Editing Large Language Models

Task Definition: Model Knowledge Editing

- Model knowledge editing modifies pre-trained LLM model outputs for specific inputs to a desired output without affecting other inputs and retraining all parameters.
- Key concepts：
  - Edit Descriptor z_e :([x_e, y_e]) specified input and output for editing

    E.g.: x_e - Who is the president of United States ? y_e - Donald Trump
  - Edit Scope: (只编辑需要修正的内容)

    In-scope Input I(x_e)： Inputs similar to the editing description.

    E.g.: I(x_e)- Who is the president of United States ?

    Out-scope Input O(x_e)： inputs unrelated to the editing description

    E.g.: O(x_e)- Why is the sky blue?

Metric

- **Reliability** : Success rate of editing based on given description Z_e, a **fundamental** requirement for model editing, with accuracy after applying edits.

$$\mathbb{E}_{x'_e,y'_e\sim\{(x_e,y_e)\}}\mathbb{1}\left\{\text{argmax}_y\, p_{\theta_e}\left(y \mid x'_e\right) = y'_e\right\}$$

- **Generalization** : Success rate **within editing scope**, with accuracy after applying edits under input set I(x_e).

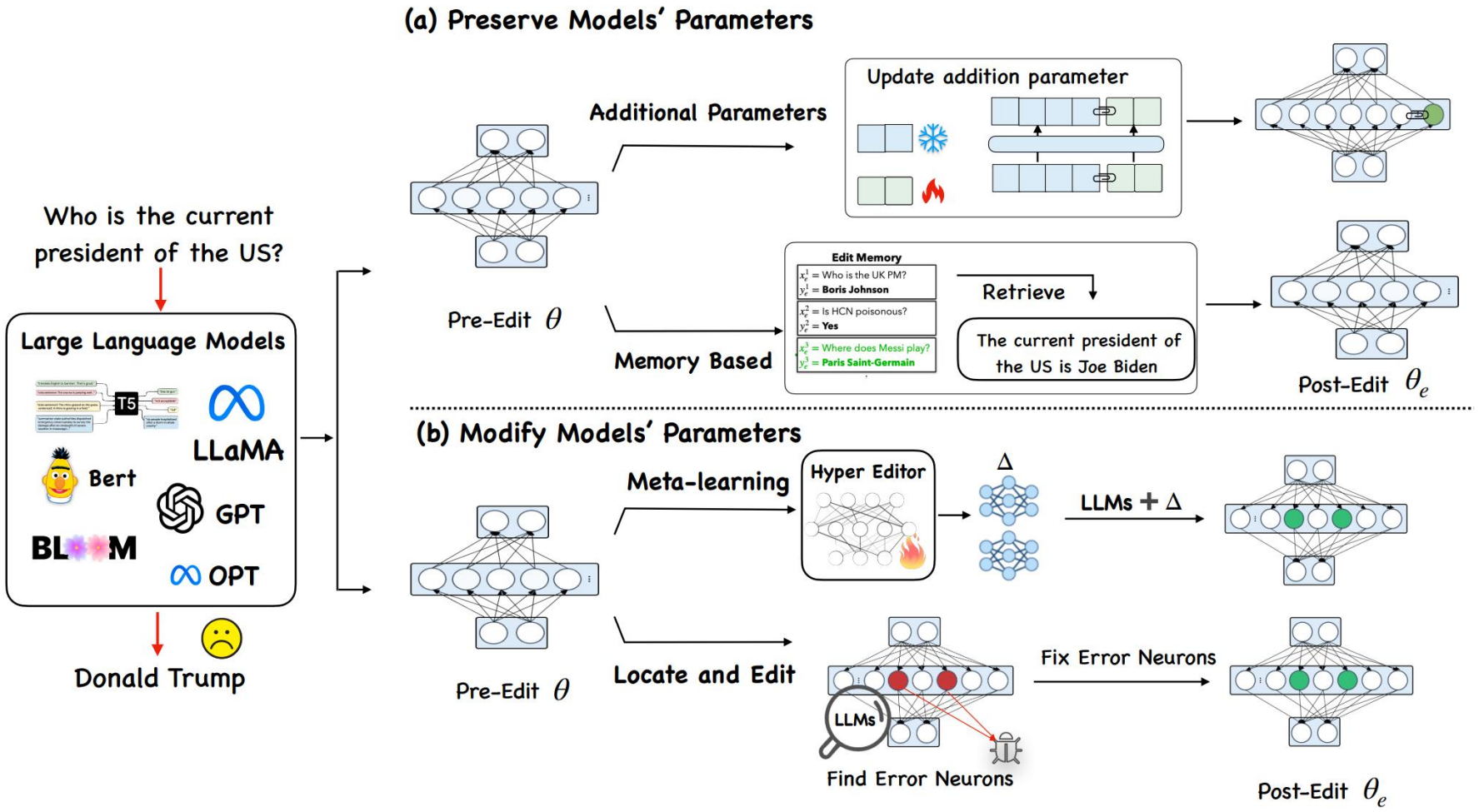
$$\mathbb{E}_{x'_e,y'_e\sim I(x_e,y_e)}\mathbb{1}\left\{\text{argmax}_y\, p_{\theta_e}\left(y \mid x'_e\right) = y'_e\right\}$$

- **Locality** : Model **controls output changes within editing scope**, without affecting external inputs. Evaluates model changes before and after model editing.

$$\mathbb{E}_{x'_e,y'_e\sim O(x_e,y_e)}\mathbb{1}\left\{p_{\theta_e}\left(y \mid x'_e\right) = p_{\theta_o}\left(y \mid x'_e\right)\right\}$$

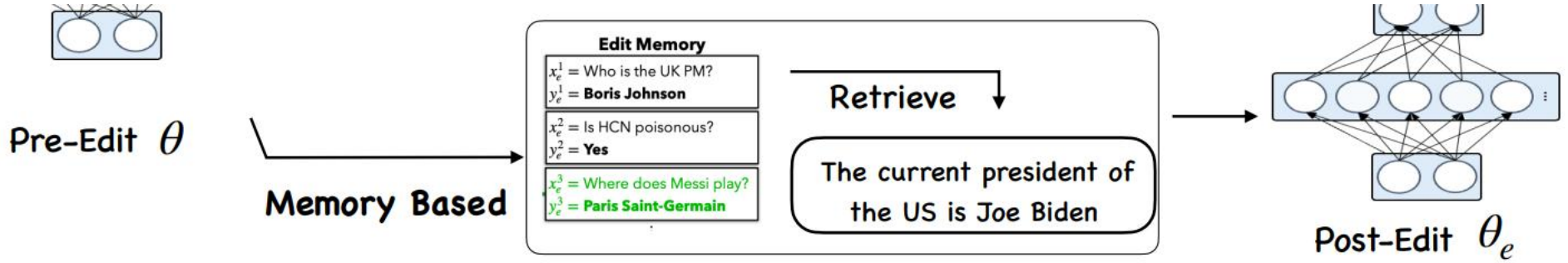
- **Efficiency** : **Time/GPU/memory consumption** for editing.

Overview for current works



An overview of two paradigms of model editing for LLMs.

Memory-based Model



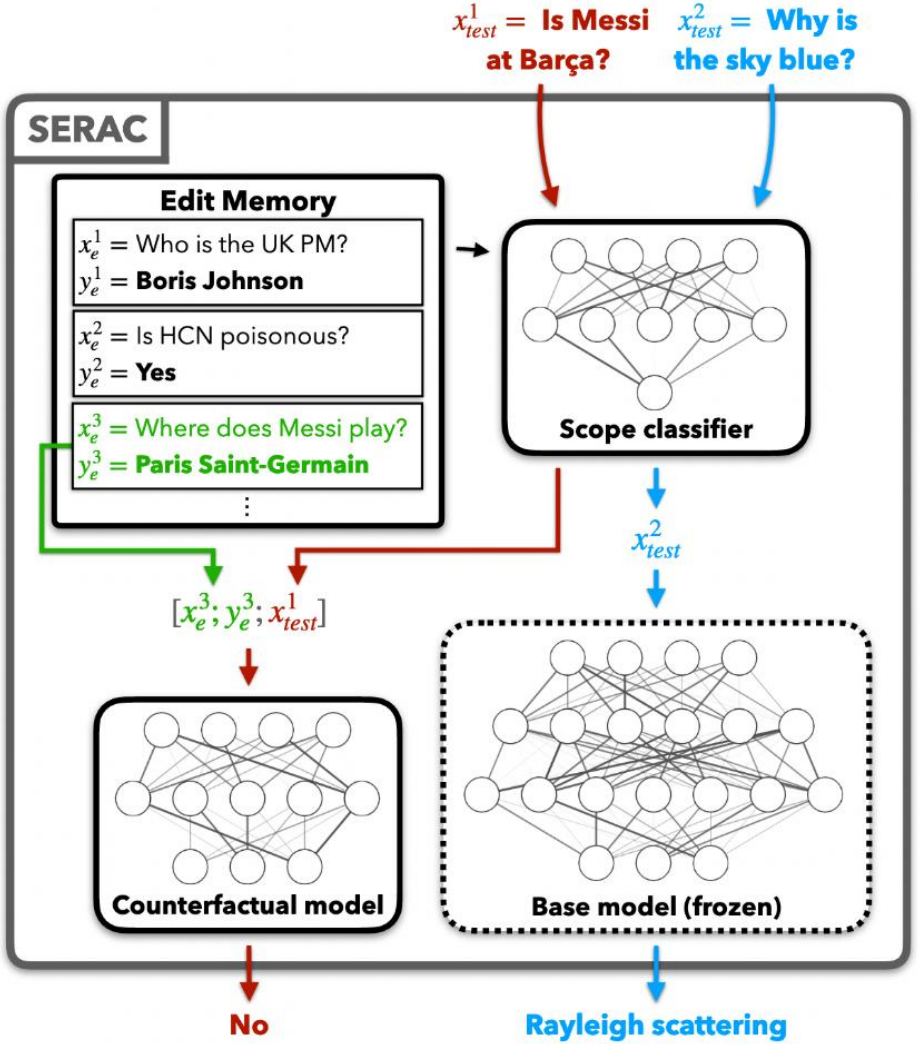
This kind of method stores all edit examples **explicitly in memory** and employs **a retriever** to extract the most relevant edit facts for each new input to guide the model to generate the edited fact.

Memory-based Model

SERAC[1] 可分为3个部分:

- Scope classifier: 分类器，用于对输入进行分类，判断是否需要更新后的知识，然后选择路由到补丁模型还是原始模型。
- Base model: 原始模型，frozen，不再更新参数，通常参数量很大。
- Counterfactual model: 补丁模型，用来储存新的知识。

具体实现很简单，在新的知识数据上，训练一个BERT分类器作为 scope classifier，训练一个T5作为 Counterfactual model 就完成了。



[1] Memory-Based Model Editing at Scale, ICML, 2022

Memory-based Model

3.2. Training SERAC

Similarly to past work (De Cao et al., 2021; Mitchell et al., 2021; Hase et al., 2021), a SERAC editor is trained using the edit dataset $\mathcal{D}_e = \{z_e^i\}$, where in-scope examples (x_{in}^i, y_{in}^i) and negative examples x_{out}^i are sampled from $I(z_e^i; \mathcal{D}_e)$ and $O(z_e^i; \mathcal{D}_e)$, respectively. The scope classifier and counterfactual model are trained completely separately, both with supervised learning as described next.

The **scope classifier** g_ϕ is trained to solve a binary classification problem where the input (z_e, x_{in}) receives label 1 and the input (z_e, x_{out}) receives label 0. The training objective for the scope classifier is the average binary cross entropy loss over the training dataset $\mathcal{D}_e$:

$$\ell(\phi) = - \mathop{\mathbb{E}}_{\substack{z_e \sim \mathcal{D}_e \\ (x_{in}, \cdot) \sim I(z_e; \mathcal{D}_e) \\ x_{out} \sim O(z_e; \mathcal{D}_e)}} \left[\log g_\phi(z_e, x_{in}) + \log(1 - g_\phi(z_e, x_{out})) \right]$$

The **counterfactual model** h_ψ considers an edit z_e and a corresponding example $(x_{in}, y_{in}) \sim I(z_e; \mathcal{D}_e)$, and is trained to minimize the negative log likelihood of y_{in} given z_e and x_{in} on average over $\mathcal{D}_e$:

$$\ell(\psi) = - \mathop{\mathbb{E}}_{\substack{z_e \sim \mathcal{D}_e \\ (x_{in}, y_{in}) \sim I(z_e; \mathcal{D}_e)}} \log p_\psi(y_{in} | z_e, x_{in}) \tag{3}$$

where in a slight abuse of notation $p_\psi(\cdot | z_e, x_{in})$ is the probability distribution over label sequences under the model h_ψ for the inputs (z_e, x_{in}).

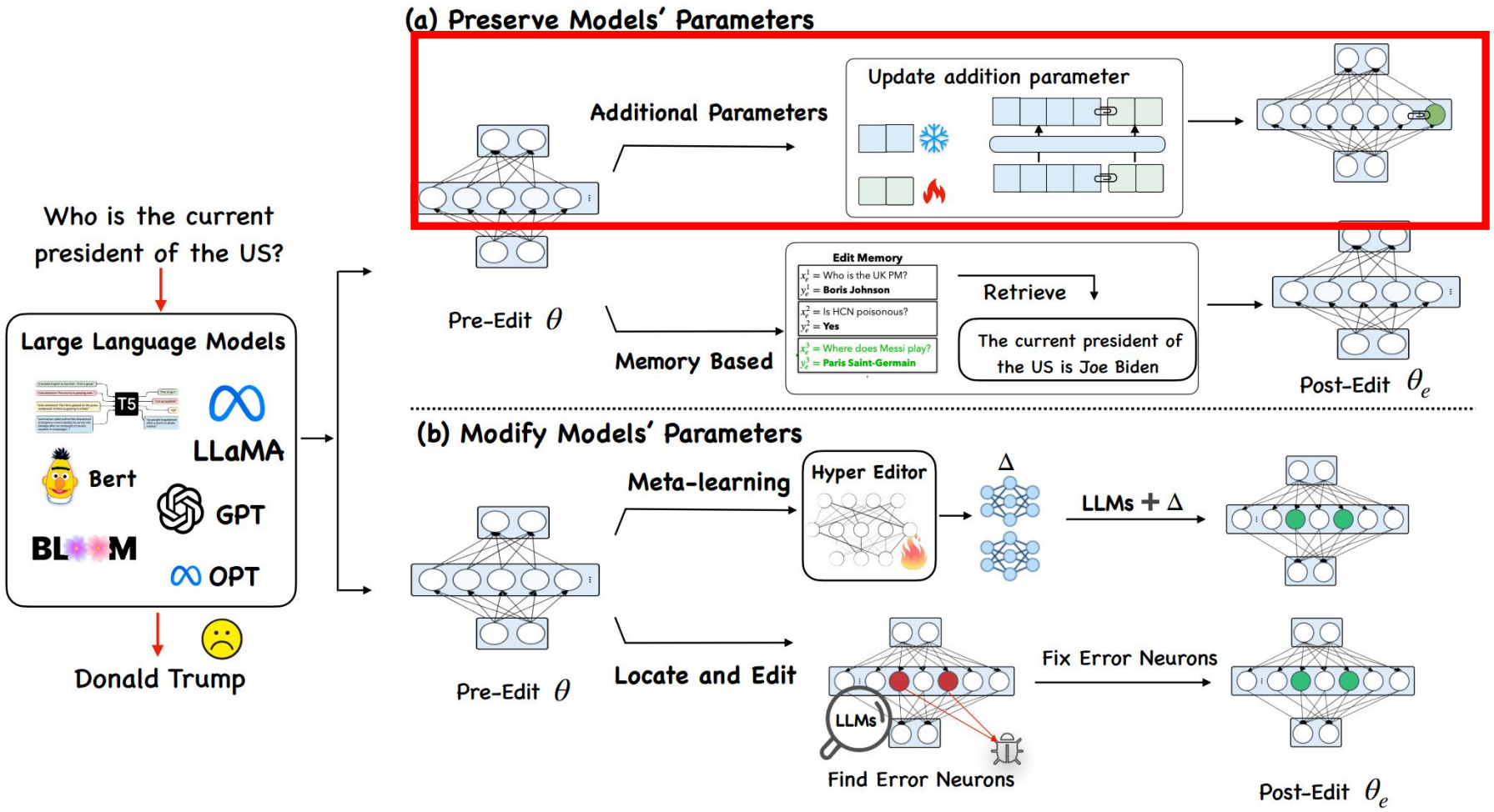
[1] Memory-Based Model Editing at Scale, ICML, 2022

Memory-based Model

- Summary:
  - **[In-context learning]** Instead of retraining an extra model with new facts, the model itself can refine its outpus with the knowledge contexts as prompts.

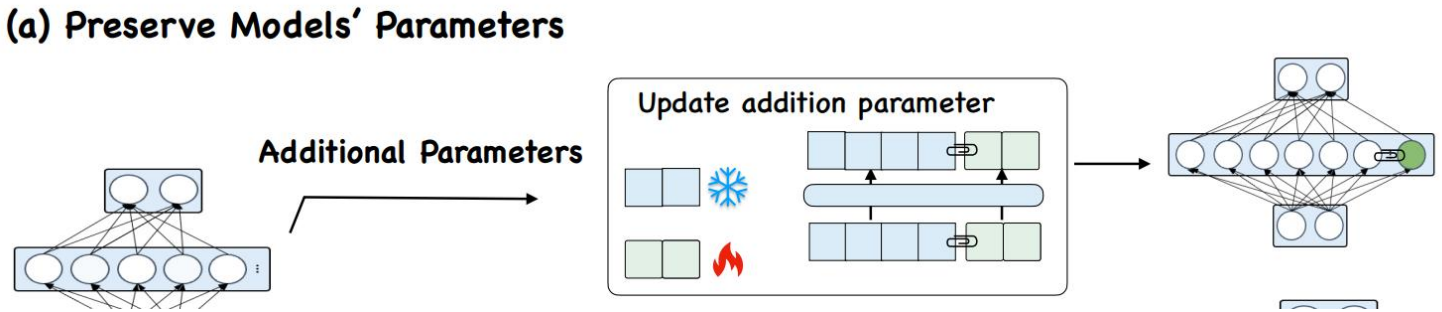
- Other Papers
  - Can We Edit Factual Knowledge by In-Context Learning?

Additional Parameters



An overview of two paradigms of model editing for LLMs.

Additional Parameters



除了显示的存储knowledge，我们是否能够将knowledge存储在模型parameters，
同时用这部分参数去更新模型？

Additional Parameters : **Merge the model's parameters which store the new/old Knowledge.**

Additional Parameters

Question: How do LLMs store Knowledge?

FFN is similar with a Neural Memory Network

1. FFN和MN结构类似[2,3]

Neural Memory Network: 在2015年的End-To-End Memory Networks中，就提出了key-value memory的结构：将 d_m 个需要存储的信息分别映射为 d 维的key向量与value向量，query向量与memory交互, 可以等价为与key, value进行attention操作。对于$K, V \in R^{d_m \times d}$,有

$$\mathrm{MN}(\mathbf{x}) = \mathrm{softmax}(\mathbf{x} \cdot K^\top) \cdot V$$

与FFN对比，对于 $W_1, W_2 \in R^{d_m \times d}$，FFN的形式为

$$FFN(x) = f(x \cdot W_1^\top) \cdot W_2$$

可以看出，FFN几乎与key-value memory相同。

[2] Transformer Feed-Forward Layers Build Predictions by Promoting Concepts in the Vocabulary Space
[3] Knowledge Neurons in Pretrained Transformers, ACL, 2022

Additional Parameters

2. 有实验证明， 知识神经元的激活和它们对应的事实的表达呈正相关的关系[2]。

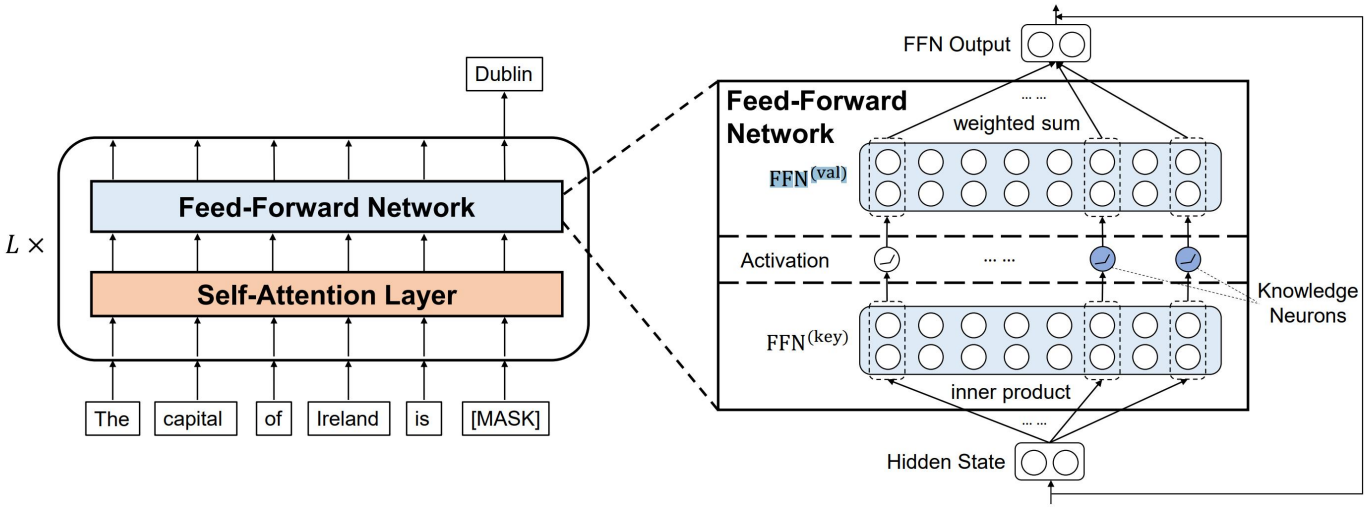


Figure 2: Illustration of how an FFN module in a Transformer block works as a key-value memory. The first linear layer $\mathrm{FFN}^{(\mathrm{key})}$ computes intermediate neurons through inner product. Taking the activation of these neurons as weights, the second linear layer $\mathrm{FFN}^{(\mathrm{val})}$ integrates value vectors through weighted sum. We hypothesize that knowledge neurons in the FFN module are responsible for expressing factual knowledge.

通过研究BERT的完形填空任务，给定一个关系事实，识别表达该事实知识的神经元。

如何找出对应当下知识的信息量大的神经元?

本文的方法是换用多个相似的prompt，
不同prompt之间的能激活的神经元之间的overleap是且仅是 对应当下知识的信息量大的神经元。

Relations	Template #1	Template #2	Template #3
P176 (`manufacturer`)	[X] is produced by [Y]	[X] is a product of [Y]	[Y] and its product [X]
P463 (`member_of`)	[X] is a member of [Y]	[X] belongs to the organization of [Y]	[X] is affiliated with [Y]
P407 (`language_of_work`)	[X] was written in [Y]	The language of [X] is [Y]	[X] was a [Y]-language work

[3] Knowledge Neurons in Pretrained Transformers, ACL, 2022

Additional Parameters

文中[2]发现这种知识神经元的激活和它们对应的事实的表达呈正相关的关系。

就可以对对应的神经元进行增强或抑制。对应来说，增强以后效果会变更好，抑制以后效果会变差。

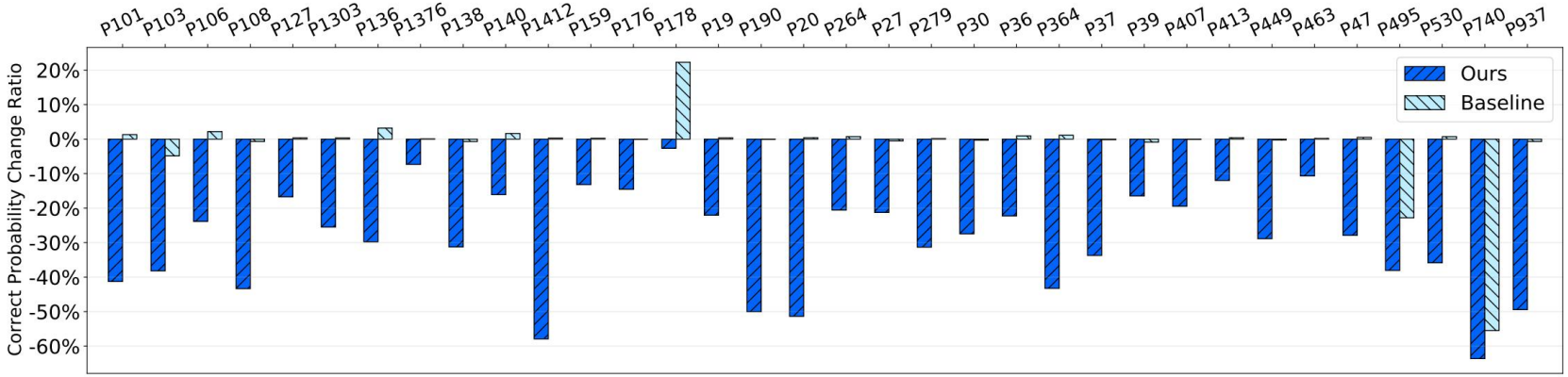


Figure 4: Results of suppressing knowledge neurons for various relations. Suppressing knowledge neurons decreases the correct probability by 29.03% on average. For the baseline, the decreasing ratio is 1.47% on average.

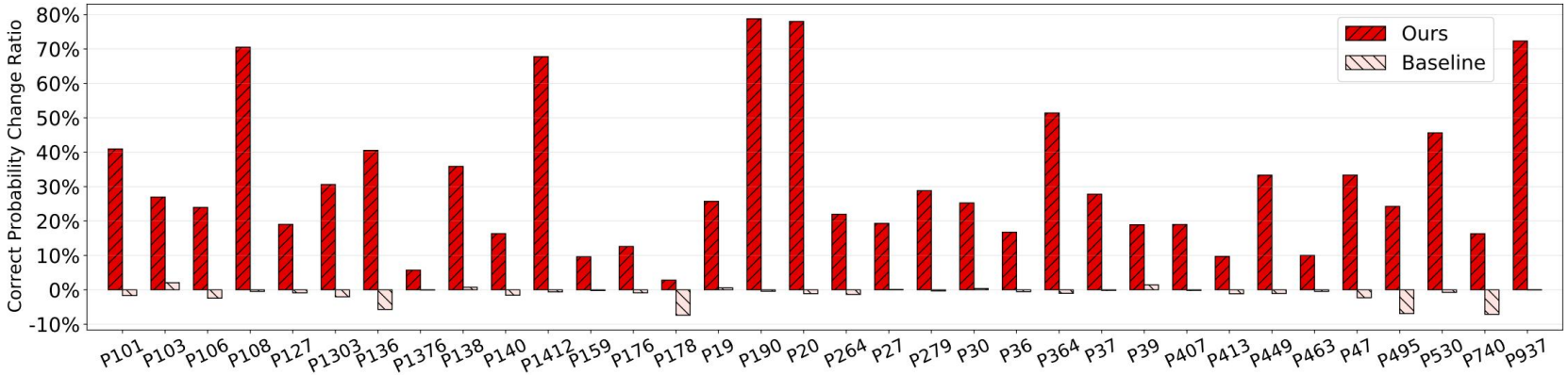


Figure 5: Results of amplifying knowledge neurons for various relations. Amplifying knowledge neurons increases the correct probability by 31.17% on average. For the baseline, the correct probability even decreases by 1.27%.

[3] Knowledge Neurons in Pretrained Transformers, ACL, 2022

Additional Parameters

如果将对应的值直接降低为0，就可以擦除这个知识[2]

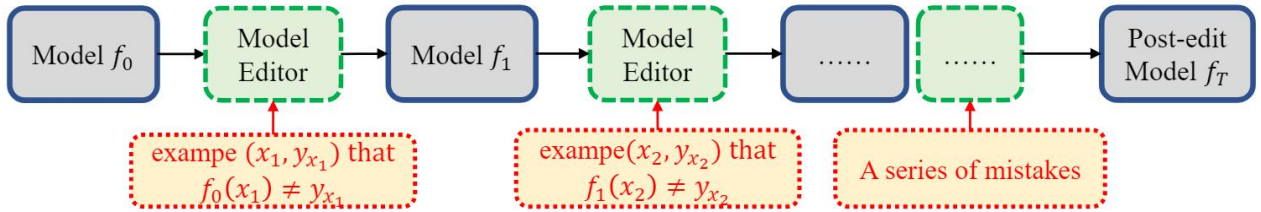
Erased Relations	Perplexity (Erased Relation)		Perplexity (Other Relations)	
	Before Erasing	After Erasing	Before Erasing	After Erasing
P19 (`place_of_birth`)	1450.0	2996.0 (+106.6%)	120.3	121.6 (+1.1%)
P27 (`country_of_citizenship`)	28.0	38.3 (+36.7%)	143.6	149.5 (+4.2%)
P106 (`occupation`)	2279.0	5202.0 (+128.2%)	120.1	125.3 (+4.3%)
P937 (`work_location`)	58.0	140.0 (+141.2%)	138.0	151.9 (+10.1%)

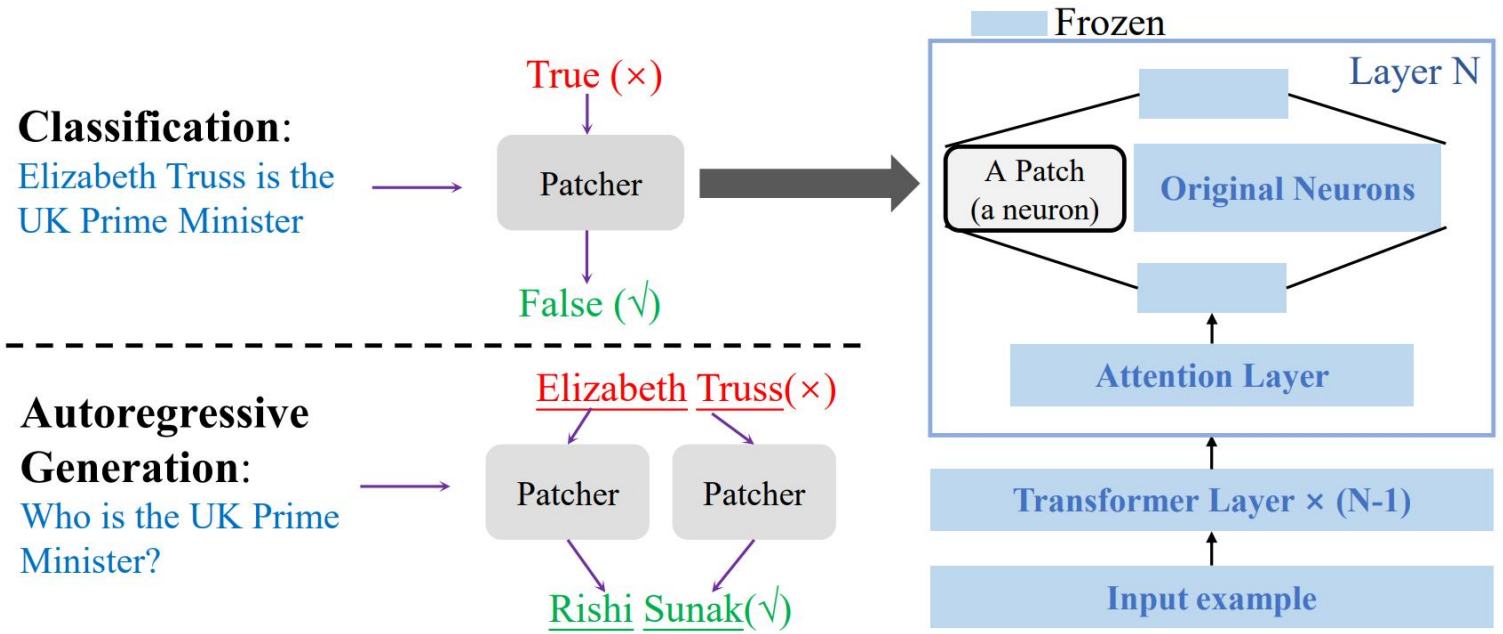
Table 5: Case studies of erasing relations. The influence on knowledge expression is measured by the perplexity change. The knowledge erasing operation significantly affects the erased relation, and has just a moderate influence on the expression of other knowledge.

$$\log(perplexity(S)_) = -\frac{1}{m}\sum_{i=1}^{m} p(w_i|w_1, w_2, \ldots, w_{i-1})$$

[3] Knowledge Neurons in Pretrained Transformers, ACL, 2022

Additional Parameters





考虑到知识存储在FFN中，作者在模型的最后一层中加入一些神经元。对于需要修改的知识，一个token的输出就加一个神经元，两个token的输出就加两个神经元。对于每个需要纠正的知识，都使用对应的增量神经元。

[4]Transformer-Patcher: One Mistake worth One Neuron, ICLR 2023

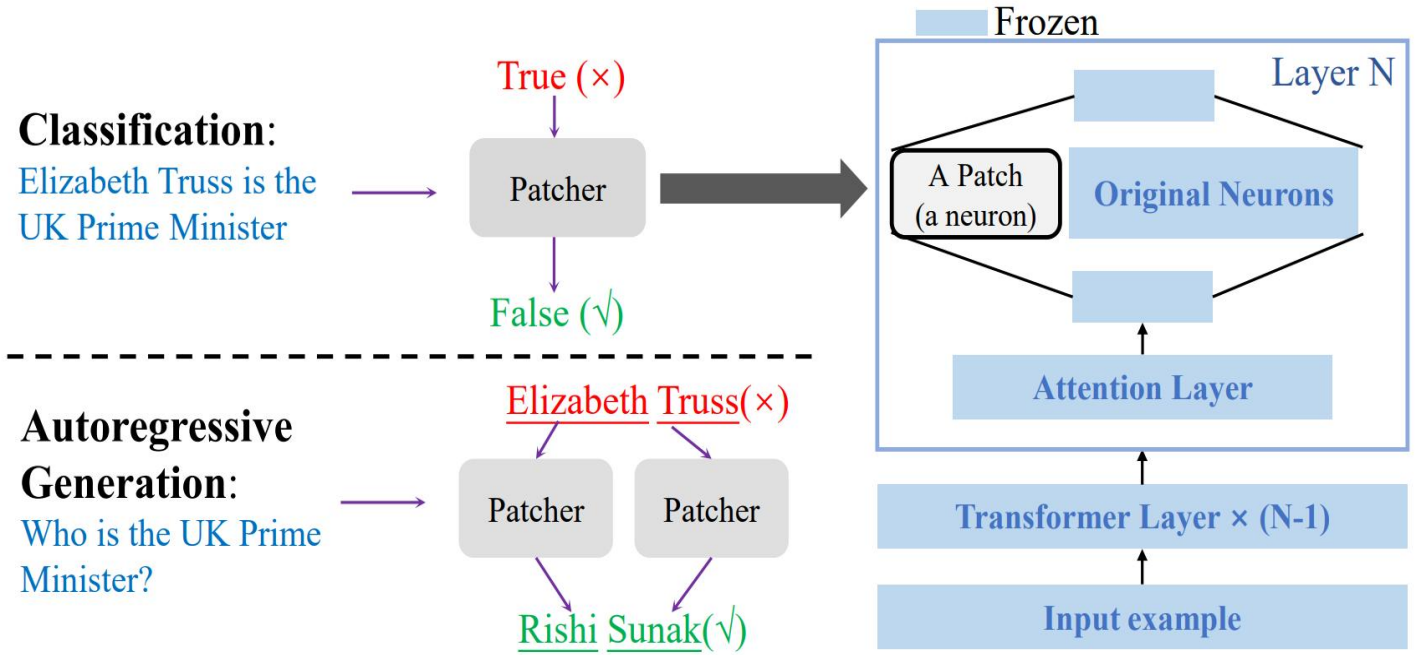
Additional Parameters

Standard FFN (query q)

$$\boldsymbol{a} = \mathrm{Act}(\boldsymbol{q} \cdot \boldsymbol{K} + \boldsymbol{b}_k)$$
$$FFN(\boldsymbol{q}) = \boldsymbol{a} \cdot \boldsymbol{V} + \boldsymbol{b}_v$$

$$[\boldsymbol{a} \quad a_p] = \mathrm{Act}(\boldsymbol{q} \cdot [\boldsymbol{K} \quad \boldsymbol{k}_p] + [\boldsymbol{b}_k \quad b_p])$$
$$FFN_p(\boldsymbol{q}) = [\boldsymbol{a} \quad a_p] \cdot \begin{bmatrix} \boldsymbol{V} \\ \boldsymbol{v}_p \end{bmatrix} + \boldsymbol{b}_v$$

$$FFN_p(\boldsymbol{q}) = FFN(\boldsymbol{q}) + a_p \cdot \boldsymbol{v}_p$$



[4]Transformer-Patcher: One Mistake worth One Neuron, ICLR 2023

Additional Parameters

Property 1 ***Reliability****: the post-edit model should output the desired prediction:*

$$f'(x_e) = y_{x_e} \tag{2}$$

Property 2 ***Generality****: given an edit example* x_e, $\mathbb{E}_{x_e} = \{x_j | y_{x_j} = y_{x_e}\}$ *is defined as the set of its equivalent inputs (e.g. rephrased sentences). Then the post-edit model* f' *should satisfy:*

$$\forall x_j \in \mathbb{E}_{x_e}, f'(x_j) = y_{x_e} \tag{3}$$

Property 3 ***Locality****: the edit should be implemented locally and precisely, which means the post-edit model should remain accurate on the irrelevant examples set* $\mathbb{I}_{x_e} = \mathbb{X} \backslash \mathbb{E}_{x_e}$*:*

$$\forall x_j \in I_{x_e}, f'(x_j) = y_{x_j} \tag{4}$$

In particular, an edit should not disrupt the results of past edits in SME setting, which means:

$$f_t(x_k) = y_{x_k}, \textit{ for } k \textit{ where } f_{k-1}(x_k) \neq y_{x_k} \tag{5}$$

[4]Transformer-Patcher: One Mistake worth One Neuron, ICLR 2023

Additional Parameters

$$FFN_p(\boldsymbol{q}) = FFN(\boldsymbol{q}) + a_p \cdot \boldsymbol{v}_p$$

1. Reliability：

the patch key $\boldsymbol{k}_p$ and patch bias b_p should satisfy:

$$a_p = \mathrm{Act}(\boldsymbol{q}_e \cdot \boldsymbol{k}_p + b_p) \neq 0 \tag{11}$$

When Act is ReLU or GeLU, the above condition can be approximated as follows:

$$\boldsymbol{q}_e \cdot \boldsymbol{k}_p + b_p > 0 \tag{12}$$

To meet the constraint 12, we propose a activation loss l_a to maximize the activation value:

$$l_a = \exp(-\boldsymbol{q}_e \cdot \boldsymbol{k}_p - b_p)) \tag{13}$$

Formally, for an edit example (x_e, y_e), the patched model's output is p_e, l_e is defined as:

$$l_e = L(y_e, p_e) \tag{14}$$

where $L(\cdot)$ is a function of label y_e and model output p_e and depends on the specific task.

2. Locality：

that all queries from irrelevant examples $\boldsymbol{q}_i$ should have a patch activation value less than or equal to a threshold β, i.e., the maximum of them is less than or equal to β:

$$\forall i \in \mathbb{I}_{x_e}, \boldsymbol{q}_i \cdot \boldsymbol{k}_p + b_p \leq \beta \rightarrow \max_i(\boldsymbol{q}_i \cdot \boldsymbol{k}_p + b_p) \leq \beta \tag{15}$$

[4]Transformer-Patcher: One Mistake worth One Neuron, ICLR 2023

Additional Parameters

$$FFN_p(\boldsymbol{q}) = FFN(\boldsymbol{q}) + a_p \cdot \boldsymbol{v}_p$$

1. Reliability：

Thus we propose the memory loss l_m to enforce the constraint 15. To imitate the distribution of queries from irrelevant examples, we randomly retain some queries from previously seen examples as memories. Each query is a d-dimensional vector and we can stack them as a matrix $\boldsymbol{M} \in \mathbb{R}^{d_m \times d}$, where d_m is the number of queries saved. Our proposed memory loss l_m is the sum of two terms. The first term l_{m1} is introduced to make the patch inactivated to all queries in $\boldsymbol{M}$:

$$l_{m1} = S(\boldsymbol{M} \cdot \boldsymbol{k}_p + b_p - \beta; k) \tag{16}$$

where $S(\cdot; k)$ is a function that receives a vector $\boldsymbol{v}$ and outputs a scalar

$$S(\boldsymbol{v}; k) = \text{Avg}[\text{TopK}(\exp(\boldsymbol{v}); k)] \tag{17}$$

In case that l_{m1} can not absolutely ensure the constraint 15, we propose l_{m2} to distance the activation value of $\boldsymbol{q}_e$ and $\boldsymbol{q}_i$. That is, the activation value of the mistaken example is larger than that of the irrelevant examples by a certain margin γ.

$$l_{m2} = S((\boldsymbol{M} - \boldsymbol{q}_e) \cdot \boldsymbol{k}_p + b_p - \gamma; k) \tag{18}$$

To sum up, the loss l_p for training a patch is defined as a weighted sum of the above losses:

$$l_p = l_e + a l_a + m l_m = l_e + a l_a + m(l_{m1} + l_{m2}) \tag{19}$$

[4]Transformer-Patcher: One Mistake worth One Neuron, ICLR 2023

Additional Parameters

- T-Patcher shows good performance for continual learning.

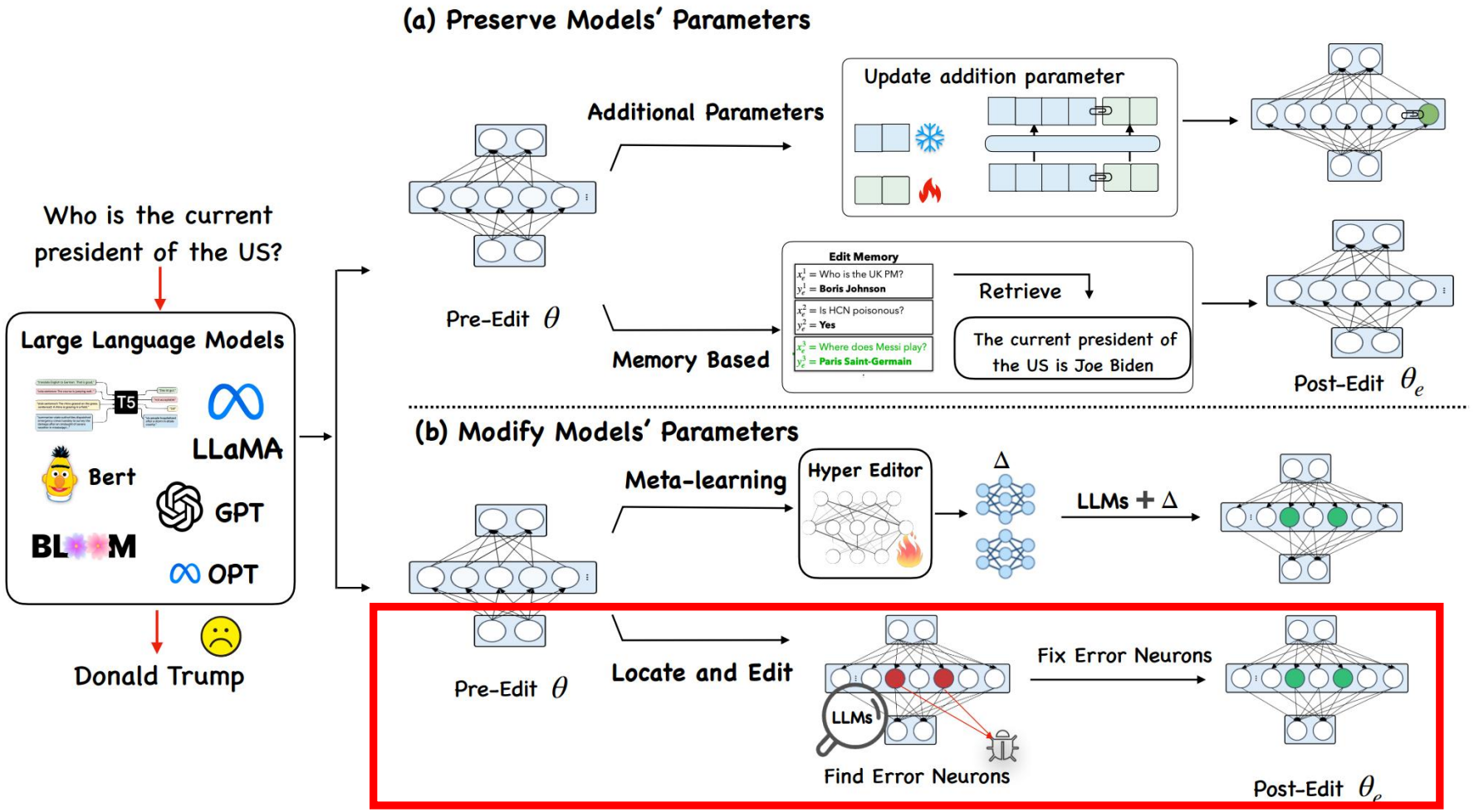
Editor	FEVER Fact-Checking BERT-base (110M)					zsRE Question-Answering BART-base (139M)				
	SR	GR	ER	TrainR	TestR	SR	GR	ER	TrainR	TestR
FT(last)	**1.00**	0.61	0.59	0.893	0.946	**1.00**	0.58	0.30	0.914	0.924
FT(all)	**1.00**	0.74	0.83	0.968	0.994	**1.00**	0.68	0.43	0.865	0.910
FT(last)+KL	**1.00**	0.53	0.45	0.968	0.998	**1.00**	0.57	0.28	0.923	0.933
FT(all)+KL	**1.00**	0.71	0.49	0.998	**1.011**	**1.00**	0.68	0.39	0.889	0.925
MEND†	0.04	0.03	0.06	0.349	0.652	0.41	0.37	0.00	0.000	0.000
KE†	0.14	0.12	0.28	0.486	0.650	0.09	0.08	0.00	0.000	0.000
SERA†	**1.00**	**0.89**	**1.00**	0.904	0.916	**1.00**	**0.90**	0.98	0.906	0.901
T-Patcher	**1.00**	0.82	**1.00**	**0.999**	1.000	1.00*	0.82	**0.99**	**0.997**	**0.996**

- But the computation is slow.

Editor	COUNTERFACT	ZsRE
FT-L	35.94s	58.86s
SERAC	5.31s	6.51s
CaliNet	1.88s	1.93s
T-Patcher	1864.74s	1825.15s
KE	2.20s	2.21s
MEND	**0.51s**	**0.52s**
KN	225.43s	173.57s
ROME	147.2s	183.0s
MEMIT	143.2s	145.6s

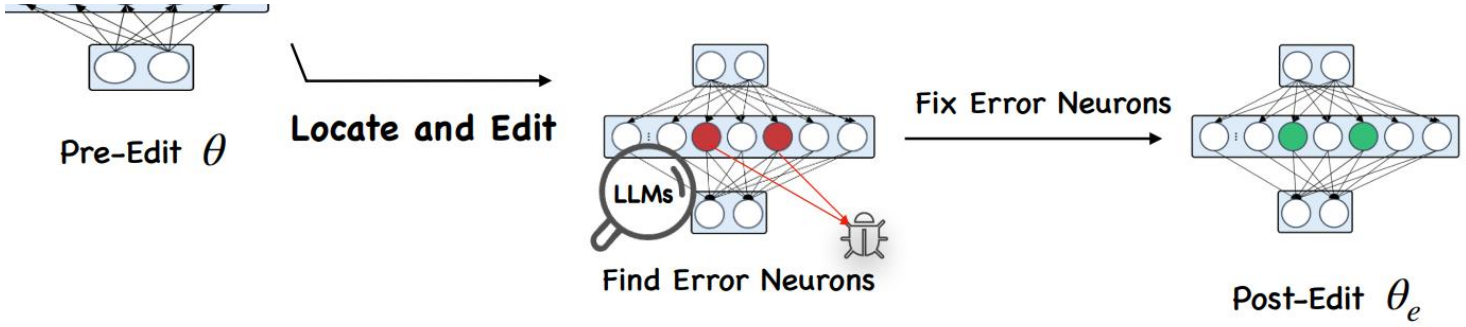
[4]Transformer-Patcher: One Mistake worth One Neuron, ICLR 2023

Locate-Then-Edit



An overview of two paradigms of model editing for LLMs.

Locate-Then-Edit



This paradigm initially **identifies parameters corresponding** to specific knowledge and modifies them through direct **updates** to the target parameters.

The effectiveness of location is still controversial !

[3] Locating and Editing Factual Associations in GPT. NeurIPS 2022

Locate-Then-Edit

This paradigm initially identifies parameters corresponding to specific knowledge and modifies them through direct updates to the target parameters.

那么就有两个比较关键的问题:
How and where a model stores its factual associations?
=> FNN

Then we can
=> Locate
=> Fix mistakes.
=> Help us to understand huge opaque neural networks.

[5] Locating and Editing Factual Associations in GPT. NeurIPS 2022

Locate-Then-Edit

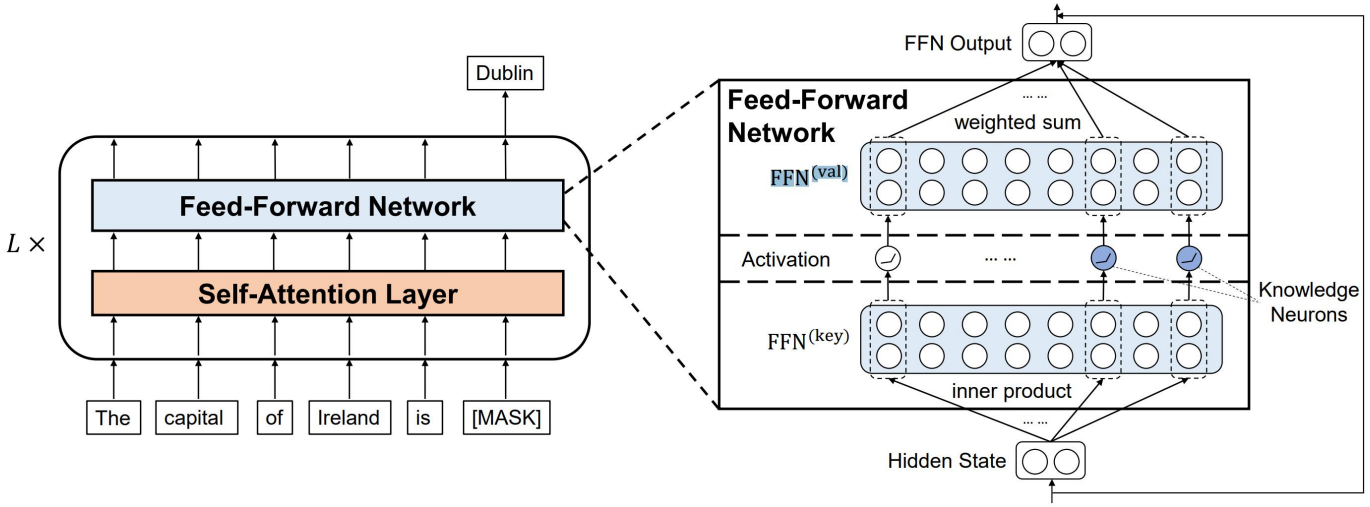


Figure 2: Illustration of how an FFN module in a Transformer block works as a key-value memory. The first linear layer $\mathrm{FFN}^{(\mathrm{key})}$ computes intermediate neurons through inner product. Taking the activation of these neurons as weights, the second linear layer $\mathrm{FFN}^{(\mathrm{val})}$ integrates value vectors through weighted sum. We hypothesize that knowledge neurons in the FFN module are responsible for expressing factual knowledge.

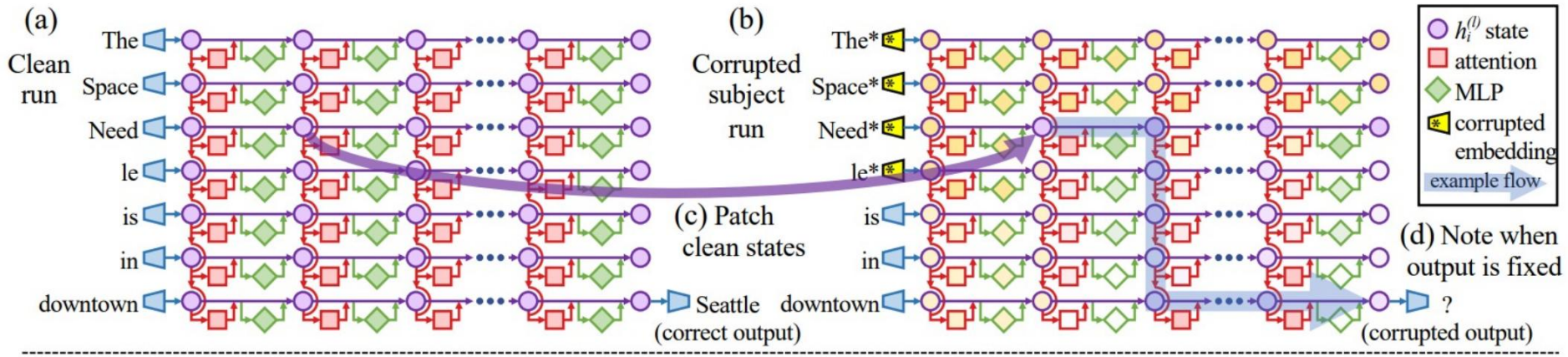
[2] Knowledge Neurons in Pretrained Transformers, ACL, 2022

$$h_i^{(l)} = h_i^{(l-1)} + a_i^{(l)} + m_i^{(l)}$$

$$a_i^{(l)} = \text{attn}^{(l)}\left(h_1^{(l-1)}, h_2^{(l-1)}, \ldots, h_i^{(l-1)}\right)$$

$$m_i^{(l)} = W_{proj}^{(l)}\,\sigma\left(W_{fc}^{(l)}\,\gamma\left(a_i^{(l)} + h_i^{(l-1)}\right)\right).$$

- **Clean run**
- **Corrupted run** $h_i^{(0)} := h_i^{(0)} + \epsilon$
- **corrupted-with-restoration run**



[5] Locating and Editing Factual Associations in GPT. NeurIPS 2022

Locate-Then-Edit

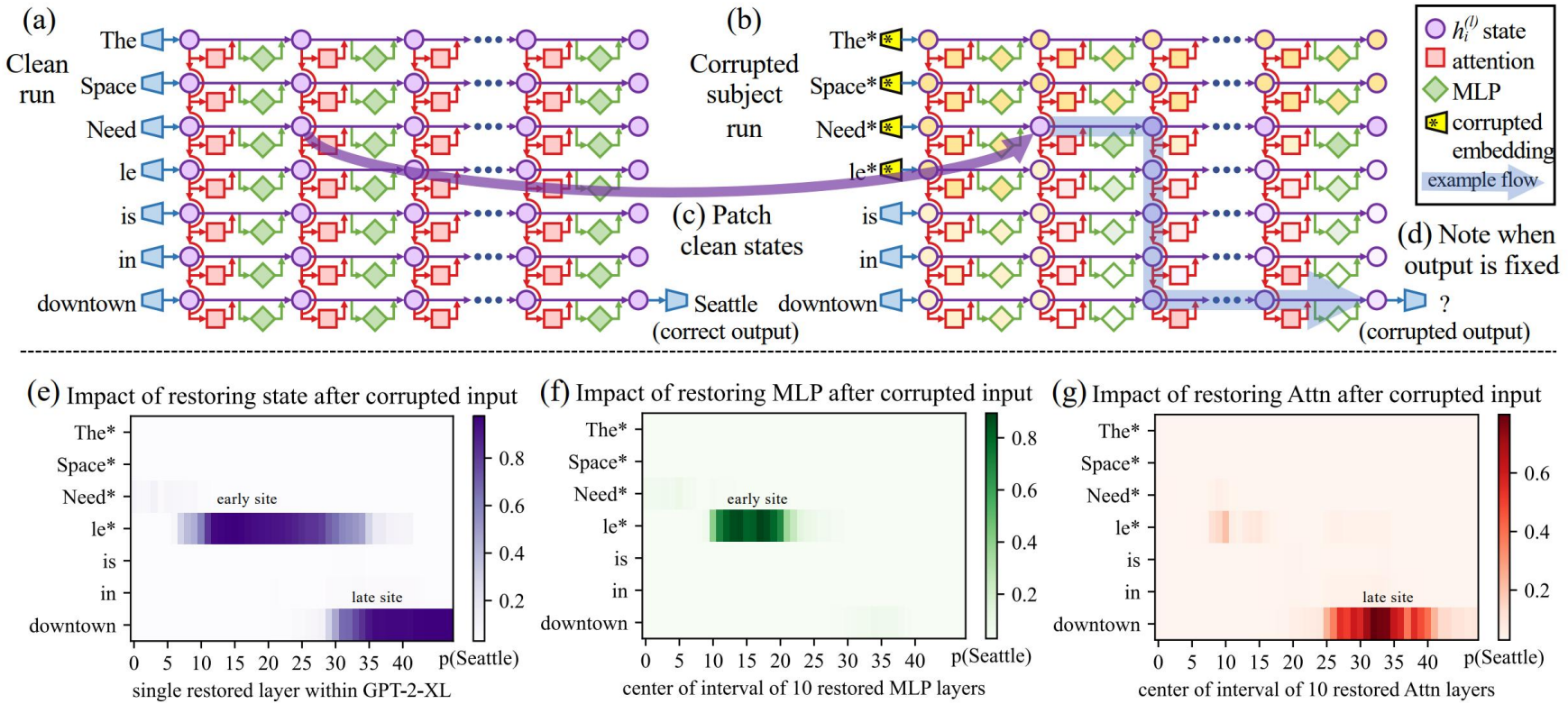


Figure 1: **Causal Traces** compute the causal effect of neuron activations by running the network twice: (a) once normally, and (b) once where we corrupt the subject token and then (c) restore selected internal activations to their clean value. (d) Some sets of activations cause the output to return to the original prediction; the light blue path shows an example of information flow. The causal impact on output probability is mapped for the effect of (e) each hidden state on the prediction, (f) only MLP activations, and (g) only attention activations.

[5] Locating and Editing Factual Associations in GPT. NeurIPS 2022

Locate-Then-Edit

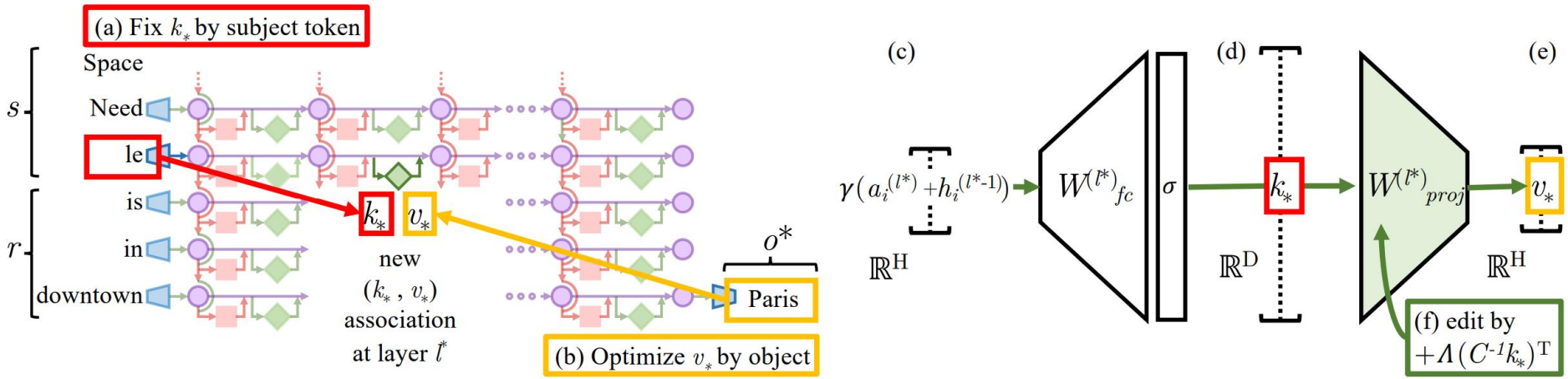


Figure 4: **Editing one MLP layer with ROME**. To associate *Space Needle* with *Paris*, the ROME method inserts a new (k_*, v_*) association into layer l^*, where (a) key k_* is determined by the subject and (b) value v_* is optimized to select the object. (c) Hidden state at layer l^* and token i is expanded to produce (d) the key vector k_* for the subject. (e) To write new value vector v_* into the layer, (f) we calculate a rank-one update $\Lambda(C^{-1}k_*)^T$ to cause $\hat{W}^{(l)}_{proj}k_* = v_*$ while minimizing interference with other memories stored in the layer.

$$k_* = \frac{1}{N}\sum_{j=1}^{N} k(x_j + s), \text{ where } k(x) = \sigma\left(W_{fc}^{(l^*)}\,\gamma(a_{[x],i}^{(l^*)} + h_{[x],i}^{(l^*-1)})\right).$$

[5] Locating and Editing Factual Associations in GPT. NeurIPS 2022

Locate-Then-Edit

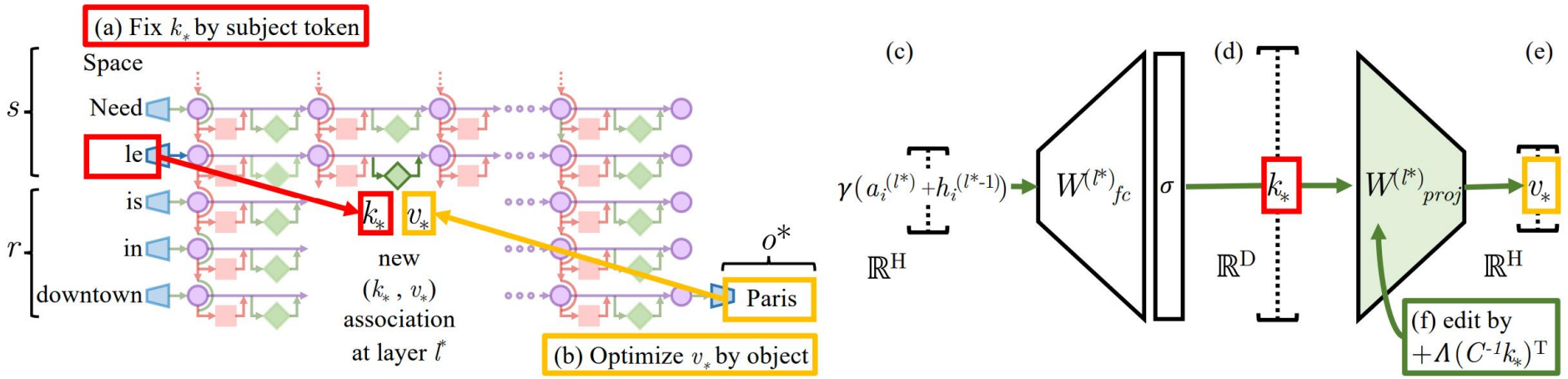


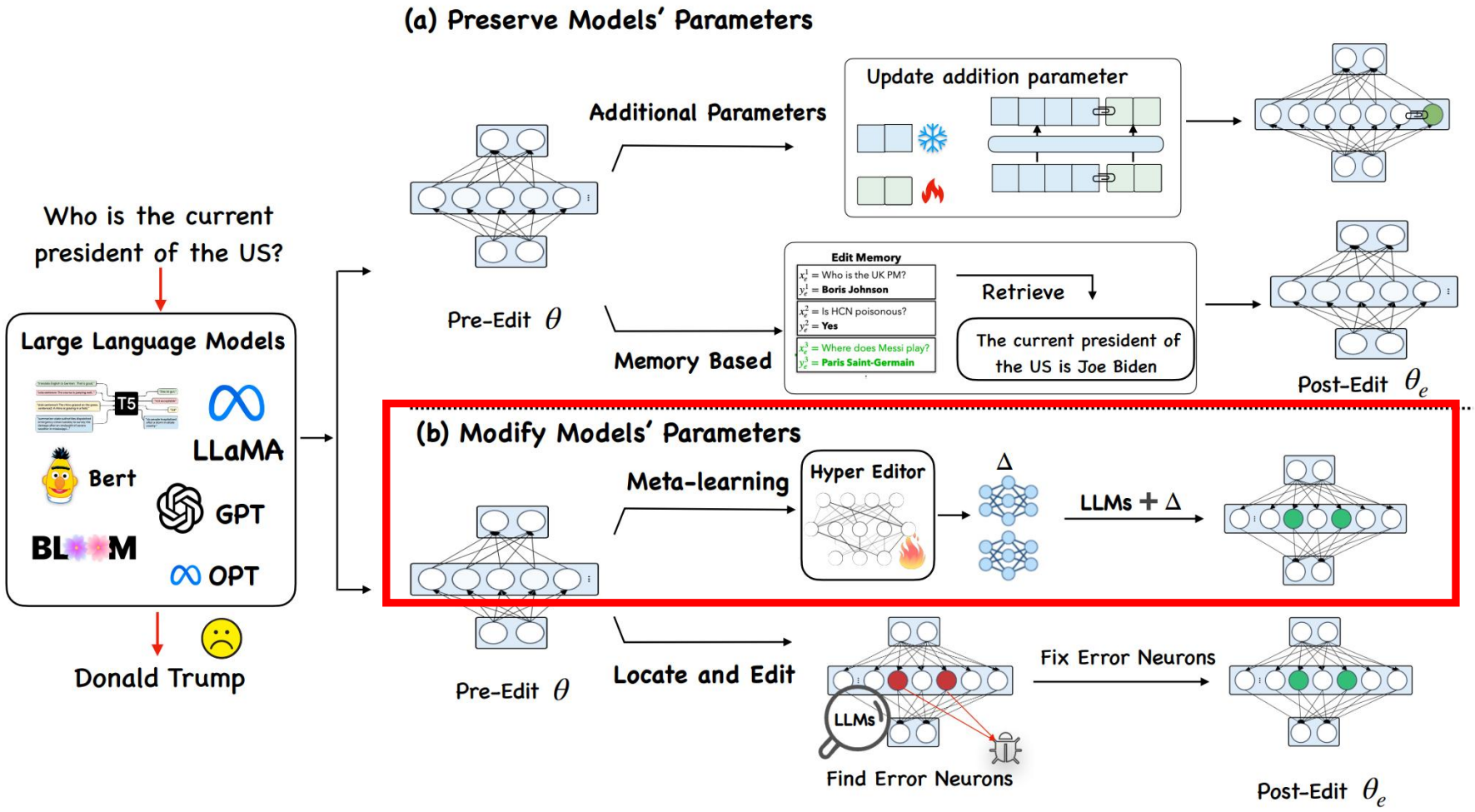
Figure 4: **Editing one MLP layer with ROME**. To associate *Space Needle* with *Paris*, the ROME method inserts a new (k_*, v_*) association into layer l^*, where (a) key k_* is determined by the subject and (b) value v_* is optimized to select the object. (c) Hidden state at layer l^* and token i is expanded to produce (d) the key vector k_* for the subject. (e) To write new value vector v_* into the layer, (f) we calculate a rank-one update $\Lambda(C^{-1}k_*)^T$ to cause $\hat{W}^{(l)}_{proj}k_* = v_*$ while minimizing interference with other memories stored in the layer.

步骤 2：选择 v_* 来回忆事实。接下来，我们希望选择一些向量值 v^*，将新关系 (r, o^*) 编码为 s 的属性。我们设置 $v_* = \arg\min_z L(z)$，其中目标 $L(z)$ 为：

$$\frac{1}{N}\sum_{j=1}^{N} \underbrace{-\log \mathbb{P}_{G(m_i^{(l^*)}:=z)}\left[o^* \mid x_j + p\right]}_{\text{(a) Maximizing } o^* \text{ probability}} + \underbrace{D_{\mathrm{KL}}\left(\mathbb{P}_{G(m_{i'}^{(l^*)}:=z)}\left[x \mid p'\right] \middle\| \mathbb{P}_G\left[x \mid p'\right]\right)}_{\text{(b) Controlling essence drift}}.$$

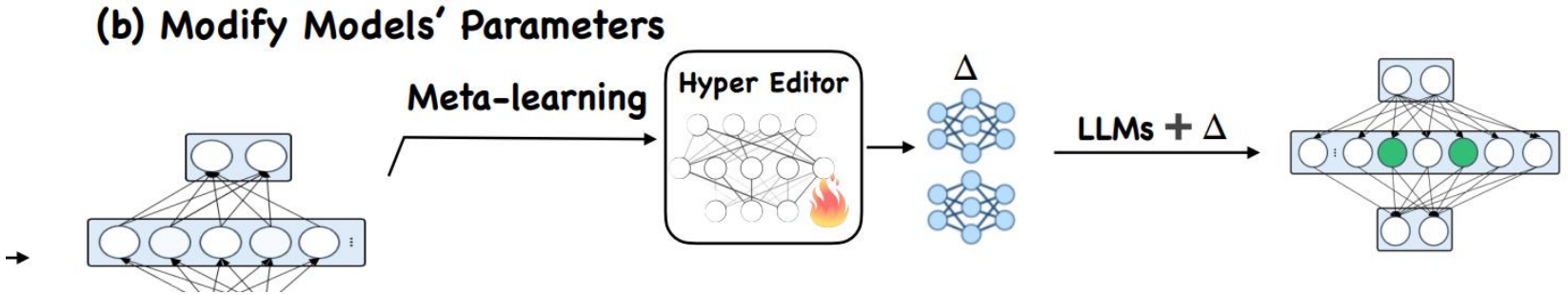
[5] Locating and Editing Factual Associations in GPT. NeurIPS 2022

Meta-learning



An overview of two paradigms of model editing for LLMs.

Meta-learning



Meta-learning methods employ **a <u>hyper-network</u>** to learn the **<u>necessary</u>** Δ for editing the LLMs.

[6] Fast Model Editing at Scale. ICLR 2022. PPT:https://iclr.cc/media/iclr-2022/Slides/6846.pdf

Meta-learning

Hyper Editor设计的难点:

- Single examples: If presented with only a single problematic input and new desired output, fine-tuning approaches tend to overfit;
- Huge amount of computation: other editing algorithms are either computationally infeasible or simply ineffective when applied to very large models.

这里简单介绍一下ICLR22的工作MENO，MENO设计了一个Edited model，Edited model可以对gradient进行低秩分解，来近似估计standard fine-tuning得到的gradient

[4] Fast Model Editing at Scale. ICLR 2022. PPT:https://iclr.cc/media/iclr-2022/Slides/6846.pdf

Meta-learning

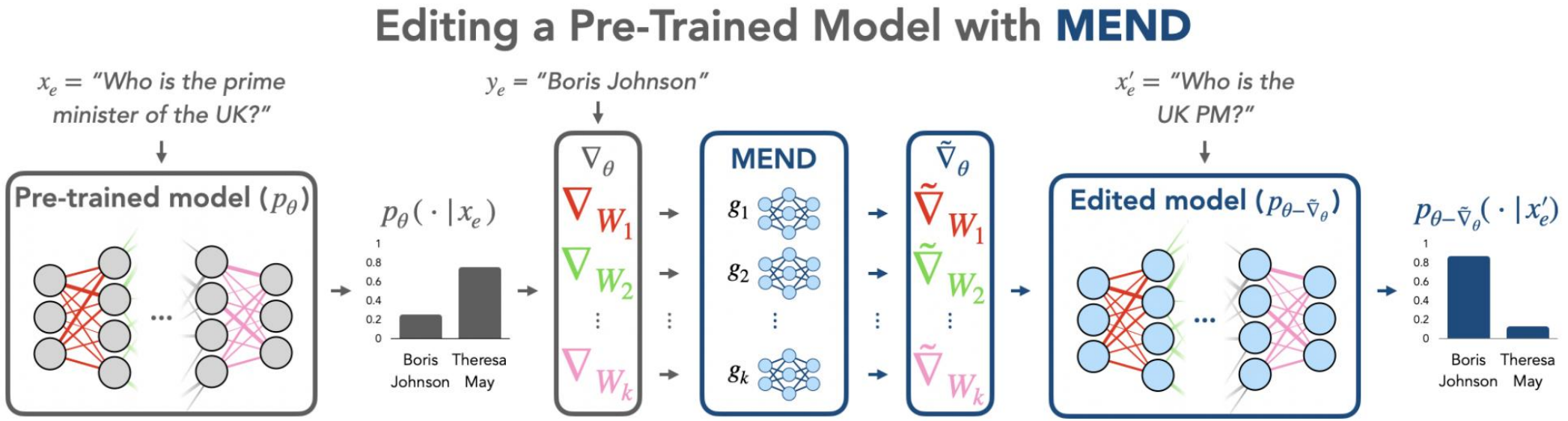


Figure 1: The proposed algorithm MEND enables editability by training a collection of MLPs to modify model gradients to produce *local* model edits that do not damage model performance on unrelated inputs. MEND is efficient to train and apply edits, even for very large models, as shown in Section 5.1.

在这里我们有两个问题需要考虑:

1. 如何更新Edited model？
2. Edited model是如何估计standard fine-tuning得到的gradient？

[4] Fast Model Editing at Scale. ICLR 2022. PPT:https://iclr.cc/media/iclr-2022/Slides/6846.pdf

Meta-learning

和之前讲的一样，Edit的对象应该是Edit example以及 example neighborhood；同时unrelated example保持不变

Requirement: an "edit dataset" $D_{edit} = \{ (z_{edit}, x_{out}, x_{in}, y_{in}) \}$

Perform edit

Promote generalization inside eq. neighborhood

Prevent changes outside eq. neighborhood

z_{edit} = "Who is the UK PM? Boris Johnson"

x_{out} = "What team does Messi play for?"

x_{in} = "The prime minister of the UK is currently who?"

y_{in} = "Boris Johnson"

[6] Fast Model Editing at Scale. ICLR 2022. PPT:https://iclr.cc/media/iclr-2022/Slides/6846.pdf

Meta-learning

Requirement: an "edit dataset" $\mathbf{D_{edit}} = \{ (\mathbf{z_{edit}}, \mathbf{x_{out}}, \mathbf{x_{in}}, \mathbf{y_{in}}) \}$

$\mathbf{z_{edit}}$: Perform edit

$\mathbf{x_{out}}$: Prevent changes outside eq. neighborhood

$\mathbf{x_{in}}$, $\mathbf{y_{in}}$: Promote generalization inside eq. neighborhood

$\mathbf{z_{edit}}$ = "Who is the UK PM? Boris Johnson"

$\mathbf{x_{out}}$ = "What team does Messi play for?"

$\mathbf{x_{in}}$ = "The prime minister of the UK is currently who?"

$\mathbf{y_{in}}$ = "Boris Johnson"

Inner loop

(run the editor)

$$\theta_e = \text{Edit}_\phi(\theta, \mathbf{z_{edit}})$$

↑ Editor parameters

Learn to transform gradient

Outer loop

(check if edit worked)

$$L_{edit} = -\log p_{\theta_e}(\mathbf{y_{in}} \mid \mathbf{x_{in}})$$

Change predictions of neighborhood!

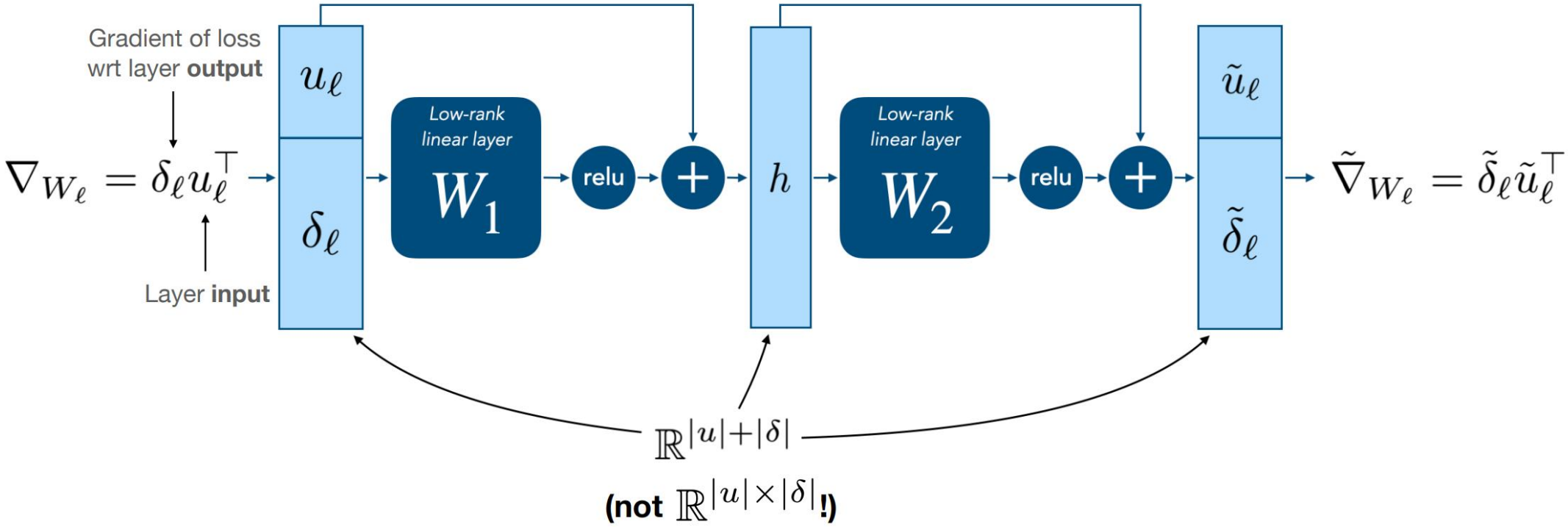
$$L_{local} = \text{KL}\left(p_\theta(\cdot \mid \mathbf{x_{out}}) \,\|\, p_{\theta_e}(\cdot \mid \mathbf{x_{out}}) \right)$$

Keep predictions the same everywhere else!

[6] Fast Model Editing at Scale. ICLR 2022. PPT:https://iclr.cc/media/iclr-2022/Slides/6846.pdf

Meta-learning

Object： an efficient, expressive gradient transform



[6] Fast Model Editing at Scale. ICLR 2022. PPT:https://iclr.cc/media/iclr-2022/Slides/6846.pdf

Meta-learning

Total algorithm

Algorithm 1 MEND Training

Input: Pre-trained $p_{\theta_{\mathcal{W}}}$, weights to make editable $\mathcal{W}$, editor params ϕ_0, edit dataset D^{tr}_{edit}, edit-locality tradeoff c_{edit}
for $t \in 1, 2, \ldots$ **do**
  Sample $x_{\text{e}}, y_{\text{e}}, x'_{\text{e}}, y'_{\text{e}}, x_{\text{loc}} \sim D^{tr}_{edit}$
  $\tilde{\mathcal{W}} \leftarrow \text{EDIT}(\theta_{\mathcal{W}}, \mathcal{W}, \phi_{t-1}, x_{\text{e}}, y_{\text{e}})$
  $L_{\text{e}} \leftarrow -\log p_{\theta_{\tilde{\mathcal{W}}}}(y'_{\text{e}}|x'_{\text{e}})$
  $L_{\text{loc}} \leftarrow \text{KL}(p_{\theta_{\mathcal{W}}}(\cdot|x_{\text{loc}}) \| p_{\theta_{\tilde{\mathcal{W}}}}(\cdot|x_{\text{loc}}))$
  $L(\phi_{t-1}) \leftarrow c_{\text{edit}} L_{\text{e}} + L_{\text{loc}}$
  $\phi_t \leftarrow \text{Adam}\,(\phi_{t-1}, \nabla_{\phi} L(\phi_{t-1}))$

Algorithm 2 MEND Edit Procedure

procedure $\text{EDIT}(\theta, \mathcal{W}, \phi, x_{\text{e}}, y_{\text{e}})$
  $\hat{p} \leftarrow p_{\theta_{\mathcal{W}}}(y_{\text{e}}|x_{\text{e}})$, **caching** input u_ℓ to $W_\ell \in \mathcal{W}$
  $L(\theta, \mathcal{W}) \leftarrow -\log \hat{p}$ ▷ Compute NLL
  for $W_\ell \in \mathcal{W}$ **do**
    $\delta_{\ell+1} \leftarrow \nabla_{W_\ell u_\ell + b_\ell} l_e(x_{\text{e}}, y_{\text{e}})$ ▷ Grad wrt output
    $\tilde{u}_\ell, \tilde{\delta}_{\ell+1} \leftarrow g_{\phi_\ell}(u_\ell, \delta_{\ell+1})$ ▷ Pseudo-acts/deltas
    $\tilde{W}_\ell \leftarrow W_\ell - \tilde{\delta}_{\ell+1} \tilde{u}_\ell^\top$ ▷ Layer ℓ model edit
  $\tilde{\mathcal{W}} \leftarrow \{\tilde{W}_1, \ldots, \tilde{W}_k\}$
  return $\tilde{\mathcal{W}}$ ▷ Return edited weights

[6] Fast Model Editing at Scale. ICLR 2022. PPT:https://iclr.cc/media/iclr-2022/Slides/6846.pdf

Can We Edit Multimodal Large Language Models?

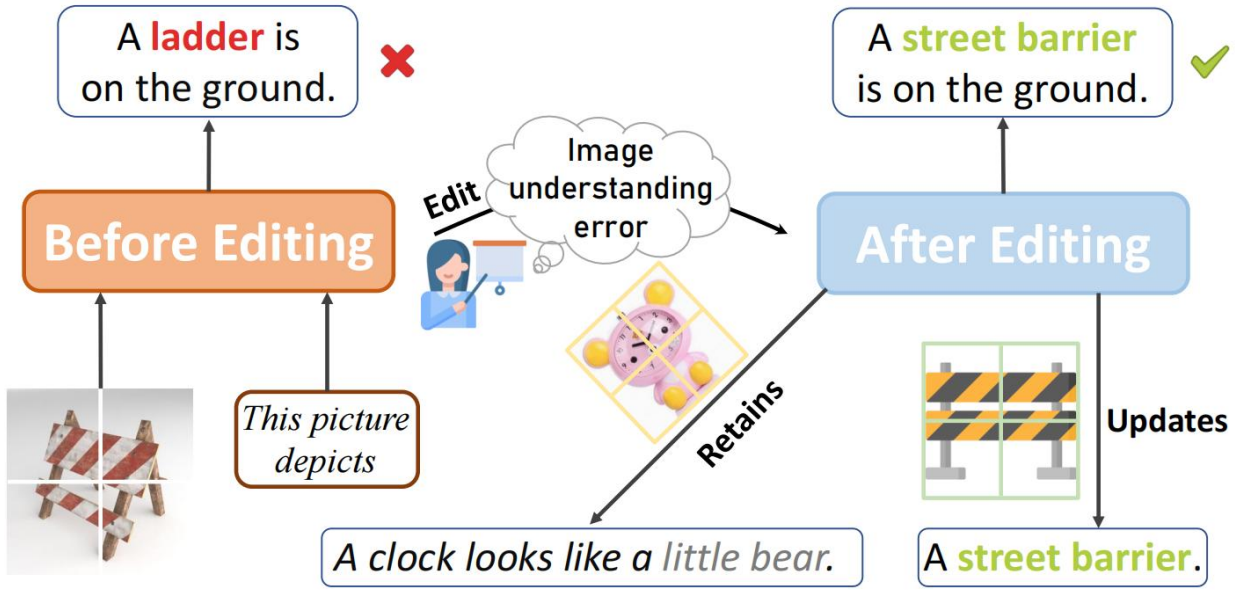


Figure 1: Overview of the **multimodal model editing** task. The editing target is to update the model's understanding of the edited input (e.g., image or text), while ensuring its interpretation of unrelated inputs remains as consistent as possible.

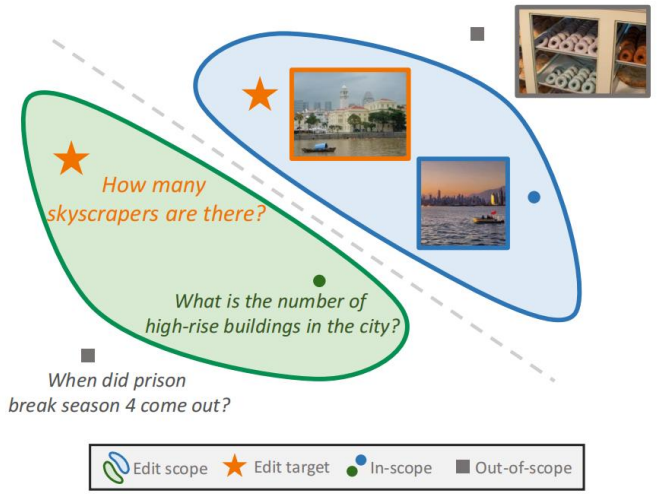


Figure 3: Taking the text modality as an example, **Edit target** and its generalization pertain to *in-scope*, which involves querying the quantity of skyscrapers in a given image, while the *out-of-scope* refers to inquiries about the publication date. In-scope inputs require editing, whereas out-of-scope inputs remain unchanged.

[7] Can We Edit Multimodal Large Language Models? EMNLP 2023.

Can We Edit Multimodal Large Language Models?

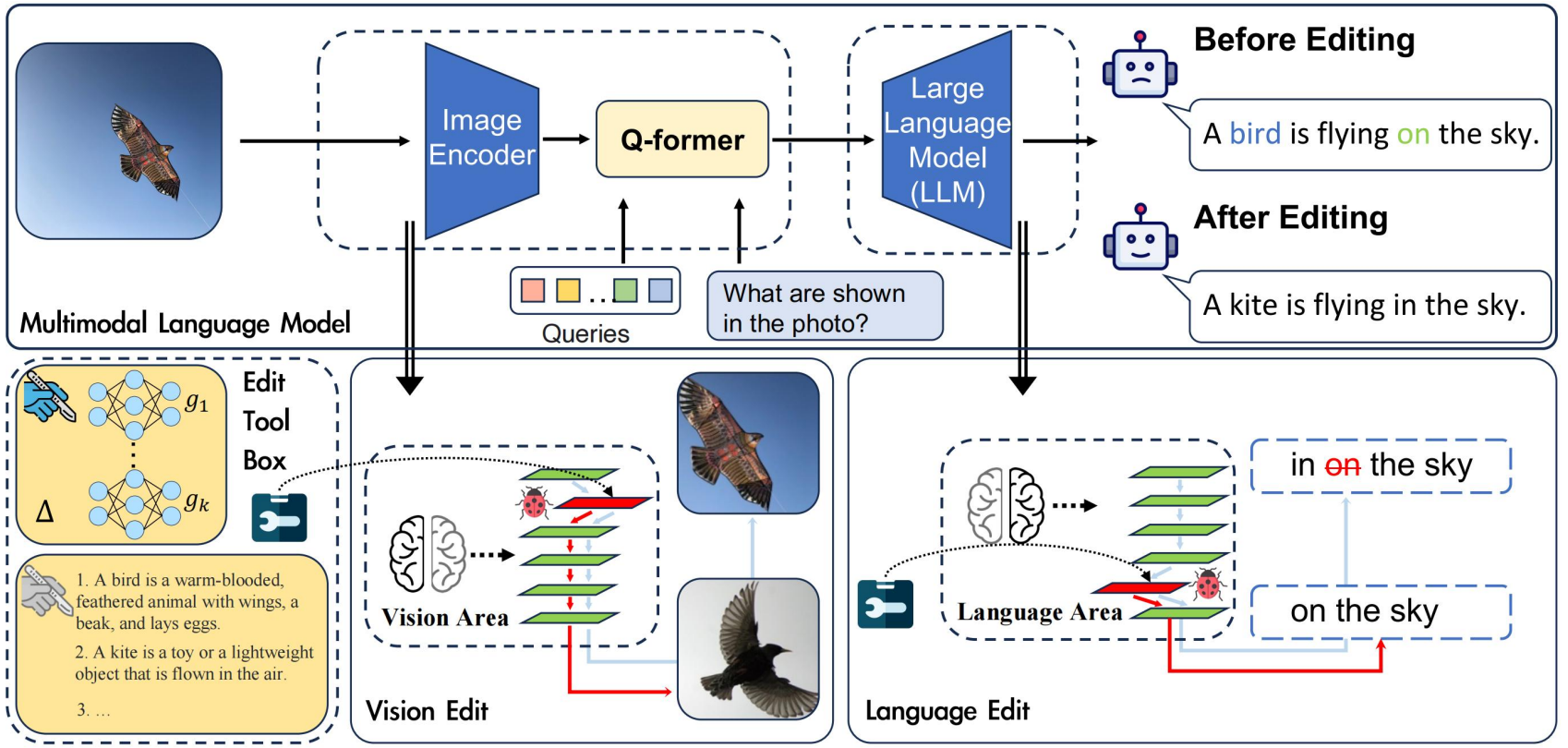


Figure 2: Utilizing multimodal LLM (e.g., BLIP-2 OPT) as an example, we dissect the comprehensive multimodal LLM into two components (Vision module and Textual module). The model's erroneous output could potentially stem from either or both of these modules. Drawing an analogy with human errors in "vision" and "speech", we apply model editing methods to these two components, thereby changing the model to refine its output.

Can We Edit Multimodal Large Language Models?

Comparison methods

Base model: BLIP-2 OPT. / MiniGPT-4.

Base Methods	Base Model
	FT (vision block)
	FT (last layer)
Model Editing	Knowledge Editor
	In-Context Editing
	SERAC
	MEND

Evaluation metric

- ✓ Reliability.

$$\mathcal{M}_{rel} = \mathbb{E}_{(i_e,x_e,y_e)\sim\mathcal{D}_{\text{edit}}}\left[\mathbb{1}_{f(i_e,x_e;\theta_e(i_e,x_e,y_e))=y_e}\right] \tag{1}$$

- ✓ Locality (text and image).

$$\mathcal{M}_{loc}^{Text} = \mathbb{E}_{\substack{(i_e,x_e,y_e)\sim\mathcal{D}_{\text{edit}} \\ (x,y)\sim\mathcal{D}_{\text{loc-t}}}}\left[\mathbb{1}_{f(x;\theta_e(i_e,x_e,y_e))=f(x,\theta)}\right]$$

$$\mathcal{M}_{loc}^{Image} = \mathbb{E}_{(i_v,x_v,y_v)\sim\mathcal{D}_{\text{loc-v}}}\left[\mathbb{1}_{f(i_v,x_v;\theta_e)=f(i_v,x_v;\theta)}\right] \tag{3}$$

where (i_v, x_v, y_v) is the out-of-scope data, and θ_e denote the parameter updated by edit data (i_e, x_e, y_e).

- ✓ Generality (text and image).

$$\mathcal{M}_{gen}^{Text} = \mathbb{E}_{(x_r)\sim\mathcal{N}(x_e)}\left[\mathbb{1}_{f(i_e,x_r;\theta_e)=f(i_e,x_e;\theta_e)}\right] \tag{4}$$

$$\mathcal{M}_{gen}^{Image} = \mathbb{E}_{(i_r)\sim\mathcal{N}(i_e)}\left[\mathbb{1}_{f(i_r,x_e;\theta_e)=f(i_e,x_e;\theta_e)}\right] \tag{5}$$

where i_r presents the rephrased image, x_r refers to the rephrased text prompt, and $\mathcal{N}(x)$ denotes to in-scope objects of x.

[7] Can We Edit Multimodal Large Language Models? EMNLP 2023.

Can We Edit Multimodal Large Language Models?

Task: VQA and Image-Caption

- Reliability $\mathcal{D}_{\text{edit}}$

  Our foundational edit data originates from suboptimal entries across two eval datasets, namely, VQAv2 and COCO Caption.

- Locality Dataset Construction

  NQ dataset & OK-VQA

- Generality Dataset Construction

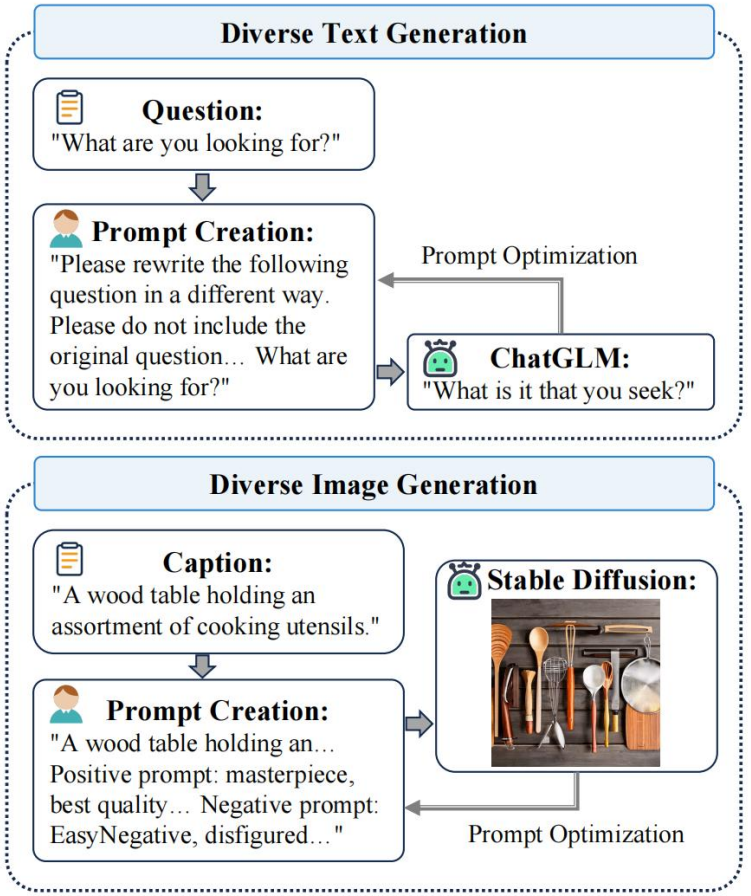


Figure 4: Generality dataset construction process.

	Method	Editing VQA				Editing Image Caption			
		Reliability ↑	T-Generality ↑	T-Locality ↑	M-Locality ↑	Reliability ↑	T-Generality ↑	T-Locality ↑	M-Locality ↑
BLIP-2 OPT									Size: 3.8B
Base Methods	Base Model	0.00	0.00	100.0	100.0	0.00	0.00	100.0	100.0
	FT (vision block)	56.28	29.88	100.0	11.32	0.08	0.00	100.0	7.31
	FT (last layer)	58.70	15.33	78.86	2.86	0.24	0.10	67.67	3.91
Model Editing	Knowledge Editor	67.80	63.00	97.32	45.89	69.00	62.80	96.21	45.55
	In-Context Editing	99.95	91.59	13.16	1.88	96.70	78.20	13.36	2.17
	SERAC	91.20	91.40	100.0	0.33	94.40	96.00	100.0	0.47
	MEND	92.60	90.80	96.07	65.15	65.00	38.00	92.67	55.72
MiniGPT-4									Size: 7.3B
Base Methods	Base Model	0.00	0.00	100.0	100.0	0.00	0.00	100.0	100.0
	FT (vision block)	39.58	0.98	100.0	3.96	0.63	0.00	100.0	5.13
	FT (last layer)	39.57	0.58	72.01	16.42	2.75	0.00	35.52	9.28
Model Editing	Knowledge Editor	87.77	86.62	97.15	55.77	35.10	24.20	96.78	52.22
	In-Context Editing	71.72	40.23	13.46	2.00	68.60	59.80	12.51	2.96
	SERAC	87.20	84.60	100.0	0.33	40.20	36.60	100.0	0.97
	MEND	95.51	95.27	98.73	71.33	87.10	84.10	98.34	59.53

Table 2: Main results on the **MMEdit**. **T-Locality, M-Locality** refer to the textual and multimodal stability. **T-Generality** represents textual generality. **Reliability** denotes the accuracy of successful editing.

Thanks!