通过差异性学习策略缓解语言模型中的近似记忆

Aly M.Kassem温莎大学 温莎，加拿大 安大略省
kasse
m6@uwindsor.ca

arXiv:2305.01550v1 [cs.CL] 2 May 2023

摘要

大语言模型（LLM）在大量数据上进行训练，其中可能包含会危及个人隐私的敏感信息。研究表明，LLM 会记忆部分训练数据，并在对手以适当方式发出提示时逐字输出这些数据。以往研究主要聚焦于数据预处理和差分隐私技术，以解决记忆问题或单独防止逐字记忆，这可能造成一种虚假的隐私安全感。然而，这些方法依赖于关于待保护数据结构的显式和隐式假设，因此往往无法完整解决这一问题。为此，我们提出了一种新颖的框架，利用强化学习方法（PPO）对 LLM 进行微调，以缓解近似记忆。我们的方法采用负相似度分数（例如 BERTScore 或 SacreBLEU）作为奖励信号，从而学习差异性策略 π_D。实验结果表明，该框架能够有效缓解近似记忆，同时保持生成样本的高水平连贯性和流畅度。此外，我们的框架在各种情况下都能稳健地缓解近似记忆，包括更长的上下文；众所周知，更长的上下文会增加 LLM 中的记忆程度。

1 引言

大语言模型近年来呈指数级增长，参数规模从数百万增至数十亿，再到数万亿（Radford 等人，2019年；Brown 等人，2020年；Chowdhery 等人，2022年；Fedus 等人，2021年）。随着这些模型规模不断扩大，其训练集也增长到包含数十亿个词元（Gao 等人，2020年），其性能也全面提升，即使采用少样本学习设置也是如此（Brown 等人，2020年）。随着模型和数据集规模扩大，性能不断提升，人们也开始关注实际问题，包括大语言模型记忆训练数据所带来的隐私风险：与预训练模型交互的对手可以提取用于训练该模型的个别序列（Carlini 等人，2021年），即使该语言模型是在公开可访问的数据集上训练的。许多研究致力于解决大语言模型中的训练数据记忆问题；研究表明，一个拥有 60 亿个参数的语言模型（GPT-J）至少会记住其训练数据的 1%（Carlini 等人，2022年）。记忆现象的原因可能是语言模型的训练策略：语言模型的目标（Radford 等人，2018年）是识别当前词元与其后续片段（自回归语言模型）或周围片段（掩码语言模型）之间的关系。记忆现象的另一个原因可能是训练语料库中存在多次重复的样本，因为一个样本重复得越多，就越有可能被记住（Lee 等人，2021年）。

目前已有多种方法被提出，用于解决大型语言模型中的记忆问题，例如数据净化、使用差分隐私算法以及数据去重。这些技术能够有效防止生成被记忆的内容，但也存在一些缺点。例如，数据净化假设隐私信息可以轻易识别，且不依赖于上下文，而差分隐私则可能导致生成模型的质量下降（Anil 等人，2021年）。

在本研究中，我们提出一种框架，通过使用强化学习方法（PPO）对大语言模型进行微调，以防止模型发生记忆（Schulman 等人，2017年）。给定语言模型原始预训练数据中的前缀和后缀样本，我们将前缀作为输入，供语言模型生成后缀；随后，我们计算负 Sacre-BLEU（Post，2018年）分数，以衡量相似度差异

真实后缀与生成的后缀之间的相似度，随后将差异度分数视为训练过程中需要最大化的奖励信号，从而确保近似记忆得到缓解。我们尝试了不同的奖励函数，例如 BERTScore（Zhang 等人，2019年）、困惑度与 SacreBLEU 或 BERTScore 的加权和。实验旨在观察这些函数如何影响生成质量和记忆率。然而，所有提出的奖励函数都确保了近似记忆的最小化。该框架旨在学习一种策略 π_D，使其能够根据给定的前缀改写后缀。例如，在“Alice Green 住在 187 Bob 街”中，前缀是“Alice Green 住在”，后缀是“187 Bob 街”。我们的目标是让微调后的语言模型将后缀改写为：12 Red 街。随着后缀被改写，前缀与后缀之间的记忆关系会减弱。我们在两种不同设置下评估该框架的有效性。第一种设置是训练过程的标准设置：向模型提供一个前缀；我们希望生成的后缀与真实后缀尽可能不同。许多研究（Carlini 等人，2021年、2022年）表明，提供更长的上下文会提高记忆率，因此第二种设置是在前缀之前额外提供 100 个上下文词元，并将其与前缀组合后输入模型，以评估这种情况下的记忆。所提出的框架不会对待保护数据的结构作出任何显式或隐式假设。此外，与差分隐私方法不同，该框架不会应用任何划分机制将数据拆分为公开数据和私人数据；由于语言数据无法划分，我们将策略应用于所有训练数据，因为在语言数据的情况下，将数据定义或划分为私人数据和公开数据可能并不可行（Brown 等人，2022年）。

我们测试了三种模型规模（125M、1.3B 和 2.7B 个参数），所有模型都属于 GPT-Neo 系列（Black 等人，2021年）。我们的主要发现如下：

- **学习到的策略能够生成与真实后缀截然不同的新后缀**，同时不会显著降低生成质量。这是因为微调后的语言模型学习到了一种策略，可以更改名称、数字，或将整个短语替换为相似的实体。
- **随着语言模型规模的增大，收敛速度会加快**。此处的收敛是指模型生成的后缀与原始后缀显著不同，同时生成示例与原始示例之间的困惑度差异逐渐缩小。在我们的实验中，我们发现 GPT-Neo 125M 在三个训练轮次后收敛，每批次进行四个 PPO 训练轮次；GPT-Neo 1.3B 模型在两个训练轮次后收敛，每批次进行四个 PPO 训练轮次；而 GPT-Neo 2.7B 模型在两个训练轮次后收敛，每批次进行两个 PPO 训练轮次。
- **随着语言模型规模增大，差异度分数也会提高**，这一变化可以通过应用该框架前后负 SacreBLEU 的差值来衡量。然而，差异度分数的提高有时并不一定是积极结果。在某些设置下，这同时伴随着生成示例的困惑度分数上升。这表明，较大的模型可能更容易快速“遗忘”记忆数据。

总体而言，我们的研究结果表明，使用所提出的框架对大型语言模型进行微调，可以减轻对训练数据的近似记忆，同时确保后缀生成质量，且不会显著损害文本的流畅度和连贯性。

2 背景

2.1 语言模型

语言模型是自然语言处理技术的核心。它们通过接收词元序列（例如单词或字符），并输出下一个词元的概率分布来运行。训练语言模型旨在最大化给定文本语料库的似然。一种流行的语言模型训练方法是采用“下一个词元预测”目标，即构建词元序列概率分布的生成模型。最先进的语言模型使用循环神经网络或基于注意力的模型等神经网络来估计该概率分布。近年来，基于 Transformer 的语言模型（Vaswani 等人，2017年；Devlin 等人，

2018年；Radford 等人，2019年）因其能够扩展到数十亿个参数，并在大型数据集上实现高性能而变得尤为流行。实证结果表明，基于 Transformer 的语言模型的性能与模型规模、数据集规模以及训练所使用的计算资源量之间遵循幂律关系（Kaplan 等人，2020年）。这意味着，要使模型性能得到显著提升，至少需要将这些因素之一增加一个数量级。例如，将模型规模扩大 10 倍，或将数据集规模扩大 10 倍，都可能显著提升模型性能。值得注意的是，使用大型模型、大型数据集以及大量计算资源，对于实现语言模型的高性能都至关重要。然而，这些大语言模型也引发了人们对其记忆和复现敏感信息（例如个人信息或较长文本序列）潜力的担忧（Carlini 等人，2021年；Brown 等人，2022年）。在我们的实验中，我们考察了三种不同的自回归大语言模型：GPT-Neo 125M、GPT-Neo 1.3B 和 GPT-Neo 2.7B，这些模型均在英语语料库上进行了预训练。

2.2 记忆定义

在大语言模型的记忆研究中，我们遵循（Lee 等人，2021年）提出的定义，该定义引入了近似记忆。给定一个字符串 S，如果存在一个提示 P，模型在给定 P 时生成 s，并且模型输出与训练集中该提示真实后续内容之间的编辑距离达到某个选定值，则该模型输出被视为已记忆。在我们的研究中，我们采用（Ippolito 等人，2022年）提出的相似度度量（BLEU）作为编辑距离，以便捕捉近似记忆，而不只是“照相式记忆”（Carlini 等人，2021年），因为逐字记忆的定义无法涵盖更微妙的记忆形式（Ippolito 等人，2022年）。例如：假设我们有一个序列 S：“My name is Alice green, I live at 187 bob street, and my phone number is 226284”，而模型生成的内容是：“My name is Alice green, I live at 187 queen street, and my phone number is 226284”。按照“逐字记忆”的定义，这个序列并未被记忆；但按照“近似记忆”的定义，我们可以说模型生成的内容是与真实续写的相似度为 84.92% SacreBLEU 分数。

2.3 相关工作

近期研究（Ippolito 等人，2022年）表明，大语言模型可能通过对提示进行合理的“风格迁移”，来规避限制逐字记忆的过滤器。他们提出了一种推理时防御机制“MemFREE 解码”，以防止大语言模型产生记忆，从而证明了这一点。该机制通过高效查询训练数据集，并检查前缀与生成的后缀组合中是否存在任何 n 元语法来实现。另一项研究表明，训练数据中的重复样本会增加语言模型记忆这些样本的可能性。因此，数据去重技术（Kandpal 等人，2022年）能够有效地从训练数据中移除这些重复样本。然而，需要注意的是，这种方法只能部分防止模型记忆样本，因为模型仍然可能记忆重复样本之外的序列。此外，差分隐私（DP）是一种广泛用于训练不会记忆单个训练样本的模型的技术。差分隐私算法（例如（Abadi 等人，2016年）中所述的算法）被视为保护机器学习模型隐私的黄金标准。然而，正如（Anil 等人，2021年）所观察到的，在实践中，这些技术的性能通常不如非隐私模型。因此，最先进的语言模型通常需要大量数据和计算资源进行训练，并且通常不会使用差分隐私进行训练。此外，与非隐私方法相比，差分隐私算法的计算成本较高，收敛速度更慢、效用更低（Anil 等人，2021年）。这对于语言模型尤其令人担忧，因为已知语言模型会记忆其训练数据的大部分内容，并且可能更容易记忆异常数据点，从而给文本作者或相关主体的隐私带来风险。在语言数据上使用差分隐私最具挑战性的方面之一，是确定隐私信息的边界（Brown 等人，2022年）。

3 数据

我们使用了 The Pile 数据集的一个子集。该数据集作为针对大语言模型的训练数据提取攻击的基准发布。通常，Pile 数据集包含来自

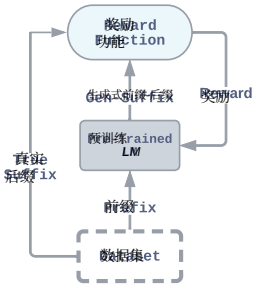


图 1：框架流程示意图，该流程可缓解语言模型中的近似记忆

各种来源的数据（例如书籍、网页抓取数据、开源代码）。我们使用了这个子集的版本[1]，，它经过专门设计，便于提取，以评估定向攻击的性能。由于完整版尚未发布，该数据集仅包含 15,000 个样本。每个样本由从 Pile 训练集中随机抽取的 200 个词元组成。该子集涵盖的主题包括代码、新闻、日志、对话-回复、版权、链接等。其中大多数内容为英语。该数据集分为 13,500 个训练样本和 1,500 个测试样本。

训练数据。在训练阶段，我们将每个字符串的第三组 50 个词元用作前缀，第四组 50 个词元用作后缀。

评估数据。我们采用了多种设置和数据集，以确保学习到的策略具有良好的泛化能力。首先，我们在 Pile 子集的测试集上评估微调后的模型：将每个字符串的第三组 50 个词元用作前缀，第四组 50 个词元用作后缀。其次，在更长上下文设置中，我们额外加入 100 个词元，并将其与样本中的第三组 50 个词元合并；前 100 个词元作为前置前缀，接下来的 50 个词元作为前缀。在语言模型中使用更长的上下文可以视为一种攻击，因为这使对手能够获取更多信息。借助这些额外信息，对手可以从语言模型中提取新的敏感信息，从而损害其安全性和完整性。

[1] https://github.com/google-research/语言模型提取基准

4 方法

我们的方法从一个自回归预训练语言模型（GPT-Neo 125M、GPT-Neo 1.3B 和 2.7B）、一个由 15,000 个样本组成且划分为前缀和后缀的数据集，以及一个奖励函数（例如负 SacreBLEU）开始，后文将对此进行讨论。具体步骤如下：将前缀输入预训练语言模型，以获得生成后缀。我们将生成后缀传入奖励函数，由其计算生成后缀与真实后缀之间的差异。最后，我们将奖励函数的输出作为标量奖励，对语言模型进行微调，并使用近端策略优化算法优化奖励。

4.1 训练环境

我们采用了基于词元生成的 MDP 环境（Ramamurthy 等人，2022年）；一般而言，该环境是一项自然语言处理任务，其中我们拥有一个由前缀和后缀组成的监督数据集。生成响应可以被视为遵循马尔可夫决策过程，其中每一步都根据当前输入和先前的动作选择一个词汇表中的词作为动作。每个回合都以特定的提示开始，并在达到一定步数（在我们的案例中，步数是指已生成的若干词元）或生成句末令牌时结束。一个回合的奖励取决于最终状态与目标输出的差异程度（例如，使用负 SacreBLEU、BERTScore 等自动评估指标）。

4.2 微调细节

我们使用 PPO 在环境中对预训练语言模型进行了微调。所用环境为生成作为令牌级马尔可夫决策过程，与老虎机环境类似：它会提供随机的客户提示，并期望模型作出回应。两种环境的唯一区别在于，老虎机环境中使用折扣因子 $\gamma = 0.95$ 而不是 $\gamma = 1$。选择 $\gamma = 0.95$ 的原因是，在训练中它能带来更高的稳定性，因为在令牌级马尔可夫决策过程中，奖励以折扣方式计算，从而降低了施加给开头所选词元的奖励幅度（Ramamurthy 等人，2022年）。给定前缀和生成的后缀后，模型会根据所选奖励函数计算奖励，并结束该回合。此外，

KL 惩罚通过参考模型按词元逐个施加（参考模型是微调前的预训练模型）。这样可以防止微调后的模型生成偏离参考语言模型过多的后缀（例如生成空格）。语言建模头旁还包含一个价值网络 V，用于估计价值函数。所有模型的批量大小均为 32；由于各模型的收敛速度不同，我们为每个模型选定了特定的训练轮次数量。这里所说的收敛，是指模型生成的后缀与原始后缀显著不同，同时困惑度没有大幅损失，也就是生成示例与原始示例之间的困惑度差异逐渐缩小。因此，我们选取了适当的训练轮次数量，以平衡这些目标。GPT-Neo 125M 采用三轮训练，每批次进行四个 PPO 训练轮次；GPT-Neo 1.3B 采用两轮训练，每批次进行四个 PPO 训练轮次；GPT-Neo 2.7B 采用两轮训练，每批次进行两个 PPO 训练轮次。所有模型的学习率均为 1.41×10^{-5}。此外，KL Beta 设为 0.2，裁剪范围设为 0.2。

4.3 奖励函数

为了有效防止训练数据被近似记忆，并鼓励模型学习多样化的语言生成策略，我们必须为预训练语言模型精心设计合适的奖励函数。该奖励函数应使模型能够生成与前缀主题相同、语义一致且句法结构正确，但彼此不相似的后缀。先前的研究表明，预训练语言模型可以通过生成全大写文本来绕过记忆检查（Ippolito 等人，2022年），因此我们的奖励函数必须解决这一问题。我们基于模型生成文本与训练数据之间的语义相似度，针对三种不同的奖励函数开展了实验。然而，我们发现，模型可能会学会通过不符合预期输出的方式来最大化奖励，例如生成空白字符，或通过学习捷径而多次重复相同的词语（Geirhos 等人，2020年）。我们选择的奖励函数必须能够应对这些潜在问题。

负 SacreBLEU。 是用于评估机器翻译文本质量的一种常用指标，是 BLEU（双语评估替补）指标的修改版本。我们的方法使用负 SacreBLEU，其计算方式为 100 与 SacreBLEU 分数之差。负 SacreBLEU 的优势在于基于语义相似度，因此适合用于学习能够最小化记忆的策略。然而，它也存在局限性：受最大 n 元语法长度限制，且不考虑语境含义，这可能会限制模型学习多样且稳健策略的能力。在以下章节中，我们将探讨解决这一局限性的方法。

负 SacreBLEU +困惑度。 为了提高模型生成文本的流畅度，我们尝试将负 SacreBLEU 分数与困惑度指标结合起来。我们发现，随着差异度分数（使用负 SacreBLEU 衡量）的增加，模型的流畅度（使用困惑度衡量）会下降。因此，我们将困惑度作为惩罚项纳入整体奖励函数，该函数通过对负 SacreBLEU 和困惑度指标的负值进行加权求和计算得出，两者的权重均为 0.5。这种指标组合使我们能够在鼓励差异性的同时，兼顾维持模型生成文本流畅度的需求。

BERTScore。 是一种基于模型的语义指标，用于评估模型生成文本的质量。它通过基于 BERT 的模型将文本转换为嵌入，并计算由此得到的两个嵌入向量之间的余弦相似度。BERTScore 根据该相似度输出召回率、精确率和 F1 分数。在我们的实验中，我们使用 DeBERTa 大型模型计算 BERTScore（He 等人，2021年），以最大化负 F1 分数；这可以理解为最小化生成后缀与真实后缀之间的余弦相似度。使用 BERTScore 的一个优点是，它依赖于上下文化嵌入，能够捕捉依赖关系并考虑句中词语的语境含义。这有助于模型学习更多样且符合语境的策略。

5 实验

为了全面评估我们的框架在防止大型语言模型记忆方面的有效性，我们开展了一系列实验。

模型	奖励函数	训练轮次	N −SacreBLEU ↑	困惑度 ↓
GPT-Neo 125M	N-SacreBLEU	12	82.78	5.25
	N-SacreBLEU + 困惑度		78.32	4.83
	BERTScore		**67.70**	**4.01**
GPT-Neo 1.3B	N-SacreBLEU	8	72.10	3.652
	N-SacreBLEU + 困惑度		52.02	2.85
	BERTScore		**53.22**	**2.64**
GPT-Neo 2.7B	N-SacreBLEU	4	71.47	6.105
	N-SacreBLEU + 困惑度		80.84	8.60
	BERTScore		**61.79**	**4.46**

表1：不同奖励函数在标准设置测试集上的结果。

在不同设置和评估指标下进行实验。我们在标准设置和更长上下文中评估了模型。评估涵盖了多种因素，包括差异度量、语义一致性和流畅度。这种综合方法使我们能够分析模型在各种场景下的性能，并了解其能力和局限性。

5.1 实验设置

为了评估我们框架的性能和灵活性，我们采用不同的设置和测试集进行了一系列实验。这些实验涵盖了模型规模、奖励函数和训练轮次数量的变化。这使我们能够分析这些因素对模型性能的影响，并确定适用于该任务的最佳配置。

模型。为了研究模型规模对我们框架的影响及其对不同规模模型的适用性，我们使用三种不同规模的模型进行了实验：125M、1.3B 和 2.7B。这些模型属于 GPT-Neo 系列，并且使用这些模型观察到了记忆问题（Carlini 等人，2022年）。在已有文献中，这些模型曾被用于多种提取或防止记忆的技术研究，因此可作为研究记忆问题的基准。GPT-Neo 模型的另一项优势是其训练集已知；它们是在公开可用的 Pile 数据集上进行预训练的，该数据集包含 825GB 数据。

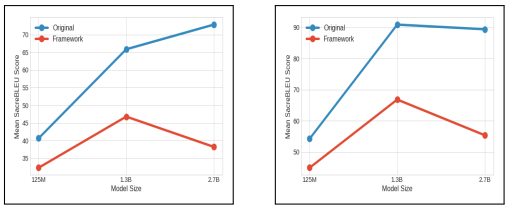
奖励函数。为了考察不同奖励函数对我们提出的框架中学习到的策略的影响，我们使用三种不同的奖励函数进行了实验：负 SacreBLEU、负 SacreBLEU 与困惑度相结合，以及负 BERTScore。这些奖励函数旨在解决近似记忆问题，并鼓励生成彼此不同但语义一致的内容。

输出。它们基于语义相似度的概念，确保生成的文本与输入在含义上相关。通过比较这些奖励函数的性能，我们旨在为给定任务确定最优选择。

训练轮次数量。训练轮次数量是我们框架中的一个关键超参数，因为它决定了模型在数据上训练的次数。每个训练轮次期间，学习到的策略都可能发生变化，因此仔细选择训练轮次数量，以找到最优策略至关重要。生成示例的差异性与流畅度之间存在权衡，稍后我们将对此进行讨论。因此，选择合适的训练轮次数量，对于在这些相互竞争的目标之间取得最佳平衡至关重要。

数据集设置。为了评估学习到的策略对预训练数据集中不同设置的泛化能力，我们使用三种不同的上下文长度进行了实验：与训练序列相同的长度，以及额外增加 100 个词元以构成更长的上下文。这使我们能够评估上下文长度对模型性能的影响，并识别其行为中的模式或趋势。通过比较不同上下文长度下的结果，我们旨在更好地了解模型的泛化能力。

评估指标。为了评估我们方法的性能，我们考察了两个关键方面：差异度分数和生成后缀的质量。（1）差异度分数：我们使用负 SacreBLEU 分数来衡量生成的后缀与真实后缀之间的差异度，因为该分数基于语义相似度，因此能够有效衡量近似记忆。（2）生成后缀的质量：为了评估文本的流畅度，我们使用了应用框架之前所用模型的困惑度分数。借助这一指标，我们能够评估生成



(a) 标准设置　　(b) 更长上下文设置

图 2：比较应用框架前（蓝色）与应用框架后（橙色）所有模型规模下的平均 SacreBLEU 分数

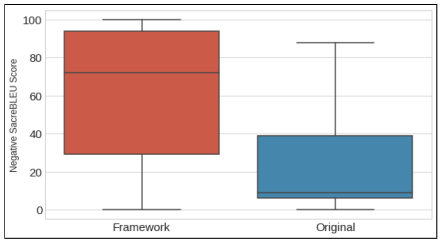


图3：展示在应用框架之前，GPT-Neo 2.7B 在标准设置下的负 SacreBLEU 分布（蓝色）和之后（橙色）应用该框架

后缀在语法正确性和整体连贯性方面的质量。综合考虑差异度分数和生成后缀的质量，我们便能全面了解模型的性能。

5.2 实验结果

本节将展示在所选数据集上使用不同配置的所提出框架所取得的结果。研究目标是考察所提出方法在不同场景下的有效性，并识别可能出现的任何模式或趋势。

GPT-Neo 125M。在实验中，我们发现，对于 GPT-Neo 125M 模型，在训练轮次为 4、8 和 12 时，负 BERTScore 奖励函数在平衡相似度与流畅度方面优于负 SacreBLEU，以及负 SacreBLEU 与困惑度结合的方案。虽然负 SacreBLEU 与困惑度结合后取得了更高的差异度分数，但随着困惑度分数上升，生成示例的流畅度也随之下降。表现最差的是负 SacreBLEU 奖励函数：如表1所示，与其他奖励函数相比，它的差异度分数和困惑度分数都更高。进一步探索后，我们发现，对于所有奖励函数，12 个 PPO 训练轮次都能在相似度与流畅度之间取得最佳平衡。所有这些实验均在标准测试集上评估，随后又在更长上下文设置下评估了排名前三的实验设置。在该设置中，采用 12 个 PPO 训练轮次的负 BERTScore 取得了最佳结果。正如预期，更长的上下文提高了记忆分数。不过，差异度分数也显著提升：应用框架前为 45.74%，采用负 SacreBLEU 分数后达到 55.04%。

如表2所示，标准设置和更长上下文设置的结果表明，该框架能够稳健地提升语言模型的性能。在标准设置中，应用框架前后负 SacreBLEU 的差值为 8.07，表明性能显著提升。即使在模型能够获取更多信息的更长上下文设置下，应用框架前后负 SacreBLEU 的差值仍为 9.3，依然有所改善。这说明，即使向模型提供更多信息，该框架仍能保持有效；对许多模型而言，这可能构成挑战，因为更多信息会增加记忆风险。此外，使用更长的上下文也对生成后缀产生了积极影响，使其流畅度得到提升。

GPT-Neo 1.3B 和 GPT-Neo 2.7B。我们在 GPT-Neo 1.3B 和 GPT-Neo 2.7B 模型上采用不同设置进行了相同的实验，发现除训练轮次数量外，表现最佳的设置相同。GPT-Neo 1.3B 模型在两个训练轮次后收敛，每批次进行四个 PPO 训练轮次；GPT-Neo 2.7B 模型则在两个训练轮次后收敛，每批次进行两个 PPO 训练轮次。这表明规模更大的模型收敛更快。我们还观察到，使用更长的上下文能够提升生成后缀的流畅度，并将差异度分数提高：GPT-Neo 1.3B 的负 SacreBLEU 分数从 9.11% 提升至 36.16%，GPT-Neo 2.7B 则从 10.61% 提升至 44.62%。然而，在这些模型中，两种设置间的差距有所扩大；如图 2所示，这表明模型规模增大后，差异度分数的差距也随之扩大。采用更长上下文设置时，困惑度没有显著损失：GPT-Neo 1.3B 的困惑度分数从 1.55 增至 1.71

模型	设置	之前		之后		Δ_{N-SB}	训练轮次
		N − SacreBLEU ↑	困惑度 ↓	N − SacreBLEU ↑	困惑度 ↓		
GPT-Neo 125M	标准	59.63	3.85	67.70	4.01	8.07	12
	LC‡	45.74	4.12	55.04	4.15	9.3	
GPT-Neo 1.3B	标准	34.16	2.18	53.22	2.64	19.09	8
	LC	9.11	1.55	36.16	1.71	27.05	
GPT-Neo 2.7B	标准	27.18	1.92	61.79	4.46	34.61	4
	LC	10.61	1.41	44.62	1.82	34.01	

表2：应用框架前后负 SacreBLEU 与困惑度均值的比较。 ‡请参阅更长上下文设置。 $\mathbf{\Delta_{N-SB}}$ 关于应用该框架前后负 SacreBLEU 均值差异的说明。

和 GPT-Neo 2.7B 分别从 1.414 增至 1.823，如应用该框架后表 2所示。

6 讨论

学习到的策略。 我们监测了模型在不同训练轮数范围内进行训练时的表现。为了增加生成的后缀与真实后缀之间的差异，模型最初采用了一种策略：输出相同的后缀，但使用不同的大小写形式（例如，全部大写或全部小写）。然而，这种做法无助于实现我们的目标，也没有降低相似度。随着训练轮次增加，模型改进了学习到的策略，从而获得更高的奖励。起初，它尝试通过改变个别词语或数字，来降低目标后缀与生成的后缀之间的相似度。然而，这种做法在降低相似度方面效果有限。随着训练的继续，模型开始修改生成的后缀中的多个词语，试图提高差异度。最终，它开始改写或替换整个短语，采用语义相近的替代表达，以达到预期的差异度分数。值得注意的是，随着训练轮次数量增加，差异度分数有所提升，但困惑度分数也随之升高，这表明两个目标之间存在权衡。此外，训练轮次过多会导致模型生成与真实后缀语义不同的后缀；在某些情况下，它还会生成问号等无意义词元，或多次重复同一个词语。

评估近似记忆。根据近期研究，当需要使用二元标签判断是否发生近似记忆时，生成后缀的 BLEU 分数达到 75% 是合适的阈值。该阈值是通过检查大量生成示例（Ippolito 等人，2022年）确定的。然而，在我们自己的研究中发现，即使将阈值降低至

应用该框架后，阈值为 50%。尽管有这一发现，我们仍选择使用更广泛认可的 75% 阈值，以展示框架的有效性。在 GPT-Neo 1.3B 和 GPT-Neo 2.7B 上实施该框架后，如图4、图5和图6所示，表现出近似记忆的生成示例数量分别从 910 个减少到 497 个，以及从 1036 个减少到 321 个。图3 以箱线图的形式展示了 GPT-Neo 2.7B 在标准设置下生成后缀和真实后缀的负 SacreBLEU 分数。生成后缀的负 SacreBLEU 分数中位数为 72.06%，这意味着约 50% 的数据的差异度分数高于 72.06%。另一方面，真实后缀的负 SacreBLEU 分数中位数为 8.70%，表明约 50% 的数据的差异度分数低于 8.70%。这一比较展示了我们的框架在减少近似记忆发生方面的有效性。

7 结论

在本文中，我们提出了一种用于解决大语言模型记忆训练数据问题的新型框架。通过在各种设置下开展的一系列评估，我们证明了该方法的有效性。我们表明，该框架能够通过显著降低 Sacre-BLEU 分数，有效减少近似记忆，同时保持生成样本的流畅度和连贯性。此外，在使用更长的上下文时，我们的框架仍表现出鲁棒性，而这可以被视为一种攻击形式。尽管许多研究已经证实，增大语言模型的规模与其记忆更多训练数据的倾向之间存在相关性，但对这些更大的模型进行微调也可能带来高昂的计算成本。我们提出的框架通过展示，提供了一种解决这一问题的新方法

语言模型规模越大，收敛速度和差异度分数也会随之提高。这些改进有效缓解了记忆增加的问题，同时降低了与微调更大模型相关的计算成本。

局限性

我们工作的局限之一是，在优化时依赖单一标量奖励，因为该问题有双重目标：差异性和困惑度。为克服这一局限，我们建议探索其他技术，例如多目标强化学习，这有望提升性能并同时优化这两个目标。此外，我们工作中使用的数据集缺少元数据和标签，而这些信息有助于进一步分析模型在不同类型文本（例如个人信息、版权内容和新闻）上的性能。使用此类元数据和标签有助于了解模型在不同文本类别上的性能，并做出必要调整以提升模型性能。

伦理声明

随着大语言模型日益受到关注，并应用于生活的多个方面，提升其隐私保护能力至关重要。确保这些模型的数据隐私十分重要，因为某些对手可能能够获取相关信息。要让这些模型得到广泛应用，我们必须保证它们不会泄露私人数据。正如早期研究所指出的，逐字记忆可能会造成隐私得到保护的假象。本文希望为开发解决大语言模型近似记忆问题的新颖方案奠定基础。我们提出的框架为解决这一问题提供了一种有前景的方法。对此领域开展进一步研究和实验，有望找到更有效的方法来减少这些模型中的记忆。我们的工作还强调，开发应对大语言模型记忆问题的新技术时，必须同时考虑计算成本与性能权衡。

参考文献

Martin Abadi、Andy Chu、Ian Goodfellow、H Brendan McMahan、Ilya Mironov、Kunal Talwar 和 Li Zhang。2016年。差分隐私下的深度学习。载于*2016*年 *ACM SIGSAC* 计算机与通信安全会议论文集，第308–318页。

Rohan Anil、Badih Ghazi、Vineet Gupta、Ravi Kumar 和 Pasin Manurangsi。2021年。大规模差分隐私 BERT。*arXiv* 预印本 *arXiv:2108.01624*。

Sid Black、Leo Gao、Phil Wang、Connor Leahy 和 Stella Biderman。2021年。GPT-Neo：使用 Mesh-Tensorflow 进行大规模自回归语言建模。如果您使用此软件，请使用这些元数据引用它。

Hannah Brown、Katherine Lee、Fatemehsadat Mireshghallah、Reza Shokri 和 Florian Tramèr。2022年。语言模型保留隐私意味着什么？*arXiv* 预印本 *arXiv:2202.05520*。

Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell 等。2020年。语言模型是少样本学习器。神经信息处理系统进展，33：1877–1901。

Nicholas Carlini、Daphne Ippolito、Matthew Jagielski、Katherine Lee、Florian Tramer 和 Chiyuan Zhang。2022年。量化神经语言模型中的记忆。*arXiv* 预印本 *arXiv:2202.07646*。

Nicholas Carlini、Florian Tramer、Eric Wallace、Matthew Jagielski、Ariel Herbert-Voss、Katherine Lee、Adam Roberts、Tom Brown、Dawn Song、Ulfar Erlingsson 等。2021年。从大语言模型中提取训练数据。载于第*30*届 *USENIX* 安全研讨会（*USENIX Security 21*），第2633–2650页。

Aakanksha Chowdhery、Sharan Narang、Jacob Devlin、Maarten Bosma、Gaurav Mishra、Adam Roberts、Paul Barham、Hyung Won Chung、Charles Sutton、Sebastian Gehrmann 等。2022年。PaLM：通过 Pathways 扩展语言建模。*arXiv* 预印本 *arXiv:2204.02311*。

Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2018年。BERT：用于语言理解的深层双向变换器预训练。*arXiv* 预印本 *arXiv:1810.04805*。

William Fedus、Barret Zoph 和 Noam Shazeer。2021年。《Switch 变换器：通过简单高效的稀疏性扩展至万亿参数模型》。

Leo Gao、Stella Biderman、Sid Black、Laurence Golding、Travis Hoppe、Charles Foster、Jason Phang、Horace He、Anish Thite、Noa Nabeshima 等。2020年。《The Pile：用于语言建模的多样化文本数据集，规模达 800 GB》。*arXiv* 预印本 *arXiv:2101.00027*。

Robert Geirhos、Jörn-Henrik Jacobsen、Claudio Michaelis、Richard Zemel、Wieland Brendel、Matthias Bethge 和 Felix A Wichmann。2020年。深度神经网络中的捷径学习。《自然·机器智能》，2(11)：665–673。

Pengcheng He、Xiaodong Liu、Jianfeng Gao 和 Weizhu Chen。2021年。DeBERTa：具有解耦注意力的解码增强型BERT。载于国际学习表征会议。

Daphne Ippolito、Florian Tramèr、Milad Nasr、Chiyuan Zhang、Matthew Jagielski、Katherine Lee、Christo- pher A Choquette-Choo 和 Nicholas Carlini。2022年。防止语言模型逐字记忆会造成一种虚假的隐私安全感。*arXiv* 预印本 *arXiv:2210.17546*。

Nikhil Kandpal、Eric Wallace 和 Colin Raffel。2022年。训练数据去重可降低语言模型中的隐私风险。*arXiv* 预印本 *arXiv:2202.06539*。

Jared Kaplan、Sam McCandlish、Tom Henighan、Tom B Brown、Benjamin Chess、Rewon Child、Scott Gray、Alec Radford、Jeffrey Wu 和 Dario Amodei。2020年。神经语言模型的缩放定律。*arXiv* 预印本 *arXiv:2001.08361*。

Katherine Lee、Daphne Ippolito、Andrew Nystrom、Chiyuan Zhang、Douglas Eck、Chris Callison-Burch 和 Nicholas Carlini。2021年。训练数据去重让语言模型表现更好。*arXiv* 预印本 *arXiv:2107.06499*。

Adam Paszke、Sam Gross、Soumith Chintala、Gregory Chanan、Edward Yang、Zachary DeVito、Zeming Lin、Alban Desmaison、Luca Antiga 和 Adam Lerer。2017年。《PyTorch 中的自动微分》。

Matt Post。2018年。呼吁清晰报告 BLEU分数。载于《第三届机器翻译会议论文集：研究论文》，第 186–191 页，比利时布鲁塞尔。计算语言学协会。

AlecRadford、KarthikNarasimhan、TimSalimans、Ilya Sutskever 等。2018年。通过生成式预训练提升语言理解能力。

Alec Radford、Jeffrey Wu、Rewon Child、David Luan、Dario Amodei、Ilya Sutskever 等。2019年。语言模型是无监督多任务学习器。*OpenAI*博客，1(8):9。

Rajkumar Ramamurthy、Prithviraj Ammanabrolu、Kianté Brantley、Jack Hessel、Rafet Sifa、Christian Bauckhage、Hannaneh Hajishirzi 和 Yejin Choi。2022年。强化学习是否适用于自然语言处理？：自然语言策略优化的基准测试、基线与构建模块。*arXiv* 预印本 *arXiv:2210.01241*。

Jeff Rasley、Samyam Rajbhandari、Olatunji Ru wase 和 Yuxiong He。2020年。DeepSpeed：系统优化使得训练拥有超过1000亿个参数的深度学习模型成为可能。发表于第*26*届*ACM SIGKDD*知识发现与数据挖掘国际会议，第3505–3506页。

John Schulman、Filip Wolski、Prafulla Dhariwal、Alec Radford 和 Oleg Klimov。2017年。近端策略优化算法。*arXiv* 预印本 *arXiv:1707.06347*。

Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017年。注意力就是你所需要的一切。神经信息处理系统进展，第30卷。

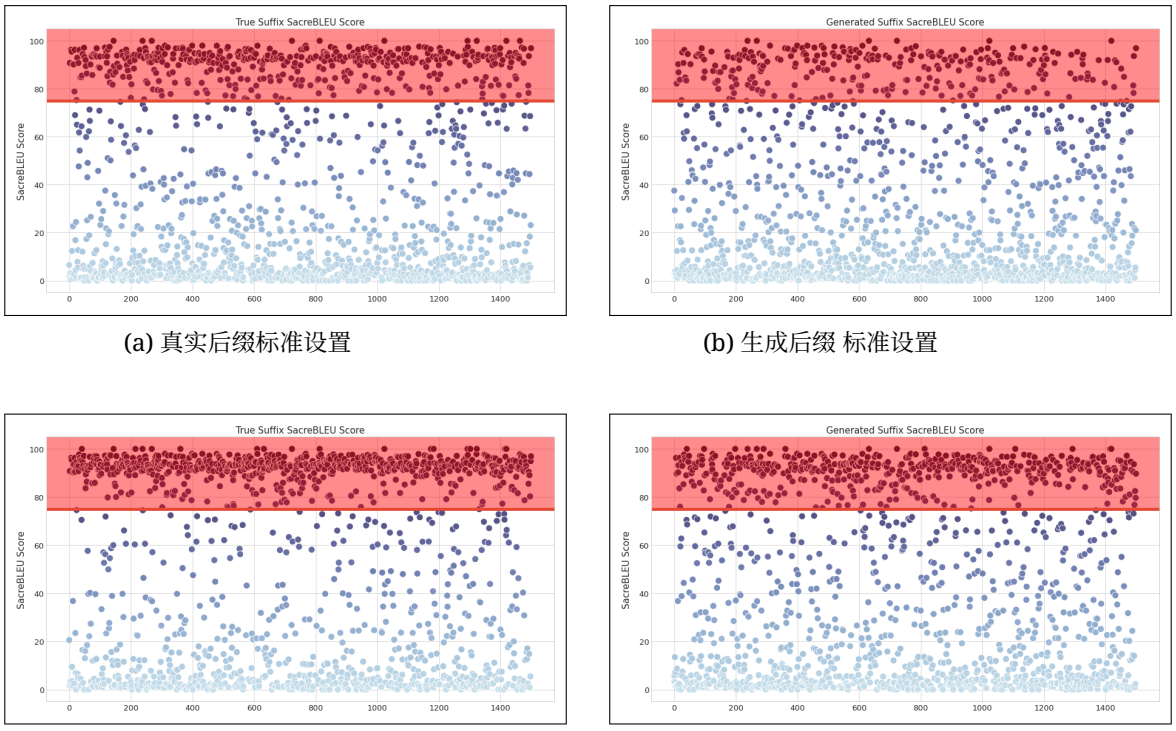
Thomas Wolf、Lysandre Debut、Victor Sanh、J ulien Chaumond、Clement Delangue、Anthony Moi、Pier- ric Cistac、Tim Rault、Remi Louf、Morgan Funtow- icz、Joe Davison、Sam Shleifer、Patrick von Platen、Clara Ma、Yacine Jernite、Julien Plu、Canwen Xu、Teven Le Scao、Sylvain Gugger、Mariama Drame、Quentin Lhoest 和 Alexander Rush。2020年。Transformers：自然语言处理领域的先进成果。收录于*2020*年自然语言处理实证方法会议论文集：系统演示，第38–45页，线上。计算语言学协会。

Tianyi Zhang、Varsha Kishore、Felix Wu、Kilian Q Weinberger 和 Yoav Artzi。2019年。BERTScore：使用 BERT 评估文本生成。*arXiv* 预印本 *arXiv:1904.09675*。

一个显示近似记忆阈值的界面

近期研究表明，当 BLEU 分数达到 75% 时会发生近似记忆；我们遵循这一观点，并在本节中通过比较应用该框架前后超过此阈值的样本数量，展示所提出框架的有效性。

$$\text{SacreBLEU}(\text{后缀}_G, \text{后缀}_T) > 0.75 \tag{1}$$

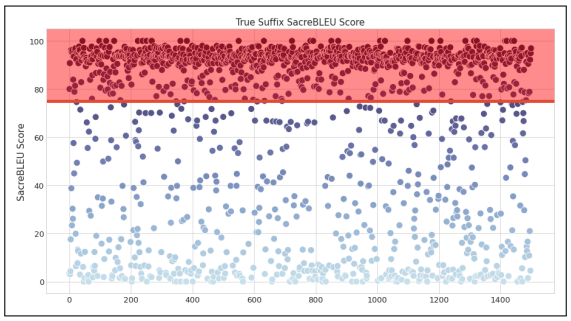


(a) 真实后缀标准设置　(b) 生成后缀 标准设置

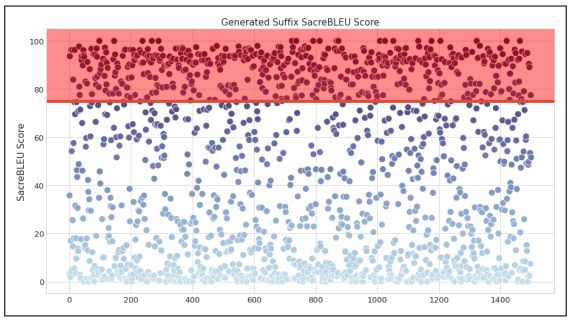
(c) 真实后缀 更长上下文设置　(d) 生成后缀更长上下文设置

图 4：75% 真实样本与生成样本的 SacreBLEU 阈值（GPT-Neo 125M 标准设置）　ng

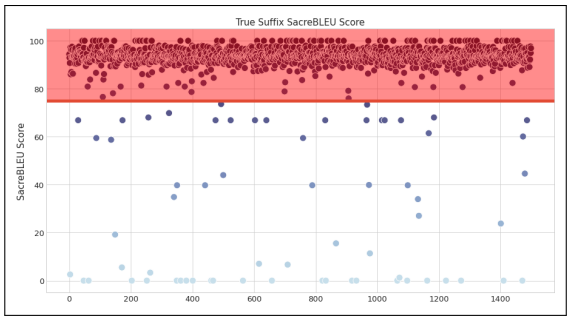
如图 4所示，GPT-Neo 125M 模型的记忆率相对较低。然而，在采用标准设置和更长上下文设置时，许多样本分布在 75% 阈值处及其以上。尽管如此，应用所提出的框架后，样本分布在各个数值区间中更加均匀，而不再仅集中于 75% 阈值以上的区域。与其他变体相比，GPT-Neo 1.3B 和 2.7B 的记忆率较高，尤其是在使用更长上下文时；从图 5 和图 6中可以明显看出，该框架对这些变体的效果，因为许多样本超过了阈值。



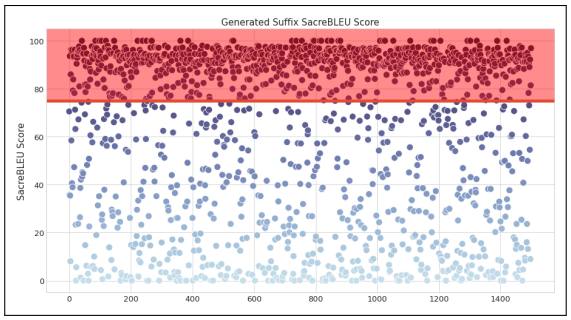
(a) 真实后缀 标准设置



(b) 生成后缀 标准设置

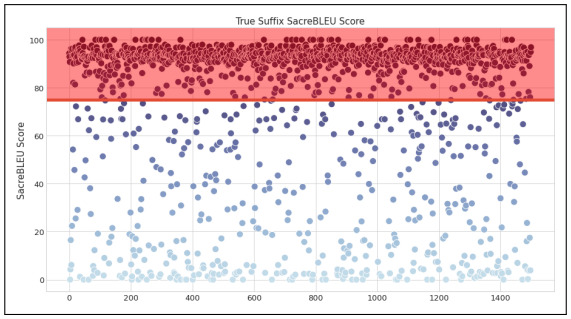


(c) 真实后缀 更长上下文设置

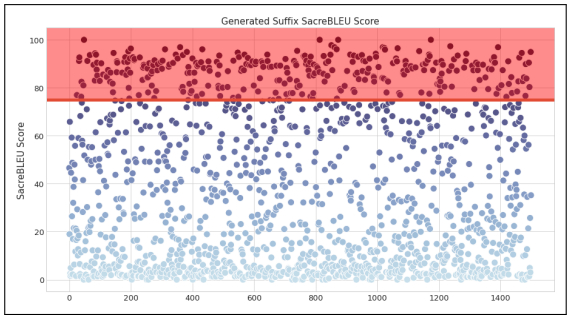


(d) 生成后缀 更长上下文设置

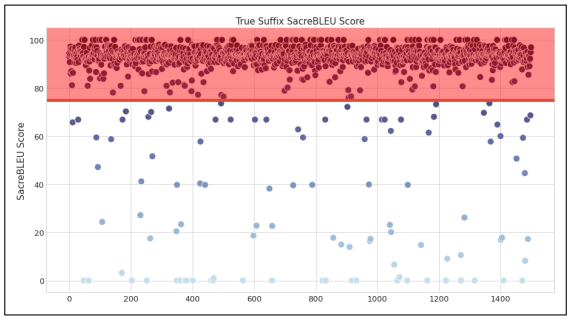
图 5： T　GPT-Neo 1.3B 标准设置下真实样本与生成样本的 SacreBLEU 75% 阈值 g



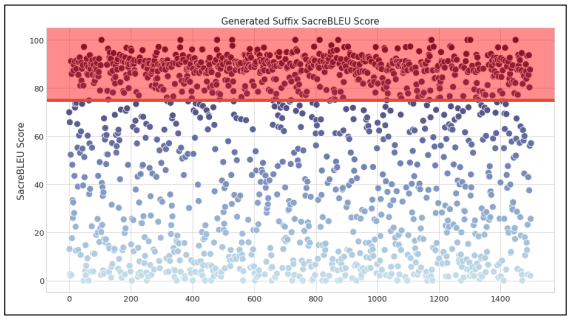
(a) 真实后缀 标准设置



(b) 生成后缀 标准设置



(c) 真实后缀 更长上下文设置



(d) 生成后缀 更长上下文设置

图 6：GPT-Neo 2.7B 标准设置下真实样本与生成样本 SacreBLEU 的 75% 阈值

B 轮次数量与奖励函数

在本节中，我们将深入探讨不同模型中各奖励函数的训练轮次数量变化所带来的影响。具体而言，我们将展示在不同设置下进行实验的结果，例如修改奖励函数以及调整各模型所使用的训练轮次数量。这将帮助我们全面了解使用不同奖励函数时，训练迭代次数会如何影响不同模型的性能。如表3所示，模型规模与收敛所需的训练轮次数量之间存在明显的反向关系。模型规模越大，达到令人满意的性能水平就越高效。例如，较小的 GPT-Neo 125M 需要 16 个训练轮次才能收敛，而较大的 GPT-Neo 2.7B 只需 4 个训练轮次即可达到最佳性能水平。

模型	奖励函数	训练轮次	N −SacreBLEU ↑	困惑度 ↓
GPT-Neo 125M	N-SacreBLEU	8	71.77	4.43
		12	82.87	5.25
		16	87.78	6.55
GPT-Neo 125M	N-SacreBLEU +困惑度	8	70.69	4.41
		12	78.32	4.83
		16	84.35	5.93
GPT-Neo 125M	BERTScore	8	64.45	3.94
		12	67.70	4.01
		16	68.61	4.11
GPT-Neo 1.3B	N-SacreBLEU	8	72.10	3.65
		12	85.53	6.43
GPT-Neo 1.3B	N-SacreBLEU +困惑度	8	52.02	2.85
		12	52.05	2.86
GPT-Neo 1.3B	BERTScore	8	53.22	2.64
		12	57.01	2.77
		16	68.47	2.95
GPT-Neo 2.7B	N-SacreBLEU	4	71.47	6.10
		8	83.09	8.423
GPT-Neo 2.7B	N-SacreBLEU +困惑度	4	80.84	8.60
		8	88.67	12.53
GPT-Neo 2.7B	BERTScore	4	61.79	4.46
		8	75.09	6.76

表3：不同奖励函数在标准设置测试集上的结果。

C 定性结果

在本节中，我们通过全面分析应用该框架前后生成的样本，展示所提出框架的有效性。为了全面评估，我们从不同模型规模中选取了样本，包括 125M、1.3B 和 2.7B，并纳入标准上下文和更长的上下文中的示例。此外，我们还展示了不同训练阶段的样本，以呈现学习到的策略随时间的演变。如前所述，该策略最初侧重于替换单个词语或数字，以降低样本之间的相似度。随着训练过程的推进，该策略变得更加激进，会替换整个短语，如图7所示。图8 表明，该框架能够学习一种策略，以减少或消除记忆的个人数据（例如电子邮件地址）。不过，需要注意的是，在某些情况下，这可能会导致困惑度上升。

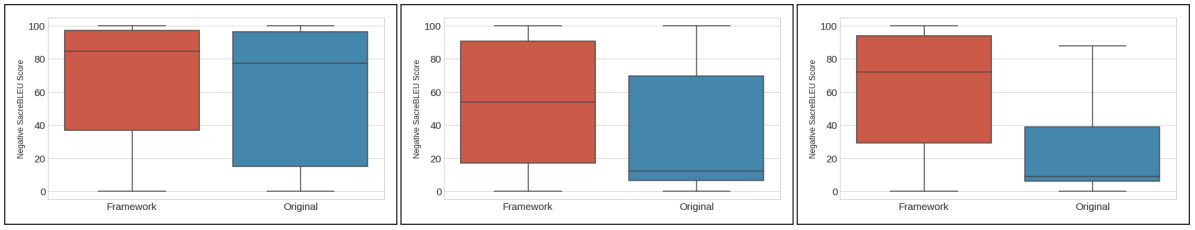
前缀	真实后缀	生成的前置后缀	生成的后置后缀	N-SacreBLEU$_B$	N-SacreBLEU$_A$	PPL$_B$	PPL$_A$
如果此前已获暂缓执行刑罚，且 获准保释释放 ，且该许可由一审法院或本院作出， 则暂时延续中止执行，期限不超过六十 天，且适用此前已缴纳的保释金。延续中止令的目的在于 使上诉人能够向 向俄亥俄州最高法院提出中止执行申请， 以在该法院诉讼程序进行期间中止执行。如果中止执行 根据本法院裁定继续执行，则将在以下两者中较早发生者终止： 六十日期间届满，或 上诉人未能向俄亥俄州最高法院提交上诉通知 的情形。	四十五天上诉期限 根据第二条第二节的规定， 《执业规则》之 俄亥俄州最高法院诉讼规则》第二条第二节， 此外，如果俄亥俄州最高法院 驳回该 在……期限届满前提出上诉 六十天，	四十五天上诉期限 根据第二条第二节 《俄亥俄州最高法院诉讼规则》 俄亥俄州最高法院。 此外，如果最高法院 俄亥俄州法院驳回了 the 到期前上诉的先决条件 六十天，t 终止	续写的情况。如果 the 俄亥俄州最高法院 驳回上诉人的动议 以中止执行；中止执行将 在以下最早时间终止： 六十日期间届满之时 ，或在 裁定录入日期，或	6.74	87.6	1.60	1.97
模块目录 = true; loadScripts(document,'script');</script> <noscript> 您的浏览器已禁用 JavaScript。 </noscript> <页眉 role="横幅	"> <导航角色="navigation"> <div class="fixedNav"> <!-- ========= 顶部开始 导航栏 ======= --> <div class="topNav"><a id="navbar.top"> <!--	"> <导航角色="导航"> <div class="固定导航"> <!-- ========= 顶部开始 导航栏 ======= --> <div class="顶部导航"><a id="导航栏.top"><!-	" class="ui-bar-nav-center d- 无 d-md-block d-lg-block"> </header> <div class="下拉菜单" bu class="b 下拉菜单切换" type="button"	3.10	88.27	3.04	3.20
。相关文章 读者互动 提交您的 评论免责声明：FitSmallBusiness.com 上的评价 是由我们的作者、 研究人员和编辑团队独立研究的成果。用户评价和 评论由独立用户提供，他们与 FitSmallBusiness.com 的编辑团队没有关联。 银行、发卡机构、信用卡公司及其他产品 & 服务提供商不对任何内容承担责任 发布于 FitSmallBusiness.com。因此，它们并不 认可或保证任何发布的评论，或 评价。取消回复 您必须登录后才能发表评论。 点击下方的“使用 LinkedIn 登录”按钮以连接 即时评论。社交 联系我们 免责声明：我们 花费数小时进行研究并	撰写文章，努力 提供准确、及时的 内容。不过，我们的 研究旨在为您提供帮助 自己的，且我们并非以 持牌专业人士的身份行事。 我们建议您 咨询您自己的律师， 会计师或其他	我们撰写文章时力求 提供准确、最新的 内容。然而，我们的 研究旨在为您提供帮助 自行承担，我们并非以 持牌专业人士的身份行事。 我们建议您 咨询您自己的律师、 会计师或其他持牌 专业人士，	撰写我们的文章，并努力 以提供准确、及时的 内容。不过，我们的 研究旨在帮助您 加深 我们不是专业机构， 也不声称自己是。 此信息可能会 随时更改。	6.31	52.61	1.34	1.92
* 以下条件：* * 上述版权声明及 本许可声明应包含在所有副本或 软件的实质性部分。* * 本软件 按“原样”提供，不附带任何保证。	类型、表达*或 默示的，包括但不限于 不限于 保证 * 适销性， 特定用途 适用性 及不侵权。 在任何情况下	类型、明示\n * 或 默示，包括但不限于 不限于 保证\n * 适销性、 特定用途 适用性 和不侵权 在任何情况下均不得 AUTH	形式，无论是明示还是 默示，包括但不限于 以下各项： 关于 * 的保证 适销性、 特定用途 适用性 且不 区域条件，	9.53	30.46	1.63	1.09

图 7：所用语言模型记忆的后缀，以及在相同前缀下生成的后缀。绿色表示这部分与真实后缀一致，即被模型记忆；红色表示这部分不相似。

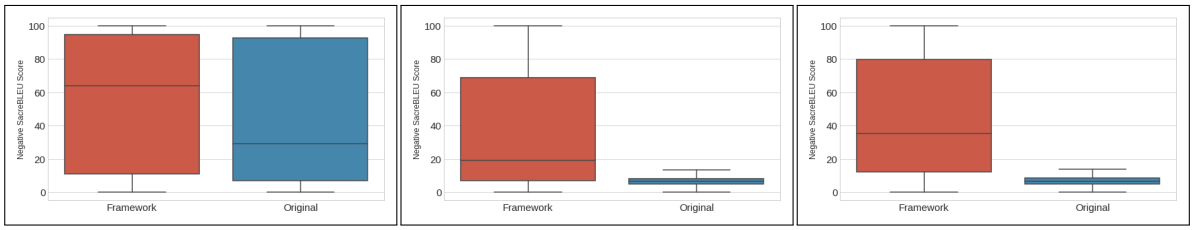
前缀	真实后缀	生成的前置后缀	生成的后置后缀	N-SacreBLEU$_B$	N-SacreBLEU$_A$	PPL$_B$	PPL$_A$
"POT 创建日期: 2017年-02-24 11:50-0800\n" "PO- 修订日期: 年-月-日 时:分+时区\n" "最后	-译者: 姓名 <电子邮件地址>\n " "语言团队: 语言 <[redacted]org>\n" "语言: \n" "MIME 版本	-译者: 全名 <电子邮件@地址>\n" "语言团队: 语言 <LL@li[redacted] "MIME 版本: "内容	-译者: 全名 <电子邮件@地址>\n" "语言团队: 英语 (印度) (http://www.transifex.com /odoo/teams/2027/英语(j)\n"	12.97	62.38	1.68	1.98
自由软件基金会的第3版 许可证，或 *（由您选择）任何更高版本。* * weid- java	-sdk 的发布希望 它会有所帮助，* 但 不提供任何保证； 即使不提供默示保证 适销性或 适用性	.org 的发布是希望 它会有所帮助，* 但是 不作任何保证； 甚至不包括默示保证 * 适销性或 适用于特	.com * *	16.41	99.95	1.93	2.11
[redacted]	.com>, S[redacted] M[redacted] <S[redacted] [redacted]om>, T--[redacted] [redacted]@[redacted].com>, R[redacted] S[redacted]<r[redacted]@	.com> [redacted]	.com> S[redacted] J[redacted] <S[redacted] J.M[redacted] 美国联邦能源监管委员会会议美国联邦能源监管委员会 会议定于 2001年6月2日，星期二，时间为	69.87	86.04	1.96	2.60
在该分布中，无论是 RC4、RSA、* lhash、DES 等， 代码；而不仅仅是安全套接层代码。安全套接层文档 * 随本发行版附带，受相同的 版权条款保护 *，但权利持有人为 Tim	Hudson（t[redacted]cr--[redacted]m）。* * 版权仍归 E 所有--- Y[redacted]- [redacted]，因此任何 版权声明位于*中。 代码不得删除。 * 如果此软件包用于某个 a 产品，E--[redacted]为 给定	Hudson（t[redacted] *	Hudson. * * 此文件的版本为 2.0.0.\r\n * \r\n * \r\n * 按常用法记法--- 记法用于	80.12	96.52	3.80	6.64

图8：所使用的语言模型记忆的后缀，以及在给定相同前缀时生成的后缀。绿色表示根据真实后缀判定该部分已被记忆，红色表示该部分不相似。

D 中位数比较



(a) GPT-N eo 125M在标准设置下 (b) GPT-Neo 1.3B 在标准设置下 (c) GPT-Neo 2.7B 在标准设置下



(d) GPT-Neo 125M 在长设置下 (e) GPT-Neo 1.3B 在长设置下 （f）长设置下的 GPT-Neo 2.7B

图9：展示模型在标准设置和长设置下应用框架前（蓝色）（蓝色）及应用后（橙色）的负 SacreBLEU 分布

E 计算资源

为了微调规模为 125M 和 1.3B 的 GPT-Neo 模型，我们使用了一个由两块 V100 GPU 组成的集群，每块 GPU 均配备 32GB 显存。125M 模型每个 PPO 训练轮次约需 0.38 分钟，六个训练轮次的总计算时间为 3.04 分钟。1.3B 模型每个 PPO 训练轮次所需的计算时间稍长，为 1.68 分钟，八个训练轮次总计耗时 13.44 分钟。对于最大的变体 GPT-Neo 2.7B，我们使用了一个由四块 V100 GPU 组成的集群，每块 GPU 配备 32GB 显存，并采用了零冗余优化器（ZeRO）3 分片策略（Rasley 等人，2020年）。该模型的每个 PPO 训练轮次需耗时 5.125 分钟，四个训练轮次的总计算时间约为 20 分钟。微调这些模型时，我们使用 HuggingFace 库（Wolf 等人，2020年）进行训练，并使用 PyTorch（Paszke 等人，2017年）实现模型并行。