Editing Memories Through Few Targeted Neurons

Wei Zhou[1, 2], **Wei Wei**[1, 2*], **Guibang Cao**[3], **Fei Wang**[4]

[1] Cognitive Computing and Intelligent Information Processing (CCIIP) Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, China
[2] Joint Laboratory of HUST and Pingan Property & Casualty Research (HPL), China
[3]Ping An Property & Casualty Insurance Company of China, Ltd
[4] Institute of Computing Technology, Chinese Academy of Sciences
zw02138@gmail.com, weiw@hust.edu.cn, sunfacy@163.com, wangfei@ict.ac.cn

Abstract

Model editing is a novel research topic in large language models (LLMs), aimed at efficiently handling various knowledge editing tasks. Since irrelevant knowledge is difficult to measure, existing editing methods often lack explicit ways to preserve it, especially for editing methods based on the fine-tuning paradigm. They generally control the locality performance of model editing by constraining the range of changes in model parameters. However, their performance improvements are not always ideal, and may even lead to a decrease in the editing reliability. In this paper, we try to explore effective editing locality control methods based on the relationship between the stored knowledge and the strongly associated model components. Based on the discovery of "knowledge neurons" and enough experimental results, we further explore the potential characteristics between knowledge and model components, confirm and point out: (1) only 1% neurons have significant contributions to specific knowledge storage, and (2) these targeted neurons often have a high overlap for knowledge with similar relational descriptions, which means that knowledge with similar relationships may be severely affected when these targeted neurons are modified. Based on these findings, we propose Targeted Neurons Fine-tuning with Data Augmentation (TNF-DA), which performs data augmentation based on the relational representation of edited knowledge to improve editing locality. By freezing most of the model parameters and only fine-tuning the highly contributing neurons corresponding to the edited knowledge, we obtain desirable results in terms of generalization and specificity compared with previous fine-tuning-based methods. Extensive experiments have demonstrated the superior editing performance achieved by our proposed method.

Code — https://github.com/lifeforzw/TNF-DA

Introduction

Pre-trained large language models (LLMs) (Radford et al. 2019; Mann et al. 2020; Vaswani et al. 2017; Zhao et al. 2023; Qiao et al. 2022) have already become commonly used tools in NLP tasks. Since their remarkable performance in understanding human-like text and the vast number of parameters they contain, LLMs are usually considered a knowledge base (Petroni et al. 2019; Roberts, Raffel, and Shazeer 2020; Jiang et al. 2020; Shin et al. 2020), with the capability to respond in natural language. However, as the world doesn't remain the same, some knowledge may change over time. It's obviously hard to strike a balance between significant computational costs and small-scale knowledge editing needs (Carlini et al. 2019). To address this challenge, **model editing** has been proposed (Zhu et al. 2020).

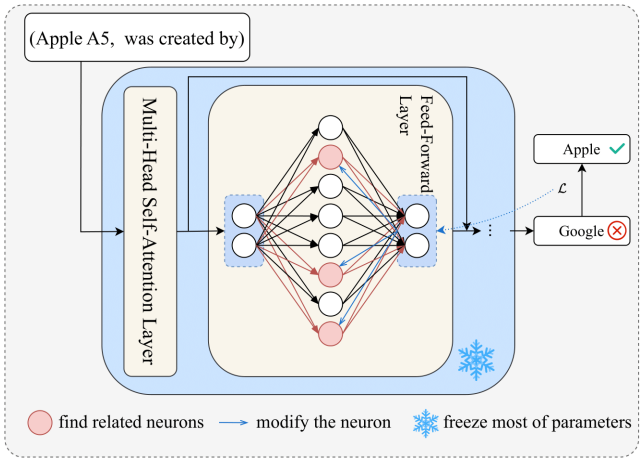

Figure 1: Finding the neurons highly contributed to the knowledge, we only modify these targeted neurons while freezing other parameters

Currently, numerous studies on model editing have been proposed. (Zheng et al. 2023; Meng et al. 2022a,b; Mitchell et al. 2021; Li et al. 2024; Zhu et al. 2020; Mitchell et al. 2022; Hartvigsen et al. 2024; Huang et al. 2023; Hernandez, Li, and Andreas 2023) These works can be categorized into two classes: adding extra information to a frozen model somewhere and directly modifying the model's parameters. The former focuses on determining when and what information should be inputted into the model, paying little attention to finding the optimal location. Among the latter, the existing parameter optimization method is optimal (Meng et al. 2022a,b; Li et al. 2024), which divides the model editing task into two parts: location and optimization, and points out where knowledge is stored.

*Corresponding authors

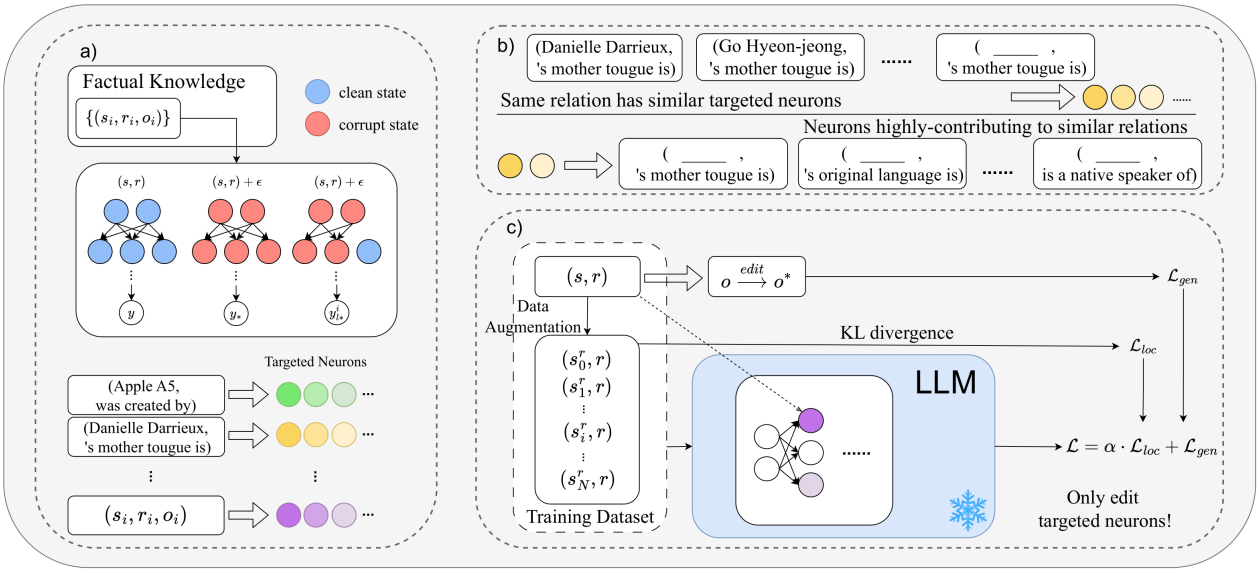

Figure 2: **TNF-DA modifies the parameters that are the targeted neurons of the edited knowledge.** We edit the knowledge stored in the model based on the relationship between neurons and knowledge: (a) The targeted neurons are collected through causal mediation analysis in FFNs. (b) Then analysis of the targeted neurons shows that the similar relational description has similar targeted neurons. (c) The targeted neurons are modified through an augmented dataset based on the relational description.

The considerations of existing methods mainly revolve around the reliability of editing, and often ignore the means of maintaining the locality of editing, which means keeping the irrelevant knowledge unaffected. A typical strategy is to limit the range of changes in weights. This goes a long way in maintaining model generation capabilities, but lacks explanation in maintaining editing locality. One of the important reasons is that it is difficult to judge whether knowledge is irrelevant. This results in the difficulty of designing some explicit method to control it. Dai et al. pointed out that there are specific "knowledge neurons" related to knowledge in the language models, and the impact on other knowledge is small when editing these neurons. Motivated by this, we try to start from the storage of knowledge in the model and the corresponding model structure. Compared with "knowledge neurons", we further compare the characteristics of strongly associated neurons corresponding to relevant knowledge. It was found that when knowledge has similar relational representations, their corresponding strongly associated neurons have high overlap, and these "targeted neurons" often have high contributions to knowledge with similar relational representations at the same time (eg. A neuron with a high contribution to "{} was created by" will also have a high contribution to "{} was developed by").

Depending on this finding, we propose TNF-DA, an editing approach through data augmentation. We try to edit these targeted neurons, which means that it may affect the knowledge expressed by the same or similar relationships. So in order to maintain this knowledge while editing, for the given edited knowledge, we construct a relevant dataset based on its relational description where the relation of all training data is the same as the edited knowledge. Using the relevant datasets to finetune a minimal number of parameters of targeted neurons, we can edit the knowledge and mitigate the overfitting problem associated with pure fine-tuning-based methods.

To summarize, our contributions are as follows:

- We confirm that in LLMs, only approximately 1% neurons have a high contribution to a knowledge representation, while the majority of neurons either have no impact or even exhibit a negative effect. We refer these highly-contributing neurons as "targeted neurons".
- We find that targeted neurons associated with knowledge sharing the same relation description exhibit a high overlap, and some of them are highly contributing to the knowledge description with similar semantic relations.
- We propose TNF-DA, a method for model editing that involves augmenting the dataset with edited knowledge based on relational descriptions and partially fine-tuning the targeted neurons related to this knowledge. Our experiments demonstrate that this approach achieves desirable performance in model editing, especially compared to previous fine-tuning-based methods.

Related Work

Model Editing

Model editing is an emerging field in recent years. So far, editing methods can be divided into two categories: adding extra input to a frozen model somewhere and modifying the model's parameters directly (Hartvigsen et al. 2024; Huang et al. 2023; Meng et al. 2022a,b). For the former, additional inputs can be added in the form of inputs as spliced context,

utilizing in-context learning (Zheng et al. 2023; Madaan et al. 2022). GRACE (Hartvigsen et al. 2024) and REMEDI (Hernandez, Li, and Andreas 2023) implements adding extra activation values by adding a bank outside the model. T-patcher (Huang et al. 2023) adds "pather", a small number of learnable parameters to the frozen model to achieve model editing. As for modifying the model's parameters directly, FT-based methods are proposed initially, like multi-loss fine-tune (Sinitsin et al. 2020) and constrained fine-tune (Zhu et al. 2020). Since these methods suffer from overfitting, meta-learning has been further used. By training hypernetworks, KE (De Cao, Aziz, and Titov 2021) and MEND (Mitchell et al. 2021) can get the post-edit parameters in a short time. ROME (Meng et al. 2022a) converts model editing into a locating and optimization task. Through "Causal Trace", it is considered that knowledge is stored in specific layers at the last subject token, where model editing can be done by optimizing the parameters there. Based on ROME, MEMIT (Meng et al. 2022b) and PMET (Li et al. 2024) further deal with the problem of mass-editing, modifying parameters of continuous adjacent layers.

Explanation of Transformers

The parameters of Transformers are basically composed of a multi-head self-attention module (MHSA) and feed-forward network (FFN) (Kovaleva et al. 2019; Hassid et al. 2022; Geva et al. 2023). It is widely believed that, considered as key-value pairs, FFN stores factual knowledge (Geva et al. 2020). FFN in each layer contributes corresponding activations and sums up through the residual network. MHSA is regarded to undertake the syntactic understanding (Voita et al. 2019). In the token dimension, MHSA processes positional information. Each token is able to obtain the above information(both information below in the encoder module) through the self-attention module. It is implied that focusing on the interaction between context input, MHSA plays a role in extracting knowledge and relationships.

Attribution on Large Language Models

Limited by the interpretability of deep learning, how the deep-network-based LLMs demonstrate excellent language capabilities remains a mystery. Currently, there are many works trying to analyze the attribution in deep networks and LLMs. "Integrated Gradients" (Sundararajan, Taly, and Yan 2017) evaluates the attribution importance by judging the impact on outputs as the inputs change. For language models, it can quickly determine which token in the input plays a key role in the output. Causal mediation analysis (Vig et al. 2020)(also known as Activation Patching) can detect which parts of the model are causally implicated in its behavior through a knock-in or knock-out way. Furthermore, Causal trace (Meng et al. 2022a) is proposed based on causal mediation analysis, which pays attention to the token dimension instead. COAR (Shah, Ilyas, and Madry 2024) transforms the attribution task into a supervised learning task to find important structures while considering interactions. Nanda et al. propose Attribution Patching(AtP), a faster, approximate, causal attribution method and AtP* (Kramár et al. 2024) further builds on this to filter out the false positives.

Methodology

Problem Definition

Model editing, aims to adjust an initial model f_θ through an edit description (x_e, y_e) (Yao et al. 2023), to modify model's some performance and get a post-edit model f_{θ_e}. Regularly, given a practical edit description with edit input x_e and edit label y_e ($f_\theta(x_e) \neq y_e$), the post-edit model f_{θ_e} is adjusted to predict the post-edit output, where $f_{\theta_e}(x_e) = y_e$. At the same time, a broad set of input associated with the edit description may get new labels from the post-edit model, and the collection of these inputs and edit description is regarded as an **editing scope** $I(x_e, y_e)$ (Yao et al. 2023), while the others remain the same as $O(x_e, y_e)$. So a successful edit operation should change the behavior of model for the inputs in editing scope rightly while remaining its performance for ones out of editing scope:

$$f_{\theta_e}(x) = \begin{cases} y_e & if\ x \in I(x_e, y_e) \\ f_\theta(x) & if\ x \in O(x_e, y_e) \end{cases} \tag{1}$$

According to the edit description (x_e, y_e), we represent them as a knowledge tuple $t = (s, r, o^*)$ (Meng et al. 2022a), composed of subject s, new object o^* and relation r, while the original object is o. Each time, we provide a natural prompt $p(s, r)$ describing (s, r) to the post-edit model and check whether the output of the model is o^* instead of o or something else.

Analysis of the Relationship Between Neurons and Knowledge

In this section, following the design of causal mediation analysis(CMA) (Vig et al. 2020) and Causal Trace(CT) (Meng et al. 2022a), we try to explore the relationship between neurons in model f_θ and knowledge (s, r, o). While CMA and CT focus on the output activations of an overall module, we are concerned about the activations of the hidden states of FFN, which is widely regarded to store factual knowledge. To make it easy to distinguish, we call the activations of the FFN "hidden states" and the activations of the hidden states of the FFN "inner states". In general, hidden size D and inner size S satisfy:

$$S = 4 \times D \tag{2}$$

To evaluate each neuron's correlation to a factual knowledge, we set three runs for each knowledge description, similar to the setting of Meng et al.:

- **clean run**: Pass a prompt x describing (s, r) into f, and collect the activations of inner states:

$$H_c = \{h_l^i \mid l \in [1, L], i \in [1, S]\} \tag{3}$$

- **corrupt run**: The whole prompt is obscured before predicting. Specifically, after embedding x as $H_{emb} = [h_0^i]$, we adjust $H_{emb} = H_{emb} + \epsilon$, where $\epsilon \sim N(0; \nu)$, then collect $P_*(o)$ (the probability of emitting o) and the corrupted activations:

$$H_* = \{h_{l*}^i \mid l \in [1, L], i \in [1, S]\} \tag{4}$$

- **corrupted-with-restoration run**: While running with the setting of a corrupt run, we keep the inner state h_l^i clean. The clean state will carry the clean information and pass it back to the final output, though others are under perturbation. We record the $P_{*,clean_l^i}(o)$ for every corrupted-with-restoration run, and calculate the indirect effect(IE) as $IE_{l,i} = P_{*,clean_l^i}(o) - P_*(o)$

We conduct experiments on GPT2-small and GPT2-medium (Radford et al. 2019)(without fine-tuning) using 1000 knowledge descriptions they know for sure. As Figure 3 shows, evaluated by IE, it can be obviously found that in the model, about 1% neurons have a high contribution to the corresponding knowledge. While most neurons have little contribution to the probability of expected output, a few of them even have a negative impact. For these neurons that have a great positive effect on associated knowledge description, we call them 'targeted neurons', which is similar to the concept in KN (Dai et al. 2021). Since each neuron has its corresponding key, value parameters in FFNs, we would like to regard the associated parameters as θ_{kn}, seen in Figure 4.

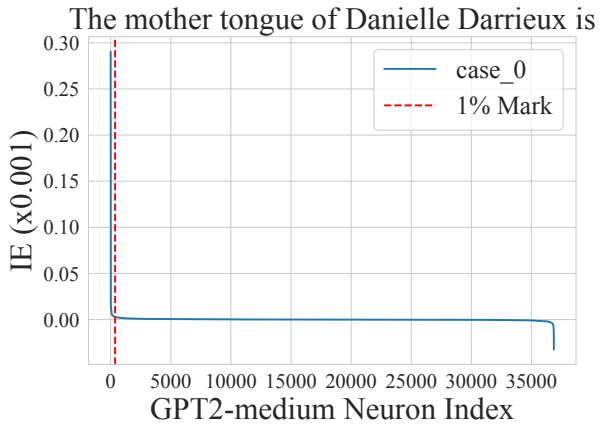

Figure 3: An instance of IE distribution characteristics of neurons in GPT2-medium.

In addition, we organize the targeted neurons corresponding to this knowledge with the same r, counting the top 2 targeted neurons with the highest frequency in each same r and calculating their average frequency in all descriptions.

model	Top1	Top2	Base	Repetition
gpt2-small	99.43	98.71	5.25	29.22
gpt2-medium	94.82	93.27	3.95	31.65

Table 1: Average repetition rate of all targeted neurons on knowledge with same relation and frequency of the top2 targeted neurons. "Base" is the lowest value of frequency.

Shown in Table 1, it is obvious that when r is the same, the knowledge probably has some repeated targeted neurons. The Repetition reflects the average number of targeted neurons with the same r (relative to the total number of targeted neurons that may appear). The lower the Repetition,

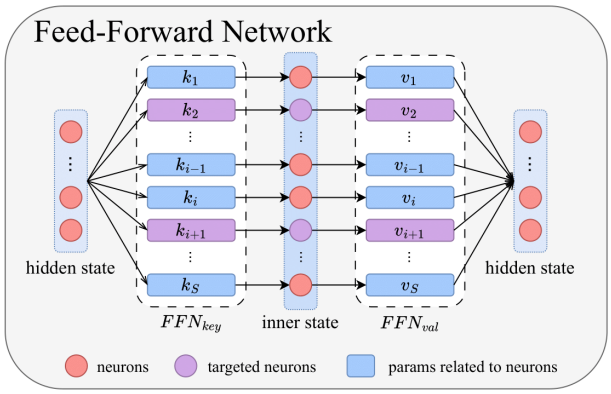

Figure 4: Relation between neurons and parameters. A FFN can be divided into FFN_{key} and FFN_{val}, as a set of key-value pairs. A targeted neuron of the inner states has a pair of k-v pairs corresponding to it, and θ_{kn} is the collection of the k-v pairs.

the fewer targeted neurons, and the higher repetition rate of targeted neurons with the same r.

Furthermore, we organize the common neurons of each r, finding that these common neurons are not only limited to the same r, but also the semantically similar ones, for example:

- "The mother tongue of {} is", "{} is a native speaker of", "The office language of {} is" , ...
- "{}, created by", "{}, developed by", "{} is a product of", ...

In summary, our results show that (1) Only 1% neurons in FFNs of the model are highly contributed to the corresponding knowledge, and (2) Knowledge descriptions with the same or similar relational semantics have similar knowledge neurons. Same experiments are conducted on GPT-J(6B) (Wang and Komatsuzaki 2021), and we also try to select such neurons by integrating gradients as Dai et al.. The experimental results and details in this section are shown in Appendix A.

TNF-DA Method

Based on the finding of "targeted neurons" distributed in different layers, we explore a simple yet effective way to modify the parameters for knowledge editing. As the FFNs in LLMs are widely regarded as key-value memories, we fine-tune the base model only on parameters corresponding to the 1% highly-contributing neurons (targeted neurons). With few iteration epochs, the model editing can be well achieved while preserving the unmodified facts.

In a standard condition, learning or editing a fact (p, o) through fine-tuning ways targets to the expected prediction:

$$L_{gen}(p\,;\theta) = -\log \mathbb{P}_\theta(o \mid p) \tag{5}$$

The cost of fine-tuning the entire model is huge, so only the parameters of the knowledge neurons θ_{kn} are collected to modify, with the rest $\theta \backslash \theta_{kn}$ frozen. However, as the model

scale increases, the time for searching targeted neurons increases. To improve the efficiency, we heuristically narrow the search place, analyzing the degree of noise influences DNI:

$$DNI_l^i = |\frac{h_l^i - h_{l*}^i}{h_l^i}| \tag{6}$$

Among neurons with large DNI, a searching subspace H_s can be located:

$$H_s = \{h_l^i \in H \mid DNI_l^i \in P_\gamma(DNI)\} \tag{7}$$

where $P_\gamma(DNI)$ represents the set of DNI values in the top $\gamma\%$. From the optimized searching space, the highly-contributing neuron is denoted as θ_{okn}, still evaluated by IE. To aid generalization, the edited description $(p(s,r),o)$ will be used for training concatenating some random prefixes x_j as $\{x_j \oplus p\}$:

$$L_{gen} = \frac{1}{N}\sum_{j=1}^{N} L_{gen}(x_j \oplus p\,;\theta_{okn}) \tag{8}$$

In order to keep the model's performance unchanged on irrelevant knowledge while editing, constrained loss is necessary (Zhu et al. 2020):

$$minimize_{\theta_{okn}}\ L_{gen}(p\,;\theta_{okn}) \tag{9}$$

$$subject\ to \sum_{p' \in O(p,o)} (L_{gen}(p')) \le \delta \tag{10}$$

Through the constraint of Eqn 10, the performance of the model on irrelevant knowledge can be controlled. Instead of building the dataset from the huge uncertain complement $O(p(s,r),o)$, we only construct it through the relation r, based on the finding that knowledge descriptions with the same or similar relational semantics have similar targeted neurons, especially the same ones. Since that, for each edit description $(p(s,r),o)$ with the subject s and relation r, an edit dataset is constructed as:

$$\begin{aligned}\mathcal{D}_{(p,o)} = \{(p_i^r, \mathbb{P}_{\theta_0}(p_i^r)) \mid p_i^r = (s_i^r, r), s_i^r \in S^r\} \\ \cup \{x_j \oplus (p,o) \mid j \in [1,N]\}\end{aligned} \tag{11}$$

Here S^r is a set of randomly chosen subjects, and whether the model knows the prediction exactly is slight, using KL divergence as training loss:

$$L_{loc} = D_{KL}(\mathbb{P}_\theta(x \mid p^r) \,||\, \mathbb{P}_{\theta_0}(x \mid p^r)) \tag{12}$$

So finally the train loss is obtained by:

$$L = L_{gen} + \alpha \cdot L_{loc} \tag{13}$$

Here, α is a hyperparameter, adjusting the ratio of L_{loc} to the final loss L.

Experiments

Baselines and Datasets

Our experiments are conducted on GPT2-XL(1.5B). Our baseline methods mainly adopt several types of editing parameters directly, including improved Constrained Fine-Tuning(FT+W) (Zhu et al. 2020), the meta-learning method MEND (Mitchell et al. 2021), and the locate-and-optimize method ROME (Meng et al. 2022a), MEMIT (Meng et al. 2022b) and PMET (Li et al. 2024). For datasets, we performed counterfactual edit experiments on the dataset, COUNTERFACT (Meng et al. 2022a). More details about datasets can be found in Appendix B.

Editing Experiments

Following Meng et al., we give the following metrics to test the editing performance. **Efficacy Success**(ES) evaluates whether the post-edit model can predict the new object o^*: $\mathbb{E}_i\{argmax_o f_{\theta_e}(o \mid p(s_i,r_i)) = o^*\}$. **Paraphrase Success**(PS) measures whether the post-edit model can give the right answer with a prompt with a rephrasing of the original statement, and **Neighborhood Success**(NS) is for the irrelevant knowledge. **Editing Score**(S) is the harmonic mean of **ES**, **PS** and **NS**. Besides the four metrics to evaluate the editing validity, other two metrics (Meng et al. 2022b) for the post-edit model are below: **Reference Score**(RS) tests the consistent of the model, checking the TF-IDF similarity between a reference Wikipedia text and the generating text about o^*, and **Generation Entropy** for fluency degradation, computing as the weighted sum of the entropy of bi- and tri-gram n-gram distributions of generated text.

In order to evaluate the performance of the editing method purely, we first filter out what the model exactly knows, as $f_\theta(p(s,r)) = o$, ignoring the uncertainty. And we finally get 2K pieces of counterfactual edits for GPT2-xl. More experiment details are shown in Appendix C.

Table 2 shows the results of all chosen methods on the 2K edits. The results show that, compared with previous fine-tuning-based methods, TNF-DA achieves a great improvement, especially in specificity. MEND has a great effect in terms of efficacy, but an obvious decrease in generalization and specificity. In terms of the fluency of the post-edit model, FT-W and MEND are both inferior to FT and TNF-DA, which is probably due to the influence of excessive changes in model parameters. In both FT-W and MEND, the range of weight changes is restricted, which can have a beneficial effect on maintaining the fluency of model generation, but will sacrifice generalization or specificity. As for optimization-based methods like ROME and PMET, there is no significant difference in final score between them and TNF-DA, since they have performed well enough on this experiment dataset. Similarly, even though they have their own advantages and disadvantages in editing performance, the fluency and consistency of the model edited by TNF-DA are reduced, similar to other baseline methods.

Case Study

For insight into TNF-DA's performance, we set up 3 different case experiments to explore the effect of targeted neurons and the possible problems with fine-tuning-based methods.

Influence of adding prefix As Eqn 8 tells, while training the model with the edits, we first concatenate some random

Editor	Score	Efficacy	Generalization	Specificity	Fluency	Consistency
GPT2-xl(1.5B)	/	0	0	100	625.26	70.35
FT	52.47	96.81	70.02	30.72	613.48	75.18
FT+W	38.70	90.04	20.85	62.37	624.71	75.35
MEND	57.92	99.13	65.45	37.90	624.23	71.46
TNF-DA	**87.65**	98.54	**87.21**	**79.42**	615.02	74.27
ROME	90.99	99.64	95.10	80.52	621.91	76.61
MEMIT	86.72	90.23	76.42	95.96	626.92	76.56
PMET	87.15	94.06	78.38	90.04	627.01	76.60
TNF-DA	87.65	98.51	87.23	79.41	615.02	74.27

Table 2: Quantitative Editing Results. The above part is the comparison result among ours and previous fine-tuning-based methods and hypernetwork-based methods. The bottom part is the comparison to the locate-and-optimize methods.

Editor	Efficacy	Generalization	Specificity
TWP	100.0	95.83	42.06
TNWP	100.0	88.89	26.98

Table 3: The results of the influence of adding prefixes. TWP represents training the edits with random prefixes. TNWP represents training without any prefix.

prefixes x_j as $\{x_j \oplus p\}$. The reason for that is to aid generalization across contexts (Meng et al. 2022b), preventing the model from overfitting to the given fixed edit description during training. In this subsection, we experimented with comparing whether or not to use these random prefixes. Based on the settings of **Ours-DA** without data augmentation, we finetune the parameters of targeted neurons with only the give edit $(p(s,r),o)$ and $(x_j \oplus p(s,r),o)$, and set a sufficient number of iterations to ensure the efficiency.

Table 3 shows the result of training with prefixes (**TWP**) or not(**TNWP**). It is obvious that training with prefixes (TWP) and without prefixes both achieve desirable results in generalization, but poor performance in specificity. This is probably due to the fact that the large number of iterations. However, we can find that when additional prefixes is concatenated to the training data, the performance in specificity will still be greatly improved, not only in generalization. This proves that training with random prefixes can mitigate the overfitting in a fine-tuning process, allowing the model to learn the knowledge paradigm rather than the fixed input prompts themselves.

Generation loss convergence and neuron chosen In order to further verify the effectiveness of the targeted neuron in the TNF-DA method and its relevance to knowledge, we conduct an experiment with the convergence speed of the generation loss when training different parameters in this section. We select some edits, and train them with standard TNF-DA and TNF-DA without targeted neurons (**Ours-RN**). We randomly select the parameters corresponding to the same number of neurons(1%) as the editable parameters and record the generation loss to compare their convergence speeds.

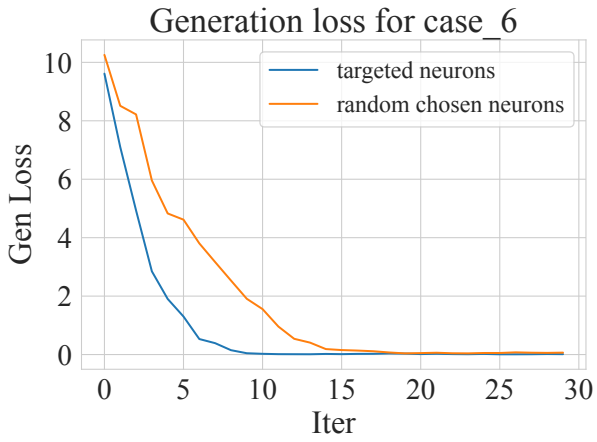

Figure 5: An instance showing the difference of convergence speed when modifying targeted neurons and randomly-chosen neurons.

Experimental results of convergence speed differences are shown in Table 5. As can be seen from the figure, the standard TNF-DA method of editing the parameters of the targeted neuron converges significantly faster than the random selection of neuron parameters in the generation loss. This largely shows that targeted editing of these high-contributed neurons is effective to a certain extent, and also proves the high correlation between the targeted neurons and its corresponding knowledge. More results are shown in Appendix D.

Different knowledge in the model Note the difference in different knowledge in Figure 3: under the same noise perturbation, the IE values given by the neuron may have significant differences. Therefore, we conduct an experiment to explore the difficulty of editing different knowledge and try to explain the difficulties of fine-tuning-based editing methods. We sample several groups of edits with the same relation from the dataset, and used TNF-DA to edit until their generation loss was less than a threshold instead of a fixed

Editor	Score	Efficacy	Generalization	Specificity	Fluency	Consistency
GPT2-xl(1.5B)	/	0	0	100	625.26	70.35
Ours	**87.65**	98.54	87.21	**79.42**	615.02	74.27
Ours-RN	52.73	94.46	66.01	32.14	608.21	71.52
Ours-DA	77.39	98.50	88.61	56.89	618.80	74.03

Table 4: The results of the ablation experiments. Ours represents the standard TNF-DA. Ours-RN represents modifying randomly-chosen parameters. Ours-DA represents that the training dataset is augmented randomly.

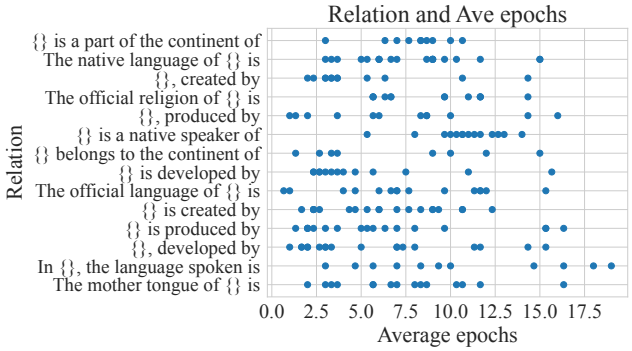

Figure 6: The average number of epochs for editing different knowledge until getting a low enough generation loss.

epochs:

$$L_{gen} \leq \delta \tag{14}$$

Each edit is tested 5 times and takes the average of the number of stopping rounds, and the results are shown in Figure 6.

It can be seen that different relations do not have similarities, and even the same relations do not have obvious characteristics. This means that it may be difficult to ensure extremely high accuracy for epoch-fixed training, that is, it may lead to overfitting for easy-to-edit knowledge, while difficult-to-edit knowledge may not be completely changed. For the dynamic epoch solution, this has a similar impact on the relevant and irrelevant knowledge, making it difficult to achieve high generalization and specificity.

Ablation Study

To demonstrate the effectiveness of our editing method with editing targeted neurons and training datasets based on data augmentation, we conduct ablation studies for TNF-DA, detailed in Table 4.

To assess the effectiveness of editing targeted neurons only, we edit parameters corresponding to a set of randomly chosen neurons (**Ours-RN**). TNF-DA's performance on specificity significantly degrades while editing the randomly chosen parameters, proving the relationship between the targeted neurons and knowledge based on the relation description. In addition, the efficacy remains at a high level, but the decrease in paraphrase is obvious, showing the relationship between the targeted neurons and knowledge based on the relation description. This is probably because the model overfits the edited knowledge description, but does not generalize to other rephrasing statements well.

Replacing the data augmentation strategy based on relation description, a random generation strategy (**Ours-RD**) is adopted for comparative experiments. While it has decreased slightly in generalization, but is obviously insufficient in specificity, indicating that the targeted neurons are more likely to contribute to knowledge with the same or similar relation description.

Conclusion

We find that approximately 1% targeted neurons in FFN of a language model are highly contributing to the correlated knowledge storage. These targeted neurons are consistently associated with the relational description of the knowledge, and exhibit a high repetition rate for knowledge with the same or similar relational descriptions. Based on the observations, we propose a fine-tuning-based method, TNF-DA, which achieves model editing by modifying the parameters of targeted neurons associated with the edited knowledge, using an augmented dataset based on the relational descriptions. Our experiments on Counterfact demonstrate that TNF-DA achieves a superior editing performance in all finetune-based methods, and produces excellent results comparable to the locate-and-optimize model editing methods. Furthermore, our ablation studies and case studies indicate that our strategies for targeted neurons and data augmentation are effective, mitigating the challenge faced by the previous fine-tuning-based methods in balancing generalization and locality. Our findings provide additional insights into the operational dynamics of FFN in language models.

Limitations

Without using the knowledge graph as an aid, TNF-DA has not been further designed for the performance of reasoning-related knowledge. The method is based on the semantics level to conduct neuron experiments, so it is difficult for the edited model to reason based on the edited knowledge. In addition, TNF-DA is a fine-tuning-based method. Since the targeted neurons are distributed across different layers of the model, it is challenging to enhance the efficiency of editing parameters through optimization like ROME and MEMIT (Meng et al. 2022a,b). In terms of targeted neurons, it can be found that there are a small number of neurons playing a negative role, rather than showing little impact. Our work has not further explored this aspect of neurons. On the other hand, due to the dispersed nature of these neurons, we do not further consider their mutual interactions.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant No. 62276110, No. 62172039, and in part by the fund of Joint Laboratory of HUST and Pingan Property & Casualty Research (HPL). The authors would also like to thank the anonymous reviewers for their comments on improving the quality of this paper.

References

Carlini, N.; Liu, C.; Erlingsson, Ú.; Kos, J.; and Song, D. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX security symposium (USENIX security 19)*, 267–284.

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. *arXiv preprint arXiv:2104.08164*.

Geva, M.; Bastings, J.; Filippova, K.; and Globerson, A. 2023. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*.

Geva, M.; Schuster, R.; Berant, J.; and Levy, O. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Hartvigsen, T.; Sankaranarayanan, S.; Palangi, H.; Kim, Y.; and Ghassemi, M. 2024. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36.

Hassid, M.; Peng, H.; Rotem, D.; Kasai, J.; Montero, I.; Smith, N. A.; and Schwartz, R. 2022. How much does attention actually attend? questioning the importance of attention in pretrained transformers. *arXiv preprint arXiv:2211.03495*.

Hernandez, E.; Li, B. Z.; and Andreas, J. 2023. Inspecting and editing knowledge representations in language models. *arXiv preprint arXiv:2304.00740*.

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*.

Jiang, Z.; Xu, F. F.; Araki, J.; and Neubig, G. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8: 423–438.

Kovaleva, O.; Romanov, A.; Rogers, A.; and Rumshisky, A. 2019. Revealing the dark secrets of BERT. *arXiv preprint arXiv:1908.08593*.

Kramár, J.; Lieberum, T.; Shah, R.; and Nanda, N. 2024. AtP*: An efficient and scalable method for localizing LLM behaviour to components. *arXiv preprint arXiv:2403.00745*.

Li, X.; Li, S.; Song, S.; Yang, J.; Ma, J.; and Yu, J. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 18564–18572.

Madaan, A.; Tandon, N.; Clark, P.; and Yang, Y. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. *arXiv preprint arXiv:2201.06009*.

Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 1.

Meng, K.; Bau, D.; Andonian, A.; and Belinkov, Y. 2022a. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35: 17359–17372.

Meng, K.; Sharma, A. S.; Andonian, A.; Belinkov, Y.; and Bau, D. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Mitchell, E.; Lin, C.; Bosselut, A.; Finn, C.; and Manning, C. D. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Mitchell, E.; Lin, C.; Bosselut, A.; Manning, C. D.; and Finn, C. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, 15817–15831. PMLR.

Nanda, N.; Rajamanoharan, S.; Kramár, J.; and Shah, R. 2023. Fact Finding: Attempting to Reverse-Engineer Factual Recall on the Neuron Level.

Petroni, F.; Rocktäschel, T.; Lewis, P.; Bakhtin, A.; Wu, Y.; Miller, A. H.; and Riedel, S. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Qiao, S.; Ou, Y.; Zhang, N.; Chen, X.; Yao, Y.; Deng, S.; Tan, C.; Huang, F.; and Chen, H. 2022. Reasoning with language model prompting: A survey. *arXiv preprint arXiv:2212.09597*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Roberts, A.; Raffel, C.; and Shazeer, N. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Shah, H.; Ilyas, A.; and Madry, A. 2024. Decomposing and editing predictions by modeling model computation. *arXiv preprint arXiv:2404.11534*.

Shin, T.; Razeghi, Y.; Logan IV, R. L.; Wallace, E.; and Singh, S. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Sinitsin, A.; Plokhotnyuk, V.; Pyrkin, D.; Popov, S.; and Babenko, A. 2020. Editable neural networks. *arXiv preprint arXiv:2004.00345*.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, 3319–3328. PMLR.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Vig, J.; Gehrmann, S.; Belinkov, Y.; Qian, S.; Nevo, D.; Singer, Y.; and Shieber, S. 2020. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33: 12388–12401.

Voita, E.; Talbot, D.; Moiseev, F.; Sennrich, R.; and Titov, I. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

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

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. *arXiv preprint arXiv:2305.13172*.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zheng, C.; Li, L.; Dong, Q.; Fan, Y.; Wu, Z.; Xu, J.; and Chang, B. 2023. Can we edit factual knowledge by in-context learning? *arXiv preprint arXiv:2305.12740*.

Zhu, C.; Rawat, A. S.; Zaheer, M.; Bhojanapalli, S.; Li, D.; Yu, F.; and Kumar, S. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.