the three selected training-based methods and w/ SSS. Note that the accuracy of MEND and SERAC on the COUNTERFACT dataset represents an out-of-domain scenario.

	Reliability				Locality				Generality			
n	**1**	**10**	**100**	**1000**	**1**	**10**	**100**	**1000**	**1**	**10**	**100**	**1000**
ROME	100	100	98.0	65.8	100	40.0	22.0	3.4	100	43.0	30.0	8.0
MEMIT	100	92.0	89.8	85.0	100	100	91.0	54.8	100	62.5	41.3	40.0
FT-L	100	99.0	97.4	92.5	100	60.0	13.0	3.0	66.7	27.0	20.0	7.5
w/ SSS	100	**100**	**98.1**	**97.0**	100	60.0	**19.0**	**3.9**	66.7	**33.3**	20.0	**8.7**
MEND	100	70.0	13.0	1.3	66.7	30.0	6.0	0.6	75.0	21.7	14.0	0.4
w/ SSS	100	**73.3**	**19.0**	**3.8**	**75.0**	**33.3**	**9.0**	**1.0**	75.0	21.0	**14.3**	**1.0**
SERAC	100	20.0	13.2	10.0	100.0	27.0	18.0	14.6	75.0	40.0	14.7	10.9
w/ SSS	100	20.0	**14.9**	**13.0**	100.0	**30.0**	**24.0**	**22.4**	**100**	**50.3**	**16.0**	**13.3**

Table 2: Sequential Editing performance on COUNTERFACT dataset.

SERAC w/ SSS outperforms SERAC across all three metrics. When tested on COUNTERFACT, SERAC demonstrates significantly inferior OOD robustness. The bottleneck may lie in the accuracy of the pre-trained classifier and the capabilities of the inference model. Despite of this, SERAC w/ SSS consistently outperforms SERAC by 2.38% on reliability, 3.52% on locality and 2.47% on generality. The significant improvement confirms the effectiveness of SSS on editing model facts and preserving neighborhood knowledge.

Sequential Editing The default evaluation procedure involves updating the knowledge of a single model, evaluating the updated model, and then rolling back the update before repeating the process for each test example. In practical scenarios, a model editor must continuously and promptly fix a series of mistakes. Therefore, we conduct experiments on sequential editing using the COUNTERFACT dataset, where models retain previous changes while making new edits. Following Yao et al. (2023), we choose the sequence number n from $[1, 10, 100, 1000]$. For comparative purposes, we also present the results of two training-free methods: ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023).

As shown in Table 2, FT-L maintains high Reliability even when $n = 1000$, but its Locality and Generality decrease significantly as n increases. MEMIT demonstrates stable performance in sequential editing, while MEND's performance deteriorates with increasing n. SSS consistently outperforms the baselines as n increases. Specifically, SERAC w/ SSS achieves up to a 7.8% improvement in Locality compared to SERAC when $n = 1000$. These results confirm that SSS is able to preserve the model's original irrelevant knowledge during sequential editing, validating the effectiveness of SSS.

7 Comprehensive Study

The impact of model editing on the language model is intricate, demanding a thorough and comprehensive evaluation to fully comprehend its effects. Consequently, in line with Yao et al. (2023), we conduct additional tests to assess SSS from both

Metric	ROME	MEMIT	FT-L	w/ SSS	MEND	w/ SSS	SERAC	w/ SSS
Reliability	98.73	83.71	97.26	**97.64**	68.71	**72.64**	10.95	**12.44**
Pre-Neigh	95.27	96.55	72.84	**73.35**	91.48	**92.00**	12.60	**13.59**
Post-Neigh	93.46	95.81	75.25	**75.72**	92.01	**92.94**	13.74	**14.53**
Distract-Neigh	58.40	63.53	57.64	**59.19**	**71.55**	70.73	**12.84**	12.61
Unrelated Attri	79.46	89.07	91.73	**92.04**	88.03	**93.44**	29.88	**36.28**

Table 3: Performance comparison between different methods and w/ SSS on locality.

Metric	Dataset	Size
Locality	counterfact	804
Subject Replace	zsre	293
Reversed Relation	zsre	385
One-Hop	counterfact	1031

Table 4: The data statistics of comprehensive study.

locality and portability perspectives. The statistics of the datasets are shown in Table 4. In this experiment, FT-L (w/ SSS) is directly fine-tuned on the specific dataset using GPT2-XL. MEND (w/ SSS) and SERAC (w/ SSS) utilize the pre-trained checkpoints from Section 6. We keep $C = 0.9$.

7.1 Evaluate Locality

Consider an edit example x_e with its original ground truth y_r and edit label y_e. To thoroughly examine the potential side effects of model editing on neighborhood examples, SSS is tested at four different levels.

Pre-Neighbor: We name a neighborhood example as Pre-Neighbor if its label is y_r.

Post-Neighbor: We name a neighborhood example as Post-Neighbor if its label is y_e.

Distract-Neighbor: Following Yao et al. (2023), we concatenate the edit example before Pre-Neighbor example to test whether the model prediction will be influenced by the edit example. The post-edit model is expected to maintain consistent behavior on Distract-Neighbor examples, predicting y_r instead of y_e.

Unrelated Attributes: The unrelated attributes of the subject in the edit example should remain unchanged after editing. For instance, if we edit the knowledge from "Alfred Kubel has citizenship from Germany" to "Alfred Kubel has citizenship from Finland", the answer of the Unrelated Attribute example "What is Alfred Kubel's sex or gender?" should remain the same.

We also present the results of ROME and MEMIT on locality. However, this locate-and-edit method heavily relies on the data format and is limited to modifying examples in triplets, where (e, r, o) represents a subject entity e, a relation r, and an object o. Therefore, we do not directly compare the accuracy with ROME and MEMIT.

Results As shown in Table 3, FT-L exhibits higher reliability compared to MEND and SERAC. It is reasonable since the latter two are tested in an OOD scenario where generalization may be somewhat reduced. In detail, MEND w/ SSS exhibits better reliability than MEND by 3.93%, highlighting the effectiveness of SSS in editing model knowledge. However, MEND w/ SSS and SERAC w/ SSS exhibit sub-optimal performance on Distract-Neighbor, showing reductions of 0.82% and 0.23%, respectively. We suspect this is attributed to the fact that training with SSS enhances the edit example's resilience to semantic perturbation, thereby making the model more responsive to the edit example when incorporated as a prompt concatenated in front of the test query. Therefore, the edit example may distract the model attention, resulting in a relatively lower performance on Distract-Neighbor. In addition, w/ SSS shows a relatively modest increase in Pre-Neighbor and Post-Neighbor, ranging from 0.47% to 0.99%. However, the improvements introduced by SSS on Unrelated Attributes are substantial, with an absolute enhancement of 6.4% compared to SERAC. It indicates that SSS is more effective in enhancing the robustness of the subject in edit example itself, protecting it from the semantic perturbation of other irrelevant examples. This validates the effectiveness of SSS.

Impact of C We apply parameter tuning method to explore the impact of C on model performance. Experiments are conducted on the locality dataset using FT-L w/ SSS with C ranging from 0.0 to 1.0. Other parameters remain unchanged. As shown in Fig 3, the best accuracy is achieved when $C = 0.9$, highlighting the importance of consid-

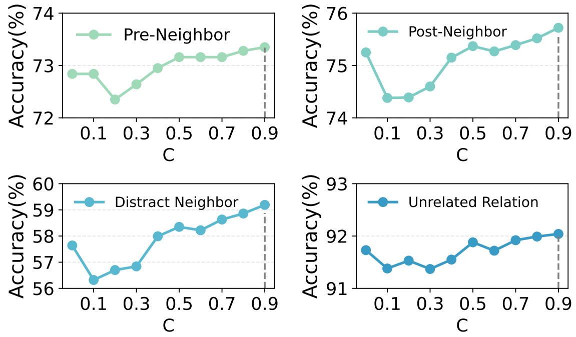

Figure 3: Ablation study on C. Experiments are conducted on FT-L w/ SSS using `GPT2-XL`.

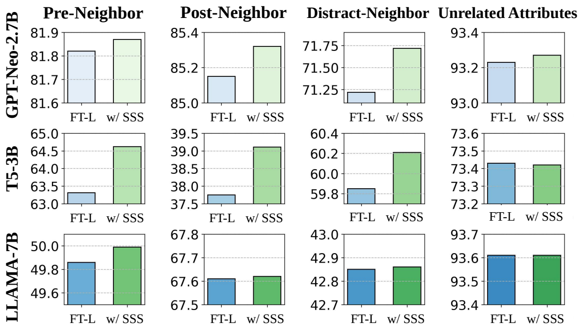

Figure 4: Experimental results across different models.

ering SSS when editing model knowledge. Additionally, the model performance can fluctuate when $C = 0.0 \sim 0.5$ where L_e takes a large proportion. However, for $C \geq 0.6$, the accuracy across all metrics consistently exhibits a stable improvement. This suggests that increasing the proportion of SSS can make the training process less dependent on L_e. A small L_e may imply that the post-edit model performs well on the edit example, but it does not necessarily ensure that the neighborhood examples are protected from disturbance. Therefore, increasing the proportion of SSS can more effectively enhance locality. Moreover, for $C = 1.0$, the reliability is 0.25% with locality reaching 100%. It demonstrates that solely relying on SSS is inadequate for editing the model knowledge.

Model Scaling To thoroughly evaluate the impact of model scaling on SSS, we conduct experiments with larger LMs on locality dataset, including `GPT-Neo-2.7B`, `T5-3B` and `LLaMA-7B`. As shown in Fig. 4, the accuracy does not exhibit a pattern with the scaling of LMs. We suspect that accuracy can be influenced by various aspects of models, including the pre-trained dataset, model architecture, and training methods. Larger models do not necessarily indicate better accuracy (Onoe et al., 2023; Hoelscher-Obermaier et al., 2023). Despite the variations, FT-L w/ SSS consistently outperforms FT-L, validating the effectiveness of SSS.

7.2 Evaluate Portability

To thoroughly evaluate the effectiveness of SSS, we test SSS on portability, assessing its ability to transfer the edited knowledge to related facts (Yao et al., 2023). There are three aspects: *Subject Replace*, *Reversed Relation* and *One-hop*. We provide detailed explanation as follows.

Subject Replace We substitute the subject in the edit example with an alias or synonym, evaluating the post-edit model's ability to generalize the edited attribute to other descriptions of the same subject.

Reversed Relation When the target of a subject and relation is edited, the attribute of the target entity also changes. We test the model's ability with the reverse question to check if the target entity is also updated. For instance, if we edit the answer of the edit example “Who is Nebaioth's father?" from “Babylon 5" to “Babur", then the post-edit model is expected to predict “Nebaioth" for the Reversed Relation example “Who is the son of Babur?".

One-Hop The post-edit model should employ the edited knowledge for downstream tasks. Therefore, we evaluate the post-edit model's ability to perform one-hop reasoning. For instance, if we edit the answer of the edit example “What company made Volvo B12M?" from “Volvo Buses" to “Volkswagen Group", then the post-edit model is expected to predict “Wolfsburg, Germany" instead of “Gothenburg, Sweden" for the One-Hop example “In which city is the headquarters of the company that made the Volvo B12M?".

Results As shown in Table 5, MEND w/ SSS presents a superior performance over MEND by 2.26% on Subject Replace. Though FT-L w/ SSS exhibits lower reliability and generality than FT-L, the accuracy difference is within 0.28%. Though SERAC w/ SSS exhibit a modest improvement of 0.85% on Reversed Relation, it demonstrates an improvement of 2.55% on One-Hop compared to SERAC. The overall performance in portability suggests that SSS can help the post-edit model to handle the implications of the edit examples and transfer the edited knowledge to associated facts for downstream applications, particularly in

Dataset	Metric	FT-L	w/ SSS	MEND	w/ SSS	SERAC	w/ SSS
Subject Replace	Reliability	54.88	**55.28**	42.27	**44.78**	47.39	47.39
	Generality	28.31	**28.80**	42.43	**44.77**	47.39	47.39
	Subject Replace	15.88	**16.27**	38.39	**40.65**	47.27	**47.45**
Reversed Relation	Reliability	**40.85**	40.72	60.94	**65.28**	71.56	**71.57**
	Generality	**34.19**	33.91	60.17	**64.24**	71.13	**71.24**
	Reversed Relation	35.57	**35.6**	33.38	**33.97**	67.16	**68.01**
One Hop	Reliability	97.19	**98.93**	60.33	**64.01**	11.35	**13.68**
	Generality	**5.04**	4.85	11.59	**16.20**	38.12	**44.28**
	One-Hop	41.62	**41.88**	37.69	**38.3**	41.75	**44.30**

Table 5: Performance comparison between the three selected methods and w/ SSS on portability.

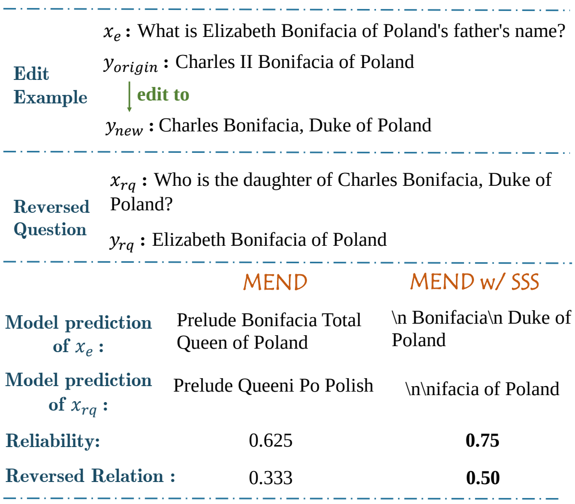

Figure 5: Case study on Reversed Relation dataset.

terms of one-hop capability. Nevertheless, the absolute portability accuracy remains relatively modest (less than 50%), calling for further exploration and innovation in future research.

Case Study We provide a case study on Reversed Relation dataset using MEND. As shown in Fig 5, the predictions made by MEND on the edit example and the reversed question significantly differ from the label, leading to lower accuracy. However, when using MEND w/ SSS, the predictions notably improve, resulting in higher accuracy, which validates the effectiveness of SSS.

8 Conclusion

In this paper, we introduce SSS, a novel evaluation metric to measure the semantic sparsity of a sentence for a specific language model without any human or machine annotation. Subsequently, we incorporate SSS into existing training-based model editing methods to enhance the locality of irrelevant neighborhood examples. Experiments conducted on two datasets across various models demonstrate the validity and scalability of SSS.

In summary, SSS presents a promising approach for mitigating the side effects of model editing, offering a notable perspective compared to existing training-based model editing methods.

Ethical Considerations

We believe that this study contributes intellectual value to the dependable application of model knowledge editing in the field of NLP, with potential broader implications for tasks in other areas. It is noteworthy that there are no direct societal consequences, and all experiments are conducted on open datasets.

Limitations

Given the constraints of computing power, incorporating language models with larger scales poses a challenge for us. While results using automatic metrics offer a fair assessment of task performance, we plan to conduct a human evaluation in the near future.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under, Grant 62201072, Grant U23B2001, Grant 62171057, Grant 62101064, Grant 62001054, and Grant 62071067; in part by the Ministry of Education and China Mobile Joint Fund under Grant MCM20200202 and Grant MCM20180101; in part by the BUPT-China Mobile Research Institute Joint Innovation Center.

References

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7747–7763. 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, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6491–6506. Association for Computational Linguistics.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2023. Evaluating the ripple effects of knowledge editing in language models. *CoRR*, abs/2307.12976.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8493–8502. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5937–5947. Association for Computational Linguistics.

Anshita Gupta, Debanjan Mondal, Akshay Krishna 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, EMNLP 2023, Singapore, December 6-10, 2023*, pages 8214–8232. Association for Computational Linguistics.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2022. Aging with GRACE: lifelong model editing with discrete key-value adaptors. *CoRR*, abs/2211.11031.

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, Toronto, Canada, July 9-14, 2023*, pages 11548–11559. 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. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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), Vancouver, Canada, August 3-4, 2017*, pages 333–342. Association for Computational Linguistics.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 9119–9130. Association for Computational Linguistics.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2023a. PMET: precise model editing in a transformer. *CoRR*, abs/2308.08742.

Zichao Li, Ines Arous, Siva Reddy, and Jackie Chi Kit Cheung. 2023b. Evaluating dependencies in fact editing for language models: Specificity and implication awareness. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 7623–7636. 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, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 2833–2861. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.

Yasumasa Onoe, Michael J. Q. Zhang, Shankar Padmanabhan, Greg Durrett, and Eunsol Choi. 2023. Can lms learn new entities from descriptions? challenges in propagating injected knowledge. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5469–5485. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry V. Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

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, EMNLP 2023, Singapore, December 6-10, 2023*, pages 10222–10240. Association for Computational Linguistics.

Chenxiao Zhao, P. Thomas Fletcher, Mixue Yu, Yaxin Peng, Guixu Zhang, and Chaomin Shen. 2019. The adversarial attack and detection under the fisher information metric. In *AAAI2019, IAAI 2019, EAAI 2019*, pages 5869–5876. AAAI Press.

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, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4862–4876. Association for Computational Linguistics.

Zexuan Zhong, Zhengxuan Wu, Christopher D. 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, EMNLP 2023, Singapore, December 6-10, 2023*, pages 15686–15702. Association for Computational Linguistics.

A Appendix

A.1 Proof

To evaluate the vulnerability of edit examples, we are inspired by Zhao et al. (Zhao et al., 2019), who adopt the Fisher Information Matrix (FIM) of the input sample as a metric tensor to measure the robustness of deep learning models in adversarial attack task. Given a labeled data example (x, y), the FIM matrix named $\mathbf{G_x}$ is defined by Eq 11:

$$\mathbf{G_x} = \sum_i p_i \left[\left(\nabla_x \mathcal{J}\left(y_i, x\right)\right) \left(\nabla_x \mathcal{J}\left(y_i, x\right)\right)^T \right], \tag{11}$$

where $\mathcal{J}\left(y_i, x\right) = -logp(y_i|x)$ is the loss function and p_i represents the probability of $p(y_i|x)$ when y takes the i-th class. $\nabla_x \mathcal{J}(y_i, x)$ is the partial differential of $\mathcal{J}(y_i, x)$ respecting to x.

Borrowing from this idea, we define a FIM-based matrix $\mathbf{H}$ to characterize the vulnerability of an edit example to the perturbation in its feature space for LM. The matrix $\mathbf{H} \in \mathbb{R}^{n \times n}$ of edit example $e = (x_e, y_e)$ is defined as follows, where n stands for the hidden dimension of LM and $\mathcal{J}(x_e, y_e)$ is the loss function:

$$\mathbf{H} = \nabla_{x_e} \mathcal{J}(x_e, y_e)^\top \nabla_{x_e} \mathcal{J}(x_e, y_e). \tag{12}$$

The matrix $\mathbf{H}$ proposed in Eq 12 can be considered as a special case of $\mathbf{G_x}$. In $\mathbf{G_x}$, each class's probability of $p(y|x)$ is weighted, and $\mathbf{G_x}$ calculates the expectation accordingly. In contrast, $\mathbf{H}$ sets the probability of $p(y|x)$ to 1 when y takes the correct class and 0 for other classes. Such difference between $\mathbf{H}$ and $\mathbf{G_x}$ originates from the different objectives. The aim of $\mathbf{G_x}$ is to find a subtle perturbation η that shifts the probability $p(y|x + \eta)$ from the correct class to an incorrect one. So each class is given a probability. However, our objective is to ensure the model consistently predicts the correct class. So other classes can be ignored with weight being 0.

Similar to the conclusion of Zhao et al. (Zhao et al., 2019), we deduce that the maximum eigenvalue of $\mathbf{H}$, denoted as λ_{max}, reflects the robustness of the edit example to LM. A smaller λ_{max} indicates a more robust edit example with higher resilience to the perturbation. We present the key derivation steps as follows.

We assume the existence of an ideally robust edit example, denoted as $\hat{x_e}$, that can yield the post-edit model prediction aligned with y_e while maintaining the distribution of other neighborhood knowledge in the embedding space unaffected. We evaluate the robustness of x_e by observing the distance between x_e and $\hat{x_e}$, denoted by $\|x_e - \hat{x_e}\|$. The embedding of x_e is denoted as $\mathbf{X} \in \mathbb{R}^{m \times n}$ where m is the length of x_e and n is the hidden dimension of LM. $\hat{\mathbf{X}}$ is the corresponding embedding of $\hat{x_e}$. Subsequently, we transform the task into observing $\|\mathbf{X} - \hat{\mathbf{X}}\|$. Given the parameters of LM expressed as $\mathbf{W} \in \mathbb{R}^{n \times n}$, the loss function can be transformed into:

$$\mathcal{J}(\mathbf{X}, \hat{\mathbf{X}}) = \left\| \mathbf{W}\mathbf{X}^\top - \mathbf{W}\hat{\mathbf{X}}^\top \right\|^2. \tag{13}$$

The partial differential of $\mathcal{J}(\mathbf{X}, \hat{\mathbf{X}})$ respecting to $\mathbf{X}$ is:

$$\nabla_{\mathbf{X}} \mathcal{J}(\mathbf{X}, \hat{\mathbf{X}}) \triangleq \left(\nabla_{X_1} \mathcal{J}(\mathbf{X}, \hat{\mathbf{X}}), \cdots, \nabla_{X_m} \mathcal{J}(\mathbf{X}, \hat{\mathbf{X}}) \right)^\top, \tag{14}$$

where X_i is the row vector standing for the ith token embedding, denoted by $X_t = (x_1, \cdots, x_n) \in \mathbb{R}^{1 \times n}$.

The maximum eigenvalue of $\mathbf{H}$, denoted as λ_{max}, can be calculated as the weighted sum of each token's maximum eigenvalue:

$$\lambda_{max} = \frac{1}{m} \sum_{i=1}^{m} a_i \lambda_i \tag{15}$$

where a_i is the weight coefficient of the ith token, and $\sum_{i=1}^{m} a_i = m$. We consider the simplest case where every token is equally important and let $a_1 = a_2 = \cdots = a_m = 1$. λ_t is the tth token's maximum eigenvalue, which can be calculated by:

$$\lambda_t = \sum_{i=1}^{n} (\nabla_{x_i} \mathcal{J}(X_t, \hat{X}_t))^2. \tag{16}$$

Substituting Eq 16 to Eq 15, we get the expression of λ_{max} as:

$$\lambda_{max} = \frac{1}{m} \sum_{z=1}^{m} \sum_{k=1}^{n} (\nabla_{x_k^z} \mathcal{J}(X_z, \hat{X}_z))^2, \tag{17}$$

where x_i^j and $\hat{x}_i^j$ represent the jth element of X_i and $\hat{X}_i$.

To calculate λ_{max}, we expand the square sum of Eq. 17 and then simplify it into a comprehensive expression. For simplicity, we introduce a notion $\Delta_i^j = x_i^j - \hat{x}_i^j$ to represent the distance between the ith dimension of the jth token embeddings. And then λ_{max} can be derived as:

$$\lambda_{max} = \frac{4}{m} \sum_{z=1}^{m} \sum_{k=1}^{n} \left(\sum_{i=1}^{n} w_{ik}^2 \cdot \Delta_i^z + \sum_{\substack{j \in \Phi \\ j \neq k}} \left(\sum_{i=1}^{n} w_{ik} w_{ij} \right) \Delta_j^z \right)^2 \tag{18}$$

where $\Phi = \{1, 2, \cdots, n\}$. And $w_{i,j}$ is the element in row i, column j of matrix $\mathbf{W}$.

We conclude from Eq 18 that λ_{max} is a function on Δ_i^j and the coefficients are only related to the model parameters. Hence, a small λ_{max} represents a relatively small Δ_i^j, leading to a small $\|\mathbf{X} - \hat{\mathbf{X}}\|$ and a more robust edit example.

A.2 Experiments

We conduct additional experiments to explore the impact of C on portability using `GPT2-XL`. Experiments are conducted on the Subject Replace dataset using FT-L w/ SSS with C ranging from 0.0 to 1.0. Other parameters remain unchanged. As illustrated in Fig. 6, the highest accuracy is observed at $C = 0.8 \sim 0.9$, which aligns with the observation in Section 7.1.

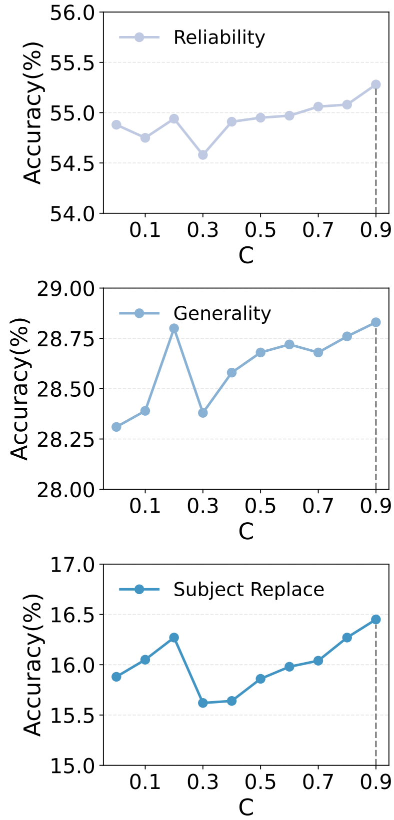

Figure 6: Ablation study of C on Subject Replace dataset.