

SetKE: 知识元素重叠的知识编辑

Yifan Wei¹, Xiaoyan Yu^{2†}, Ran Song³, Hao Peng¹ and Angsheng Li^{1†}

¹State Key Laboratory of CCSE, School of Computer Science and Engineering, Beihang University

²Beijing Institute of Technology, ³Kunming University of Science and Technology

{ weiyifan, angsheng } @buaa.edu.cn, xiaoyan.yu@bit.edu.cn, song_ransr@163.com

Abstract

大型语言模型 (LLMs) 在检索和问答等任务中表现出色，但需要更新以整合新知识并减少错误和幻觉。传统的更新方法，如微调和增量学习，面临过拟合和高计算成本等挑战。知识编辑 (KE) 提供了一个有前景的替代方案，但通常忽视了知识元素重叠 (KEO) 现象，其中多个三元组共享共同元素，导致编辑冲突。我们识别了现有知识编辑数据集中 KEO 的普遍性，并表明其对当前知识编辑方法产生了显著影响，导致处理此类三元组的性能下降。为了解决这一问题，我们提出了一种新的公式化，即知识集编辑 (KSE)，并引入 SetKE，一种同时编辑成套三元组的方法。实验结果表明，SetKE 在主流大型语言模型的 KEO 场景中胜过现有方法。此外，我们介绍了 EditSet，一个包含 KEO 三元组的数据集，提供了一个全面的基准。

1 介绍

大型语言模型 (LLMs) 作为强大的知识库，在执行信息检索和问答等任务时表现出色 [Petroni et al., 2019; Geva et al., 2020]。然而，事实信息的动态性要求不断更新以防止部署后的不准确和幻觉。虽然参数高效微调和增量学习技术为更新 LLMs 中的知识提供了方法，但这些范式可能导致潜在的缺陷，如过拟合和显著的计算成本。为应对这些挑战，知识编辑 (KE) 的概念应运而生，该概念涉及在不进行广泛再训练的情况下直接修改 LLMs 中的特定知识，成为一个关注的焦点 [Dong et al., 2022; Wei et al., 2023; Gupta et al., 2023; Song et al., 2024; Yao et al., 2024]。

当前关于知识编辑的研究将大语言模型中的事实知识形式化为三元组 (s, r, o) ，由主体 s 、客体 o 和它们的关系 r 组成。这些研究可以分为三种主要方法：1) 传统编辑专注于修改单个事实知识条目 [Sinitin et al., 2019; De Cao et al., 2021; Rawat et al., 2021; Dai et al., 2022] 的能力。他们只对模型参数进行单一修改，且没有后续变化。2) 顺序编辑涉及逐一迭代更新知识三元组 [Hartvigsen et al., 2024; Hu et al., 2024; Yu et al., 2024; Wang et al., 2024a]。他们支持流编辑，确保知识更新连续而有序。3) 批量编辑支持同时编辑多个知识实例，

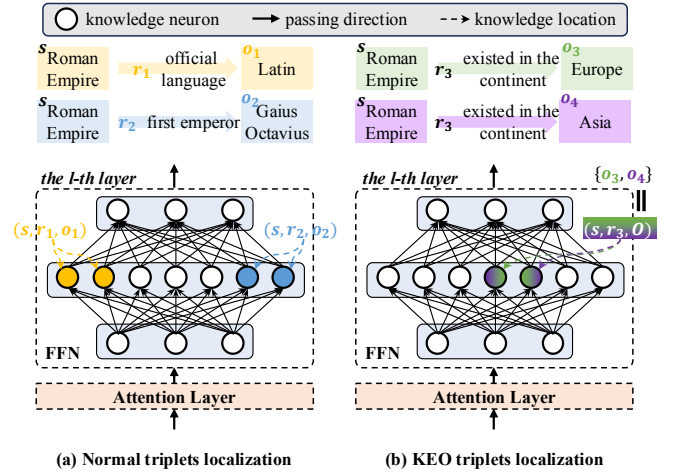


Figure 1: 正常三元组和 KEO 三元组的演示，以及一个展示它们如何在基于 Transformer 的 LLM 中定位的简单示例，其中正常三元组被映射到不同的神经元，而 KEO 三元组被映射到重叠的神经元。

每个实例被分配一个唯一的编辑目标 (例如 $o \rightarrow o^*$) [Mitchell et al., 2022; Tan et al., 2023; Li et al., 2024]。总体而言，当前方法集中于知识前缀 $t_r(s)$ (由主体 s 和关系 r 组成) 对应于单个客体的情况，将其视为编辑的单一目标。如图 1 (a) 所示，这两个三元组具有不同的关系和客体，导致不同的前缀 $t_{r_1}(s)$ 和 $t_{r_2}(s)$ ，当前方法可以有效处理。

在现实世界的场景中，一个主体通常对应于多个在同一个关系下的客体。例如，如图 1 (b) 所示，罗马帝国跨越了多个大陆，导致了共享相同主体和关系但客体不同的三元组。我们称之为知识元素重叠 (KEO) 三元组。在这种情况下，编辑知识前缀 $t_{r_3}(s) = t_{r_4}(s)$ ，例如“罗马帝国存在于哪个大陆？”对应了多个有效答案，包括欧洲和亚洲。目前的方法常常忽视这种重叠，这存在问题，因为在同一个事实背景下共享元素的知识三元组需要同时考虑，以确保在知识编辑期间的一致性。

为了进一步研究这种忽视的后果，我们分析现有主流的知识编辑 (KE) 数据集，以评估知识元素重叠 (KEO) 的普遍性，并发现其广泛存在。然后，我们将包含 KEO 的实例与不包含的实例分开，并进行实验以评估当前 KE 方法在这些实例上的表现。如预期，结果表明，包含 KEO 的实例表现显著下降。进一步分析揭示了具有重叠

[†] Corresponding Authors.

元素的三元组在 FFN 层中共享知识神经元, 如图 1 (b) 所示, 这导致编辑者无意中覆盖了知识。例如, 修改欧洲为非洲可能会无意中导致欧洲和亚洲都被更改为非洲。基于此观察, 我们从 Wikidata 中收集 KEO 实例, 以构建一个新的数据集 EditSet, 允许更全面地探讨 KE 中的 KEO。数据集包含 700 多种关系类型, 我们的研究集中在 31 种最常见的类型上, 与之前的研究一致 [Levy et al., 2017; Elazar et al., 2021; Meng et al., 2022a; Zhong et al., 2023; Wei et al., 2024; Yin et al., 2024; Ma et al., 2024]。由于当前方法未能有效处理 KEO 情况, 本文介绍了一种名为 Set Knowledge Editors (SetKE) 的编辑框架, 该框架采用二分匹配进行优化, 以实现涉及重叠知识元素的场景中的知识编辑。实验结果表明, SetKE 在 KEO 场景中显著优于现有的 KE 方法。此外, 深入分析证实 SetKE 有效缓解了知识覆盖问题。我们的主要贡献总结如下:

- 我们提出了一种新颖的知识集合编辑 (KSE) 形式化, 并构建了一个新数据集 EditSet, 以促进对知识元素重叠 (KEO) 的深入探索。
- 我们引入了一种新的集合编辑框架——集合知识编辑器 (SetKE), 通过二分图匹配实现编辑目标, 将知识条目视为集合。

- 大量实验表明, SetKE 在 KEO 场景中显著优于现有方法, 奠定了它作为编辑知识集合的强大范式。

2 预备知识

本节概述了知识元素重叠 (KEO) 的定义、当前知识编辑任务的表达方式以及我们对知识集编辑的表达方式。

2.1 知识元素重叠

一个事实知识条目 $\mathcal{K}$ 可以表示为一个三元组 (s, r, o) , 其中知识元素指的是主体 s , 客体 o , 以及它们之间的关系 r 。基于定义为共享相同的 s 、 o 或 r 的知识元素间的重叠程度, 知识三元组可以分类为四种类型 (受 Sui et al. [2023] 的分类启发): 除了正常类型之外, 所有其他类型都被视为知识元素重叠 (KEO) 的实例。

2.2 知识编辑

广泛采用的知识编辑 [Geva et al., 2021, 2022] 公式表示知识 $\mathcal{K}$ 以三元组 (s, r, o) 的形式存储在语言模型 f_θ 中。知识编辑的目标是将模型 f_θ 修改为 f_θ^* , 以实现转换 $(s, r, o) \rightarrow (s, r, o^*)$, 其中输入 $x = t_r(s)$ 与其编辑后的输出 $y = o^*$ 相关联。这里, $t_r(s)$, 被称为知识前缀, 是用于描述与主题 s 的关系 r 的模板。

然而, KEO 在现实场景中很常见, 给当前使用知识前缀 $t_r(s)$ 来编辑单个对象的 KE 框架带来了挑战。在这种情况下, KEO 的 RSO 类型会影响编辑结果, 因为其他类型会生成不同的前缀 $t_r(s)$ 。因此, 在本文中, KEO 通常指 RSO 类型。在这种情况下, 多个三元组共享相同的前缀 $t_r(s)$, 表示为 $O = \{o_1, o_2, \dots, o_N\}$, 其中 N 表示不同对象的数量。当前的 KE 表述未能区分以下场景: $(s, r, o) \rightarrow (s, r, o^*)$ 和 $(s, r, O) \rightarrow (s, r, O^*)$ 。在后者情况下, 修改单个对象通常不足以达到预期结果。因此, 现有的 KE 方法表现不佳, 如在第 3 节中所证明的。这凸显出需要一个新的表述来解决这些局限性。

2.3 我们的表述: 知识集编辑

在我们新的知识集编辑 (KSE) 表述中, 编辑目标被定义为 $(s, r, O) \rightarrow (s, r, O^*)$, 其中对象是

一组实体 $O^* = \{o_1^*, o_2^*, \dots, o_N^*\}$ 。在 KEO 场景中, 如 $\langle \text{Roman Empire, continent, Europe} \rangle$ 和 $\langle \text{Roman Empire, continent, Asia} \rangle$, 表明罗马帝国横跨多个大陆, 编辑这些知识 (变换 $O \rightarrow O^*$) 需要集合操作。例如, 考虑当前的值 $s = \text{Roman Empire}$ 、 $r = \text{existed in the continent}$ 和 $O = \{\text{Europe, Asia}\}$ 。为了将 Europe 修改为 Africa, 更可靠的方式是执行 $(O = \{\text{Europe, Asia}\}) \rightarrow (O^* = \{\text{Africa, Asia}\})$ 而不是 $(o_1 = \text{Europe}) \rightarrow (o_1^* = \text{Africa})$ 。这凸显出需要基于集合的操作, 而不是像以往方法中那样修改单个对象。我们提出的表述通过考量 KEO 场景满足了这一需求。

评估指标 新的 KSE 公式的评估标准与以前的作品 (其中对象是单数) 保持一致 [Meng et al., 2022a, b]。这些标准从以下角度评估编辑性能: 有效性衡量后编辑模型 f_θ^* 是否为目标编辑生成预期的预测, 这通过有效性得分 (ES) 量化; 泛化能力衡量后编辑模型 f_θ^* 在等效输入中泛化编辑的能力, 通过泛化得分 (GS) 评估; 局部性衡量后编辑模型 f_θ^* 是否保留对编辑范围外输入 (即未编辑输入) 的原始预测, 这通过局部性得分 (LS) 评估。较高的分数表明在各自维度上的更好性能。请注意, 指标是基于每个实例中提供的提示进行评估的。有关详细的得分计算, 请参见附录 A.5。

3 试点分析 & 数据集构建

我们对现有的 KE 数据集进行初步分析, 以检查 KEO 实例的普遍性并评估其对当前 KE 方法的影响。根据这些发现, 我们构建了一个新的 KE 数据集以进一步研究 KEO 问题。

我们分析并量化了以下 KE 数据集中具有 KEO 的实例。

- zsRE [Levy et al., 2017] 是一个最普遍的问答数据集, 由 De Cao et al. [2021]; Mitchell et al. [2021, 2022] 扩展并用于 KE 评估。

- ParaRel [Elazar et al., 2021] 是一个由专家精心策划的数据集, 包含针对从 T-REx 数据集 [Elsahar et al., 2018] 获取的 38 个关系的不同提示模板。

- MQuAKE-t 和 MQuAKE-cf [Zhong et al., 2023] 是从 Wikidata 构建的, 用于评估 KE 方法在多跳问题上的有效性。

- CounterFact [Meng et al., 2022a] 来源于 ParaRel, 每个样本包括一个知识三元组和精心制作的提示模板。

表 1 总结了上述知识增强 (KE) 数据集中知识三元组中 KEO 的分布, 显示了 KEO 在所有数据集中都存在。在分析的数据集中, zsRE 数据集正常三元组的比例最高, 占 80.66%。相比之下, 在数据集 ParaRel 和 CounterFact 中, 正常类型分别仅占 7.29% 和 5.92%。此外, 数据集 MQuAKE-t 和 MQuAKE-cf 不仅显示了 KEO 问题, 还包含大量重复的知识三元组, 样本量分别为 1772 和 3486。这些统计数据突显了当前主流 KE 数据集中知识元素重叠的普遍性。然而, 这一问题的实际影响尚未得到充分考虑。

KEO 的影响 考虑到之前的编辑器主要是以正常类型设置为重点设计的, 忽视了 KEO 的存在, 我们使用 CounterFact 数据集的 500 个正常三元组和 500 个 KEO 三元组进行评估, 以研究 KEO 对编辑器性能的影响。正如实验结果所示于图 2, 当处理 KEO 类型的知识时, 与正常类型知识相比, 当前的 SOTA 编辑器表现

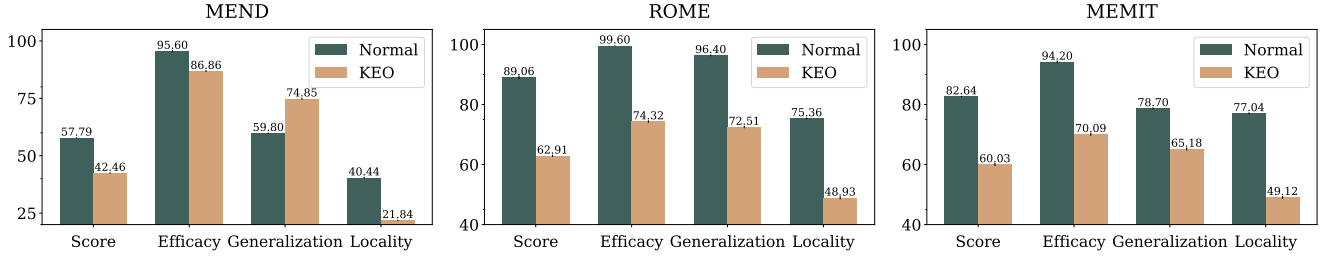


Figure 2: MEND、ROME 和 MEMIT 在 Normal 和 KEO 类型下的编辑性能比较。

Table 1: 当前主流知识图谱数据集的统计数据。注意，一个知识三元组可以属于不同的类别。重复项 (Dup.) 是完全相同的知识三元组。比例计算所有实例中正常 (Norm.) 实例的比例。

Dataset	Norm.	RSO	ROO	SOO	Dup.	Ratio	ALL
zsRE	8,066	0	902	1,103	0	80.66	10,000
ParaRel	2,023	2,242	23,241	742	21	7.29	27,738
MQuAKE-t	79	0	3	14	1,772	4.23	1,868
MQuAKE-cf	1,748	28	3,970	0	3,486	18.96	9,218
CounterFact	592	315	9,376	10	11	5.92	10,000

出较低的有效性 (MEND 的泛化指标除外)。进一步的分析揭示了性能差异来自于将知识处理为储存在模型前馈网络 (FFN) 中的键值对。当编辑具有元素重叠的知识时，这些方法往往导致知识覆盖，其中与编辑目标关联的值向量被覆盖。结果是，这导致了有效性和泛化性能的下降。此外，知识更新的连锁效应 [Cohen et al., 2024] 也导致了 KEO 实例局部性性能的下降。总之，上述观察强调了现有方法在处理包含 KEO 的知识时存在不足，呼吁从新的角度解决这些挑战。

为了进一步探索提升具有 KEO 实例的 KE，我们引入了一个全新的数据集 EditSet，该数据集专为 KEO 样本设计，以提供一个集中的评估环境。

数据收集 所有知识三元组都从 Wikidata 中收集，Wikidata 是一个包含与数百万实体相关的事实三元组的知识库。为了收集 KEO 实例，我们首先从 Wikidata 中抽取共享共同关系属性的主-宾对。随后，我们利用 Wikidata 的转存来提取数据，识别出 710 个特定关系，仅保留那些与给定事实陈述关联的具有多个对象的实例 $t_r(s)$ 。使用这种方法，收集了知识三元组。最后，GPT-4 被用于根据知识三元组生成评估提示，包括反事实、释义和邻域提示（有关提示的详细信息，请参见附录 A.1）。

EditSet 的统计数据 EditSet 数据集包含总共 710 个关系。此外，我们构建了一个子集，包括 31 个与 CounterFact [Meng et al., 2022a] 和 ParaRel [Elazar et al., 2021] 数据集相交的最常用关系。在 EditSet 中，每个数据样本都与 N 对象相关联，即在每个知识三元组（单个数据实例）中，主语 s 和关系 r 都是唯一的，同时它们对应于 N 个对象 o 。该子集总共包括 40,904 个实例。关于该子集的详细统计信息，包括主体和对象的总数，详见表格 2。反事实提示用于评估效能，复述提示用于泛化，邻域提示用于局部性。

数据集专门为 KSE 设计，旨在支持探索涉及 KEO 实例的 KE。

Table 2: EditSet 数据集在 31 个常用关系上的统计数据。EditSet 数据集包括三种类型的提示，Counter.P.、Para.P. 和 Neigh.P. 分别表示反事实提示、释义提示和邻域提示。每种提示对应一个包含 N 个对象的事实知识陈述。

Overlap	N=3	N=4	N=5	N=6	N=7	N>=8	Total
Subjects	21,256	9,203	4,389	2,161	1,160	682	35,301
Relations	31	28	28	22	22	19	31
Objects	18,497	13,985	10,570	7,891	6,019	4,444	26,144
Counter.P.	22,770	9,574	4,503	2,193	1,175	687	40,900
Para.P.	43,164	18,410	8,674	4,219	2,249	1,325	78,031
Neigh.P.	3,780	2,768	2,240	1,757	1,509	1,221	3,988

4 SetKE: 集合知识编辑器

本节详细介绍了针对 KEO 问题量身打造的建议集知识编辑器 (SetKE)。传统的 KE 方法尚未考虑 KEO，因此无法解决这一问题，正如在第 3 节经验性展示的那样。对此，我们抛开先前工作在 KE 任务中确立的模式，从全新的视角来解决这个问题。具体来说，我们在 KSE 中的目标是将集合 O 转变为新集合 O^* ，以确保模型成功被编辑。这可以视为对模型预测（实际输出集合）的编辑（或对齐），使其与真实结果（编辑后输出）相匹配，而在当前集合和目标集合之间建立映射。受集合预测任务 [Sun et al., 2021] 启发，我们旨在约束和优化此映射以实现所需的转换。为此，我们采用二分图匹配找到最佳对应关系，匹配使用匈牙利算法计算。然后根据优化结果更新模型权重，完成 KE 过程，成功编辑模型。图 3 展示了从集合视角编辑模型的简化框架。

4.1 二部匹配约束

基于前面提出的背景和考虑因素，我们使用二分图匹配问题来优化编辑 KEO 三元组。在二分图匹配中，给定一组预测和一组编辑目标，目标是找到这两组之间的最优匹配，以最大化整体匹配分数，同时最小化总匹配成本。在此基础上，KSE 问题定义如下：给定一组编辑目标 $y = \{y_j\}_{j=1}^M$ 和一组模型预测对象 $\hat{y} = \{\hat{y}_j\}_{j=1}^N$ ，其中 $M < N$ ，并使用占位符 ($\emptyset$) 来填充未匹配的预测，任务是找到一个最优匹配 $\hat{\pi}$ ，使得总匹配成本最小化，由以下公式确定：

$$\hat{\pi} = \arg \min_{\pi \in \Pi_N} \sum_{j=1}^N C_{\text{match}}(y_j, \hat{y}_{\pi(j)}), \quad (1)$$

，其中 $\hat{y}_{\pi(j)}$ 是 y_j 在匹配 π 下的匹配， Π_N 表示所有匹配的集合。 $C_{\text{match}}(y_j, \hat{y}_{\pi(j)})$ 是成对匹配成本，在这项工作中，定义为：

$$C_{\text{match}}(y_j, \hat{y}_{\pi(j)}) = -\mathbb{1}_{\{y_j \neq \emptyset\}} P_{\pi(j)}(y_j), \quad (2)$$

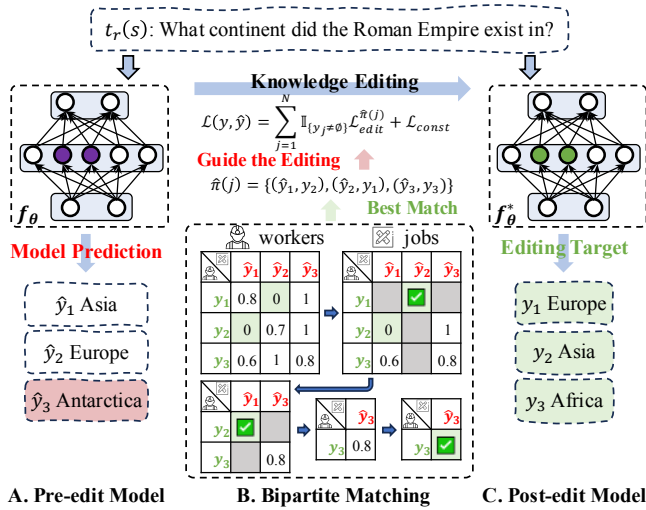


Figure 3: SetKE 框架的简化说明。

，其中 $P_{\pi(j)}(y_j)$ 表示模型预测对象 $\hat{y}_{\pi(j)}$ 正确匹配编辑目标 y_j 的概率。

基于这个二分图匹配问题，我们采用匈牙利算法 [Kuhn, 1955] 来寻找最佳匹配。图 3 B 展示了使用匈牙利算法获得的最佳匹配的一个简单示例。该算法将真实物体集合 $\{y_j\}_{j=1}^M$ 视为一组工人，而预测物体集合 $\{\hat{y}_j\}_{j=1}^N$ 视为一组工作。在一个填充了 0 的矩阵 $A^{N \times N}$ 中，每个元素 A_{ij} 表示将工作 j 分配给工人 i 的成本。首先，该算法通过从每行所有元素中减去该行的最小值来缩减矩阵，确保每个工人的最低成本最小化。然后，通过从每列所有元素中减去该列的最小值来进一步简化矩阵并减少总体成本值。接下来，当前值为零的元素表示潜在的最佳分配。该算法将这些零赋予形成最佳匹配的一部分，并迭代调整矩阵，直到所有工人分配到总成本最小的工作。匈牙利算法保证以 $O(N^3)$ 时间复杂度找到最佳匹配，如附录算法 1 中所示。

一旦找到最佳匹配（或分配） $\hat{\pi}$ ，我们定义总体损失为：

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{j=1}^N \mathbf{1}_{\{y_j \neq \emptyset\}} \mathcal{L}_{\text{edit}}^{\hat{\pi}(j)} + \mathcal{L}_{\text{const}}, \quad (3)$$

，其中 $\mathcal{L}_{\text{edit}}^{\hat{\pi}(j)}$ 是衡量编辑成功的编辑目标损失， $\mathcal{L}_{\text{const}}$ 是优化编辑局部性指标的约束损失。整体过程针对每个真实值 y_j 及其对应的预测 $\hat{y}_{\hat{\pi}(j)}$ ，计算如下：

$$\begin{aligned} \mathcal{L}_{\text{edit}}^{\hat{\pi}(j)} &= -\log P_{\theta}(h_i^l + \delta_i) [y_{ij} | x_i]_{j=\hat{\pi}(j)}, \\ \mathcal{L}_{\text{const}} &= D_{\text{KL}}(P_{\theta}(h_i^l + \delta_i) [y' | x'] \| P_{\theta} [y' | x']). \end{aligned} \quad (4)$$

第二项 $\mathcal{L}_{\text{const}}$ 最小化输入 x' （形式为“主题 is a”）的预测与未改变模型之间的 KL 散度，这有助于保持模型对主体本质的理解。直观上，当模型成功更新其输出时， $\mathcal{L}_{\text{edit}}$ 很小，而当编辑不影响不相关的输入时， $\mathcal{L}_{\text{const}}$ 很小。

现有的知识编辑方法 [Meng et al., 2022b; Li et al., 2024] 仅对模型的特定部分（包含待编辑的知识的部分）进行优化，以避免影响非编辑目标。本研究遵循这一查

找然后编辑的范式。根据第 4.1 节中介绍的优化策略，编辑过程如下：

步骤 1：定位编辑组件。当前的知识编辑工作 [Meng et al., 2022a; Li et al., 2024] 已表明，知识通常存在于特定的模型层中，例如基于 Transformer 的模型中的前馈网络 (FFN)。因此，编辑通常发生在这些层内。在此步骤中，我们建立在前面的工作 [Meng et al., 2022a] 上，以追踪需要编辑的组件的位置。

步骤 2：执行编辑。给定追踪到的编辑组件，特别是在模型 f_{θ} 中执行编辑的层 l ，和给定的编辑提示 x_i （描述一组 KEO 三元组 (s_i, r_i) 的主题和关系），我们在提示通过第 l 个 FFN 层后计算隐藏状态 h_i^l ，如下所示：

$$h_i^l = \text{FFN}^l(x_i) = \sigma(x_i \cdot W_{fc}^l) \cdot W_{proj}^l. \quad (5)$$

由于目标是编辑特定知识而不影响模型的其他部分，我们引入一个残差向量 δ_i 来调整模型的输出，并使用该向量进行优化。这个残差使得知识能够从原始值 v_i^l 转移到编辑值 z_i ，从而实现所需的编辑：

$$z_i = v_i^l + \delta_i = h_i^l + \arg \min_{\delta_i} \mathcal{L}(\delta_i). \quad (6)$$

步骤 3：在多个层之间传播编辑。为了最小化副作用（如非预期的修改或覆盖）和减少过多的修改，我们将残差向量 δ_i 分布到模型的多个层中，而不是将其应用于单一层。这遵循之前的工作 [Meng et al., 2022b; Li et al., 2024]，通过均匀地在关键层之间传播更新来实现：

$$r_i^l = \frac{\delta_i}{L - l + 1}, \quad \mathcal{R}^l \triangleq [r_1^l | r_2^l | \dots | r_n^l]. \quad (7)$$

步骤 4：权重更新。最后，通过将增量权重 Δ^l 加到原始权重上来更新模型的权重矩阵 W_{proj}^l ，计算如下：

$$\Delta^l = \mathcal{R}^l K^{l\top} (C^l + K^l K^{l\top})^{-1}, \quad (8)$$

其中 $\hat{W}_{proj}^l = W_{proj}^l + \Delta^l$ 表示更新的权重， K^l 表示新的键， $C^l = K_0^l K_0^{l\top}$ 是通过采样获得的先前键集的估计值， $\mathcal{R}^l$ 是旧值向量和新值向量之间的残差。

5 实验

在本节中，我们评估了基线和 SetKE 关于 EditSet 关于 KEO 知识的编辑性能。具体来说，我们的目标是回答以下问题：Q1：与其他基线相比，SetKE 在 EditSet 上的表现如何？Q2：KEO 重叠的数量如何影响知识编辑方法？Q3：KEO 现象如何导致知识重写问题？Q4：二部匹配约束如何对结果产生贡献？

5.1 实验设置

数据集 编辑过程是利用提出的数据集的一个子集 EditSet 进行的，该子集包含 31 个常用关系。关于这个子集的具体统计数据和详细配置请参阅第 ?? 节。在这个数据集中，所有 KEO 类型都是 RSO 类型，每个实例都表示为 $\{s, r, O = \{o_1, o_2, \dots\}\}$ ，与编辑对象不是单一的目标相一致。我们采用两种广泛使用的自回归语言模型，即 GPT2-Large (760M)、GPT2-XL (1.5B) 和 GPT-J (6B) [Radford et al., 2019]，作为基础语言模型进行编辑并评估 KE 方法的有效性。我们选择了以下方法：FT-W 是一个基本的微调方法。KN [Dai et al., 2022] 利用知识归因来实现知识更新。MEND [Mitchell

Table 3: 在 EditSet 上的数值结果, 涉及 10,000 次编辑 (括号内为 95 % 置信区间)。

GPT2 Large (760M)					GPT2 XL (1.5B)			
Editor	Score	Efficacy	Generalization	Locality	Score	Efficacy	Generalization	Locality
FT-W	51.46	58.13 (0.5)	45.29 (0.5)	52.58 (0.4)	55.18	65.66 (0.5)	50.97 (0.5)	51.24 (0.4)
KN	42.21	38.22 (0.5)	37.46 (0.5)	54.89 (0.4)	40.65	35.93 (0.5)	35.86 (0.5)	55.29 (0.4)
MEND	—	—	—	—	38.75	89.58 (0.3)	81.58 (0.4)	18.52 (0.3)
PMET	43.34	40.82 (0.5)	39.24 (0.5)	51.98 (0.4)	44.13	41.15 (0.5)	40.08 (0.5)	53.39 (0.4)
MEMIT	49.65	49.53 (0.5)	45.64 (0.5)	54.59 (0.4)	56.60	61.12 (0.5)	55.07 (0.5)	54.11 (0.4)
ROME	64.08	72.03 (0.4)	69.63 (0.4)	53.84 (0.4)	65.89	75.27 (0.4)	73.58 (0.4)	53.62 (0.4)
SetKE	71.47	88.57 (0.3)	80.91 (0.4)	54.77 (0.4)	75.28	95.90 (0.2)	91.68 (0.2)	54.14 (0.4)

et al., 2021] 使用梯度的低秩分解学习新的知识。ROME [Meng et al., 2022a] 首先应用因果中介分析来更新参数。MEMIT [Meng et al., 2022b] 扩展 ROME 以编辑大量事实。PMET [Li et al., 2024] 扩展 MEMIT 以同时优化 FFN 和注意模块的隐藏状态。

表格 3 展示了在 EditSet 中关于 GPT2-Large (760M) 和 GPT2-XL (1.5B) 的 10,000 个案例的数值结果。在此实验中, 我们与最新的基线方法以及我们提出的方法 SetKE 进行了比较。我们观察到 SetKE 在不使用任何额外参数的情况下明显优于现有的编辑方法。具体来说, 我们在大多数情况下超越了最新的高级批量编辑基线 MEMIT, 在 GPT2-Large 上在效能和泛化指标方面分别提高了最多 39.04 % 和 35.27 %。在 GPT2-XL 上的相同实验条件下, SetKE 在得分、效能和泛化指标方面也分别实现了 9.39 %、20.63 % 和 18.1 % 的性能提升, 相对最新技术方法, 这表明我们的方法在准确改变模型行为以编辑重叠的事实知识方面而不相互干扰中是有效的。

图 ?? 展示了在不同方法编辑的 GPT2-XL 上, 编辑器性能随知识三元组重叠数量变化的表现。我们分析了共计 3000 个 KEO 案例, 从这些结果可以看出, 随着重叠三元组数量的增加, 所有编辑方法的性能都有所下降。图 ?? 展示了在使用不同方法编辑的 GPT2-XL 中, 编辑器性能随知识三元组重叠程度变化的表现。结果显示, 随着重叠三元组数量的增加, 所有编辑技术的性能普遍下降。具体而言, ROME 和 MEMIT 在 KEO 问题上表现出更高的敏感性, 编辑性能显著下降。另一方面, MEND 对 KEO 的影响显得相对不那么敏感。这种差异可以归因于 ROME 和 MEMIT 的知识定位方法, 这通常会在 KEO 场景中导致更多的知识覆盖实例。此外, 虽然 SetKE 也在知识定位领域运作, 但它表现出的脆弱性较 ROME 和 MEMIT 小。这表明 SetKE 中的二部匹配约束通过减少知识覆盖的发生, 提高了 KEO 案例中的编辑性能。最后, 随着重叠数量的增加, 所有方法的局部性指标继续改善。我们将这一趋势归因于 $\mathcal{L}_{\text{const}}$ 的拟合。

在本研究中, 我们对 500 个 KEO 案例进行了分析, 以调查 KEO 知识三元组是否会导致知识覆盖问题, 如图 1 (b) 所示。我们采用基于积分梯度的知识归属方法 [Dai et al., 2022], 确定模型中具体事实知识 (知识神经元) 的存储位置, 分析与 KEO 知识对应的知识神经元之间的重叠程度, 以评估与 KEO 相关的知识三元组是否可能触发知识覆盖。我们的分析结果, 如图 ?? 中的一个代表性示例所示, 揭示了在 GPT2 [Radford et al., 2019]

模型中表示三个 KEO 事实知识的知识神经元之间存在显著重叠。这一观察与我们的初步分析一致 (见图 2), 表明对这些特定知识神经元的重复修改常常导致知识覆盖, 导致编辑性能下降。

5.2 消融研究分析 (Q4)

为了评估通过二分匹配约束优化的 SetKE 的有效性, 我们将其与普通的知识集合编辑 (对象串联) 进行比较。这种设置将编辑目标视为一个由多个串联对象构成的长序列, 旨在同时优化多个对象。结果显示在表 ?? 中。表的左侧部分 (对象集) 展示了编辑器在主要实验配置下的表现, 而右侧部分则展示了普通 (对象串联) 设置的结果。在 KEO 场景中, 我们观察到同时优化多个对象通常优于对 MEND、ROME 和 MEMIT 等编辑器的逐个对象优化。我们推测这是因为同时优化多个对象有助于缓解知识覆盖的问题。在对象串联设置下, SetKE 的表现低于 ROME。然而, 所有编辑器在对象集配置中都不及 SetKE 的表现, 这突显了专为 SetKE 在集合场景中设计的二分匹配约束的有效性。

6 相关工作

知识编辑 (KE) 涉及修改语言模型以修正从预训练语料学习到的事实知识的表达。当前关于 KE 任务的研究可以分为以下三种类型: 元学习方法利用额外的可训练参数来存储记忆或学习所需的调整 (Δ), 以更新 LLM [De Cao et al., 2021; Huang et al., 2022; Tan et al., 2024; Cheng et al., 2024] 中的知识。“定位然后编辑”方法首先采用因果中介分析来定位与知识表达正相关的知识神经元, 然后进行相应修改 [Dai et al., 2022; Meng et al., 2022a,b; Huang et al., 2024a,b; Wang et al., 2024b]。“上下文编辑”方法是一种无训练范式, 其中知识编辑是通过直接在输入上下文中连接演示来实现的 [Zheng et al., 2023; Zhong et al., 2023; Qi et al., 2024]。然而, 现有的 KE 方法主要局限于修改标准三元组, 并且通常忽视了知识重叠现象。据我们所知, 本论文是首个引入一个新的数据集 EditSet, 以评估 KEO 编辑性能, 并提出了一个通用框架以减轻知识覆盖。

7 结论

在本文中, 我们分析了主流的知识编辑 (KE) 数据集, 并识别出一个广泛存在但之前被忽视的问题: 知识元素重叠 (KEO) 现象。我们还证明了现有的编辑方法不足以有效地解决这一挑战。为了解决这一局限性, 我们提

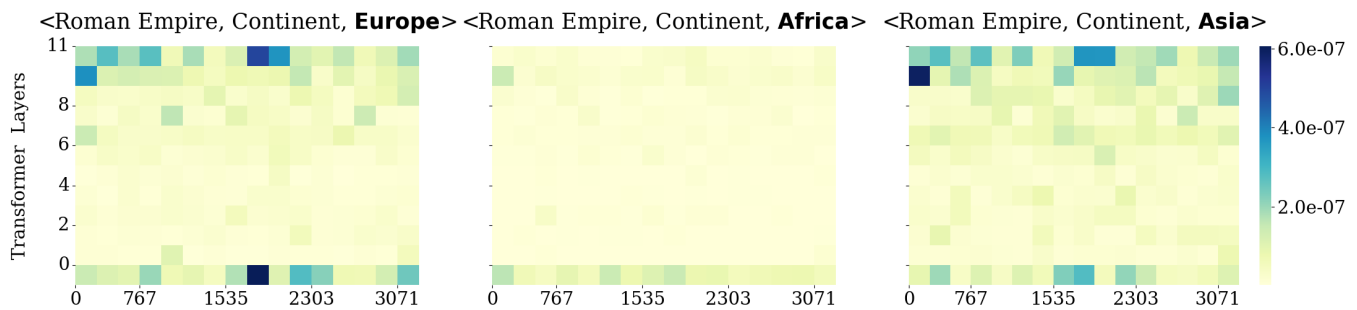


Figure 4: GPT2 上 KEO 类型知识的知识定位结果。

Table 4: GPT-J (6B) 在 EditSet 上进行 7,500 次编辑的结果 (括号中为 95 % 置信区间)。

Object Set					Object Concatenation			
Editor	Score	Efficacy	Generalization	Locality	Score	Efficacy	Generalization	Locality
FT-W	46.65	51.62 (0.5)	39.92 (0.5)	50.29 (0.4)	43.36	44.53 (0.5)	37.20 (0.5)	50.40 (0.4)
MEND	37.44	91.33 (0.3)	82.60 (0.4)	17.52 (0.3)	43.05	67.64 (0.5)	63.37 (0.5)	25.56 (0.4)
MEMIT	55.95	67.51 (0.5)	56.04 (0.5)	47.71 (0.4)	57.07	82.31 (0.4)	64.00 (0.4)	40.34 (0.4)
ROME	59.48	80.54 (0.4)	79.90 (0.4)	39.21 (0.4)	61.30	85.86 (0.3)	82.57 (0.3)	39.71 (0.4)
SetKE	73.68	90.08 (0.3)	90.76 (0.3)	53.77 (0.4)	59.50	89.43 (0.3)	68.68 (0.4)	40.52 (0.4)

出了一种新颖的公式化方法，称为知识集编辑 (KSE)，并介绍了一种专门为 KEO 场景设计的方法——SetKE。SetKE 利用二分图匹配来优化对象集编辑，有效解决编辑冲突并提高准确性。实验结果证实，SetKE 优于现有方法，在编辑多个主流 LLM 时达到了最先进的性能。此外，我们开发了 EditSet 数据集，它作为一个综合基准，用于评估 KEO 场景中的知识集编辑。

References

- Siyuan Cheng, Ningyu Zhang, Bozhong Tian, Xi Chen, Qingbin Liu, and Huajun Chen. Editing language model-based knowledge graph embeddings. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 17835–17843, 2024.
- Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. Transactions of the Association for Computational Linguistics, 11:283–298, 2024.
- Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pre-trained transformers. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8493–8502, 2022.
- Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 6491–6506, 2021.
- Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. Calibrating factual knowledge in pretrained language models. In Findings of the Association for Computational Linguistics: EMNLP 2022, pages 5937–5947, 2022.
- Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. Transactions of the Association for Computational Linguistics, 9:1012–1031, 2021.
- Hady Elsahar, Pavlos Vougiouklis, Arslan Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. T-rex: A large scale alignment of natural language with knowledge base triples. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), 2018.
- Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. arXiv preprint arXiv:2012.14913, 2020.
- Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 5484–5495, 2021.
- Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 30–45, 2022.

- Anshita Gupta, Debanjan Mondal, Akshay Sheshadri, Wenlong Zhao, Xiang Li, Sarah Wiegrefe, and Niket Tandon. Editing common sense in transformers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8214–8232, 2023.
- Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36, 2024.
- Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. *arXiv preprint arXiv:2402.10987*, 2024.
- Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*, 2022.
- Kiusheng Huang, Jiaxiang Liu, Yequan Wang, and Kang Liu. Reasons and solutions for the decline in model performance after editing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- Kiusheng Huang, Yequan Wang, Jun Zhao, and Kang Liu. Commonsense knowledge editing based on free-text in llms. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14870–14880, 2024.
- Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
- Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *21st Conference on Computational Natural Language Learning, CoNLL 2017*, pages 333–342. Association for Computational Linguistics (ACL), 2017.
- Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572, 2024.
- Jun-Yu Ma, Zhen-Hua Ling, Ningyu Zhang, and Jia-Chen Gu. Neighboring perturbations of knowledge editing on large language models. In *Forty-first International Conference on Machine Learning*, 2024.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.
- Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2022.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2021.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR, 2022.
- Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, 2019.
- Si yuan Qi, Bangcheng Yang, Kailin Jiang, Xiaobo Wang, Jiaqi Li, Yifan Zhong, Yaodong Yang, and Zilong Zheng. In-context editing: Learning knowledge from self-induced distributions. *arXiv preprint arXiv:2406.11194*, 2024.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- Ankit Singh Rawat, Chen Zhu, Daliang Li, Felix Yu, Manzil Zaheer, Sanjiv Kumar, and Srinadh Bhojanapalli. Modifying memories in transformer models. In *International Conference on Machine Learning (ICML)*, volume 2020, 2021.
- Anton Sinitin, Vsevolod Plokhhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. Editable neural networks. In *International Conference on Learning Representations*, 2019.
- Ran Song, Shizhu He, Shuting Jiang, Yantuan Xian, Shengxiang Gao, Kang Liu, and Zhengtao Yu. Does large language model contain task-specific neurons? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7101–7113, 2024.
- Dianbo Sui, Xiangrong Zeng, Yubo Chen, Kang Liu, and Jun Zhao. Joint entity and relation extraction with set prediction networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.
- Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris M Kitani. Rethinking transformer-based set prediction for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3611–3620, 2021.
- Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language model via meta learning. In *The Twelfth International Conference on Learning Representations*, 2023.
- Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language models via meta learning. In *In-*

ternational Conference on Learning Representations, 2024.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. arXiv preprint arXiv:2405.14768, 2024.

Xiaohan Wang, Shengyu Mao, Shumin Deng, Yunzhi Yao, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. Editing conceptual knowledge for large language models. In Findings of the Association for Computational Linguistics: EMNLP 2024, pages 706–724, Miami, Florida, USA, 2024. Association for Computational Linguistics.

Yifan Wei, Xiaoyan Yu, Huanhuan Ma, Fangyu Lei, Yixuan Weng, Ran Song, and Kang Liu. Assessing knowledge editing in language models via relation perspective. arXiv preprint arXiv:2311.09053, 2023.

Yifan Wei, Xiaoyan Yu, Yixuan Weng, Huanhuan Ma, Yuanzhe Zhang, Jun Zhao, and Kang Liu. Does knowledge localization hold true? surprising differences between entity and relation perspectives in language models. In Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, pages 4118–4122, 2024.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. Knowledge circuits in pretrained transformers. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024.

Xunjian Yin, Jin Jiang, Liming Yang, and Xiaojun Wan. History matters: Temporal knowledge editing in large language model. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 19413–19421, 2024.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. Melo: Enhancing model editing with neuron-indexed dynamic lora. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 19449–19457, 2024.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 4862–4876, 2023.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. Mquake: Assessing knowledge editing in language models via multi-hop questions. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 15686–15702, 2023.

A 附录

A.1 使用 GPT-4 生成问题

我们从维基数据 RSO 收集形式为三元组的事实知识。这个例子可以在提供的链接 * 找到。我们有一个属性 ID (关系 ID) 和来自维基数据的此关系的相应关系描述。使用这些信息,我们提示 GPT-4 基于属性 ID 的描述和相关实体对自动生成问题。用于此目的的提示在表格 8 和表格 9 中显示。我们展示了一个 EditSet 的示例,如图 24 所示。

```
{
  "case_id": 3587,
  "requested_rewrite": {
    "id": "P30",
    "prompt": "The {} spanned across multiple continents, including",
    "target_new": {
      "list": ["Oceania", "Antarctica", "North America"],
      "ids": ["Q55643", "Q51", "Q49"]
    },
    "target_true": {
      "list": ["Europe", "Africa", "Asia"],
      "ids": ["Q46", "Q15", "Q48"]
    },
    "subject": "Byzantine Empire"
  },
  "paraphrase_prompts": [
    "One of the continents where the Byzantine Empire existed was",
    "The continent where the Byzantine Empire was situated in was"
  ],
  "neighborhood_prompts": [
    "The Ottoman Empire spanned across multiple continents, including",
    "The Hispanic Monarchy One was located in the continent of"
  ],
  "neighborhood_ans": [
    ["Asia", "Europe", "Africa"],
    ["Europe", "Asia", "Africa", "North America", "South America"]
  ]
}
```

Figure 5: EditSet 数据集的样本形式。

A.2 实现细节

实验使用 PyTorch 框架实现,并在一台装有八个 NVIDIA GeForce RTX 3090 GPUs 的机器上运行。我们随机抽取 50 个% 和 50 个% 进行知识编辑任务的训练和测试。在评估中,我们遵循 ROME 作为编辑指标。我们将每个 KEO 案例的批量大小设置为三重态重叠数,学习率设置为 $5e-1$,并采用 Adam 优化器。我们报告 5 次重复实验的平均值和标准差。所有实现均可在 <https://anonymous.4open.science/r/SetKE-11B4> 获得。

通过匈牙利算法(如算法 1 所示),可以很容易地计算出总成本最小的最优分配。

A.3 匈牙利算法

通过匈牙利算法(如算法 1 所示),可以很容易地计算出总成本最小的最优分配。

A.4 默认位置设置

对于基于不同 LLM 骨干网的编辑任务,SetKE 的层位置可以被视为超参数。以下表格展示了我们实验中的默认位置设置(GPT2-Large 的表格 5, GPT2-XL 的表格 6)。

* https://www.wikidata.org/wiki/Property_talk:P1302

Algorithm 1 匈牙利算法

Input: A cost matrix $C \in \mathbb{R}^{N \times N}$
Output: $\hat{\pi}$
1: Step 1: Subtract row minima:
2: for each row i in C do
3: $C[i, :] = C[i, :] - \min(C[i, :])$
4: end for
5: Step 2: Subtract column minima:
6: for each column j in C do
7: $C[:, j] = C[:, j] - \min(C[:, j])$
8: end for
9: Step 3: repeat
10: Cover all zeros with a minimum number of horizontal and vertical lines
11: Find the smallest entry not covered by any line
12: Subtract this entry from all uncovered entries
13: Add this entry to all entries covered twice
14: until the number of lines equals N
15: Step 4: Find an optimal assignment among the zeros using DFS or BFS
16: return $\hat{\pi}$

Table 5: 默认设置 KE 目标模块用于 GPT2 大型

Model	Target Editing Modules
GPT2 Large	transformer.h.[1].mlp.c_proj
	transformer.h.[2].mlp.c_proj
	transformer.h.[3].mlp.c_proj
	transformer.h.[4].mlp.c_proj
	transformer.h.[5].mlp.c_proj

A.5 评估指标

我们编制了一系列具有挑战性的错误陈述 (s, r, o^*) 。这些假设情境最初比正确陈述 (s, r, o) 的排名较低。我们的有效性得分 (ES) 表示在编辑后我们观察到 $P[o^*] > P[o]$ 的情况所占的比例。为了评估泛化能力，我们为每个反事实生成与 (s, r) 等价的替代提示，并以类似于 ES 的方式计算泛化评分 (GS)。为了评估局部性，我们收集了一组相关主题 s_n ，其中 (s_n, r, o) 仍然有效。

- 有效性得分 (ES) 是指 o_i^* 在概率上超过 o_i 的情况所占的比例。请注意，提示与编辑方法在运行时看到的内容完全匹配：

$$\mathbb{E}_i [\mathbb{P}[o_i^* | p(s_i, r_i)] > \mathbb{P}[o_i | p(s_i, r_i)]] . \quad (21)$$

- 泛化得分 (GS) 是在原始陈述的改写中， o_i^* 在概率上超过 o_i 的案例比例：

$$\mathbb{E}_i [\mathbb{E}_{p \in \text{paraphrases}(s_i, r_i)} [\mathbb{P}[o_i^* | p] > \mathbb{P}[o_i | p]]] . \quad (22)$$

- 局部性分数 (LS) 是邻域提示中模型对正确事实赋予更高概率的比例：

$$\mathbb{E}_i [\mathbb{E}_{p \in \text{neighborhood}(s_i, r_i)} [\mathbb{P}[o_i^* | p] < \mathbb{P}[o_i | p]]] . \quad (23)$$

为了探讨泛化与局部性之间的取舍，我们引入了一种复合测量指标，即编辑分数 (S)，这是 ES、GS 和 LS 的调和平均数。在处理多个对象时，我们使用编辑分数的算术平均值来表示集合编辑的性能。

Table 6: 默认 SetKE 目标模块用于 GPT2 XL

Model	Target Editing Modules
GPT2 XL	transformer.h.[13].mlp.c_proj
	transformer.h.[14].mlp.c_proj
	transformer.h.[15].mlp.c_proj
	transformer.h.[16].mlp.c_proj
	transformer.h.[17].mlp.c_proj

A.6 在 CounterFact 上的表现

表 ?? 展示了在 CounterFact 数据集上的 GPT2-XL (1.5B) 超过 3000 个案例的数值结果。在这个实验中，我们与最近的基线和 SetKE 进行了比较。我们观察到 SetKE 在之前的知识编辑数据上也取得了最佳性能。

Table 7: GPT2-XL (1.5B) 在 CountFact 上的结果，涉及 3,000 次编辑 (括号中为 95 % 置信区间)。

Editor	Score	ES	GS	LS
FT	36.82	28.30 (0.4)	30.97 (0.4)	72.17 (0.3)
KN	29.76	21.57 (0.4)	24.02 (0.3)	78.06 (0.3)
MEND	41.26	35.27 (0.5)	34.40 (0.4)	65.38 (0.4)
ROME	53.57	56.67 (0.5)	52.08 (0.4)	52.22 (0.3)
PMET	46.19	44.33 (0.5)	36.47 (0.4)	66.85 (0.3)
SETKE	66.58	70.53 (0.5)	56.38 (0.4)	76.08 (0.3)

A.7 神经元激活的热图

我们旨在探讨知识覆盖问题与 KEO 三元组之间的关联。为了实现这一点，我们将知识归因方法 [Dai et al., 2022] 应用于 GPT-2。热图被用来可视化神经元激活值，其中较深的颜色代表较高的激活值。x 轴对应神经元的索引，y 轴对应它们所在的层。

通过这种方法，我们旨在识别模型中存储知识的位置。具体来说，我们发现与 KEO 三元组相关的神经元激活有很大的重叠，而涉及非 KEO 实例的神经元则没有表现出强相关性。

Table 8: 由 GPT-4 (gpt-4-turbo) 生成的问题示例。我们使用 Python 脚本过滤掉不符合格式的问题模板。

The Generation of ChatGPT :	
{	"template": "The { } extended its influence to multiple continents, including" }
{	"template": "The { } spanned across multiple continents, including" }
{	"template": "One of the continents where the { } existed was" }
{	"template": "The continent where the { } was situated in was" }
{	"template": "The { } was located in the continent of" }
{	"template": "The { } had a presence in various continents, such as" }

Table 9: 使用 GPT-4 (gpt-4-turbo) 从 Wikidata 三元组生成问题的一个例子。我们手动编写了 3 个演示作为提示以便查询 ChatGPT。

Instruction: Please help me generate 10 templates by imitating the following example:	
Input:	
subject = 'Roman Empire'	
relation = 'continent'	
object = ['Asia','Europe','Africa']	
Output:	
template_list=	['The Roman Empire spanned across multiple continents, including',
	'The Roman Empire was situated in the continents of',]
In this context, the relation string must appear in the template.	
The predicate in template is generated by relation,	
and must satisfy all objects at the same time.	
And the elements left blank at the end correspond to the object:	
Asia, Europe, Africa, which cannot appear in template.	

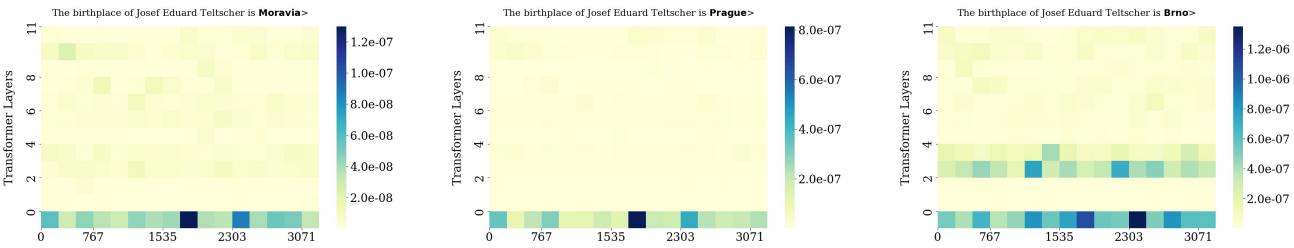


Figure 6: 使用案例 1 对知识覆盖的 分析。

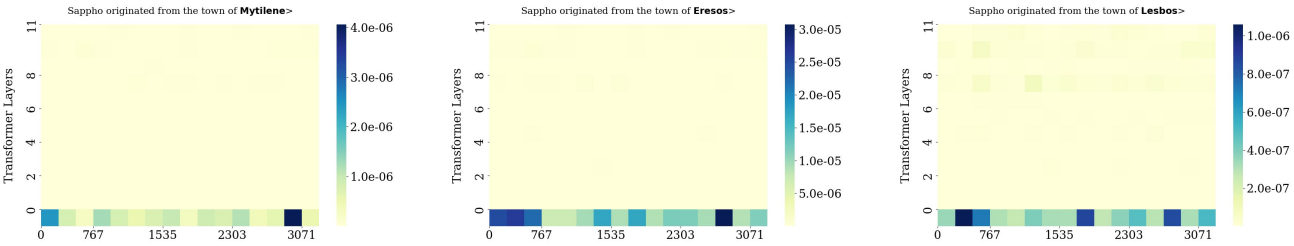


Figure 7: 用于知识重写的案例 2 分析。

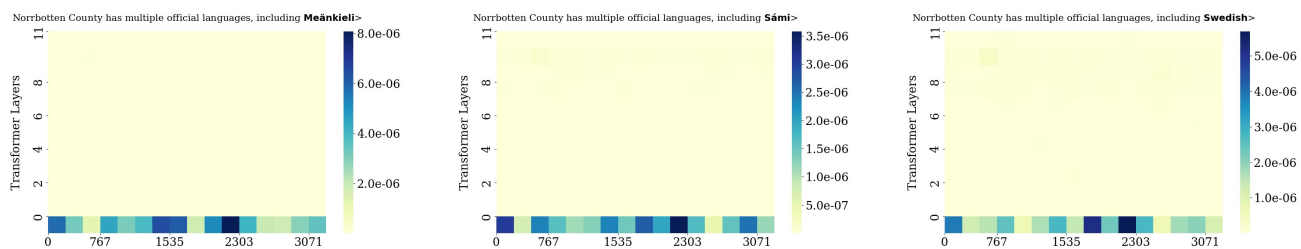


Figure 8: 使用情境 3 进行知识覆盖分析。

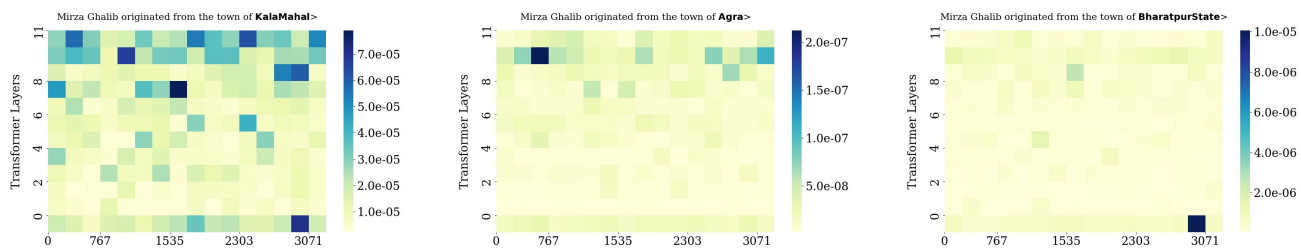


Figure 9: 使用案例 4 进行知识覆盖的分析。

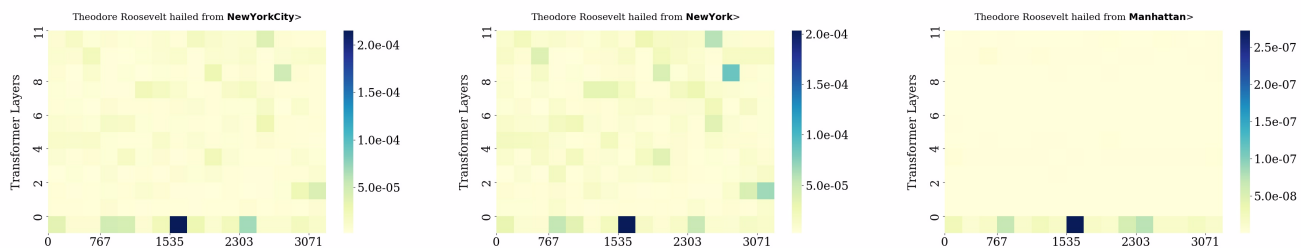


Figure 10: 分析使用情况五进行知识重写。

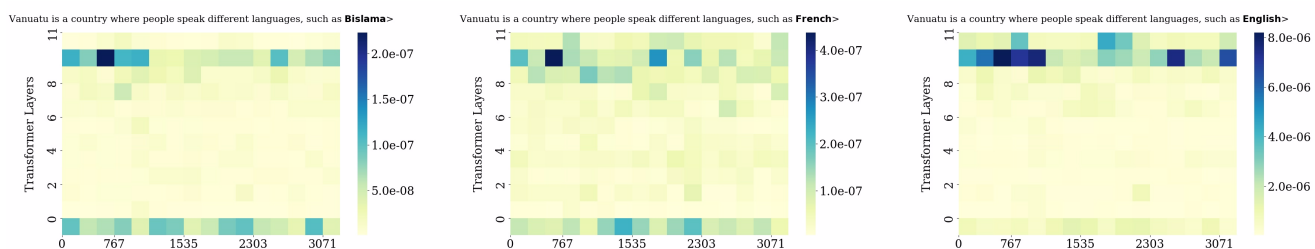


Figure 11: 对于使用案例 6 的知识覆盖分析。

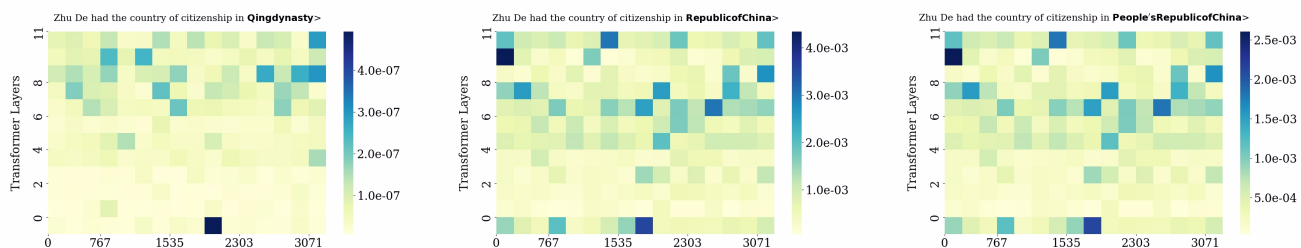


Figure 12: 分析使用案例 7 进行知识覆盖

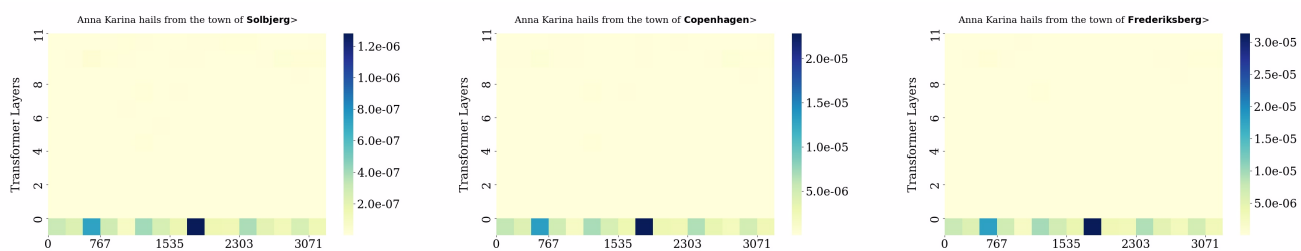


Figure 13: 使用案例 8 进行知识覆盖的分析。

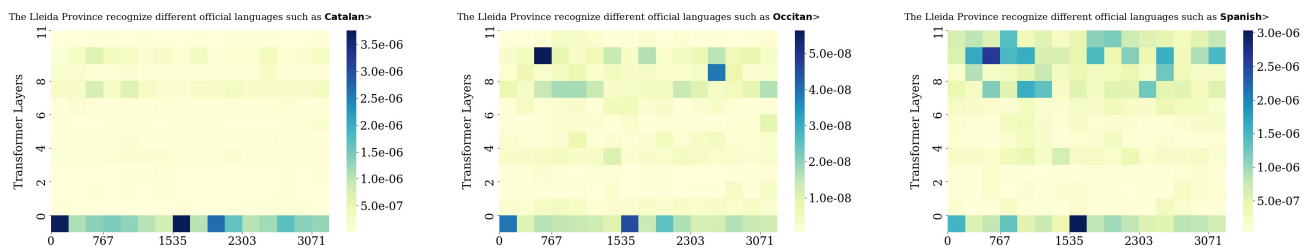


Figure 14: 使用案例 9 进行知识覆盖分析。

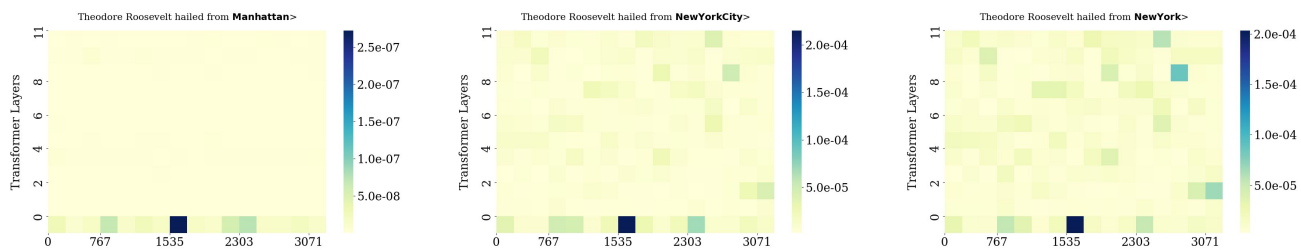


Figure 15: 使用案例 10 进行知识覆盖的分析。

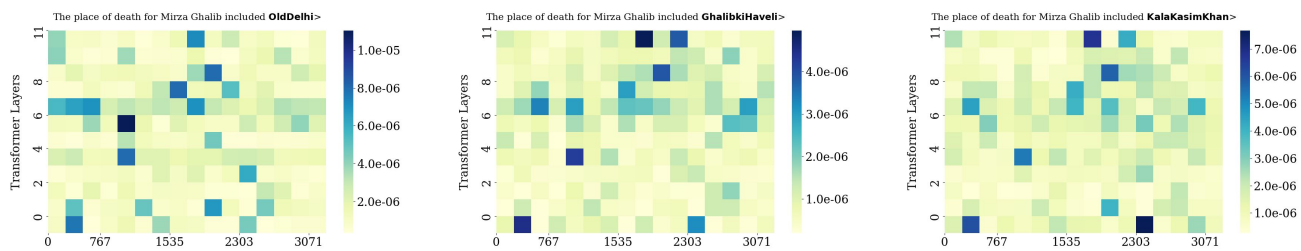


Figure 16: 使用案例 11 进行知识覆盖的分析。

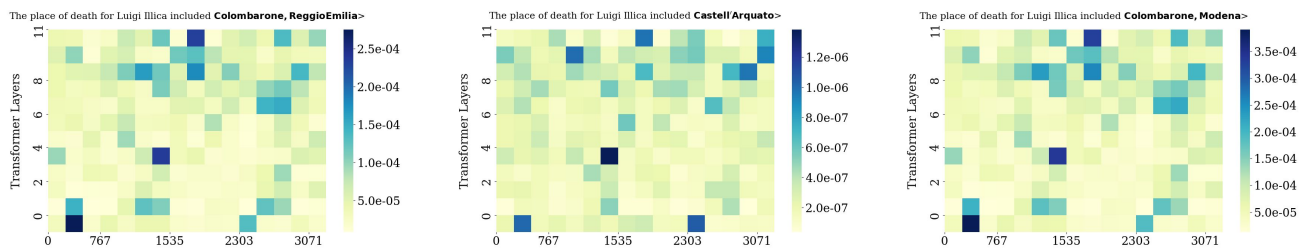


Figure 17: 使用案例 12 进行知识重写的分析。

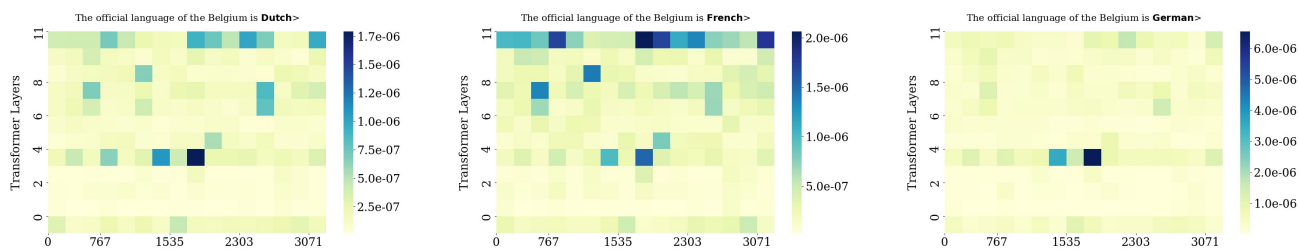


Figure 18: 使用案例 13 的知识覆盖 分析。

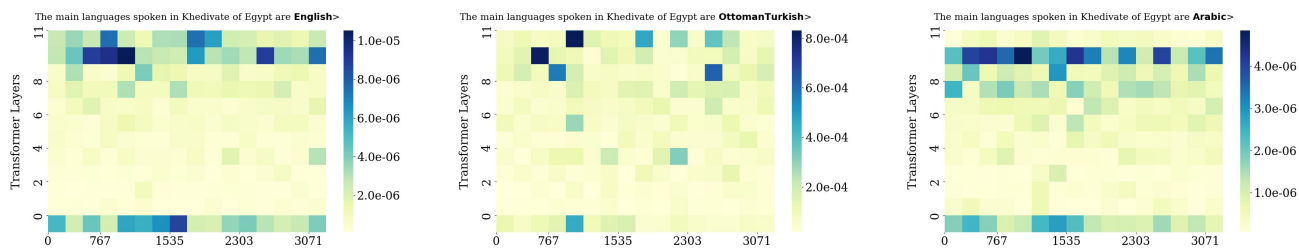


Figure 19: 使用案例 14 进行知识覆盖的 分析。

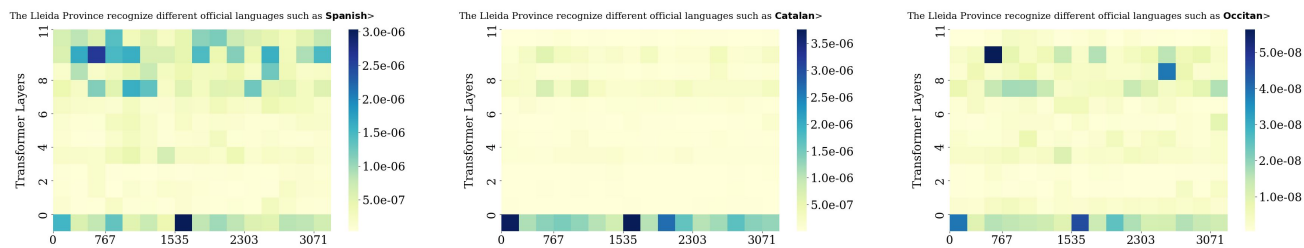


Figure 20: 使用案例 15 进行知识覆盖分析。

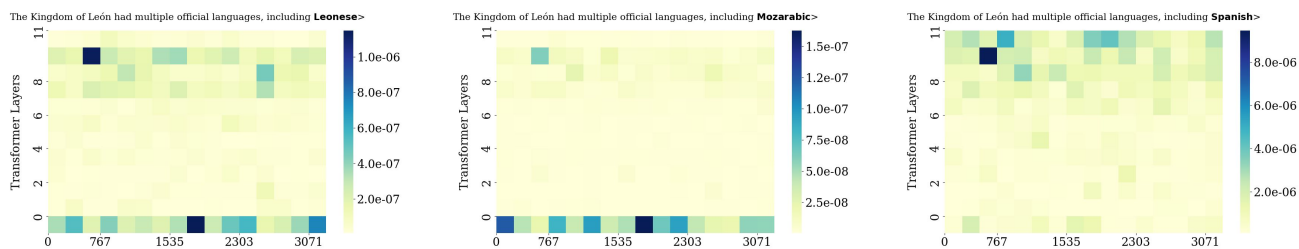


Figure 21: 使用案例 16 进行知识覆盖分析。

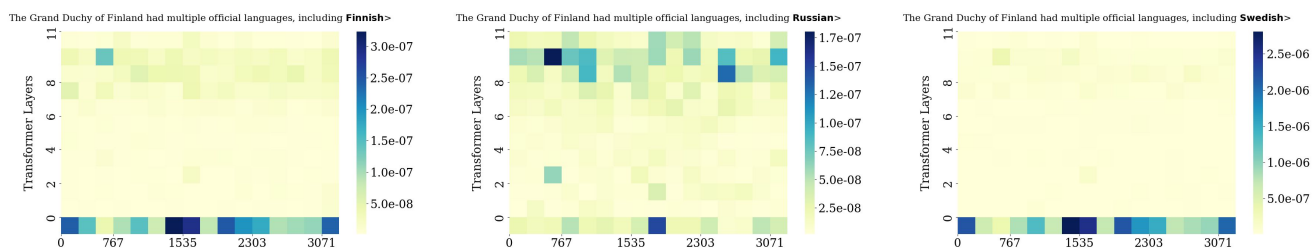


Figure 22: 分析用于案例 17 的知识覆盖。

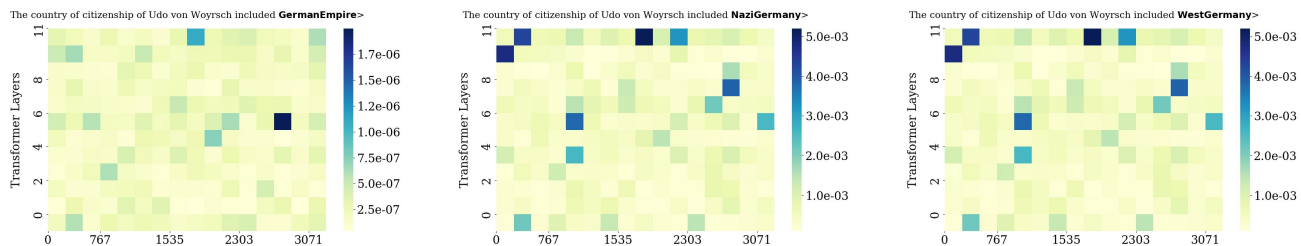


Figure 23: 使用案例 18 进行知识覆盖的分析。

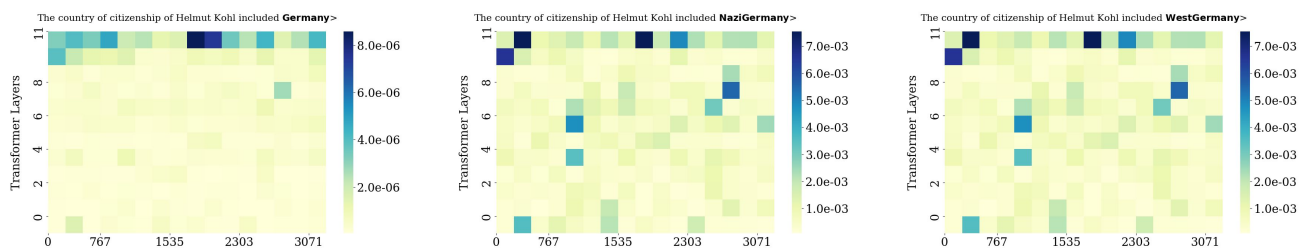


Figure 24: 对案例 19 的知识覆盖分析。

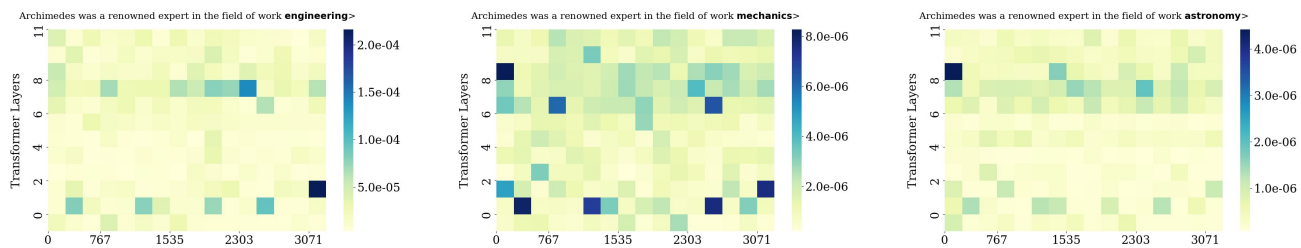


Figure 25: 使用案例 20 进行知识覆盖分析。