DAFNet: Dynamic Auxiliary Fusion for Sequential Model Editing in Large Language Models

Taolin Zhang[1] *, **Qizhou Chen**[2,1*], **Dongyang Li**[2,1*], **Chengyu Wang**[1†], **Xiaofeng He**[2†]
Longtao Huang[1], **Hui Xue**[1], **Jun Huang**[1]
[1] Alibaba Group [2] East China Normal University
{zhangtaolin.ztl, chengyu.wcy}@alibaba-inc.com, hexf@cs.ecnu.edu.cn

Abstract

Recently, while large language models (LLMs) have demonstrated impressive results, they still suffer from hallucination, i.e., the generation of false information. Model editing is the task of fixing factual mistakes in LLMs; yet, most previous works treat it as a one-time task, paying little attention to ever-emerging mistakes generated by LLMs. We address the task of sequential model editing (SME) that aims to rectify mistakes continuously. A **D**ynamic **A**uxiliary **F**usion **Net**work (DAFNet) is designed to enhance the semantic interaction among the factual knowledge within the entire sequence, preventing catastrophic forgetting during the editing process of multiple knowledge triples. Specifically, (1) for semantic fusion within a relation triple, we aggregate the intra-editing attention flow into auto-regressive self-attention with token-level granularity in LLMs. We further leverage multi-layer diagonal inter-editing attention flow to update the weighted representations of the entire sequence-level granularity. (2) Considering that auxiliary parameters are required to store the knowledge for sequential editing, we construct a new dataset named **DAFSet**, fulfilling recent, popular, long-tail and robust properties to enhance the generality of sequential editing. Experiments show DAFNet significantly outperforms strong baselines in single-turn and sequential editing. The usage of DAFSet also consistently improves the performance of other auxiliary network-based methods in various scenarios [1].

1 Introduction

Transformer-based models, particularly LLMs (Devlin et al., 2019; Brown et al., 2020; Touvron et al., 2023; Roumeliotis and Tselikas, 2023) have become backbones of modern NLP and delivered promising results in various downstream tasks (Li et al., 2022; Zheng et al., 2022; Blevins et al., 2023; Blinova et al., 2023). However, LLMs still produce undesirable outputs occasionally (Basta et al., 2021; An et al., 2023). The cost of such mistakes is non-negligible and exhibits an inclination to generate hallucinations (Shi et al., 2023; Tam et al., 2023), resulting in seemingly plausible yet factually unsupported contents. To alleviate these problems, there has been growing interest in integrating knowledge into LLMs through model editing (Cao et al., 2021; Madaan et al., 2022; Meng et al., 2023). It updates the knowledge stored in relevant parameters in LLMs without fine-tuning the whole model.

In the literature, previous model editing methods[2] can be divided into two categories, including `Single-turn Editing` and `Sequential Editing`. (1) `Single-turn Editing` edits one or a batch of knowledge triples at a time via modifying the original parameters of LLMs based on meta-learning (Cao et al., 2021; Mitchell et al., 2022a) and locate-then-edit methods (Hartvigsen et al., 2022; Meng et al., 2022, 2023). These works struggle to handle factual knowledge that is continuously updated in real-world scenarios. (2) `Sequential Model Editing` (SME) based approaches learn the updated knowledge via adding extra modules (Dong et al., 2022; Mitchell et al., 2022b; Huang et al., 2023). The editing process of parameters is not achieved through back-propagation, but instead through weights independently calculated by extra modules associated with each fact. However, these methods do not take into account the effects of mutual semantic influence between these sequentially input facts. Another overlooked problem is that extra parameters are randomly initialized and can not

* T. Zhang, Q. Chen and D. Li contribute equally to this work.

† C. Wang and X. He are co-corresponding authors.

[1]The code and pre-trained models will be available at https://github.com/qizhou000/DAFNet

[2]"Model editing" and "knowledge editing" of LLMs share the same meaning. Thus, we use both terms interchangeably.

Findings of the Association for Computational Linguistics ACL 2024, pages 1588–1602
August 11-16, 2024

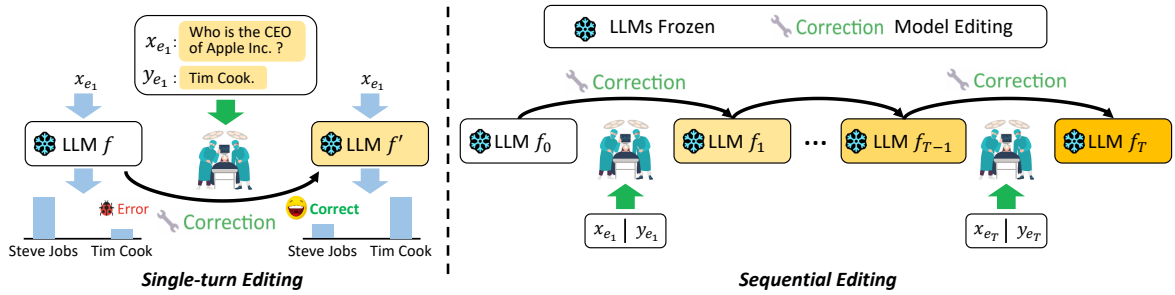

Figure 1: The comparison between different model editing scenarios for LLMs including single-turn and sequential editing with T steps. The single-turn editing model only edits one fact into the LLM at a time. In the sequential editing scenario, it requires to edit a series of facts continually (Best viewed in color).

provide a good starting point to cover the editing properties (Yao et al., 2023) of the testing data.[3] As shown in Figure 1, previous model editing methods edit the sequential facts independently similar to single-turn editing in SME scenario. Thus, subsequent LLMs need to wait for previous LLMs to finish editing before they can be edited.

In this paper, we propose a dynamic auxiliary fusion network named **DAFNet** to explicitly capture the correlation among the input sequential triples for the update of knowledgeable parameters in LLMs. In addition, an auxiliary dataset named **DAFSet** for learning the meta-edit weights is constructed to compensate for the learning gap between extra cold-start parameters and testing properties in the training stage. To our knowledge, this is currently the first comprehensive training set in model editing[4]. Specifically, we introduce the above two main contributions as follows:

DAFNet: Unlike previous methods that update parameters independently, we aggregate the representations of facts into intra- and inter-editing auto-regressive flows. The intra-editing attention flow gathers each token-level fact representation auto-repressively via assigning different semantic weights to tokens, decomposing the LLMs' high-dimensional outputs to two low-rank representations as editing signal. The inter-editing attention flow obtains interactive representations of sentence-level sequential inputs through multi-layer auto-regressive iterative diagonal attention between facts. Finally, our designed loss based on the desired editing properties is leveraged to train the auxiliary network.

DAFSet: In previous works (Cao et al., 2021; Mitchell et al., 2022a; Tan et al., 2023), weights of the original auxiliary network are usually randomly initialized and then combined with LLMs in multi-task training. Therefore, the lack of training set for existing editing methods causes the inconsistency with the distributional characteristics of the final editing target solely by editing the facts (Bickel et al., 2007; Cai et al., 2023). The goal of constructing our auxiliary dataset is to learn auxiliary meta-weights to compensate for the bias caused by random weight distributions. DAFSet is designed to include four different properties (i.e., `Recency`, `Popularity`, `Long-tailness` and `Robustness`) based on the test editing properties. We collect data for these properties via subject frequency and output likelihood in various domains to make the learned auxiliary weights equipped with more generalized editing abilities.

2 Related Work

In this section, we briefly overview the related works of model editing for LLMs in four aspects.

Adding Extra Modules: This method stores all edit examples in memory and uses a retriever to extract the most relevant facts for each new input, guiding the model in generating the edited fact. SERAC (Mitchell et al., 2022b) adopts a distinct counterfactual model while leaving the original model unchanged. Other methods edit LLMs by prompting the model with the edited fact and retrieved edit demonstrations from the memory such as MemPrompt (Madaan et al., 2022), IKE (Zheng et al., 2023) and MeLLo (Zhong et al., 2023).

Additional Parameters: This paradigm introduces additional trainable parameters in LLMs, which are trained using a modified knowledge dataset while keeping the original parameters unchanged. T-Patcher (Huang et al., 2023) and CaliNET (Dong

[3]The post-edit model should satisfy the following properties: reliability, generality and locality (Yao et al., 2023).

[4]Previous public datasets are used for testing editing ability of LLMs directly (Levy et al., 2017; Meng et al., 2022, 2023).

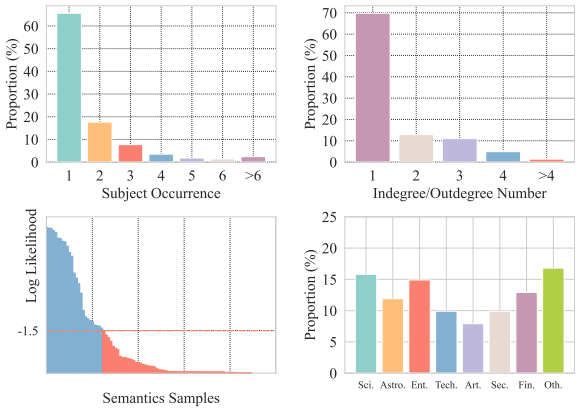

Figure 2: Statistical results of the collected DAFSet.

et al., 2022) integrate a new single neuron (patch) for each error in the last layer of the FFN. GRACE (Hartvigsen et al., 2022) maintains a discrete codebook module for the middle layer of the LLMs.

Locate-Then-Edit: The initial step involves identifying parameters that correspond to specific knowledge and then modifying them through direct updates to the target parameters. The work (Dai et al., 2022) introduces a method for identifying the specific "knowledge neuron" (a key-value pair in the FFN matrix) that represents the knowledge, and subsequently updating these neurons. ROME (Meng et al., 2022) applies causal mediation analysis to locate the editing area. MEMIT (Meng et al., 2023) expands on the setup of ROME, realizing the situation of synchronous editing for multiple cases.

Meta-learning: It utilizes a hyper-network to acquire the essential updated weights for editing. Knowledge Editor (KE) (Cao et al., 2021) utilizes a hyper-network to predict the weight updated for each data point. MEND (Mitchell et al., 2022a) learns to edit LLMs fastly improving the performance via employing a low-rank decomposition of gradients as the input. MALMEN (Tan et al., 2023) accommodates editing multiple facts with limited memory budgets via separating the computation on the hyper-network and LM enabling arbitrary batch size on both neural networks. Note that, all the above editing methods focus on single-turn edits with one or a batch of facts, not taking into account semantic connections among the facts in a sequential order.

3 Preliminaries of SME

In this section, we provide a brief introduction to SME for LLMs. A model $f \in \mathbb{F}$ can be defined as a function $f : \mathbb{X} \mapsto \mathbb{Y}$ that maps an input x to its prediction $f(x)$. Then, given a model f and an edit example pair (x_e, y_e) that $f(x_e) \neq y_e$, a model editor (ME) outputs a post-edit model f'.

$$\text{ME} : \mathbb{F} \times \mathbb{X} \times \mathbb{Y} \mapsto \mathbb{F} \tag{1}$$

$$f' = \text{ME}(f, x_e, y_e) \tag{2}$$

Given a sequence of facts $(x_{e_1}, y_{e_1}), ..., (x_{e_T}, y_{e_T})$ and an initial model f, a model editor ME needs to conduct edits successively when the model makes undesirable output:

$$f_t = \text{ME}\left(f_{t-1}, x_{e_t}, y_{e_t}\right), t = 1, ..., T \tag{3}$$

where assume $f_0 = f$. Every edit in SME should satisfy the following three properties:

Reliability A reliable edit holds when the post-edit model f_t gives the target answer for the every cases $(x_{e_\tau}, y_{e_\tau}), \tau \leq t$ to be edited. The reliability is measured as the average accuracy on the edit cases:

$$\mathbb{E}_{(x_e, y_e) \sim \{(x_{e_\tau}, y_{e_\tau})\}_{\tau=1}^{t}} \mathbb{I}\left\{\underset{y}{\text{argmax}}\, f_t\left(y \mid x_e\right) = y_e\right\} \tag{4}$$

Generality The post-edit model f_t should also satisfy the relevant neighbours $N(x_{e_\tau}, y_{e_\tau}), \tau \leq t$. It is evaluated by the average accuracy of f_t on examples drawn uniformly from the relevant neighborhood:

$$\mathbb{E}_{(x_e, y_e) \sim \{(x_{e_\tau}, y_{e_\tau})\}_{\tau=1}^{t}} \mathbb{E}_{(x_g, y_g) \sim N(x_e, y_e)}\, \mathrm{G}(x_g, y_g)$$
$$\text{s.t. } \mathrm{G}(x_g, y_g) = \mathbb{I}\left\{\underset{y}{\text{argmax}}\, f_t\left(y \mid x_g\right) = y_g\right\} \tag{5}$$

Locality Editing should be implemented locally, which means the post-edit model f_t should not change the output of irrelevant examples in out-of-scope $O(x_{e_\tau}, y_{e_\tau}), \tau \leq t$. Hence, the locality is evaluated by the rate at which the post-edit model f_t's predictions are unchanged as the pre-edit model f:

$$\mathbb{E}_{(x_e, y_e) \sim \{(x_{e_\tau}, y_{e_\tau})\}_{\tau=1}^{t}} \mathbb{E}_{(x_l, y_l) \sim O(x_e, y_e)}\, \mathrm{L}(x_l, y_l)$$
$$\text{s.t. } \mathrm{L}(x_l, y_l) = \mathbb{I}\left\{f_t\left(y \mid x_l\right) = f\left(y \mid x_l\right)\right\} \tag{6}$$

4 The DAFSet Dataset

Due to the lack of training editing sets for various meta-learning based methods, we propose DAFSet to provide the initial editing ability of LLMs. It enables the model to better emerge with knowledge generalization ability (Yao et al., 2023; Cohen et al., 2023) in the test editing stage with four properties.

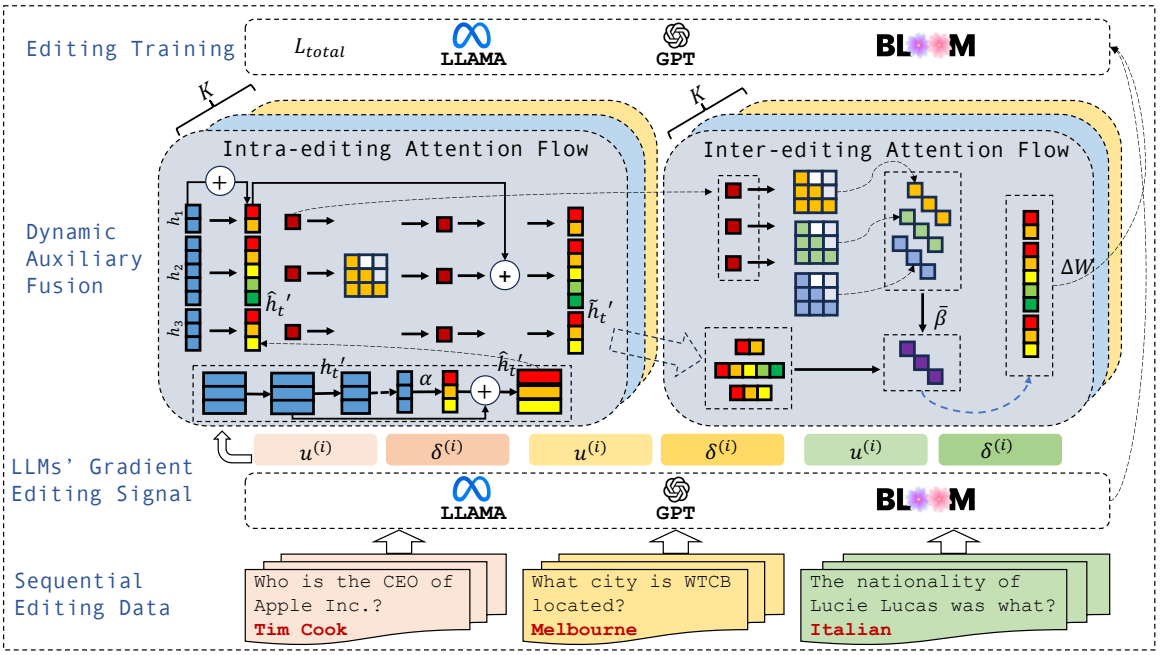

Figure 3: Model overview. Our DAFNet model mainly includes three steps: `Gradient Editing Signal Acquisition`, `Dynamic Auxiliary Fusion` and `Editing Training`. Particularly, `Dynamic Auxiliary Fusion` is designed with intra and inter-editing attention flows to capture the interaction between editing facts.

4.1 Data Collection

We leverage `Wikidata`[5], a knowledge graph (KG) with relation triples represented as (e_h, r, e_t), where e_h is the head entity, r is the relation predicate and e_t is the tail entity. The specific collection steps for these properties are as follows.

Recency: We gather triplets that have been recently added to Wikidata. The head e_h and tail entities e_t are often associated with numerous redundant triples. To address this, we only collect relation triples associated with a set of 48 common relation predicates. We use the off-the-shelf tool[6] to search the triples where each triple has been modified in the last 7 days. Then, we use templates to map the triples to construct the data samples, with templates shown in Appendix C.1.

Popularity: We collect triples corresponding to popular entities, where head entities are from top-viewed pages in Wikipedia. Next, we perform multi-hop selection of tail entities within 2-hops. Finally, we also leverage the above templates to construct the training samples.

Long-tailness: LLMs often lack sufficient learning of low-frequency data, and thus editing effect on such knowledge is poor. We identify and construct training data related to long-tail knowledge from three perspectives: (1) `Frequency`: we first collect head entities and count their corresponding frequencies in Wikidata. Then, we set a threshold to select approximately 80% of entities with low frequencies. (2) `In-out KG Degree`: we calculate the connectivity of the corresponding head entities in KG, and then set the threshold to select low-frequency entities. (3) `Likelihood`: unlike the two intuitive statistics, our approach to identify long-tail data focuses on the semantics of the model's output. Specifically, we input the sentence related to the head entity into LLMs and evaluate the model's comprehension of the entity by examining the likelihood probability of its position[7].

Robustness: We employ three robustness properties (Omar et al., 2022) for constructing the training data, including text length, context, and emotions. To control the length of input prompts, we utilize the "loc" and "rephrase" fields of each edited data (Levy et al., 2017; Mitchell et al., 2022a; Cohen et al., 2023). Templates are used for training sentence construction, specifically for context and emotions, to prompt the LLMs for generation. To further enhance the robustness, we generate two opposite data attributes for each training data point, such as "positive" and "negative" in emotions.

[5] https://en.wikipedia.org/wiki/Wikidata:Main_Page

[6] https://pypi.org/project/qwikidata/0.2.0/

[7] Detailed construction process is shown in Appendix C.1.

To learn auxiliary parameters and provide an initial meta-weight for editing, the data labeled as "recent" and "popular" is predominantly stable and frequently used to insert factual information. This leads to improved reliability and locality of test editing. In terms of the generality metric for test editing, the "long-tailness" and "robustness" categories can enhance the meta-weights, considering the sparsity of editing data. Overall, there are a total of 30,000 final constructed data samples for long-tailness, 13,000 for robustness, 30,000 for popularity, and 30,000 for recency.

4.2 Statistical Analysis on Our Dataset

We further conduct analysis on two complicated properties, i.e., long-tailness and robustness. As shown in Figure 2, we visualize the statistical results of our training data. The first subgraph on the left shows the frequency of head entities. We can see that the lower the frequency, the greater the proportion such as the sum of occurrence 1 and 2 has exceeded 80%. The second subgraph shows the proportion of head entities with different in-out kG' degrees. We see that most of the data has neighboring edges with high degrees and thus we select long-tail triples with low in-out degrees. The third subgraph shows the output likelihood probability of target entities in editing sentences. We select data samples with low log likelihood (i.e., < -1.5) output corresponding to a large proportion as our long-tail training sentences. In the fourth subgraph, given that robustness needs to be applicable across various domains, we analyze the distribution of domains in the generated training data for robust editing.

5 The DAFNet Model

In this section, we formally introduce our DAFNet model, with an overall architecture shown in Figure 3.

In DAFNet, to obtain rich semantic representation for each knowledge triple, we perform a token-level granularity fusion modeling by intra-editing attention flow in Section 5.2. Next, we leverage the auto-regressive modeling process to naturally capture the semantic interaction among sequential facts by inter-editing attention flow in Section 5.3.

5.1 Editing Signal Acquisition

In auxiliary network-based methods, we need to first determine the editing position before performing editing. We heuristically select linear layers at the last few layers of LLMs (Meng et al., 2022; Mitchell et al., 2022a). Then, we obtain the editing signal input to the auxiliary network by decomposing gradients of the editing weight matrix.

Specifically, the gradient of the editing loss w.r.t. the weight matrix (to be edited) of a certain linear layer W in a language model f is a summation of B rank-1 matrix, where B is the token count of an editing sample (x_e, y_e). Formally, we have

$$\begin{aligned} \nabla_W \mathcal{L}_{edit}(x_e, y_e) &= \sum_{i=1}^{B} u^{(i)\top} \cdot \delta^{(i)} \\ \text{s.t. } \mathcal{L}_{edit}(x_e, y_e) &= -\log f(y_e|x_e) \end{aligned} \tag{7}$$

where $u^{(i)} \in \mathbb{R}^{1\times d_{in}}, \delta^{(i)} \in \mathbb{R}^{1\times d_{out}}$ are the input and the output gradient of the linear layer at the i_{th} token position, respectively. Thus, the editing signal of weight W for editing sample (x_e, y_e) is defined as $h = [u; \delta] \in \mathbb{R}^{B\times(d_{in}+d_{out})}$. Note that editing signals w.r.t. different Ws are independently fed into auxiliary networks.

5.2 Intra-editing Attention Flow

In SME, the semantic modeling among facts has a significant influence on the overall performance. We first separately fuse the edited signal after dimensional reduction w.r.t. each token, and then aggregate the fused edited sequence fact into auto-regressive self-attention to further enhance the interaction between fact tokens.

Let the editing signal of the t_{th} editing fact as $h_t^0 \in \mathbb{R}^{B_t\times(d_{in}+d_{out})}$, which is the input to the first intra-editing attention module. Assume the input to the k_{th} layer is h_t^{k-1}. For the sake of simplicity, the layer symbol k is omitted below in this subsection. We obtain the token attention score α_i via allocating the representation importance. The aggregated fact representations $\widehat{h}_t \in \mathbb{R}^{d_{in}+d_{out}}$ fuse the tokens' representations together with the token attention score. Specifically, we first transform the editing signal output into fused representation $h'_t \in \mathbb{R}^{B\times(d_{in}+d_{out})}$ through a residual module as:

$$h'_t = (\sigma(h_t W_1 + b_1))W_2 + b_2 \tag{8}$$

where $W_1 \in \mathbb{R}^{(d_{in}+d_{out})\times d_{down}}$ and $W_2 \in \mathbb{R}^{d_{down}\times(d_{in}+d_{out})}$ are the linear layer weights. σ is ReLU function (Agarap, 2018). Then, we calculate the token attention weight $\alpha \in \mathbb{R}^{B_t\times 1}$ using the transformed representation h'_t. The fused editing representation $\widehat{h}_t$ is learned from the token-span

representation h'_t as follows:

$$\alpha = \varphi(\sigma(h'_t W_3 + b_3)W_4 + b_4), \widehat{h}_t = \alpha \otimes h'_t \tag{9}$$

$$\bar{h}_t = \sum_{i=1}^{B_t} \widehat{h}_t^{(i)}, \quad \widehat{h}'_t = \widehat{h}_t + h_t \tag{10}$$

where $W_3 \in \mathbb{R}^{(d_{in}+d_{out})\times d_{down}}$, $W_4 \in \mathbb{R}^{d_{down}\times 1}$. φ is the softmax non-linear activation function. $\otimes$ is the element-wise multiplication. Finally, each fused fact representation $\bar{h}_t \in \mathbb{R}^{1\times(d_{in}+d_{out})}$ is aggregated by the token importance. After obtaining the fused representations of all facts in the sequence, we transform the fused representations into the auto-regressive model for iteration learning to enhance the semantic information modeling between the fused facts:

$$\bar{h}'_1, \bar{h}'_2, ..., \bar{h}'_T = f_{intra}(\bar{h}_1, \bar{h}_2, ..., \bar{h}_T) \tag{11}$$

$$\tilde{h}_t = \widehat{h}'_t \oplus \bar{h}'_t, \quad t = 1, 2, ..., T \tag{12}$$

where f_{intra} is the auto-regressive self-attention layer to fuse the facts sequentially. $\bar{h}'_t \in \mathbb{R}^{1\times(d_{in}+d_{out})}$ is the i_{th} auto-regressive fact representation. T is the number of facts for sequential editing modeling. $\oplus$ is the element-wise addition. $\tilde{h}_t \in \mathbb{R}^{B_t\times(d_{in}+d_{out})}$ is the final intra-editing attention fusion representation. The intra-editing attention flow module is stacked by K layers. Finally, $\tilde{h}_t$ is fed as the input into the next module.

5.3 Inter-editing Attention Flow

Considering the effect of enhancing the interaction between facts in a sequence on updating iteration weights, we use the multi-layer auto-regressive diagonal attention to fuse the previous edited fact representations within a sequence. For each achieved editing fact weight score, we aggregate the last layer editing fusion weight representation $\tilde{h}_t, t = 1, ..., T$ to update the original LLMs.

Specifically, we use another multi-layer auto-regressive self-attention layer to learn the relationship of facts for sequential editing modeling:

$$\beta^k = \mathrm{diag}\left(f_{sa}^k\left(\bar{h}_1, \bar{h}_2, \ldots, \bar{h}_T\right)\right), k = 1, ..., K \tag{13}$$

where f_{sa}^k is the function to obtain the auto-regressive self-attention matrix at k_{th} layer. $\mathrm{diag}(*)$ is the function to achieve diagonal values of a matrix. $\beta^k \in \mathbb{R}^T$. Then, we aggregate the diagonal results of K layers into the averaged facts' importance: $\bar{\beta} = \frac{1}{K}\sum_{k=1}^{K}\beta^k$ where $\bar{\beta} \in \mathbb{R}^T$. Finally, the inter-editing fusion representations of each fact is $\tilde{h}_t^K \in \mathbb{R}^{B_t\times(d_{in}+d_{out})}$ at last K_{th} layer. We recover it to the original LLM's weight shape to obtain updated weights of the T sequential facts and then sum them weighted by $\bar{\beta}$:

$$[\tilde{u}_t; \tilde{\delta}_t] = \tilde{h}_t^K, \quad \Delta W_t = \frac{\tilde{u}_t^\top \otimes \tilde{\delta}_t}{B_t} \tag{14}$$

$$\Delta\tilde{W}_T = \sum_{t=1}^{T}\left(\prod_{\tau=t+1}^{T} 1 - \bar{\beta}_\tau\right)\bar{\beta}_t \cdot \Delta W_t \tag{15}$$

where $\tilde{u}_t \in \mathbb{R}^{B\times d_{in}}$ and $\tilde{\delta}_t \in \mathbb{R}^{B\times d_{out}}$ are the decomposed weight representations, respectively. $\Delta\tilde{W}_T \in \mathbb{R}^{d_{in}\times d_{out}}$ aggregates all the token weight representations together to achieve the fact updated weights' representation. Finally, the process of sequential editing by T facts can be formulated as: $f_T = \Gamma(f_{T-1}, \Delta\tilde{W}_T)$ where Γ indicates adding $\Delta\tilde{W}_T$ to the corresponding matrix to be edited. Algorithm 2 in the appendix describes the implementation of how DAFNet performs each edit one by one in sequential editing. Next we formulate the losses of f_T to model T edits into DAFNet.

5.4 Sequential Editing Training

Our auxiliary network considers three editing properties including reliability, generality and locality. Hence, the total loss with a T sequential editing facts is defined as follows:

$$\mathcal{L}_{rel}(f_T) = \sum_{t=1}^{T} -\log f_T(y_e^{(t)}|x_e^{(t)}) \tag{16}$$

$$\mathcal{L}_{gen}(f_T) = \sum_{t=1}^{T}\sum_{j=1}^{N_g^{(t)}} -\log f_T(y_{g_j}^{(t)}|x_{g_j}^{(t)}) \tag{17}$$

$$\mathcal{L}_{loc}(f, f_T) = \sum_{t=1}^{T}\sum_{j=1}^{N_l^{(t)}} \mathrm{KL}(f(x_{l_j}^{(t)})||f_T(x_{l_j}^{(t)})) \tag{18}$$

$$\mathcal{L}_{total} = \mathcal{L}_{rel}(f_T) + \mathcal{L}_{gen}(f_T) + \mathcal{L}_{loc}(f, f_T) \tag{19}$$

where $(x_e^{(t)}, y_e^{(t)})$ is the reliability sample of the t_{th} fact, i.e., the editing sample itself. $(x_{g_j}^{(t)}, y_{g_j}^{(t)})$ and $x_{l_j}^{(t)}$ are the j_{th} generality and locality sample of t_{th} fact, respectively. $N_g^{(t)}$ and $N_l^{(t)}$ are the corresponding loss sample number of t_{th} fact. KL is the Kullback-Leibler Divergence function. Algorithm 1 describes the training process of DAFNet.

Backbone	# Editing	Editor	ZSRE				CounterFact				RIPE			
			Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
GPT-J (6B)	10	FT	10.3	10.8	0.3	$7.1_{(\pm 0.1)}$	56.2	24.2	2.1	$27.5_{(\pm 0.5)}$	7.8	4.3	1.4	$4.5_{(\pm 0.1)}$
		TP	85.2	78.3	77.2	$80.2_{(\pm 1.2)}$	96.0	54.3	3.6	$51.3_{(\pm 1.2)}$	80.8	56.7	32.4	$56.6_{(\pm 1.7)}$
		KN	1.0	1.1	1.9	$1.3_{(\pm 0.0)}$	1.2	0.7	2.3	$1.4_{(\pm 0.0)}$	0.1	0.3	0.2	$0.2_{(\pm 0.0)}$
		ROME	81.1	78.8	94.6	$84.8_{(\pm 1.7)}$	95.9	59.4	90.0	$81.8_{(\pm 1.9)}$	98.2	41.9	39.1	$59.7_{(\pm 0.8)}$
		MEMIT	82.1	76.0	94.7	$84.2_{(\pm 2.0)}$	96.0	38.1	**95.5**	$76.5_{(\pm 2.5)}$	98.5	37.7	47.3	$61.2_{(\pm 1.2)}$
		GRACE	81.8	78.4	94.5	$84.9_{(\pm 1.6)}$	95.2	60.3	91.2	$82.2_{(\pm 1.6)}$	98.0	40.9	38.7	$59.2_{(\pm 0.4)}$
		KE♠	0.0	0.0	0.7	$0.3_{(\pm 0.0)}$	0.0	0.0	0.2	$0.1_{(\pm 0.0)}$	0.0	0.0	0.1	$0.0_{(\pm 0.0)}$
		MEND♠	0.4	0.4	0.5	$0.4_{(\pm 0.0)}$	0.6	0.2	0.2	$0.3_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		MALMEN♠	99.1	95.3	92.8	$95.8_{(\pm 1.6)}$	90.0	32.9	77.1	$66.7_{(\pm 2.2)}$	89.7	52.1	51.3	$64.4_{(\pm 1.8)}$
		DAFNet♠	**99.6**	**97.6**	**94.8**	**97.3**$_{(\pm 1.5)}$	**96.2**	**65.8**	85.2	**82.4**$_{(\pm 1.6)}$	**98.7**	**57.6**	**57.6**	**71.3**$_{(\pm 1.6)}$
	100	FT	2.2	1.9	0.3	$1.4_{(\pm 0.0)}$	35.9	10.8	1.6	$16.1_{(\pm 0.3)}$	5.7	1.6	0.1	$2.5_{(\pm 0.1)}$
		TP	68.5	59.3	52.8	$60.2_{(\pm 1.3)}$	76.0	31.9	2.2	$36.7_{(\pm 0.8)}$	64.2	36.4	23.7	$41.4_{(\pm 1.0)}$
		KN	0.6	0.4	0.8	$0.6_{(\pm 0.0)}$	0.2	0.5	0.8	$0.5_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		ROME	77.4	75.6	85.0	$79.3_{(\pm 2.2)}$	78.8	38.4	52.2	$56.5_{(\pm 1.0)}$	**95.7**	36.0	32.2	$54.6_{(\pm 1.0)}$
		MEMIT	77.9	74.1	90.2	$80.7_{(\pm 2.5)}$	**94.1**	40.2	85.1	**73.1**$_{(\pm 1.3)}$	86.6	33.3	33.5	$51.1_{(\pm 1.3)}$
		GRACE	77.8	74.6	85.9	$79.4_{(\pm 2.0)}$	76.3	39.2	51.6	$55.7_{(\pm 0.8)}$	94.8	36.7	31.5	$54.3_{(\pm 0.8)}$
		KE♠	0.0	0.0	0.7	$0.2_{(\pm 0.0)}$	0.0	0.0	0.1	$0.0_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		MEND♠	0.2	0.1	0.0	$0.1_{(\pm 0.0)}$	0.2	0.2	0.0	$0.1_{(\pm 0.0)}$	0.0	0.0	0.1	$0.0_{(\pm 0.0)}$
		MALMEN♠	50.6	40.7	59.3	$50.2_{(\pm 0.8)}$	29.7	31.8	68.0	$43.2_{(\pm 0.4)}$	39.9	27.8	53.2	$40.3_{(\pm 0.8)}$
		DAFNet♠	**89.5**	**76.5**	**90.2**	**85.4**$_{(\pm 1.6)}$	81.8	**40.3**	**87.3**	$69.8_{(\pm 1.5)}$	78.5	**38.9**	**64.4**	**60.6**$_{(\pm 1.5)}$
LLAMA2 (7B)	10	FT	38.3	37.4	57.9	$44.5_{(\pm 0.9)}$	19.3	13.6	22.3	$18.4_{(\pm 0.2)}$	30.5	21.8	28.0	$26.8_{(\pm 0.7)}$
		TP	57.3	52.4	36.7	$48.8_{(\pm 1.1)}$	85.9	58.6	21.5	$55.4_{(\pm 0.9)}$	63.4	41.2	30.4	$45.0_{(\pm 0.8)}$
		KN	0.0	0.0	0.7	$0.2_{(\pm 0.0)}$	0.8	0.7	4.4	$1.9_{(\pm 0.1)}$	0.0	0.0	0.3	$0.1_{(\pm 0.0)}$
		ROME	41.1	39.6	93.0	$57.9_{(\pm 1.4)}$	38.6	24.9	83.6	$49.1_{(\pm 1.0)}$	33.4	20.3	29.5	$27.7_{(\pm 0.6)}$
		MEMIT	24.3	24.1	51.1	$33.2_{(\pm 0.9)}$	18.6	15.4	62.9	$32.3_{(\pm 0.7)}$	18.4	13.6	10.1	$14.1_{(\pm 0.3)}$
		GRACE	42.5	39.7	92.5	$58.2_{(\pm 1.5)}$	38.1	24.5	82.6	$48.4_{(\pm 0.8)}$	31.4	20.8	29.1	$27.1_{(\pm 0.5)}$
		KE♠	0.5	0.5	1.4	$0.8_{(\pm 0.0)}$	0.0	0.0	0.2	$0.1_{(\pm 0.0)}$	0.1	0.2	1.2	$0.5_{(\pm 0.0)}$
		MEND♠	0.3	0.3	3.3	$1.3_{(\pm 0.0)}$	0.0	0.0	0.1	$0.0_{(\pm 0.0)}$	0.3	0.2	1.6	$0.7_{(\pm 0.0)}$
		MALMEN♠	96.2	88.3	92.6	$92.4_{(\pm 1.8)}$	79.5	45.8	36.2	$53.8_{(\pm 1.1)}$	84.7	47.5	70.9	$67.7_{(\pm 2.3)}$
		DAFNet♠	**97.2**	**92.0**	**93.3**	**94.1**$_{(\pm 1.2)}$	**87.3**	**59.6**	**85.9**	**77.6**$_{(\pm 1.4)}$	**88.8**	**56.4**	**83.2**	**76.1**$_{(\pm 1.9)}$
	100	FT	7.6	7.0	4.1	$6.3_{(\pm 0.2)}$	1.0	0.2	3.6	$1.6_{(\pm 0.0)}$	1.8	0.8	1.0	$1.2_{(\pm 0.1)}$
		TP	46.1	41.2	9.7	$32.3_{(\pm 0.5)}$	70.0	40.8	4.5	$38.4_{(\pm 0.8)}$	44.7	28.9	11.6	$28.4_{(\pm 0.7)}$
		KN	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		ROME	9.6	10.5	22.0	$14.0_{(\pm 0.4)}$	33.6	22.1	68.0	$41.2_{(\pm 1.2)}$	5.9	4.2	5.2	$5.1_{(\pm 0.1)}$
		MEMIT	0.7	0.7	0.9	$0.8_{(\pm 0.0)}$	0.6	0.6	3.7	$1.6_{(\pm 0.0)}$	0.2	0.5	0.3	$0.3_{(\pm 0.0)}$
		GRACE	9.3	8.5	23.0	$13.6_{(\pm 0.5)}$	31.6	21.1	69.0	$40.6_{(\pm 1.0)}$	5.7	4.9	5.1	$5.2_{(\pm 0.2)}$
		KE♠	0.0	0.0	0.1	$0.0_{(\pm 0.0)}$	0.0	0.0	0.9	$0.3_{(\pm 0.0)}$	0.1	0.1	0.0	$0.1_{(\pm 0.0)}$
		MEND♠	0.0	0.0	0.1	$0.0_{(\pm 0.0)}$	0.0	0.0	0.1	$0.0_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		MALMEN♠	54.3	51.8	65.3	$57.1_{(\pm 0.9)}$	48.1	22.4	47.2	$39.2_{(\pm 0.6)}$	41.6	31.7	38.5	$37.3_{(\pm 0.7)}$
		DAFNet♠	**84.7**	**72.0**	**93.6**	**83.5**$_{(\pm 1.4)}$	**72.8**	**41.5**	**76.4**	**63.6**$_{(\pm 1.1)}$	**57.7**	**41.2**	**87.5**	**62.2**$_{(\pm 1.7)}$

Table 1: The overall results of DAFNet and baselines in sequential edits. "# Editing" indicates the length of sequential editing. "Rel.", "Gen." and "Loc." are the reliable, generality and locality editing metrics, respectively. The t-tests demonstrate the improvements of DAFNet are statistically significant with $p < 0.05$ level. The editors marked with ♠ are methods requiring training before editing, which are all augmented with DAFSet in this table.

Backbone	# Editing	Editor	ZSRE				CounterFact				RIPE			
			Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
LLAMA2 (7B)	1	FT	52.8	53.2	92.1	$66.0_{(\pm 0.3)}$	34.1	25.4	49.8	$36.4_{(\pm 0.5)}$	49.2	33.7	71.0	$51.3_{(\pm 0.5)}$
		TP	86.4	84.0	86.4	$85.6_{(\pm 1.4)}$	91.4	68.6	39.0	$66.3_{(\pm 0.3)}$	77.0	55.1	51.3	$61.1_{(\pm 0.5)}$
		KN	20.2	20.8	52.4	$31.1_{(\pm 0.1)}$	12.3	9.2	67.9	$29.8_{(\pm 0.4)}$	21.8	15.4	55.4	$30.9_{(\pm 0.5)}$
		ROME	53.5	51.6	94.0	$66.4_{(\pm 0.7)}$	41.1	21.8	91.8	$51.6_{(\pm 0.8)}$	48.3	27.1	42.5	$39.3_{(\pm 0.9)}$
		MEMIT	49.7	49.4	91.9	$63.7_{(\pm 0.5)}$	45.4	29.3	92.9	$55.9_{(\pm 0.4)}$	58.4	29.6	38.7	$42.2_{(\pm 0.3)}$
		GRACE	52.3	50.8	95.7	$66.3_{(\pm 0.9)}$	44.6	28.5	93.4	$55.5_{(\pm 0.5)}$	56.7	30.6	41.1	$42.8_{(\pm 0.2)}$
		KE♠	12.9	8.6	90.6	$37.4_{(\pm 0.2)}$	8.0	2.9	90.3	$33.7_{(\pm 0.4)}$	9.9	4.4	42.8	$19.0_{(\pm 0.2)}$
		MEND♠	73.8	70.3	66.1	$70.1_{(\pm 0.5)}$	81.1	67.2	77.1	$75.1_{(\pm 0.3)}$	66.4	29.4	29.7	$41.8_{(\pm 0.5)}$
		MALMEN♠	66.4	67.8	43.7	$59.3_{(\pm 0.5)}$	52.4	42.3	36.6	$43.8_{(\pm 0.6)}$	51.5	33.8	20.5	$35.3_{(\pm 0.9)}$
		DAFNet♠	**97.5**	**97.4**	**94.9**	**96.6**$_{(\pm 1.5)}$	**92.0**	**86.7**	**94.3**	**91.0**$_{(\pm 0.8)}$	**97.8**	**66.4**	**72.2**	**78.8**$_{(\pm 0.7)}$

Table 2: The overall results in single-turn editing.

6 Experiments

In this section, we extensively evaluate the proposed method and compare it with strong baselines. Due to space limitation, we describe experimental settings including datasets, baselines and implementation details in Appendix A.

6.1 General Results of Sequential Editing

We first evaluate our DAFNet model over ZSRE (Levy et al., 2017), CounterFact (CF) (Meng et al., 2022) and RIPE datasets (Cohen et al., 2023), which are editing benchmarks. We compare the editing performance with 10, 100 and 1000 edits.[8]

[8]Due to the space limitations, the experimental results of single-turn and 1000 edits in Appendix C.3

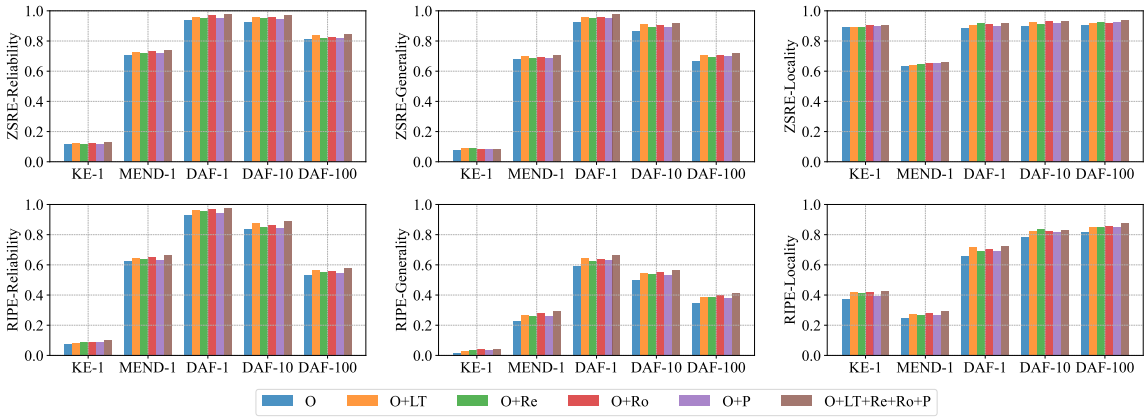

Figure 4: The influence of different properties in DAFSet for editing results. "O", "LT", "Re", "P" and "Ro" indicate original, long-tailness, recency, popularity and robustness data, respectively.

Table 1 shows the overall performance. We can observe that: (1) Extra-module-based and meta-learning methods show poor results in all metrics. We conjecture that meta-learning methods do not consider the sequential fusion modeling of facts. These methods only focus on single-turn editing with one fact. KN (Dai et al., 2022) achieves editing by amplifying the activation of located neurons, and thus multiple facts can lead to excessive amplification of activation. (2) Locate-then-edit methods in sequential editing scenario show normal performance. The reason may be that multiple back-propagation gradient signals can partially capture the inherent connections between the corresponding sequences. (3) DAFNet achieves significant improvement and the vast majority of results are the best, which proves the effectiveness of our method.

We further evaluate our DAFNet in single-turn editing scenario shown in Table 2 with LLAMA2 (7B) backbone. Our model also achieves the competitive performance over the specifically designed baselines for single-turn model editing.

6.2 Detailed Analysis

Influence of Properties in DAFSet. We discuss different properties of DAFSet. We choose typical meta-learning methods such as KE (Cao et al., 2021) and MEND (Mitchell et al., 2022a) as baselines to compare with DAFNet. Other baselines do not need extra data to learn random initialized parameters. We choose the entity-centric ZSRE (Levy et al., 2017) and the relation-centric dataset RIPE (Cohen et al., 2023) as our testing sets.

Based on Figure 4, we can see that (1) For `Reliability` and `Generality`, all training data with each property is beneficial for performance improvement on the testing data. The KE model shows the performance on individual and combined data is poor. We suggest that the training of KE is based on a fixed batch of editing and modeling on the original model and thus the auxiliary network cannot adapt to new models generated by more than one edit. As the number of edits increases, the performance of these two indicators gradually decreases due to the increase in modeling complexity. (2) From the `Locality` results, we can also observe that the model performance can be further improved by adding our data to learn the meta weights. The locality results increase as the number of model edits increases.

Influence of Attention Flows. In intra- and inter-attention flows, we use the auto-regressive attention mapping to all previous token and fact granularities. Considering that different window sizes may have different impacts on semantic interaction, we control different window sizes to test the model during editing. For the inter-attention flow, we need to explore whether the model really pays attention to the previous editing data during the sequence editing process. Specifically, we evaluate our DAFNet model on ZSRE using LLAMA2.

From Figure 5, we explore the window size influence of intra-editing attention flow. We can see that as the number of attention modeling window size increases, the results of these two indicators steadily improve. The reason is that these two editing metrics are highly relevant to the data of the test edited facts and thus the performance can be greatly improved based on the generality of the data. From Figure 6, the multi-layer inter-editing attention flow shows the self-attention importance with 1,000 edits. We can see that the importance

Data	intra	Inter	10			100		
			R	G	L	R	G	L
ZSRE			98.5	95.7	93.5	81.7	68.7	89.9
	✓		99.5	97.2	93.7	88.7	75.1	90.1
		✓	99.2	96.5	93.8	83.5	71.3	90.1
	✓	✓	**99.8**	**98.0**	**94.3**	**90.1**	**76.8**	**90.3**
RIPE			80.7	44.6	45.9	54.7	29.1	45.5
	✓		94.0	56.2	55.8	70.5	37.3	60.5
		✓	82.6	47.8	48.7	58.1	31.7	52.3
	✓	✓	**99.2**	**59.8**	**58.6**	**76.3**	**40.6**	**64.1**

Table 3: Ablation study for GPT-J on ZSRE and RIPE. "R, G, L" means the three test editing metrics.

of semantic modeling for each interactive sentence is the greatest, which is consistent with the self-attention mechanism. With the increase of editing interactions, we can see that the data distribution of self-attention becomes more uniform, and the model can better focus on previously edited data.

6.3 Ablation Study

To assess the impact of intra- and inter-editing attention flows, we conduct an ablation study where we remove each module. This study demonstrates the performance on ZSRE (Yao et al., 2023) and RIPE (Cohen et al., 2023) using GPT-J. From Table 3, when the intra-editing attention flow module is removed, the editing performance of LLMs deteriorates rapidly. The results suffer when the dynamic token modeling process is removed, as it hampers the effective transfer of previously edited semantic memory to the editing of the next token in the facts. When we remove two main modules simultaneously, our model degenerates to only use the autoregressive attention modeling to perform the input gradient signal. Hence, it is easier to capture the semantic connections between sequentially edited data than other meta learning-based methods.

7 Conclusion

In this paper, we propose a dynamic auxiliary fusion model (DAFNet) for sequential editing, including the intra-editing and inter-editing attention flow modules. To obtain better meta weights for updating LLMs' original weights in the auxiliary network, we further propose the DAFSet dataset to enhance the editing ability of LLMs. Experimental results show that our model achieves state-of-the-art results over the strong baselines.

Limitations

For DAFSet, the raw data we currently use only contains factual knowledge from the Wiki. Therefore, in the future, we will consider building a more widely distributed training set on other types of raw data. For DAFNet, due to the need for training, the preparation work before editing is more tedious and time-consuming compared to locate-then-editing based methods. In addition, due to limitations in machine resources, our model has only been tested at a parameter scale of around 10B. If there are more resources, we can experimentally demonstrate our results on a larger parameter scale.

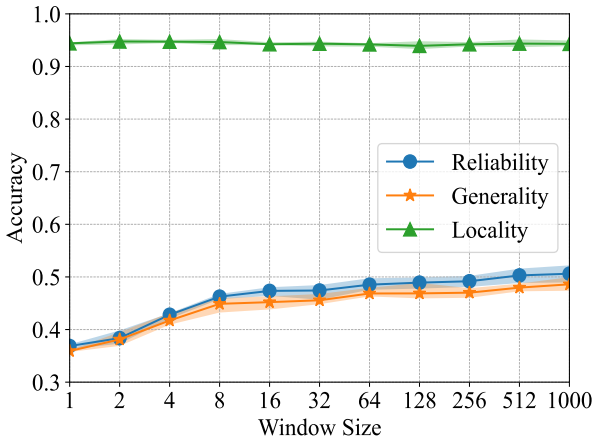

Figure 5: The window size influence of intra-editing attention flow.

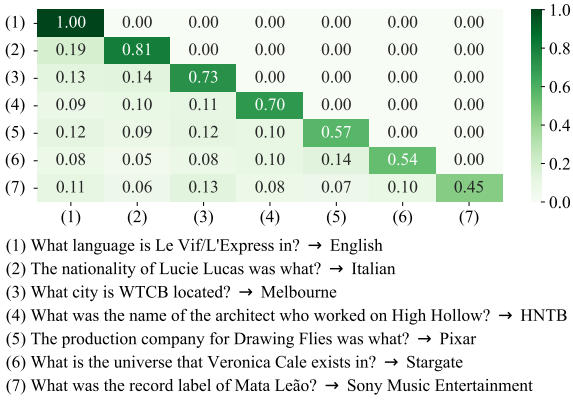

Figure 6: The inter-editing attention flow scores of 1000 sequential edits.

Acknowledgements

We would like to thank anonymous reviewers for their valuable comments. This work is supported by Alibaba Group through Alibaba Research Intern Program.

References

Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2023. Learning from mistakes makes LLM better reasoner. *CoRR*, abs/2310.20689.

Christine Basta, Marta R. Costa-jussà, and Noe Casas. 2021. Extensive study on the underlying gender bias in contextualized word embeddings. *Neural Comput. Appl.*, 33(8):3371–3384.

Steffen Bickel, Michael Brückner, and Tobias Scheffer. 2007. Discriminative learning for differing training and test distributions. In *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 81–88.

Terra Blevins, Hila Gonen, and Luke Zettlemoyer. 2023. Prompting language models for linguistic structure. In *ACL*, pages 6649–6663.

Sofia Blinova, Xinyu Zhou, Martin Jaggi, Carsten Eickhoff, and Seyed Ali Bahrainian. 2023. SIMSUM: document-level text simplification via simultaneous summarization. In *ACL*, pages 9927–9944.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Tiffany Tianhui Cai, Hongseok Namkoong, and Steve Yadlowsky. 2023. Diagnosing model performance under distribution shift. *CoRR*, abs/2303.02011.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *EMNLP*, pages 6491–6506.

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 *ACL*, pages 8493–8502.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. In *EMNLP*, pages 5937–5947.

Ameya Godbole and Robin Jia. 2023. Benchmarking long-tail generalization with likelihood splits. In *EACL*, pages 933–953.

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.

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

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *ICML*, volume 202, pages 15696–15707.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *CoNLL*, pages 333–342.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880.

Huihan Li, Tianyu Gao, Manan Goenka, and Danqi Chen. 2022. Ditch the gold standard: Re-evaluating conversational question answering. In *ACL*, pages 8074–8085.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *EMNLP*, pages 2833–2861.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *NeurIPS*.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *ICLR*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale. In *ICLR*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *ICML*, pages 15817–15831.

Marwan Omar, Soohyeon Choi, Daehun Nyang, and David Mohaisen. 2022. Robust natural language processing: Recent advances, challenges, and future directions. *IEEE Access*, 10:86038–86056.

Konstantinos I. Roumeliotis and Nikolaos D. Tselikas. 2023. Chatgpt and open-ai models: A preliminary review. *Future Internet*, 15(6):192.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *ICML*, pages 31210–31227.

Derek Tam, Anisha Mascarenhas, Shiyue Zhang, Sarah Kwan, Mohit Bansal, and Colin Raffel. 2023. Evaluating the factual consistency of large language models through news summarization. In *ACL*, pages 5220–5255.

Chenmien Tan, Ge Zhang, and Jie Fu. 2023. Massive editing for large language models via meta learning. *CoRR*, abs/2311.04661.

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.

Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, and Huajun Chen. 2023. Easyedit: An easy-to-use knowledge editing framework for large language models. *CoRR*, abs/2308.07269.

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 *EMNLP*, pages 10222–10240.

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? *CoRR*, abs/2305.12740.

Heqi Zheng, Xiao Zhang, Zewen Chi, Heyan Huang, Yan Tan, Tian Lan, Wei Wei, and Xian-Ling Mao. 2022. Cross-lingual phrase retrieval. In *ACL*, pages 4193–4204.

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 *EMNLP*, pages 15686–15702.

A Experimental Settings

A.1 Training Data

Following Yao et al. (2023), we use the ZSRE training data containing 162555 entries, the CF training data containing 10000 entries, and our proposed enhanced dataset DAFSet to train meta-learning based models, including KE (Cao et al., 2021), MEND (Mitchell et al., 2022a), and our DAFNet model.

A.2 Evaluation Data

ZSRE (Levy et al., 2017): It uses BART (Lewis et al., 2020) to answer questions and manually filtering, where each piece of data contains an editing sample, rephrased counterpart and an irrelevant sample corresponding to the reliability, generality and locality indicators, respectively. Inspired by (Yao et al., 2023), we divide it into a training set and a testing set with 162555 and 19009 entries.
CF (Meng et al., 2022): The characteristic is that the facts to be edited are all false facts. Hence, the probability of the model answering correctly before editing is low, thereby increasing the difficulty of editing evaluation. Similar to ZSRE, each data contains an editing sample, rephrased data and an irrelevant sample. Following (Yao et al., 2023), both the training and testing sets contain 10000 entries.
RIPE (Cohen et al., 2023): It finely divides the generality and locality into multiple parts. The generality includes logical generalization, combination I, combination II, and subject aliasing (Cohen et al., 2023). The locality includes forgetfulness and relation specificity. It is also an dataset editing false fact like CF, coupled with its fine-grained evaluation making it a difficult and comprehensive dataset. After pre-processing, a total of 4388 entries are collected.

A.3 Baselines

In this work, besides using fine-tuning as the basic baseline, we mainly compare our DAFNet with three types of editing methods:
Adding Additional Parameters: T-Patcher (Huang et al., 2023) attaches and trains additional neurons in the FFN of the last layer of the model to be edited. GRACE (Hartvigsen et al., 2022) proposes a General Retrieval Adapters for Continuous Editing (GRACE), which maintains a dictionary like structure to construct new mappings for potential representations that need to be modified.
Locate-then-Edit: (1) KN (Dai et al., 2022) uses an integral gradient-based method to locate neurons in FFN, achieving editing by amplifying the activation of the located neurons. (2) ROME (Meng et al., 2022) first uses causal mediation analysis to locate the layer that has the greatest impact on the editing sample. They propose Rank One Model Editing

(ROME) to modify the FFN weight of the located layer. (3) MEMIT (Meng et al., 2023) expands the editing scope to multiple layers based on ROME, which improves editing performance and supports batch editing.

Meta learning-Based: (1) KE (Cao et al., 2021) trains a bidirectional LSTM auxiliary network to predict weight updates of the editing samples. (2) MEND (Mitchell et al., 2022a) trains an MLP to transform the low-rank decomposition of the gradients of the model to be edited with respect to the editing samples, and updates the model with the transformed gradients to achieve editing.

We conduct a comprehensive comparison including the methods with additional or without additional data to train the auxiliary network. Some methods require additional data, while others inherently do not require additional data. Each method can be divided into three categories based on the different editing modes: (1) adding extra parameters modules (2) locate-then-editing and (3) meta-learning based approach. Both meta-learning based methods and locate-then-editing based methods require additional data at different stages such as "ROME, MEMIT, KE, MEND". Our DAFSet dataset aims to enhance meta-learning based editing methods. The meta-learning methods in our main experiment of Table 1 are all trained on data enhanced with DAFSet such as "KE" and "MEND". From Figure 4, we can observe that our DAFNet model also achieves SOTA competitive performance without using additional DAFSet data. If DAFSet data is used, our modeling performance for SME can be further improved.

A.4 Model Settings and Training Details

DAFNet (1) Hyperparameter Settings: We use GPT-J [9] and LLAMA2 [10] as our backbone models to edit and the same hyperparameter settings for the DAFNet auxiliary network. The basic module (including intra-editing attention flow and inter-editing attention flow) of the auxiliary network has 2 layers. We set $d_{down} = 1024$. All self-attention's head numbers and middle dimensions (including K, Q, V, O) are set as 2 and 1024, respectively. Regarding the selection of editing weights, we use settings consistent with MEND and KE: GPT-J and LLAMA2 both use the FFN weights of the last three layers of the model. Different editing matrices with the same shape and a shared DAFNet. Embedding layers are used to remap the representations of editing matrices of different FFN layers inputting into the same DAFNet.

(2) Training Details: As shown in Algorithm 1,

Algorithm 1 Training of DAFNet

Input: Language model to be edited f, initialized DAFNet $\mathcal{M}$, training set $\mathcal{D} = \left\{ \left(x_e^{(i)}, y_e^{(i)}, \left\{ x_{g_j}^{(i)}, y_{g_j}^{(i)} \right\}_{j=1}^{N_g^{(i)}}, \left\{ x_{l_j}^{(i)} \right\}_{j=1}^{N_l^{(i)}} \right) \right\}_{i=1}^{N}$, maximum number of sequential editing modeling T_{max}, EMA loss coefficient α, EMA loss initial value L_{ini}, iteration number to increase sequential modeling number I_{inc}, increment scale γ for increasing sequential editing modeling number, maximum iteration number I_{max}, DAFNet learning rate η.
Output: trained DAFNet $\mathcal{M}$.
 # Current sequential editing modeling number.
$T_{now} = 1$
 # Initialize EMA loss and minimum EMA loss.
$L_{min} = L_{ema} = L_{ini}$
 # Set the iteration number of minimum EMA loss.
$i_{min} = 1$
 # Training iterations for DAFNet $\mathcal{M}$.
for $i \leftarrow 1$ to I_{max} **do**
  # Randomly sample editing number T smaller than current modeling number T_{now}.
  Sample integer T from $\mathbf{Uniform}(1, T_{now})$
  $D \leftarrow$ Sample T data from $\mathcal{D}$
  # Collect editing signals for T editing samples.
  $S_{edit} = []$
  for $(x_e^{(t)}, y_e^{(t)}, \_, \_)$ in D **do**
    # Get editing signal by hook functions.
    $u_t, \delta_t = \mathbf{Hook}(f, (x_e^{(t)}, y_e^{(t)}))$
    $S_{edit}.\mathbf{append}([u_t; \delta_t])$
  end for
  # Input editing signals of the T sequential editing modeling samples and obtain corresponding editing weights.
  $[\Delta W_1, ..., \Delta W_T], [\bar{h}_1, ..., \bar{h}_T], \bar{\beta} = \mathcal{M}(S_{edit}, [])$
  # Compute the fused editing weight of the T editing samples and update the language model f.
  Compute $\Delta \tilde{W}_T$ by formula 15
  $f_T = \Gamma(f, \Delta \tilde{W}_T)$
  # Compute loss using data from D.
  $\mathcal{L}_{total} = \mathcal{L}_{rel}(f_T) + \mathcal{L}_{gen}(f_T) + \mathcal{L}_{loc}(f, f_T)$
  # Update DAFNet $\mathcal{M}$.
  $\mathcal{M} \leftarrow \mathrm{Adam}(\nabla_{\mathcal{M}} \mathcal{L}_{total}, \eta)$
  # Update EMA loss and minimum EMA loss.
  $L_{ema} = (1 - \alpha) L_{ema} + \alpha \mathcal{L}_{total}$
  if $L_{ema} < L_{min}$ **then**
    $L_{min} = L_{ema}$
    $i_{min} = i$
  end if
  # Update current sequential modeling number T_{now}, which will increase exponentially until T_{max}.
  if $i - i_{min} > I_{inc}$ and $T_{now} < T_{max}$ **then**
    $T_{now} = T_{now} + \max(10, \lfloor \gamma T_{now} \rfloor)$
    $L_{min} = L_{ema} = L_{ini}$
    $i_{min} = i$
  end if
end for
return $\mathcal{M}$

[9] https://huggingface.co/docs/transformers/model_doc/gptj

[10] https://huggingface.co/docs/transformers/model_doc/llama2

Algorithm 2 The t_{th} Edit of DAFNet in Sequential Editing Scenario

Input: Language model to be edited f, trained DAFNet $\mathcal{M}$, the t_{th} edit sample $(x_e^{(t)}, y_e^{(t)})$, the editing weight of previous $t-1$ edits $\Delta\tilde{W}_{t-1}$ (zero when $t=1$), the fused fact representations of previous $t-1$ edits $\bar{H}_{t-1} = [\bar{h}_1, ..., \bar{h}_{t-1}]$ (empty list then $t=1$).
Output: Edited model f_t, the updated editing weights $\Delta\tilde{W}_t$, the updated fact representations $\bar{H}_t$.
 # Get editing signal by hook functions.
$u_t, \delta_t = \mathbf{Hook}(f, (x_e^{(t)}, y_e^{(t)}))$
 # Below input current editing signal $[u_t; \delta_t]$, and past fused fact representations for Intra-editing Attention $\bar{H}_{t-1}$. Output the editing weight ΔW_t of the current fact, the fused representations $\bar{h}_t$ of current fact, and the vector $\bar{\beta} \in \mathbb{R}^t$ described in subsection 5.3.
$\Delta W_t, \bar{h}_t, \bar{\beta} = \mathcal{M}\left([u_t; \delta_t], \bar{H}_{t-1}\right)$
 # Update editing weight.
$\Delta\tilde{W}_t = (1-\bar{\beta}_t)\Delta\tilde{W}_{t-1} + \beta_t \Delta W_t$
 # Add updated editing weight to f.
$f_t = \Gamma(f, \Delta\tilde{W}_t)$
 # Append fused representation of current fact into the list.
$\bar{H}_t = [\bar{h}_1, ..., \bar{h}_{t-1}, \bar{h}_t]$
return $f_t, \Delta\tilde{W}_t, \bar{H}_t$

we define the initial sequential editing modeling number $T_{now} = 1$. The moving coefficient of exponential moving average (EMA) loss $\alpha = 0.01$, and set $I_{inc} = 1000$. We set the scale to increase the current sequential editing modeling number T_{now} as 0.25, i.e., $\gamma = 0.25$. The upper limit for the sequential editing modeling number is 1000, i.e., $T_{max} = 1000$. When the sequential editing modeling number reaches the maximum value, we perform an additional 20000 iterations before stopping. We store checkpoints every 1000 iterations and the checkpoint with the lowest loss would be selected for evaluation. The learning rate η is set as 1e-6. The training process takes 7 days on 8 NVIDIA A800 GPUs. These experiments are presented on average with 5 random runs with different random seeds and the same hyper-parameters.

Baseline Models For the baselines, we use the same settings in EasyEdit (Wang et al., 2023) to train and evaluate other editing methods.

B The Editing Algorithm

In order to facilitate readers to better understand the model training and editing process, we have presented the algorithm pipeline of the training and editing in Algorithm 1 and Algorithm 2.

C Additional Experimental Results

C.1 Dataset Construction

The collected dataset samples are shown in Table 4, including the manual templates used to prompt the LLMs. The detailed long-tail dataset construction process is shown as follows:

- We use the LLMs to perform language modeling on each data in the dataset to obtain the log likelihood probability for each token position in the sentence.
- The sentence semantic representation is obtained by multiplying the log likelihood probability of all tokens.
- We select the subject in the sentence that is lower than the threshold as the long tail sentence.

Since the editing samples are all obtained through a single triple transformation, each editing sample only contains one entity and relation such as above samples. Therefore, the semantic of this sentence is usually dominated by entity and corresponding relation. If the log likelihood probability of the sentence is relatively low, it indicates that the semantic of the entity triple is not well memorized in the LLMs and the entity triple is low-frequency sparse knowledge (Godbole and Jia, 2023; Kandpal et al., 2023). Hence, it can be used for capturing the long-tailnesses of those subject entities. Note that the construction process of long-tail data involves concatenating knowledge triples with conjunctions or articles to form a natural language. Their training samples lengths are basically the same. Meanwhile, we multiply by the corresponding sample length to maintain the fairness of the prediction probability as much as possible. Most samples have lengths between 6 and 8, and thus there is no unfairness in comparison after multiplication.

C.2 Computational Resource Analysis

In order to evaluate the computational cost of our model and baselines, we compare the scores of different editors including "No training" and "Training" before editing on all datasets. The average score is performed on the editing numbers of 1, 10 and 100. Specifically, we evaluate the model's overhead on machine resources by comparing Training time, GPU memory and Inference time.

From the Table 5, we can conclude that although our DAFNet model is not the optimal design in

Recency	**subject:** 'syukuro manab' **src:** 'The employer of syukuro manabe is' **tgt:** 'Princeton University' **subject:** 'Giancarlo González' **src:** 'The member of sports team of Giancarlo González is' **tgt:** 'Alajuelense' **subject:** 'polonia bytom' **src:** 'The league of polonia bytom is' **tgt:** 'III liga'
Popularity	**subject:** 'Canon de 75 mle TR ' **src:** 'What year did Canon de 75 mle TR come into use?' **tgt:** '1904' **subject:** 'Walker Pond' **src:** 'What state is Walker Pond located?' **tgt:** 'Maryland' **subject:** 'Golden Bay Air' **src:** 'What airport is Golden Bay Air associated with?' **tgt:** 'Barnstable Municipal Airport'
Long-tailness	**subject:** 'Austin-Healey' **src:** 'The manufacturer of Austin-Healey is' **tgt:** 'British Motors Corp' **subject:** 'Erkin Hadımoğlu' **src:** 'The league of polonia bytom is' **tgt:** 'Painoist' **subject:** 'Isaac Barrow' **src:** 'The place of birth of Isaac Barrow is' **tgt:** 'London'
Robustness	**subject:** 'Sandy High School' **src:** 'What state is Sandy High School located?' **tgt:** 'Florida' **prefix:** 'A student asking a friend for the location of Sandy High School.', 'A parent inquiring about the state where Sandy High School is situated.', 'Sandy High School is a great educational institution.', 'There are some negative comments about Sandy High School.' **subject:** 'Purabirbal' **src:** 'Which state is Purabirbal located?' **tgt:** 'Tiruchir district' **prefix:** 'The sentence is asked in a geography quiz, where participants are being tested on their knowledge of different states.', 'The sentence is part of a conversation between two friends who are discussing a place called Purabirbal.', 'I'm excited to know where Purabirbal is located!', 'I have no idea where Purabirbal is located, and I don't care to find out.' **subject:** 'Coca-Cola Telecommunications' **src:** 'What year was Coca-Cola Telecommunications formed in?' **tgt:** '1926' **prefix:** 'The sentence indicates a query about the company's past and origins.', 'The sentence suggests an interest in the company's history and possibly its involvement in the specific industry.', 'It demonstrates that the company's commitment to expanding its reach and diversifying its offerings.', 'It's hard to believe they even bothered with such a pointless venture.'

Table 4: Samples of the DAFSet dataset.

Training Type	Model	Training Time (Day)	GPU Memory (GB)	Inference Time (s)	Avg.
No Training Before Editing	FT	N/A	**27.80**	1.73	25.09
	TP	N/A	32.30	5.56	55.63
	KN	N/A	31.50	20.41	9.08
	ROME	N/A	36.80	16.10	56.85
	MEMIT	N/A	42.90	31.65	50.40
	GRACE	N/A	35.20	0.16	54.12
Training Before Editing	KE	3	41.10	0.26	10.53
	MEND	1	59.40	**0.08**	19.79
	MALMEN	2	56.20	2.18	52.40
	DAFNet	7	37.20	0.29	**80.42**

Table 5: The overall comparison of computation efficiency.

Backbone	# Editing	Editor	ZSRE				CounterFact				RIPE			
			Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
GPT-J (6B)	1000	FT	4.3	3.0	0.1	$2.5_{(\pm 0.1)}$	12.9	5.1	1.1	$6.4_{(\pm 0.1)}$	3.1	0.9	0.8	$1.6_{(\pm 0.0)}$
		TP	45.7	40.4	10.5	$32.2_{(\pm 0.8)}$	47.3	17.0	1.4	$21.9_{(\pm 0.7)}$	48.1	29.1	15.2	$30.8_{(\pm 0.6)}$
		KN	0.8	0.0	2.2	$1.0_{(\pm 0.0)}$	0.1	0.4	1.0	$0.5_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		ROME	57.2	53.9	29.9	$47.0_{(\pm 1.1)}$	0.2	0.2	0.0	$0.1_{(\pm 0.0)}$	47.5	16.9	13.4	$26.0_{(\pm 0.5)}$
		MEMIT	56.8	54.6	54.9	$55.4_{(\pm 1.3)}$	**82.3**	36.4	30.7	$49.8_{(\pm 1.3)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		GRACE	56.2	51.3	28.4	$45.3_{(\pm 1.2)}$	0.3	0.4	0.1	$0.3_{(\pm 0.1)}$	46.7	16.3	13.8	$25.6_{(\pm 0.7)}$
		KE♠	0.0	0.0	1.1	$0.4_{(\pm 0.0)}$	0.0	0.0	0.1	$0.0_{(\pm 0.0)}$	0.0	0.0	0.2	$0.1_{(\pm 0.0)}$
		MEND♠	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$	0.2	0.1	0.1	$0.1_{(\pm 0.0)}$
		MALMEN♠	43.0	35.1	39.3	$39.1_{(\pm 0.4)}$	15.0	12.4	25.1	$17.5_{(\pm 0.4)}$	31.1	19.1	35.3	$28.5_{(\pm 0.6)}$
		DAFNet♠	**60.0**	**57.6**	**88.0**	$\mathbf{68.5}_{(\pm 1.9)}$	53.1	**38.1**	**82.3**	$\mathbf{57.8}_{(\pm 1.2)}$	**48.3**	**31.3**	**57.3**	$\mathbf{45.6}_{(\pm 1.2)}$
LLAMA2 (7B)	1000	FT	7.9	6.7	4.6	$6.4_{(\pm 0.1)}$	1.5	0.1	1.7	$1.1_{(\pm 0.0)}$	2.7	1.0	2.2	$2.0_{(\pm 0.0)}$
		TP	47.7	44.1	4.4	$32.0_{(\pm 0.6)}$	**64.7**	32.5	11.6	$36.3_{(\pm 0.9)}$	42.3	26.8	9.9	$26.3_{(\pm 0.6)}$
		KN	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		ROME	1.6	1.5	0.6	$1.2_{(\pm 0.0)}$	0.2	0.1	0.1	$0.1_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		MEMIT	0.2	0.2	0.1	$0.2_{(\pm 0.0)}$	0.1	0.1	1.0	$0.4_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		GRACE	1.5	1.6	0.8	$1.3_{(\pm 0.1)}$	0.1	0.2	0.1	$0.1_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		KE♠	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$	0.0	0.0	0.3	$0.1_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		MEND♠	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$	0.0	0.0	0.0	$0.0_{(\pm 0.0)}$
		MALMEN♠	32.0	28.5	28.1	$29.6_{(\pm 0.6)}$	15.8	16.4	22.5	$18.3_{(\pm 0.3)}$	42.3	38.4	38.5	$39.8_{(\pm 0.9)}$
		DAFNet♠	**50.5**	**48.6**	**93.6**	$\mathbf{64.2}_{(\pm 1.3)}$	50.4	**35.8**	**76.9**	$\mathbf{54.4}_{(\pm 1.6)}$	**44.1**	**34.3**	**85.8**	$\mathbf{54.7}_{(\pm 0.9)}$

Table 6: Results with 1000 edits of DAFNet and baselines.

terms of machine resources overhead, our model results have achieved significant improvement under the GPU memory. Our inference time also has strong competitiveness. Since the LLMs are usually trained once and can be reused, our training time is also acceptable. The high memory overhead is mainly due to the need for two different auxiliary networks to model the semantic interaction within and between facts. We can unify the modeling of the fusion process in two scenarios to save memory costs in the future.

C.3 Results of 1000 Sequential Edits

The results of 1000 sequential edits are presented in Table 6. They also show the similar conclusion with general results, which prove the effectiveness of our approach.