

NeuralDB: 通过神经 KV 数据库扩展 LLM 中的知识编辑到 100,000 个事实

Weizhi Fei^{*} Hao Shi^{*} Jing Xu[†] Jingchen Peng^{*} Jiazheng Li[‡]
Jingzhao Zhang^{† ¶} Bo Bai[§] Wei Han[§] Zhenyuan Chen[§] Xueyan Niu^{§ ¶}

Abstract

高效地编辑存储在大型语言模型 (LLM) 中的知识,可以在不进行大规模训练的情况下更新模型。一种可能的解决方案是 Locate-and-Edit (L & E), 它允许同时修改大量事实。然而,此类编辑可能会损害 LLM 的一般能力,甚至在扩展到上千次编辑时,可能会导致遗忘已编辑的事实。在本文中,我们将现有的线性 L & E 方法建模为查询一个键-值 (KV) 数据库。从这个角度出发,我们提出了 NeuralDB,这是一种编辑框架,明确将已编辑的事实表示为一个神经 KV 数据库,并配备了一个非线性门控检索模块,有效地保留了 LLM 的一般能力。在 ZsRE 和 CounterFacts 数据集上进行了涉及编辑 10,000 个事实的综合实验,使用了 GPT2-XL、GPT-J (6B) 和 Llama-3 (8B)。结果表明,NeuralDB 不仅在编辑效率、泛化、特异性、流畅性和一致性方面表现出色,而且在六个具有代表性的文本理解和生成任务中的整体表现也得到了保留。进一步的实验表明,NeuralDB 即使在扩展到 100,000 个事实 (比以往工作多 50 倍) 时,仍然保持其有效性。

更新大型语言模型 (LLMs) 参数中存储的知识对于刷新过时的信息和整合特定领域的知识以促进定制化非常重要。然而,从头开始重新训练 LLMs 通常是不切实际的,因为这需要大量的计算资源和时间。微调虽然是一种更可行的方法,但可能导致灾难性遗忘。为了解决这些挑战,知识编辑 (KE) 方法出现了,这些方法被认为是有前途的解决方案,可以在 LLMs 中进行精确且成本效益高的特定事实关联的修改。在不损害 LLMs 的情况下编辑大量知识是 KE 中的一个重要但具有挑战性的任务。定位并编辑 (L & E) 方法是实现这一目标的主要解决方案,这些方法首次实现了对大量事实的高效和可扩展的编辑。L & E 范式旨在通过引入一个线性扰动到前馈网络 (FFN) 层中的目标参数来纠正模型对新事实的激活。新激活从新事实中学习。此外,与编辑内容无关的“保留知识”的激活应被保留。通常,保留知识是从维基百科中抽样的。然后,使用最小二乘法求解修改后的参数,以确保为新事实生成所需的残差而不影响要保留的样本。值得注意的是,AlphaEdit 引入了零空间投影来增强采样知识的保留,有效地维护了 LLMs 的一般能力。受其工作启发,我们为这些方法提出了一个统一框架,使我们能够进一步提升编辑能力。

尽管已有进展,现有方法只能编辑数百个事实而不影响模型的总体能力。在编辑成千上万个事实时,经后期编辑的模型通常会遭受从广泛训练中发展出的有价值的总体能力的下降 (见图 1)。该下降可归因于从维基百科采样的子集不足以代表总体能力。此外,随着编辑的事实增多,以前更新的信息会逐渐消退,因为编辑方法使用的线性系统在容量上不是最优的。

Preprint. Under review.

^{*}Department of Mathematical Sciences, Tsinghua University; { fwz22, shih22, pjcz22 } @mails.tsinghua.edu.cn

[†]IIS, Tsinghua University; { xujing21, jingzhaoz } @mails.tsinghua.edu.cn

[‡]College of AI, Tsinghua University; foreverlasting1202@outlook.com

[§]Huawei Technologies Co., Ltd.; { baibo8, harvey.hanwei, chenchenyuan, niuxueyan3 } @huawei.com

[¶]Corresponding authors.

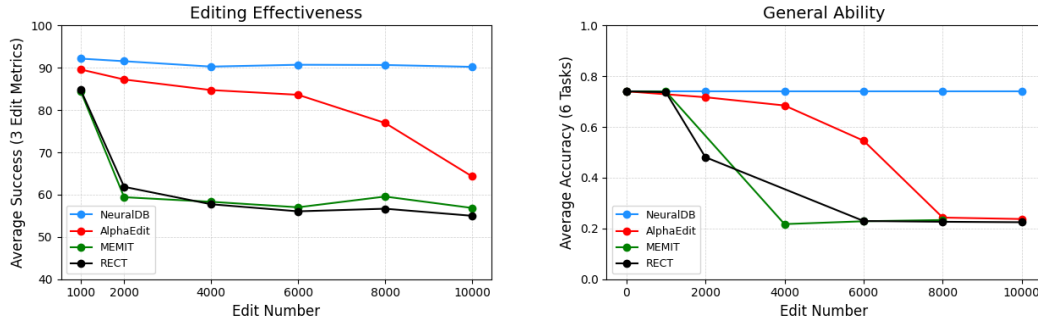


Figure 1: 所提出的 NeuralDB 能够扩展到编辑最多 10,000 个事实，几乎没有性能损失。左图：在 MCF 数据集上与 AlphaEdit、MEMIT 和 RECT 比较的效能、泛化和特异性的平均值。随着编辑数量的增加，NeuralDB 仅出现轻微下降，而其他三种方法则逐步出现较大幅度的下降。右图：lm-evaluation-harness 基准上的六项任务（MMLU、SciQ、Commonsense QA、ARC Challenge、Lambada、WSC273）的平均性能。保持这些通用能力是至关重要的，因为它们是通过大规模数据集的大规模预训练所获得的。即使在编辑事实数量增加的情况下，NeuralDB 也完美地保持了这些通用能力，而其他三种方法几乎在编辑数量增加时立即失去了这些能力。

在本文中，我们证明了大多数 L & E 方法，包括 MEMIT [?]、D4S [?] 和 AlphaEdit [?]，可以从键值 (KV) 数据库的角度理解。我们将这些方法概念化为查询一个 KV 数据库，其中某个隐藏状态作为查询以检索相应的学习残差。形式上，这些方法的更新参数可以被解读为与编辑事实相关联的残差矩阵的加权平均。我们通过实证研究多个事后编辑模型中的这些权重，并表明它们在推理过程中呈现出极端稀疏的形式。具体而言，当在编辑的事实上进行推理时，权重与一个独热向量非常相似，只有与编辑事实对应的权重是非零的，从而返回相关联的残差。相反，当在无关内容上进行推理时，权重呈现为一个零向量，从而防止干扰。

鉴于这一新视角，我们提出了一个神经 KV 数据库 (NeuralDB) 编辑框架，该框架将目标 FFN 层与一个替代原线性扰动 Δ 的门控非线性检索模块集成在一起。NeuralDB 首先从编辑后的事实中构建神经 KV 数据库，然后利用检索模块将该数据库集成到 LLMs 中。只有当后编辑模型涉及编辑后的事实时，门控模块才返回最兼容的学习残差，因此，它克服了 L & E 方法中的线性限制，享有更大的编辑能力。通过门控机制，我们的方法既能保护通用能力，又能减少从维基百科采样的计算和缓存成本。此外，NeuralDB 易于管理，支持添加、修改和删除等操作。

为了验证 NeuralDB 的能力，我们对三种模型进行了全面的实验：GPT-2 XL [?]、GPT-J (6B) [?] 和 Llama-3-Instruct (8B) [?]。这些模型根据其在编辑过的事实和一般能力方面的表现进行了评估。我们的结果表明，NeuralDB 在编辑后的事实、重述的编辑事实以及邻近事实方面的表现明显更好，并且在生成后的模型保持了流畅性和一致性。此外，NeuralDB 表现出良好的可扩展性——当编辑事实从 2000 条扩展到 10,000 条时，性能仅下降了 3%。为了进一步验证 NeuralDB 的有效性，我们使用广泛采用的文本理解和生成任务评估了编辑后的模型。结果表明，NeuralDB 能够在不影响 Llama-3-8B 生成文本质量的情况下成功编辑数以万计的事实。在 100,000 条编辑事实（比 AlphaEdit 高出 45 倍的编辑事实）的广泛扩展实验中，证明了我们的方法能够在完全保留一般语言理解和生成能力的同时保持高编辑精度。大容量和生成质量不下降的结合，凸显了 NeuralDB 在 LLMs 的可信赖和可定制部署中的潜力。

1 背景

当前的知识嵌入方法通常集中于更新基于变换器的大型语言模型 (LLMs) 的事实知识，这些知识可以表示为三元组 (s, r, o) ，其中 s 表示主语， r 代表关系谓词， o 是宾语。例如，事实“最新的世界杯在卡塔尔举行。”可以表示为 (“The latest World Cup”, “was located in”, “Qatar”)。反过来，三元组可以转化为自然语言句子，我们在后续过程中将这两种表示方式视为可互换的。我们将编辑后的事实表示为一组修订后的元组 $\mathcal{F}^* = \{(s_i, r_i, o_i \rightarrow \hat{o}_i)\}$ ，其中 $\hat{o}_i$ 表示替换原始 o_i 的目标新对象。值得注意的是，知识嵌入不应损害模型的通用能力 [?]，因为这些能力通常是通过广泛的预训练发展出来的。如图所示 2，在推理过程中，大型语言模型中第 l 层 FFN 层的预测的隐藏状态 h^l 是根据以下递归形式计算的：

$$h^l = h^{l-1} + a^l + m^l, \quad m^l = W_{out}^l \sigma(W_{in}^l (\mathcal{N}(h^{l-1} + a^l))), \quad (1)$$

其中 $\mathbf{a}^l$ 和 $\mathbf{m}^l$ 分别是注意力块和 FFN 层的输出, $\mathbf{W}_{in}^l$ 和 $\mathbf{W}_{out}^l$ 分别代表第 l 层 FFN 层的权重矩阵, $\mathcal{N}(\cdot)$ 表示层归一化, $\sigma(\cdot)$ 代表激活函数。我们遵循先前的研究 [??], 通过如下操作性键值存储的模型化 FFN 层:

$$\mathbf{k}^l = \sigma(\mathbf{W}_{in}^l(\mathcal{N}(\mathbf{h}^{l-1} + \mathbf{a}^l))), \quad \mathbf{v}^l = \mathbf{W}_{out}^l \mathbf{k}^l. \quad (2)$$

然后, 文本知识 (s, r, o) 可以通过从推理过程推导出的激活链接到大型语言模型的参数知识。在这种情况下, 键向量 $\mathbf{k}$ 可以解释为对查询 (s, r) 的编码, 而宾语 o 随后由模型根据与键 [?] 关联的值向量 $\mathbf{v}$ 解码。

L & E 方法旨在通过修改参数来调整新事实的激活 $\mathbf{v}$ 。在这一范式中, 通过监督学习在指定层 l 中向激活 $\mathbf{v}^l$ 添加一个可学习的扰动, 从而产生一个新的激活 $\hat{\mathbf{v}}^l$, 使模型能够生成新的答案 $\hat{o}$ 。然后, 扰动矩阵 Δ^l 应满足

$$(\mathbf{W}_{out}^l + \Delta^l) \mathbf{k}_i^l = \hat{\mathbf{v}}_i^l \quad \text{and} \quad (\mathbf{W}_{out}^l + \Delta^l) \mathbf{k}_j^l = \mathbf{v}_j^l \quad (3)$$

, 其中 $\hat{\mathbf{v}}_i^l$ 对应于新事实, $(\mathbf{k}_j^l, \mathbf{v}_j^l)$ 对应于应保存的采样知识。

为了简化符号, 我们用 $\mathbf{W}$ 表示将要更新的参数 $\mathbf{W}_{out}^l \in \mathbb{R}^{d_2 \times d_1}$, 其中 d_1 和 d_2 分别表示 FFN 中间层和输出层的维度。为了更新大量事实, 我们通过叠加向量来获得键和值矩阵:

$$\mathbf{K}_1 = [\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_m] \in \mathbb{R}^{d_1 \times m}, \quad \hat{\mathbf{V}}_1 = [\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \dots, \hat{\mathbf{v}}_m] \in \mathbb{R}^{d_2 \times m}, \quad (4)$$

, 其中 m 是编辑的事实数量, 我们在附录 ?? 中提供了从目标编辑事实计算 $\mathbf{k}_i$ 和 $\hat{\mathbf{v}}_i$ 的详细信息。此外, 我们定义了残差矩阵和残差向量为

$$\mathbf{R}_1 = \hat{\mathbf{V}}_1 - \mathbf{W} \mathbf{K}_1 \quad \text{and} \quad \mathbf{r}_i = \mathbf{R}_1[:, i]. \quad (5)$$

。为了保留编辑模型的一般能力, 当前的方法 [??] 需要从维基百科⁶ 中抽取大量事实来构建矩阵 $\mathbf{K}_0$, 该矩阵通常由 10 万个叠加的键向量组成, 用于代表保留的一般知识。

2 通过查询键值数据库重新思考定位和编辑方法

在本节中, 我们展示了大多数 L & E 方法可以被视为查询一个键值 (KV) 数据库。我们通过理论推导和实验验证研究这些支持大规模知识更新的编辑方法。具体而言, 我们在执行 MEMIT [?] 和 AlphaEdit [?] 时, 在单步中更新所有目标事实, 这已被证明可以有效缓解 D4S [?] 造成的一般能力的下降。为简化分析, 我们仅讨论单个 FFN 层的参数更新。

我们得出结论, 用 Δ_{upd} 更新参数的操作机制可以写成

$$(\mathbf{W} + \Delta_{\text{upd}}) \mathbf{k} = \mathbf{v} + \mathbf{R}_1 \omega, \quad \omega = \mathbf{K}_1^T \mathbf{S} \mathbf{k} \in \mathbb{R}^{m \times 1}, \quad (6)$$

, 其中 $\mathbf{k}$ 和 $\mathbf{v}$ 是来自原始激活的键和值向量, $\mathbf{K}_1^T$ 是使用编辑后的事实计算的键矩阵的转置, $\mathbf{S}$ 是从特定编辑方法获得的核矩阵。 $\mathbf{R}_1 \omega$ 是 $\mathbf{R}_1$ 的加权平均, 其中加权分数是自相似性 $\omega = \mathbf{K}_1^T \mathbf{S} \mathbf{k}$ 。这意味着 $\mathbf{k}$ 充当查询以从学习到的残差中检索信息 $\mathbf{R}_1$ 。然后我们推导出 MEMIT 和 AlphaEdit 的解决方案, 并呈现它们的结构。

MEMIT 的目标是以下有约束的最小二乘优化:

$$\arg \min_{\Delta} \|(\mathbf{W} + \Delta) \mathbf{K}_1 - \hat{\mathbf{V}}_1\|_2^2 + \beta_1 \|\Delta \mathbf{K}_0\|_2^2, \quad (7)$$

, 其中项 $\|\Delta \mathbf{K}_0\|_2^2$ 确保更新的参数保持保留的知识。MEMIT 的闭式解可以如下推导:

$$\Delta_{\text{upd}}^{\text{MEMIT}} = \mathbf{R}_1 \mathbf{K}_1^T (\mathbf{K}_1 \mathbf{K}_1^T + \beta_1 \mathbf{K}_0 \mathbf{K}_0^T)^{-1}. \quad (8)$$

设 $\mathbf{S}_1 = (\mathbf{K}_1 \mathbf{K}_1^T + \beta_1 \mathbf{K}_0 \mathbf{K}_0^T)^{-1}$, 那么更新可以表示为 $\Delta_{\text{upd}}^{\text{MEMIT}} = \mathbf{R}_1 \mathbf{K}_1^T \mathbf{S}_1$ 。

同样地, 我们讨论 AlphaEdit, 该方法利用零空间投影将要保留的知识的软约束转化为硬约束。零空间投影矩阵 $\mathbf{P}$ 是通过将 $\mathbf{K}_0^T \mathbf{K}_0$ 进行 SVD 分解计算出来的。然后, AlphaEdit 构建 $\Delta = \delta \mathbf{P}$ 使得 $\|\Delta \mathbf{K}_0\|$ 趋于零, 并用 L2 范数求解以下最小二乘问题:

$$\arg \min_{\delta} \|(\mathbf{W} + \delta \mathbf{P}) \mathbf{K}_1 - \hat{\mathbf{V}}_1\|_2^2 + \beta_2 \|\delta \mathbf{P}\|_2^2. \quad (9)$$

⁶<https://huggingface.co/datasets/wikimedia/wikipedia>

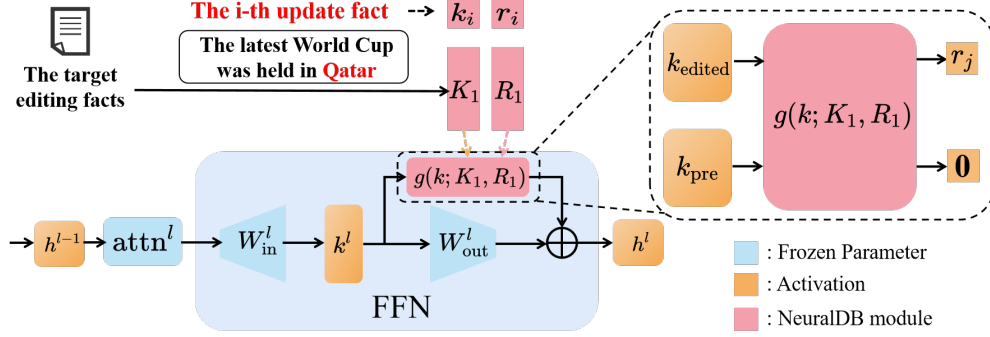


Figure 2: NeuralDB 编辑框架概述。我们在 FFN 层中以非线性门控检索模块替换了 LMATH-XAD E 中的线性系统。通过方程 4 和方程 5 定义的键 K_1 和残差矩阵 R_1 ，被计算用于构建神经 KV 数据库。在推理过程中，我们的非线性门控函数 $g(\cdot; K_1, R_1)$ 仅返回与后编辑模型推断出的一个编辑事实最匹配的残差 r_j ，并涉及键向量 k_{edited} 。当涉及到保留知识的键向量 k_{pre} 时，函数 $g(\cdot; K_1, R_1)$ 会恢复零向量 0 。

。AlphaEdit 的闭式解为

$$\Delta_{\text{upd}}^{\text{AlphaEdit}} = R_1 K_1^T P^T (P K_1 K_1^T P^T + \beta_2 I)^{-1} P. \quad (10)$$

。令 $S_2 = P^T (P K_1 K_1^T P^T + \beta_2 I)^{-1} P$ ，这可以重写为 $\Delta_{\text{upd}}^{\text{AlphaEdit}} = R_1 K_1^T S_2$ ，这也是 R_1 的加权平均，其中 $K_1^T S_2 k$ 表示加权得分 ω 。特别地，当 k 属于要保留知识的零空间 K_0 时，AlphaEdit 返回 0 。因此， $S k = P^T (P K_1 K_1^T P^T + \beta_2 I)^{-1} (P k) = 0$ ，这有效地保留了通用能力。

我们通过 MEMIT 和 AlphaEdit 对三个后编辑模型在推理新事实时进行经验可视化，显示加权得分 $\omega = K_1^T S k$ 。具体而言，我们针对 Counterfactuals 数据集的 1,000 个事实执行 KE，并评估这些经编辑事实上的后编辑模型。在测试 i -th 编辑知识时，我们将 ω_i 的 i -th 组件视为正样本，而其余组件则视为负样本。正如图 ?? 所示，标记为负样本的加权得分接近 0，而正样本的加权得分则显著提高，其中 AlphaEdit 的接近 1。这些实证结果表明，只有与编辑事实对应的残差被激活，而其他与目标事实无关的残差向量则有效地被抑制。附录 11.2 提供的额外结果表明，当前方法在对未修改事实进行推理时通常产生 $\omega = 0$ 。

3 方法

从查询 KV 数据库的角度来看，我们可以通过显式构建 KV 数据库来提升编辑能力并保持 LLMs 的一般能力。为此，我们提出了一个神经 KV 数据库 (NeuralDB) 编辑框架，通过用非线性门控检索模块替换线性扰动，来作为插入即用的模块。具体来说，NeuralDB 首先计算目标编辑事实的键向量和剩余向量，并构建一个神经 KV 数据库。然后，该构建的 KV 数据库配备有非线性检索功能，用于在推理编辑事实时查询适当的剩余向量。

3.1 神经 KV 数据库

给定目标事实 $\mathcal{F}^* = \{(s_i, r_i, o_i \rightarrow \hat{o}_i)\}$ ，我们首先计算它们的关键和残差矩阵以获得 K_1 和 R_1 。下面，我们正式将它们定义为一个神经 KV 数据库。

Definition 1 (Neural Key-Value Database). 给定 $\mathcal{F}^*$ ，构建的神经 KV 数据库可以表示为 (K_1, R_1) ，其中 $K_1 \in \mathbb{R}^{d_1 \times m}$ 和 $R_1 \in \mathbb{R}^{d_2 \times m}$ 分别表示经过编辑事实的键矩阵和剩余矩阵，如方程 (4) 和方程 (5) 所定义。 K_1 和 R_1 在数据库中作为键和值使用， $k_i = K_1[:, i]$ 与 $r_i = R_1[:, i]$ 相关联。

我们提出了以下充足条件用于有效编辑。

Editing Rules. Given a neural KV database (K_1, R_1) for target facts $\mathcal{F}^*$. Let k denote the key vector obtained through inference on a sequence x , KE should then adhere to the following rules:

- I. 如果序列 x 与第 i 个编辑过的事实相关联，那么 k 将在 K_1 内与 k_j 匹配，并返回相应的残差向量 r_j ；

II. 如果序列 $\mathbf{x}$ 与任何已编辑的事实无关，则 $\mathbf{k}$ 不会匹配 $\mathbf{K}_1$ 中的任何键，并返回零向量 $\mathbf{0}$ 。

规则 确保目标事实的有效更新，而规则 确保与已编辑事实无关的知识得以保留。为了更严格地实现这些规则，我们提出以下非线性检索函数。

3.2 非线性门控检索模块

给定一个由目标编辑事实构造的神经 KV 数据库 $(\mathbf{K}_1, \mathbf{R}_1)$ ，我们提出了以下函数，该函数返回与目标编辑事实的最大相似残差：

$$g(\mathbf{k}; \mathbf{K}_1, \mathbf{R}_1) = \mathbf{r}_j * \overbrace{\mathbf{1}_{\cos(\mathbf{k}, \mathbf{k}_j) > \gamma}}^{\text{Gate}}, j = \arg \max \cos(\mathbf{k}, \mathbf{k}_i), \quad (11)$$

，其中 $\cos(\cdot, \cdot)$ 表示余弦相似性， $\mathbf{1}$ 代表指示函数， γ 是控制门控机制的参数。我们采用余弦相似性，因为它提供了良好的可解释性，有助于轻松设置 γ 。

如图 2 所示，非线性函数被整合到目标 FFN 层中如下：

$$\mathbf{v}^l = \mathbf{W}^l \mathbf{k}^l + g(\mathbf{k}^l; \mathbf{K}_1, \mathbf{R}_1). \quad (12)$$

。当后编辑模型利用保留的知识时，涉及的键向量 $\mathbf{k}$ 通常与矩阵 $\mathbf{K}_1$ 中的所有键向量 $\mathbf{k}_i$ 表现出低相似性。因此，该附加模块不会提供扰动，确保原始模型的工作流程保持不变。因此，原编辑的 LLM 的通用能力可以成功保留。相反，当后编辑模型遇到修正事实时，我们的检索函数可以识别并调用最密切相关的剩余向量，因为键之间具有高度相似性。这种精确的检索机制有效地更新了指定的参数知识。

我们提出的框架的另一个显著优势是其易于部署。当前的 L & E 方法通常通过优化来更新参数，这给增量更新或恢复已修改的事实带来了挑战。相比之下，我们的框架维护与目标事实具体对应的神经 KV 数据库。此设计促进了模型中编辑事实的添加、删除和修改，从而提高了编辑过程的灵活性。

单层编辑与多层编辑 我们仅在一个单独的 FFN 层中实现了我们的模块。尽管之前的方法通常采用多层策略，但我们发现这种策略提供的性能提升有限。我们在附录 10 中提供了相关讨论。

在 NeuralDB 模块中引入额外的参数会导致计算和存储需求增加。具体来说，矩阵 $\mathbf{K}_1 \in \mathbb{R}^{d_1 \times m}$ 和 $\mathbf{R}_1 \in \mathbb{R}^{d_2 \times m}$ 的存储空间复杂度为 $O((d_1 + d_2) \times m)$ ，其中 m 表示事实的数量。当使用 Llama 3 8B (Instruct) 编辑 10,000 个事实时，额外参数的大小为 150M，约占原始模型大小的 2.2%。与原始模型相比，在 Counterfact 数据集中评估 10,000 个事实的时间仅增加了 1.5%。关于各种模型的额外内存使用和计算时间的详细信息在附录 9 中提供。

4 实验

在本节中，我们对大规模 KE 方法进行全面评估。为了进行公平的比较，我们主要遵循 AlphaEdit [?] 的实验设置来基准测试我们的方法。具体来说，我们检查编辑后的模型是否能够有效地平衡对编辑事实的掌握与其通用能力的保留。NeuralDB 编辑框架的详细实现见附录 ??。此外，附录 10 中包含了消融研究，我们探讨了层选择、参数 γ 的设置以及多层配置的影响。

模型和方法 我们选择了三个代表性的大型语言模型，包括 GPT-2 XL [?]、GPT-J (6B) [?] 和 Llama3 (8B) [?]。我们将我们的方法与以下 KE 方法进行比较，包括微调 (FT) [?]、MEND [?]、ROME [?]、MEMIT [?]、MELO [?]、RECT [?] 和 AlphaEdit [?]。我们在附录 ?? 中详细介绍了这些基线方法和模型。

用于知识编辑的数据集 为了评估知识编辑 (KE) 方法，我们使用了两个广泛认可的基准：Counterfact 数据集 [?] 和 ZsRE 数据集 [?]。与之前的研究一致 [??]，我们采用以下评估指标：效果（编辑事实的成功率）、泛化（改写事实的成功率）、特异性（相邻事实的成功率）、流畅度（生成熵）以及一致性（参考得分）。数据集和指标的详细解释见附录 ??。

Table 1: NeuralDB 与现有方法在大规模 KE 任务上的比较。我们评估了编辑 2,000 个事实的性能，其中 Pre-edited、FT、MEND、InstructEdit、MELO 和 ROME 的结果来源于 [?]。对于 10,000 个事实，MEMIT、RECT、AlphaEdit 和 NeuralDB 的结果使用箭头 (→) 标记。2,000 个和 10,000 个事实的最佳结果分别以粗体突出显示。

Method	Model	Counterfact					ZsRE		
		Efficacy ↑	Generalization ↑	Specificity ↑	Fluency ↑	Consistency ↑	Efficacy ↑	Generalization ↑	Specificity ↑
Pre-edited		7.9	10.6	89.5	635.2	24.1	37.0	36.3	31.9
FT	LLaMA3	83.3	67.8	46.6	233.7	8.8	30.5	30.2	15.5
MEND		63.2	61.2	45.4	372.2	4.2	0.9	1.1	0.5
InstructEdit		66.6	64.2	47.1	443.9	7.3	1.6	1.4	1.0
MELO		65.3	58.6	63.4	609.0	32.4	25.2	24.1	30.4
ROME		64.4	61.4	49.4	449.1	3.3	2.0	1.8	0.7
MEMIT		63.5→63.4	62.8→56.6	52.0→50.55	466.6→460.4	6.5→6.5	36.7→0.1	32.9→0.1	19.1→1.5
RECT		64.2→60.0	62.5→53.9	58.9→51.2	502.8→399.1	12.9→1.6	86.8→0.0	82.3→0.0	31.9→0.0
AlphaEdit		99.1→75.8	94.0→63.1	68.6→54.0	622.7→417.8	32.8→7.0	94.4→90.5	91.3→85.9	32.6→30.3
NeuralDB		99.9→99.2	86.6→85.9	88.2→85.6	632.7→631.02	32.9→32.6	96.3→95.9	92.0→91.0	31.9→31.8
Pre-edited		16.2	18.6	83.1	621.8	29.7	26.3	25.8	27.4
FT	GPT-J	92.2	72.4	43.4	297.9	6.7	72.4	68.9	19.7
MEND		46.2	46.2	53.9	242.4	3.9	0.7	0.7	0.5
InstructEdit		50.6	51.7	56.3	245.9	4.2	0.9	0.9	0.7
MELO		78.3	60.5	66.8	610.8	24.3	82.2	32.9	26.7
ROME		57.5	54.2	52.1	589.4	3.2	56.4	54.7	9.9
MEMIT		98.6→48.8	95.4→49.3	66.1→51.9	557.8→281.5	36.5→5.1	90.5→0.2	84.7→0.1	30.9→0.2
RECT		98.8→76.3	86.3→70.6	74.4→54.9	618.1→517.3	41.2→25.4	96.6→53.5	91.5→49.6	29.0→21.9
AlphaEdit		99.8→91.6	96.3→79.6	76.2→60.3	618.5→517.8	41.9→6.9	99.7→94.2	95.9→86.1	28.8→22.5
NeuralDB		99.7→99.1	94.6→93.2	80.0→75.7	619.8→620.0	41.4→41.3	99.2→98.2	95.9→95.0	27.5→27.0
Pre-edited		22.2	24.3	78.5	626.6	31.9	22.2	31.3	24.2
FT	GPT2-XL	63.6	42.2	57.1	519.4	10.6	37.1	33.3	10.4
MEND		50.8	50.8	49.2	407.2	1.0	0.0	0.0	0.0
InstructEdit		55.3	53.6	53.3	412.6	1.1	3.5	4.3	3.2
ROME		54.6	51.2	52.7	366.1	0.7	47.5	43.6	14.3
MELO		72.6	53.6	63.3	588.6	23.6	93.5	45.3	23.5
MEMIT		93.0→58.5	83.3→55.8	58.9→56.1	481.8→496.2	23.2→8.1	74.4→3.5	66.9→2.8	25.87→2.07
RECT		91.8→86.9	79.5→69.5	64.0→55.0	482.1→517.8	20.3→10.9	82.6→27.5	74.7→25.1	24.6→13.1
AlphaEdit		99.4→92.2	93.8→76.5	65.6→56.5	584.0→580.9	37.9→29.6	93.2→57.1	83.5→47.5	25.3→13.5
NeuralDB		99.8→99.1	97.2→95.7	74.1→70.9	621.5→619.9	42.2→41.7	96.3→94.6	92.8→91.0	25.0→24.2

通用能力数据集 我们使用以下典型数据集评估编辑后的 LLMs 的一般能力: SciQ [?] (科学问答)、MMLU [?] (大规模多任务语言理解)、Commonsense QA [?] (常识知识理解)、ARC Challenge [?] (需要推理的挑战任务)、WSC273 [?] (共指消解)、Lambada [?] (预测文本段落的结尾)。诸如 SciQ、MMLU 和 Commonsense QA 等数据集主要评估基于知识的问题回答，侧重于模型理解和保持事实信息的能力。相比之下，ARC Challenge、WSC273 和 Lambada 旨在评估超越单纯知识记忆的能力，如推理和文本生成。关于这些数据集的详细信息，参见附录 ??。

4.1 后编辑模型的编辑效果

我们评估了编辑 2,000 条事实后的性能，并在括号中包括了 MEMIT、RECT、AlphaEdit 和 NeuralDB 编辑 10,000 条事实的结果。结果呈现于表格 1 中。我们的方法在三个模型和两个数据集的所有指标上都表现出卓越的性能。特别是，当将编辑事实的数量从 2,000 扩展到 10,000 时，我们的方法仅表现出微不足道的性能下降，相较之下，基准方法则观察到显著的退化。

我们的方法在各种情况下展示了卓越的效果。即使在编辑了 10,000 个事实之后，我们的后编辑模型仍保持较强的效果和特异性。相比之下，所有基线方法在效果和特异性指标上都显著下降。这突出显示了我们方法在有效编辑方面的优秀可扩展性。

就特异性和流畅性指标而言，我们的后编辑模型表现出与前编辑模型几乎相同的有效性，相比于基准模型，这在 CounterFacts 数据集上有了显著的改进。此外，我们的方法确保生成的文本在一致性上保持高度的一致性。这表明我们的方法能够在保持生成能力的同时，保留未修改事实的记忆。

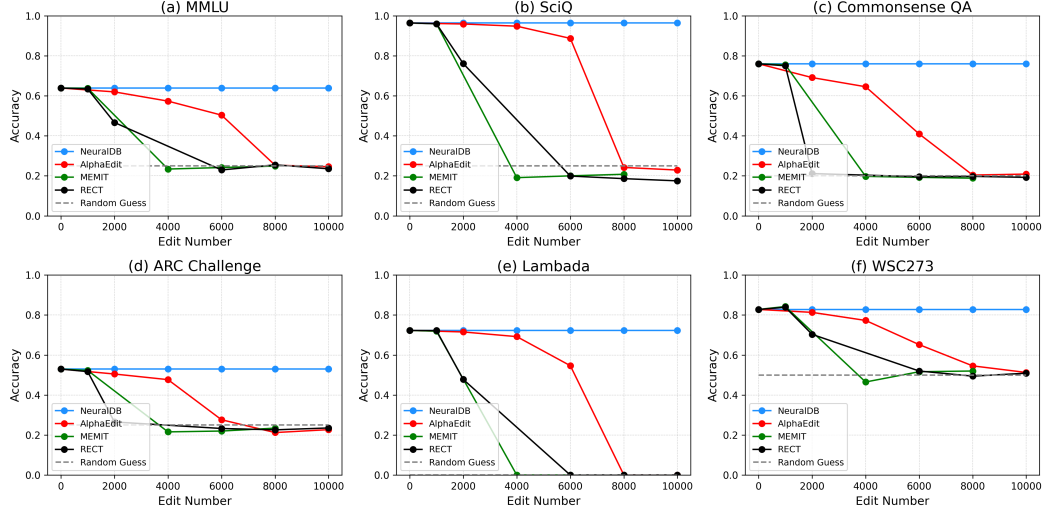


Figure 3: 大规模编辑后的常规能力结果。NeuralDB 的性能与基线方法进行比较，包括 MEMIT、RECT 和 AlphaEdit。多项选择数据集的随机猜测基线用虚线表示。NeuralDB 在编辑数量增加时表现出卓越的常规能力保持性。

4.2 后编辑模型的一般能力

我们评估了不同配置的后编辑模型，其性能在 2,000、4,000、6,000、8,000 和 10,000 个事实上进行了评估，如图 3 所示。评估是在 lm-evaluation-harness [?] 上进行的。结果表明，我们的方法能有效编辑大量事实，同时不影响模型在各种任务上的整体能力。相比之下，现有的 L & E 方法在编辑 4,000 个事实时表现不佳，并且随着编辑事实数量的增加，其一般能力迅速下降。值得注意的是，这些基线方法在 SciQ 数据集上取得了不错的结果，这可能是因为该数据集的内容在维基百科中被很好地表示，因此被采样的 K_0 所捕捉。然而，它们在其他任务上的性能下降，突显出依赖于维基百科采样的 K_0 的局限性。我们的方法直接结合了门控机制，与从维基百科得出的近似值相比，提供了更精确和有效的方法。有关更多模型的结果，请参见附录 11。

4.3 扩大量化编辑事实的数量

我们进一步研究了 NeuralDB 在应用于大量知识时的可扩展性。为了获得足够大的事实集用于此研究，我们使用了 ZsRE 数据集的训练集进行模型编辑。Llama3 8B (Instruct) 模型的结果在表 2 中展示。这些结果表明，随着被编辑事实的数量从 10,000 增加到 100,000，NeuralDB 的性能保持高度稳定，仅观察到轻微的下降。在模型的一般能力评估中，我们发现将编辑事实的数量扩大到 100,000 并没有损害一般能力的表现，并且在基准数据集中有 0.7% 的提升。这强调了我们的框架的优越可扩展性。

Table 2: 在对极大规模事实集进行编辑时，NeuralDB 在 Llama 3 (8B) 上的编辑准确性和后编辑模型的整体性能。

Number of edits	0k	10k	20k	30k	40k	50k	60k	70k	80k	90k	100k
Efficacy ($\uparrow$)	37.0	96.9	96.6	96.6	96.4	96.1	96.0	95.9	95.8	95.6	95.5
Generalization ($\uparrow$)	36.3	91.4	91.4	91.2	91.0	90.7	90.6	90.6	90.5	90.4	90.2
Specificity ($\uparrow$)	31.9	35.1	35.3	35.2	35.2	35.2	35.2	35.2	35.1	35.1	35.1
MMLU ⁷ ($\uparrow$)	56.2	56.2	56.2	56.2	56.2	56.2	56.2	56.9	56.9	56.9	56.9

⁷The MMLU results are evaluated by the AlphaEdit project rather than the lm-evaluation-harness. Although they use different metrics, both sets of results reflect the general capabilities of the LLMs.

5 相关工作

5.1 通过参数修改进行知识编辑

定位与编辑 L & E 范式源于随意的轨迹实验 [?]，表明 Transformer 模型的事实记忆主要与 FFN 层相关 [?]，因此可以使用联想记忆来理解 Transformers [?]。ROME [?] 被提出用于通过修改目标 FFN 层的参数来编辑模型的事实记忆。MEMIT [?] 扩展了 ROME，以支持多层和批量编辑版本，允许编辑成千上万的事实知识。为了解决编辑后模型失去其通用能力的挑战 [?]，提出了几种解决方案，包括顺序编辑缓存的转储 [?]、零空间投影 [?]、权重正则化 [?] 和奇异值正则化 [?]。

超网络 KE [?] 训练一个轻量级的 biLSTM-MLP 编辑器，将单样本梯度转化为一个低秩权重变化。MEND [?] 将两层梯度分解为秩为 1 的向量，并通过一个共享的 MLP 处理，从而实现节省内存的批量编辑。InstructEdit [?] 在前面添加 '[Task][Description][Input]' 提示，以便梯度自我聚类，使低秩编辑器能够在不同的 OOD 任务中执行可靠的更新。所有这些方法都需要利用大型数据集对超网络进行额外的微调，从而产生相当大的计算开销。

5.2 无需参数修改的知识编辑

外部模块 使用外部内存缓存的这一系列工作始于 SERAC [?]，该方法通过加入显式检索内存、范围分类器和轻量级反事实 Seq2Seq 生成器来增强一个冻结的模型，从而通过查找而非梯度更新来应用编辑。T-Patcher [?] 在每个错误的最终 FFN 层注入一个稀疏的“键-值-偏置”三元组，确保每个修补程序仅在其预期触发器上激活。GRACE [?] 将错误的隐藏状态存储为动态码本中的离散键，每当当前激活落入一个 ϵ -球内时，其值会覆盖所选择的 Transformer 层，从而实现毫秒级、成千上万次编辑的热修复，并附加 $< 0.4\%$ 的额外参数。MELO [?] 同样使用一个隐藏状态索引的数据库来按需激活低秩、每次编辑的适配器块。MindBridge [?] 将每次编辑编码为一个独立的“记忆模态”，该模态可以在未来模型版本之间共享，从而在升级基础 LLM 时防止知识丢失。类似地，这些方法的外部模块需要大量数据集用于微调，这导致了高昂的成本。

基于提示的方法 最近的研究利用提示工程来促进高效的知识编辑。例如，MemPrompt [?] 和 IKE [?] 将更新后的知识嵌入提示中以利用上下文学习。对于多跳问答任务，MQAKE [?] 引入了一个基准来评估知识编辑性能。MeLlo [?] 将编辑后的事实存储在外部，并迭代性地提示模型生成与更新一致的答案。PokeMQA [?] 通过提示分解多跳问题以提高检索和答案准确性。RAE [?] 检索精炼的事实并通过使用知识图谱的上下文学习增强语言模型。为了应对多语言知识编辑，ReMaKE [?] 将新检索的多语言知识整合到提示中以更好地适应。

6 结论

在本文中，我们介绍了 NeuralDB，这是一种可扩展的知识编辑框架，旨在从编辑后的事实中构建一个神经 KV 数据库，并使用非线性门控函数将其集成到 LLM 中的目标 FFN 层中。此集成确保了 LLM 的通用能力得以保留。该神经数据库被设计为易于维护，便于在模型中高效地添加和修改编辑后的事实。我们在各种 LLM 中进行了全面的实验以验证我们框架的有效性。我们在 ZsRE 和 CounterFacts 数据集上的结果，利用 GPT2-XL、GPT-J (6B) 和 Llama-3 (8B)，表明 NeuralDB 的编辑可以有效修改数十万条事实而不降低生成文本的质量。此外，我们在六个通用文本理解和生成任务中的发现证实了我们的方法保留了与目标编辑事实无关的 LLM 的通用能力。这些结果突出显示了 NeuralDB 编辑的稳健性和可扩展性，使其成为提高 LLM 在各种应用中适应性和准确性的有价值工具。

7 局限性

尽管 NeuralDB 能有效编辑成千上万条知识，但相应的存储开销随着编辑条目数量的增加呈线性增长。这可能导致额外的内存使用，尤其是在编辑大量知识时。例如，为 Llama3 8B 编辑 100,000 条知识后，额外的内存使用量达到了原模型的 20 %。

8 边界影响

随着信息在社会中迅速演变，大型语言模型也必须快速适应。预训练一个新的大型语言模型需要大量的时间和资源。知识编辑使模型能够更新信息并增强其特定领域的知识。我们的方法显著扩展了在不损害模型性能的情况下可进行的编辑数量的上限。这一进展有潜力推动知识编辑的更广泛应用。

更广泛的影响包括对社会的积极贡献。通过提高知识更新的效率，模型可以更迅速地适应不断变化的环境和信息。这不仅有利于研究和教育，还在医疗保健和司法等关键领域提供最新的信息支持，促进科学和及时的决策。此外，知识编辑的普及将鼓励跨学科合作，使来自不同领域的专家更有效地共享和整合知识，以解决复杂的社会问题。

随着知识编辑能力的扩展，解决潜在的伦理和社会影响是势在必行的。确保信息的准确性和可信性对于防止错误信息的传播至关重要。此外，透明的编辑过程和可追溯性对于建立公众信任和确保技术的负责任使用至关重要。所提出的 NeuralDB 编辑框架明确从更新的知识中得出，促进了解释性编辑。因此，我们的研究不仅着眼于技术进步，还强调了这些创新的更广泛应用和社会影响。

在本节中，我们展示了在我们工作中评估的六种基线方法。对于每种方法，我们采用相应论文官方代码中提供的默认超参数设置。

在本节中，我们描述了在实验中使用的数据集和评估指标。我们在两种类型的数据集上评估我们的方法：CounterFact 和 ZsRE 用于评估 KE，以及包括 SciQ、MMLU、CommonsenseQA、ARC Challenge、WSC273 和 LAMBADA 在内的六个基准，用于评估后编辑模型的通用能力。这里，我们介绍了为 CounterFact 和 ZsRE 数据集选择的评估指标，这些指标基于之前的工作选取。

给定一个语言模型 f ，一个查询 (s_i, r_i) ，一个被编辑的对象 $\hat{o}_i$ ，以及原始对象 o_i ，CounterFact 的评估指标定义如下。

- 有效性（编辑事实的成功）：在提示 (s_i, r_i) 时，模型更倾向于编辑后的对象 $\hat{o}_i$ 而非原始对象 o_i 的实例比例：

$$\mathbb{E}_i [\mathbb{P}_f [\hat{o}_i | (s_i, r_i)] > \mathbb{P}_f [o_i | (s_i, r_i)]] . \quad (13)$$

- 泛化（释义事实的成功）：模型为 $\hat{o}_i$ 而非 o_i 分配更高可能性的释义提示 $F_i(s_i, r_i)$ 的比例，这些释义提示是对原始查询 (s_i, r_i) 的改写：

$$\mathbb{E}_i [\mathbb{P}_f [\hat{o}_i | F_i(s_i, r_i)] > \mathbb{P}_f [o_i | F_i(s_i, r_i)]] . \quad (14)$$

- 特异性（邻域事实的成功）：模型对正确对象 o_i 赋予的可能性高于对编辑对象 $\hat{o}_i$ 的可能性的邻域提示 $N_i(s_i, r_i)$ 的比例。这些提示涉及与原始主体 s_i 在语义上相关但不同的主体：

$$\mathbb{E}_i [\mathbb{P}_f [\hat{o}_i | N_i(s_i, r_i)] < \mathbb{P}_f [o_i | N_i(s_i, r_i)]] . \quad (15)$$

- 流畅性（生成熵）：基于模型输出中 n -gram 分布的熵来衡量输出的重复性。具体来说，它计算了二元组和三元组熵的加权组合，其中 $g_n(\cdot)$ 表示 n -gram 频率分布：

$$-\frac{2}{3} \sum_k g_2(k) \log_2 g_2(k) + \frac{4}{3} \sum_k g_3(k) \log_2 g_3(k) . \quad (16)$$

- 一致性（参考分数）：一致性通过以主题 s 提示模型 f ，并计算其生成输出的 TF-IDF 向量与关于对象 o 的参考维基百科段落的余弦相似度来衡量。

给定一个语言模型 f ，一个查询 (s_i, r_i) ，一个编辑对象 $\hat{o}_i$ ，以及一个原始对象 o_i ，ZsRE 的评价指标定义如下：

- 效能（编辑事实的成功）：在编辑样本上的 Top-1 准确率，测量在给定提示 (s_i, r_i) 的情况下，模型将编辑对象 $\hat{o}_i$ 排为最可能预测的比例：

$$\mathbb{E}_i \left[\hat{o}_i = \arg \max_o \mathbb{P}_f (o | (s_i, r_i)) \right] \quad (17)$$

- 泛化（释义事实的成功）：在释义提示 $F_i(s_i, r_i)$ 上的 Top-1 准确率，这些提示是原始查询 (s_i, r_i) 的重述，衡量模型在给定的重述提示中将编辑对象 $\hat{o}_i$ 排名为最可能预测的情况比例：

$$\mathbb{E}_i \left[\hat{o}_i = \arg \max_o \mathbb{P}_f (o | F_i(s_i, r_i)) \right] \quad (18)$$

- 特异性 (邻域事实的成功率): 在邻域提示 $N_i(s_i, r_i)$ 上的 Top-1 准确率, 这些提示涉及与 s_i 相关但不同的主题。特异性反映了模型是否在不受影响的输入上保持正确预测, 仍然优先选择 o_i 而不是 $\hat{o}_i$:

$$\mathbb{E}_i \left[o_i = \arg \max_o \mathbb{P}_f(o \mid N_i(s_i, r_i)) \right] \quad (19)$$

我们提供了实验实现的详细信息。要重现我们的方法, 需要一个具有 40G 内存的 GPU。我们的框架是基于 L & E 方法构建的, 例如 MEMIT [?] 和 AlphaEdit [?]。我们首先顺序计算目标编辑事实的键向量 $\mathbf{k}$ 和残差向量 $\mathbf{r}$ 。计算的详细信息可以在附录 ?? 中找到。然后, 我们将它们堆叠为键矩阵 $\mathbf{K}_1$ 和残差矩阵 $\mathbf{R}_1$, 并构建一个神经 KV 数据库 $(\mathbf{K}_1, \mathbf{R}_1)$ 。接着, 我们将非线性检索模块与目标 FFN 层 l^* 集成。我们的模块仅涉及一个超参数 γ 来控制门机制。我们在以下表格 3 中提供了设置的详细信息。

Table 3: 主要实验中各种模型的 NeuralDB 超参数

	GPT2-xl	GPT-J (6B)	Llama 3 Instruct (8B)
Layer found by casual trace	17	17	17
Layer l^* used by NeuralDB	17	8	7
γ	0.65	0.65	0.65

我们遵循之前的定位和编辑方法 [???], 从给定的编辑事实 $(s_i, r_i, o_i \rightarrow \hat{o}_i)$ 中导出关键向量和残差向量。让 l^* 表示需要更新的 FFN 层。

对于关键向量 $\mathbf{k}_i$, 我们从 LLM 推理提示中检索指定的激活。我们将 $\mathbf{k}^{l^*}(\mathbf{x})$ 表示为层 l^* 中提示 $\mathbf{x}$ 的关键激活。然后目标关键向量通过以下对随机前缀 $\mathbf{x}_j$ 的平均计算得到:

$$\mathbf{k}_i = \frac{1}{N} \sum_{j=1}^N \mathbf{k}^{l^*}(\mathbf{x}_j + \mathbf{s}_i), \quad (20)$$

, 其中 $\mathbf{s}_i$ 是编辑后的事实主体。 $\mathbf{x}_j$ 是由语言模型 f 随机生成的前缀, 用以提高 $\mathbf{k}_i$ 表达能力的鲁棒性。

对于目标向量 $\mathbf{r}_i$, 我们希望找到一个向量来解码新的答案 $\hat{o}_i$ 。我们利用监督学习来导出 $\mathbf{r}_i = \arg \min_{\mathbf{r}} L(\mathbf{r})$, 其中损失对象 $L(\mathbf{r})$ 定义如下:

$$\frac{1}{N} \sum_{j=1}^N (-\log \mathbb{P}_{f(\mathbf{h}^{l^*} + \mathbf{r})}[o^* | \mathbf{x}_j + p] + D_{\text{KL}}(\mathbb{P}_{f(\mathbf{h}^{l^*} + \mathbf{r})}[x | p'] \| \mathbb{P}_f[x | p'])). \quad (21)$$

p 是事实提示, 而 p' 是其变体 (形式为 “主题 是 a”)。 $f(\mathbf{h}^{l^*} + \mathbf{r})$ 指的是用一个附加的可学习参数 $\mathbf{r}$ 替换 i -th MLP 的输出。此优化还使用随机前缀文本 $\mathbf{x}_j$ 来增强鲁棒性。

9 额外的内存使用和计算时间

在本节中, 我们将详细讨论我们新模块的附加资源。

我们缓存关键矩阵 $\mathbf{K}_1$ 和残差矩阵 $\mathbf{R}_1$, 并构建新模块, 该模块总共使用了 $(d_1 + d_2) \times m$ 个参数, 其中 m 表示已编辑事实的数量。对于 Llama 3 8B 模型, 给定 $d_1 = 14,336$, $d_2 = 4,096$, 存储 10,000 个事实大约需要 1.5 亿个参数。与总共 8B 个参数相比, 额外存储 100 万个事实仅需 2.2%。此外, 我们的存储随着事实数量线性增长, 且能够轻松扩展以存储更多的事实。

我们在表格 4 中报告了三个模型和两个数据集的平均评估时间。结果显示, 与没有额外模块的方法相比, 平均时间仅有略微改善。

10 消融研究

Table 4: 在 CounterFacts 和 ZsRE 上评估后编辑模型的平均时间

Model	Llama3			GPTJ-6B		
Method	MEMIT	AlphaEdit	NeuralDB	MEMIT	AlphaEdit	NeuralDB
CounterFacts	4.12	4.11	4.18	3.81	3.76	3.90
ZsRE	0.22	0.22	0.22	0.16	0.16	0.17

Algorithm 1 新的多层方法

Require: Input Transformer model f , target layers list $L = [l_1, \dots, l_n]$, request facts $\mathcal{F}$, Function COMPUTE_KEY to compute the keys of facts at layer l , COMPUTE_RESIDUAL compute the residual of facts at layer l .

- 1: **for** l in L **do**
- 2: $K_i \leftarrow \text{COMPUTE_KEY}(f, \mathcal{F}, l)$
- 3: $R_i \leftarrow \text{COMPUTE_RESIDUAL}(f, \mathcal{F}, l)$
- 4: Perform KE at layer l with (K_i, R_i)
- 5: **end for**

10.1 层选择的影响和参数 γ 的设置

我们进行了一项消融研究，以研究超参数的选择，包括 γ 和目标层，结果见表 ??。对于表 1 中显示的配置，其中 $\gamma = 0.65$ 和 $L = 7$ 用于 Llama3，我们将 γ 在 0.55 到 0.75 之间变化，并在 8 到 9 之间变化。虽然第 8 层由因果追踪确定，但我们的结果表明其结果不是次优的。增加 γ 可以提高特异性，同时损害生成过程，这与门控机制一致。总体而言，结果表明有必要搜索最佳的超参数。

Table 5: 不同层设置下的编辑性能

Model	Layer Setup	Efficacy	Generalization	Specificity	Fluency
GPT-J	[8] baseline	99.08	93.48	75.52	620.53
	[6,7,8] new multi layers	94.44	91.72	75.93	617.44
	[6,7,8] old multi layers	99.31	93.23	76.78	616.00
GPT2-XL	[17] baseline	99.04	95.96	70.72	621.90
	[15,16,17] new multi layers	94.81	92.68	70.26	618.51
	[15,16,17] old multi layers	99.08	94.01	71.33	624.48

我们还提出了两种用于多层更新的算法，并在表格 5 中比较了这两种算法在两个不同预训练模型上的性能。实验结果表明，尽管会有一些性能损失，但新的多层方法可以大幅度扩展可编辑事实的数量，而旧的多层方法虽然在编辑准确性上表现更好，但需要显著更多的存储。

11 附加实验**11.1 语言模型评估难度**

我们使用 GPT2-XL 和 GPT-J 进行了更多关于 lm-evaluation-hardness 的实验，详细信息见表 6。

11.2 释义和邻近事实的加权得分可视化

我们进一步在重述的事实和邻域事实上进行实验，以检查在 MEMIT 和 AlphaEdit 下加权分数的分布。如图 4 所示，重述事实中的正样本分数始终较高，而负样本的分数仍然接近 0。

Table 6: 在不同模型-算法配置下, 各任务在不同编辑预算 (1,000、2,000、4,000、6,000、8,000、10,000) 下的表现。

(a) GPT-J (AlphaEdit)

Task	1,000	2,000	4,000	6,000	8,000	10,000
sciq	0.9110	0.9080	0.9060	0.8900	0.8850	0.7410
logiq_a	0.2151	0.2243	0.2089	0.2181	0.2304	0.2181
commonsense_qa	0.2080	0.2146	0.2048	0.1884	0.1925	0.1785
arc_easy	0.6658	0.6477	0.6326	0.6010	0.5804	0.4870
MMLU	0.2660	0.2688	0.2622	0.2592	0.2587	0.2535
arc_challenge	0.3276	0.3148	0.2901	0.2782	0.2611	0.2261
lambada	0.6722	0.6604	0.6057	0.5158	0.4036	0.2203
winogrande	0.6346	0.6227	0.6093	0.5991	0.5730	0.5635
wsc273	0.8425	0.8352	0.7985	0.7399	0.7179	0.6264

(b) GPT-J (NeuralDB)

Task	1,000	2,000	4,000	6,000	8,000	10,000
sciq	0.9160	0.9160	0.9160	0.9160	0.9160	0.9160
logiq_a	0.2120	0.2120	0.2120	0.2120	0.2120	0.2120
commonsense_qa	0.2080	0.2080	0.2080	0.2080	0.2080	0.2080
arc_easy	0.6692	0.6692	0.6692	0.6692	0.6692	0.6692
MMLU	0.2695	0.2697	0.2697	0.2695	0.2695	0.2698
arc_challenge	0.3396	0.3396	0.3404	0.3404	0.3404	0.3404
lambada	0.6829	0.6827	0.6827	0.6821	0.6821	0.6819
winogrande	0.6409	0.6417	0.6417	0.6417	0.6417	0.6409
wsc273	0.8242	0.8242	0.8242	0.8242	0.8242	0.8242

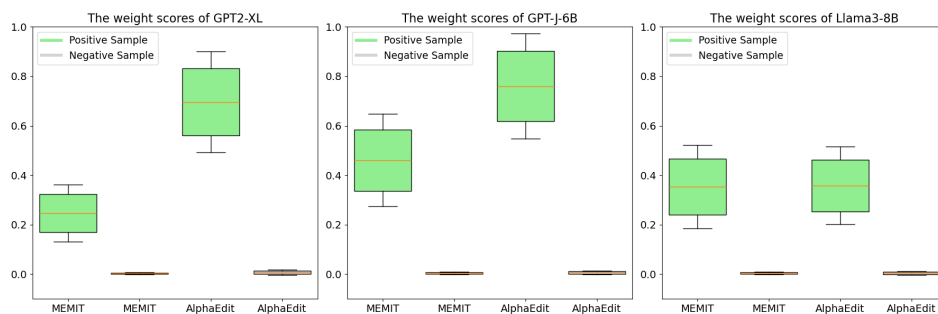
(c) GPT-2 XL (AlphaEdit)

Task	1,000	2,000	4,000	6,000	8,000	10,000
sciq	0.8250	0.8230	0.7920	0.7440	0.6390	0.4920
logiq_a	0.2289	0.2273	0.2012	0.2104	0.1951	0.1889
commonsense_qa	0.1908	0.1957	0.1916	0.1974	0.2080	0.1933
arc_easy	0.5682	0.5484	0.4987	0.4693	0.4066	0.3493
MMLU	0.2618	0.2562	0.2464	0.2312	0.2369	0.2315
arc_challenge	0.2423	0.2346	0.2398	0.2108	0.1887	0.2065
lambada	0.4881	0.4170	0.2610	0.1467	0.0767	0.0231
winogrande	0.5904	0.5549	0.5564	0.5272	0.5201	0.5067
wsc273	0.6520	0.6227	0.5714	0.5861	0.5678	0.5421

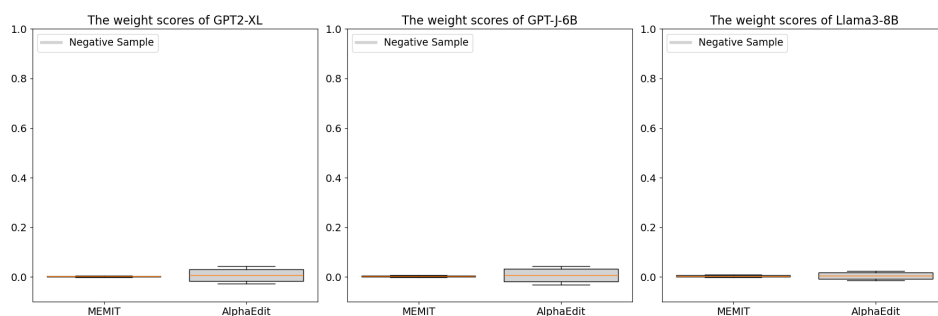
(d) GPT-2 XL (NeuralDB)

Task	1,000	2,000	4,000	6,000	8,000	10,000
sciq	0.8240	0.8290	0.8280	0.8280	0.8280	0.8280
logiq_a	0.2212	0.2181	0.2181	0.2181	0.2181	0.2197
commonsense_qa	0.1900	0.1933	0.1941	0.1941	0.1941	0.1941
arc_easy	0.5770	0.5785	0.5848	0.5848	0.5848	0.5848
MMLU	0.2532	0.2545	0.2547	0.2546	0.2543	0.2544
arc_challenge	0.2509	0.2509	0.2491	0.2500	0.2500	0.2517
lambada	0.5055	0.5053	0.5077	0.5069	0.5065	0.5053
winogrande	0.5770	0.5785	0.5848	0.5848	0.5848	0.5848
wsc273	0.6777	0.6667	0.6850	0.6850	0.6850	0.6850

对于邻域事实, 其中所有组件均视为负数, 分数同样始终接近 0。这些结果证实, 在推理过程中, 与编辑事实无关的残差保持不活跃, 导致加权分数接近于零。



(a) 改述的事实



(b) 邻域事实

Figure 4: 使用 MEMIT 和 AlphaEdit 在三个模型中可视化转述事实和邻域事实的加权得分。箱线图由权重得分的均值和方差生成，中心线表示均值，箱体显示 ± 1 标准差，须显示 ± 1.5 倍标准差。