Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

大模型也不安全：小心信息被泄露

硕士研究生 皮佳伟

2025年08月24日

问题回溯

- 总结反思
  - 讲述过程中部分阶段语速较快
- 相关内容
  - 2025.03.16 皮佳伟《提示词怎么在别人兜里：提示词窃取攻击》
  - 2024.10.27 皮佳伟《针对文本嵌入模型的模型反演攻击方法研究》
  - 2024.01.28 皮佳伟《偷走你的训练数据：模型反演攻击方法研究》
  - 2024.01.01 徐程柯《大模型调研》

内容提要

- 预期收获
- 题目内涵解析
- 研究背景与意义
- 研究历史与现状
- 知识基础
- 算法原理
  - Zlib
  - DSP
- 特点总结与工作展望
- 参考文献

背景简介

- 预期收获
  - 了解大模型隐私的重要性
  - 了解大模型面临的信息泄露风险
  - 理解两种关于大模型信息泄露攻击的算法
  - 了解模型信息泄露攻击研究的局限性和未来方向

内涵解析与研究目标

- 题目内涵解析（大模型信息泄露攻击）
  - 隐私信息：隐私信息是大模型发布者或用户不想被其他人获知的信息（个人联系方式等）
  - 泄露攻击：未经所有者许可前提下，通过精心设计的手段获取大模型包含的敏感信息
- 研究目标
  - 面向大模型的隐私安全研究
  - 结合生成模型、大模型记忆等理论
  - 窃取精心设计的攻击方法，揭示大模型面临的隐私安全风险

研究背景

- 大语言模型（Large Language Models）
  - 是一类基于深度学习的人工智能模型，具备强大的自然语言理解和生成能力。
  - 它们通过在海量文本数据上进行预训练，学习语言的语法、语义、常识和推理能力，从而能够完成多种任务，如问答、写作、翻译、代码生成等。
- 大模型的应用频率
  - ChatGPT: 2025年8月数据，日均活跃用户约1.8亿
- 预训练语料
  - 网络文本合集（网页内容）、GitHub代码、第三方非公开数据

研究意义

- 研究意义
  - 验证大语言模型面临的隐私泄露风险
    - 大语言模型的市场体量
    - 预训练语料中包含的敏感信息
    - 大模型信息泄露
  - 促进防御方法发展
    - 以攻击促进防御手段的发展
    - 提高大模型服务提供商对隐私保护的重视
    - 促进服务提供商、AI模型设计者设计有效防御手段

大模型被广泛使用，但背后正在悄悄泄露你的隐私

【 USENIX 】

Extracting Training Data from Large Language Models

Zlib TIPO

T	目标	证明攻击者可查询大模型提取出训练数据中的个别样本
I	输入	目标模型接口*1，生成文本的前缀
P	处理	1. 由语言模型中生成大量文本样本 2. 从生成样本中筛选出真正来自于训练数据的样本 3. 对筛选后的样本进行模糊去重
O	输出	文本片段*1

P	问题	能否从大模型中提取训练数据中的样本尚不可知
C	条件	1. 目标模型黑盒设置
D	难点	1. 如何定义模型的记忆
L	水平	USENIX 2020（CCF A）

Zlib 记忆的定义

- 正常建模和隐私敏感的记忆
  - 一段文本如果只在一两个网页中出现，却能被模型准确复现，则极有可能是不希望的记忆（敏感记忆）
  - 类似“the cat sat on the mat”，模型准确复现也算不上隐私泄露
- 可提取性
  - 假设存在一个前缀c，模型在贪心或者beam解码中会把S作为最有可能后续，则S是可提取的

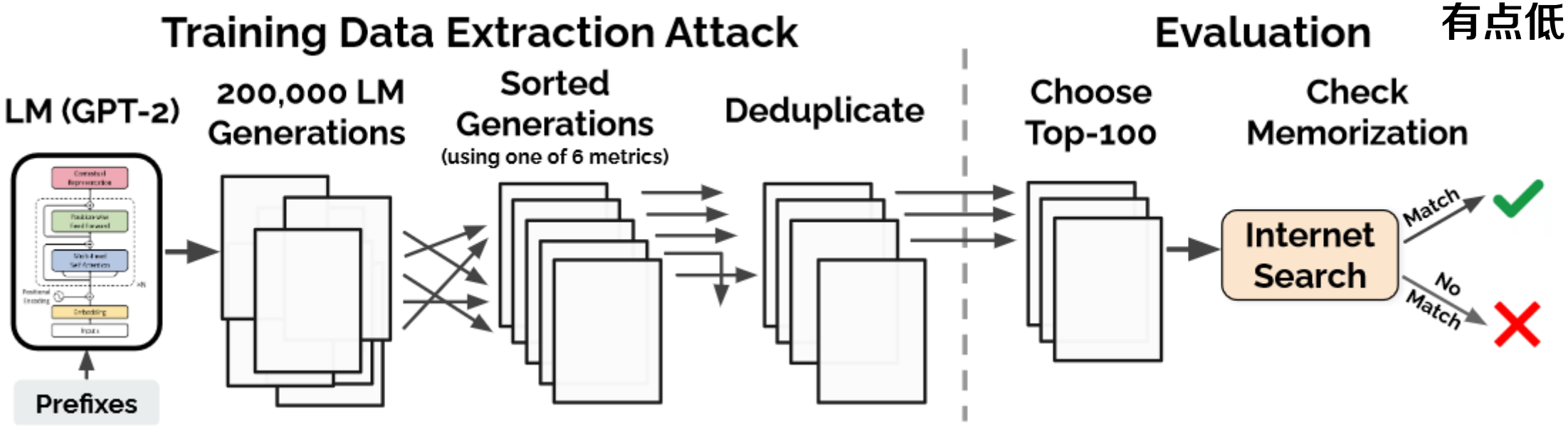
$$s = argmax f_\theta(s|c)$$

- k-eidetic记忆
  - S是可提取的；S在训练集中至多出现在K个不同文档中
  - K越小，则风险越高；K=1是极限情况，即旨在一个文档中出现

Zlib 算法原理图

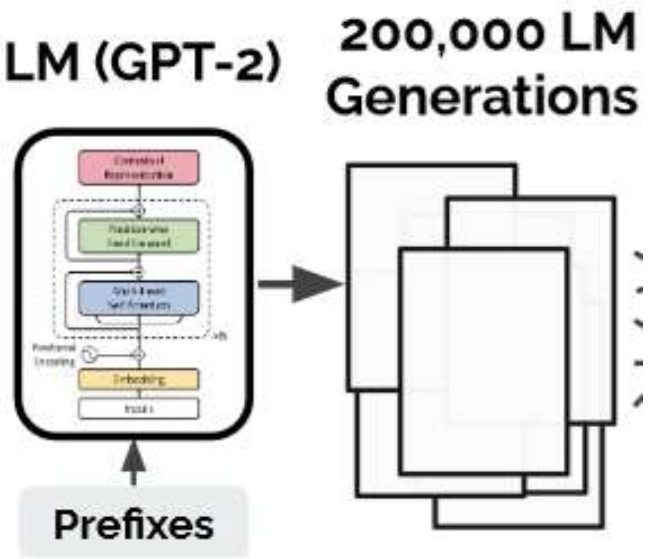
- 基线方法
  - 目的：用以验证大模型是否会泄露敏感信息
  - 样本生成：以固定起始token进行Top-n采样，获得200,000样本
  - 样本筛选：以样本困惑度筛选，训练集见过的语句会被给出更低困惑度
  - 样本人工验证

9%成功率
有点低……



Zlib 算法原理图

- 改进文本生成方法
  - 温度衰减策略
    - **改进思路**：若全程使用高温度，可能导致生成偏离记忆内容；全程低温度，多样性不足
    - 实施方案：初始温度 t=10（高多样性，探索更多前缀），在前 20 个 token 内逐渐衰减至 t=1（高置信度，锁定记忆内容路径）
  - 互联网文本条件采样
    - **改进思路**：用外部互联网文本片段作为前缀，引导模型生成与训练数据分布相似的内容
    - 实施方案：从 Common Crawl（公共网页存档）中抽取文本；随机选取 5-10 个 token 作为前缀，再用 top-n 采样继续生成后续内容



Zlib 算法原理图

- 改进样本筛选方法
  - 与参照模型对比
    - 计算大小模型困惑度比值，越低则越可能是记忆内容
  - 与zlib压缩对比
    - 计算 “GPT-2 困惑度” 与 “zlib 熵” 的比值，比值低的样本（模型似然高但 zlib 熵高）更可能是真实记忆内容（而非重复模式）。
  - 与小写文本对比
    - 计算 “原始文本困惑度” 与 “小写文本困惑度” 的比值，比值低的样本（小写后困惑度显著上升）更可能是记忆内容。（部分记忆内容可能依赖大小写）
  - 滑动窗口困惑度
    - 计算 50 个 token 滑动窗口的最小困惑度，捕捉局部高似然的记忆子串，避免因整体低似然而遗漏（样本整体困惑度高，但是包含局部记忆字串）

实验设计

- 样本生成：使用三种策略各生成200,000个样本（每个256token）
  - Top-n 采样：基于空序列的基础采样
  - 温度衰减采样：通过温度变化提升多样性
  - 互联网条件采样：以外部互联网文本为前缀引导生成
- 样本筛选
  - 采用六种指标排序，从每个策略和指标组合前1000个样本中选择100个，共1800个样本
- 去重处理
  - “三元组多集”判断：若两个样本的三元组重叠率大于50%，则去除
- 记忆验证
  - 手动：互联网搜索确认样本是否为训练样本
  - 官方：与GPT-2作者合作，以3-gram匹配确认样本是否在样本中存在

实验设计

- 目标模型：GPT-2XL 1.3B
- 总体结果：1800个样本中，有604个样本被确定为敏感记忆（33.5%）
- 记忆内容的类别
  - 常见内容：
    - 新闻标题、日志文件、许可证
  - 敏感内容：
    - 个人联系信息（32条）
    - 高熵序列（35条，如UUID）
    - 代码片段（31条）
    - 有效URLs（50条）

Category	Count
US and international news	109
Log files and error reports	79
License, terms of use, copyright notices	54
Lists of named items (games, countries, etc.)	54
Forum or Wiki entry	53
Valid URLs	50
Named individuals (non-news samples only)	46
Promotional content (products, subscriptions, etc.)	45
High entropy (UUIDs, base64 data)	35
Contact info (address, email, phone, twitter, etc.)	32
Code	31
Configuration files	30
Religious texts	25
Pseudonyms	15
Donald Trump tweets and quotes	12
Web forms (menu items, instructions, etc.)	11
Tech news	11
Lists of numbers (dates, sequences, etc.)	10

实验设计

- 目标模型：GPT-2XL 1.3B
- 攻击策略的有效性
  - 文本生成策略
    - 基于互联网文本的条件采样效果最佳（273）
    - 温度衰减采样最差（140）
  - 文本筛选策略
    - 对比类指标显著优于直接使用模型困惑度
      - zlib熵比值在top-n采样时获得了59个样本
      - 模型困惑度在top-n采样时获取了9个样本

Inference Strategy	Text Generation Strategy		
	Top-*n*	Temperature	Internet
Perplexity	9	3	39
Small	41	42	58
Medium	38	33	45
zlib	59	46	67
Window	33	28	58
Lowercase	53	22	60
Total Unique	191	140	273

- 内容差异
  - zlib策略通常获得非罕见文本，例如新闻标题、许可文件
  - 大小写转换策略通常找到存在不规则大小写的内容，如新闻标题或错误日志
  - 小型和大型参照模型策略通常会找到罕见内容

实验设计 记忆与模型大小、插入频率的关系

- 目标：验证影响语言模型的记忆行为的两个因素
- 实验设置：
  - 设定目标字符串：指定格式的URL，在文档中出现次数不同
  - 目标模型：GPT-2（1.5B，345M，117M参数量）
  - 攻击方法
    - 直接提示：使用URL前缀作为提示，生成10,000个结果，检查是否包含目标URL
    - 增强提示：额外提供URL前6个字符作为前缀
- 实验结果
  - 模型大小：更大的模型记忆能力更强，直接提示下，1.5B模型能记住所有出现33次及以上的URL
  - 插入频率：对于1.5B模型，字符串出现33次及以上就会被完全记忆
- 实验结论
  - 模型大小与记忆量正相关；重复次数是关键

算法总结 Zlib

- 算法流程
  - 通过生成策略获得候选文本
  - 对候选文本进行筛选
- 算法优势
  - 首次提出并验证了大模型的隐私泄露风险
- 算法不足
  - 对上下文的强依赖
  - 样本筛选方案均具有局限性

【 arXiv 】

Unlocking Memorization in Large Language Models with Dynamic Soft Prompting

DSP TIPO

T	目标	更精准高效的提取大模型记忆
I	输入	目标模型查询接口*1，前缀*1
P	处理	1. 将前缀进行映射 2. 使用训练好的生成器生成动态软提示 3. 利用动态软提示和前缀嵌入拼接，输入目标模型获得样本
O	输出	提取的记忆样本*1

P	问题	1. 固定的提示并不能适用于所有前缀
C	条件	拥有特定样本的给定前缀
D	难点	1. 如何针对性构建软提示
L	水平	arXiv 2024

DSP 问题定义

- 可被发现的记忆
  - 对于目标模型f_θ和数据x，如果存在生成策略G满足如下公式，则称x为可发现记忆

$$f_\theta(G(p)) = s \qquad x = [p][s]$$

  - 其中，p表示前缀，s表示后缀
- 优化目标
  - 目标是学习生成策略，以最大化测试集上的可发现记忆率

$$max \frac{1}{|D_{test}|} \sum_{x_i \in D_{test}} 1_{f_\theta(G(p_i))=s_i}(p_i)$$

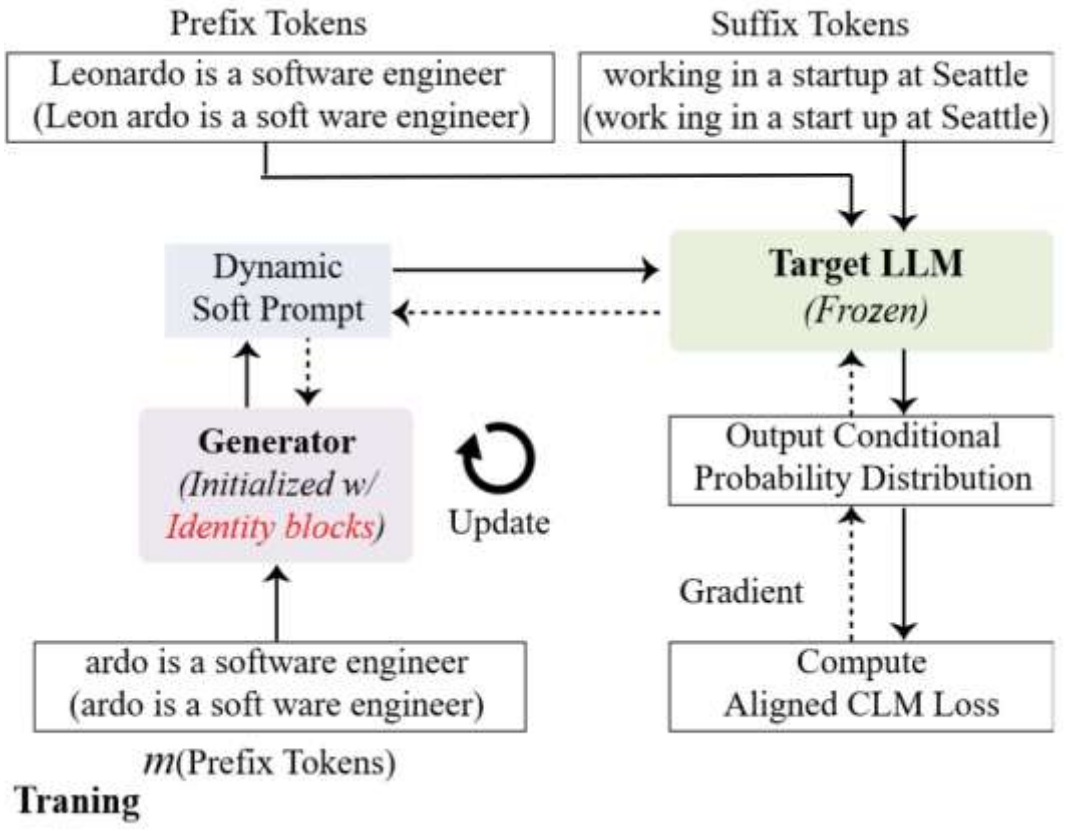
  - 其中，D_{test}是采样的测试集，$1(\cdot)$是指示函数，条件成立则为1

DSP 算法原理图

- 训练阶段
  - 对训练集的每个序列，生成器生成软提示o_i
  - 将o_i、前缀后缀嵌入$E(p_i)$ $E(s_i)$拼接输入目标模型
  - 最小化损失函数，优化生成器参数

$$L = -\sum_{x_i \in D_{tr}} \sum_{t=k_i}^{|q_i|-1} logP_{\theta,\omega}(q_{i,t}|q_{i,1}, \cdots, q_{i,t-1})$$

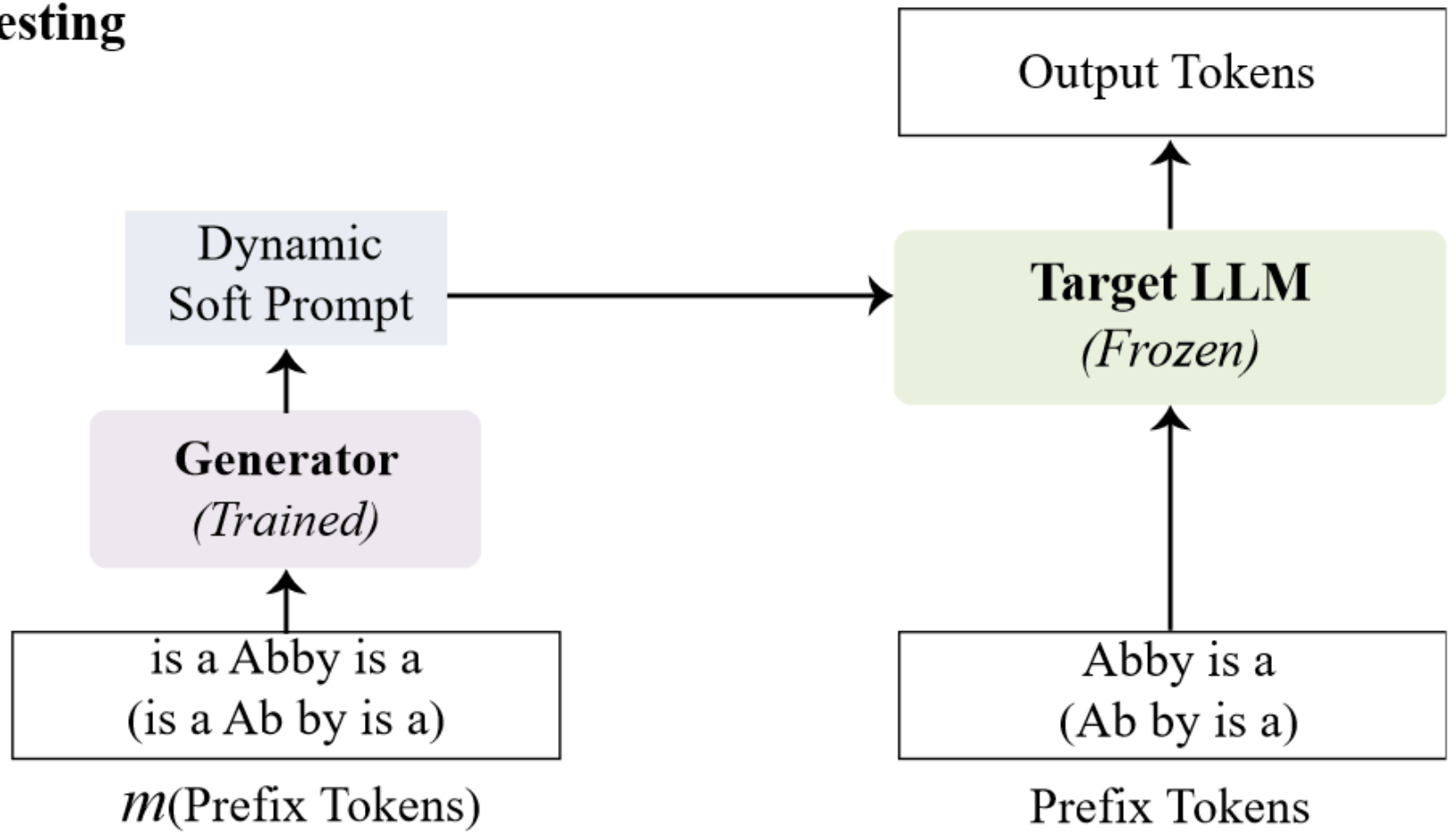
  - 其中，$q_i = [o_i, E(p_i), E(s_i)]$，$\theta$为目标模型参数，$\omega$是生成器参数，$k_i$是后缀$s_i$的起始索引



DSP 算法原理图

- 攻击阶段
  - 对测试集合的每个前缀p_i，生成器生成软提示o_i
  - 将o_i、前缀$E(p_i)$拼接输入目标模型
  - 目标模型获得输出token

$$y_i = f_\theta([o_i, E(p_i)])$$



实验设计　对比实验

- 目标模型
  - GPT-Neo（125M、1.3B、2.7B）
    - 基于 Pile 数据集预训练的文本生成模型
  - Pythia（410M、1.4B、2.8B、6.9B）
    - 基于 Pile 数据集预训练的文本生成模型
  - StarCoderBase（1B、3B、7B）
    - 基于 The Stack 数据集预训练的代码生成模型，支持 80 多种编程语言
- 对比算法
  - No Prompt：仅使用前缀令牌输入 LLM
  - Constant Hard Prompt：前置固定硬提示（取 LLM 词汇表的前 N 个token）
  - Dynamic Hard Prompt：基于前缀映射生成提示（不经过生成器）
  - CSP：使用固定软提示（SOTA）

实验设计　对比实验

- 数据集
  - Language Model Extraction Benchmark（文本生成任务， Pile 子集）
  - the-stacksmol（代码生成任务，The Stack 子集）
- 默认参数
  - 提示长度、前缀长度、后缀长度均设置为50
- 评估指标
    - Exact ER（精确提取率）： 生成的输出token与后缀完全匹配的比例
    - Fractional ER（部分提取率）： 生成的输出token与后缀部分匹配的比例
    - Test Loss（测试损失）
    - Test Preplexity（测试困惑度）
      - 与Test Loss用于衡量生成结果的整体拟合程度

实验设计　对比实验

- 文本生成任务
  - GPT-Neo 系列：所提方法在所有模型规模上均显著优于基线
    - 125M 模型的 Exact ER 从 0.189（No Prompt）提升至 0.42
    - 2.7B 模型的 Exact ER 从 0.54 提升至 0.702
  - Pythia 系列：同样全面超越基线
    - 410M 模型的 Exact ER 从 0.236 提升至 0.513
    - 6.9B 模型的 Exact ER 从 0.561 提升至 0.702

Table 2: Main Results on GPT-Neo Suite

Model	Method	Dynamic Prompt?	Exact ER	Exact ER Gain	Fractional ER	Fractional ER Gain	Test Loss	Test Perplexity (PPL)
GPT-Neo (125M)	No Prompt	N/A	0.189	N/A	0.369	N/A	0.953	2.594
	Constant Hard Prompt	✗	0.144	-23.81%	0.326	-11.65%	1.002	2.725
	Dynamic Hard Prompt	✓	0.056	-70.37%	0.153	-58.54%	1.122	3.071
	CSP (Ozdayi et al., 2023)	✗	0.239	26.46%	0.421	14.09%	0.858	2.359
	Ours	✓	**0.421**	**122.75%**	**0.557**	**50.89%**	**0.665**	**1.945**
GPT-Neo (1.3B)	No Prompt	N/A	0.46	N/A	0.643	N/A	0.202	1.224
	Constant Hard Prompt	✗	0.392	-14.78%	0.581	-9.64%	0.24	1.271
	Dynamic Hard Prompt	✓	0.1	-78.26%	0.194	-69.83%	0.394	1.483
	CSP (Ozdayi et al., 2023)	✗	0.532	15.65%	0.698	8.55%	0.133	1.142
	Ours	✓	**0.651**	**41.52%**	**0.772**	**20.04%**	**0.114**	**1.121**
GPT-Neo (2.7B)	No Prompt	N/A	0.54	N/A	0.702	N/A	0.127	1.135
	Constant Hard Prompt	✗	0.473	-12.41%	0.651	-7.26%	0.158	1.171
	Dynamic Hard Prompt	✓	0.117	-78.33%	0.213	-69.66%	0.291	1.338
	CSP (Ozdayi et al., 2023)	✗	0.641	18.70%	0.779	10.97%	0.084	1.087
	Ours	✓	**0.702**	**30.00%**	**0.820**	**16.83%**	**0.075**	**1.077**

Table 3: Main Results on Pythia Suite

Model	Method	Dynamic Prompt?	Exact ER	Exact ER Gain	Fractional ER	Fractional ER Gain	Test Loss	Test Perplexity (PPL)
Pythia (410M)	No Prompt	N/A	0.236	N/A	0.458	N/A	0.473	1.605
	Constant Hard Prompt	✗	0.161	-31.78%	0.361	-21.18%	0.595	1.812
	Dynamic Hard Prompt	✓	0.039	-83.47%	0.119	-74.02%	0.704	2.022
	CSP (Ozdayi et al., 2023)	✗	0.318	34.75%	0.526	14.90%	0.392	1.48
	Ours	✓	**0.513**	**117.37%**	**0.683**	**49.09%**	**0.283**	**1.328**
Pythia (1.4B)	No Prompt	N/A	0.416	N/A	0.648	N/A	0.199	1.22
	Constant Hard Prompt	✗	0.293	-29.57%	0.526	-18.83%	0.288	1.333
	Dynamic Hard Prompt	✓	0.067	-83.89%	0.159	-75.46%	0.412	1.51
	CSP (Ozdayi et al., 2023)	✗	0.497	19.47%	0.714	10.16%	0.126	1.135
	Ours	✓	**0.617**	**48.32%**	**0.786**	**21.36%**	**0.109**	**1.115**
Pythia (2.8B)	No Prompt	N/A	0.517	N/A	0.735	N/A	0.144	1.155
	Constant Hard Prompt	✗	0.401	-22.44%	0.611	-16.87%	0.214	1.239
	Dynamic Hard Prompt	✓	0.091	-82.40%	0.198	-73.06%	0.33	1.39
	CSP (Ozdayi et al., 2023)	✗	0.585	13.15%	0.783	6.57%	0.090	1.094
	Ours	✓	**0.669**	**29.40%**	**0.827**	**12.57%**	**0.080**	**1.084**
Pythia (6.9B)	No Prompt	N/A	0.561	N/A	0.781	N/A	0.104	1.11
	Constant Hard Prompt	✗	0.446	-20.50%	0.674	-13.70%	0.165	1.179
	Dynamic Hard Prompt	✓	0.122	-78.25%	0.231	-70.42%	0.262	1.3
	CSP (Ozdayi et al., 2023)	✗	0.648	16.04%	0.831	6.67%	0.063	1.065
	Ours	✓	**0.702**	**25.13%**	**0.858**	**9.89%**	**0.062**	**1.064**

实验设计　对比实验

- 代码生成任务
  - 所提方法在代码生成任务中仍表现最优
    - 1B 模型的 Exact ER 从 0.062 提升至 0.082
    - 7B 模型的 Exact ER 从 0.091 提升至 0.11

Table 4: Main Results on StarCoderBase Suite

Model	Method	Dynamic Prompt?	Exact ER	Exact ER Gain	Fractional ER	Fractional ER Gain	Test Loss	Test Perplexity (PPL)
StarCoderBase (1B)	No Prompt	N/A	0.062	N/A	0.232	N/A	0.836	2.306
	Constant Hard Prompt	✗	0.035	-43.55%	0.206	-11.21%	0.959	2.608
	Dynamic Hard Prompt	✓	0.006	-90.32%	0.066	-71.55%	0.958	2.605
	CSP (Ozdayi et al., 2023)	✗	0.071	14.52%	0.235	1.29%	0.815	2.259
	Ours	✓	**0.082**	**32.26%**	**0.244**	**5.17%**	**0.806**	**2.238**
StarCoderBase (3B)	No Prompt	N/A	0.071	N/A	0.254	N/A	0.745	2.106
	Constant Hard Prompt	✗	0.043	-39.44%	0.232	-8.66%	0.834	2.302
	Dynamic Hard Prompt	✓	0.018	-74.65%	0.093	-63.39%	0.838	2.312
	CSP (Ozdayi et al., 2023)	✗	0.081	14.08%	0.249	-1.97%	0.734	2.084
	Ours	✓	**0.094**	**32.39%**	**0.268**	**5.51%**	**0.713**	**2.039**
StarCoderBase (7B)	No Prompt	N/A	0.091	N/A	0.277	N/A	0.67	1.954
	Constant Hard Prompt	✗	0.021	-76.92%	0.243	-12.27%	0.765	2.149
	Dynamic Hard Prompt	✓	0.037	-59.34%	0.137	-50.54%	0.744	2.104
	CSP (Ozdayi et al., 2023)	✗	0.1	9.89%	0.278	0.36%	0.657	1.928
	Ours	✓	**0.11**	**20.88%**	**0.289**	**4.33%**	**0.641**	**1.898**

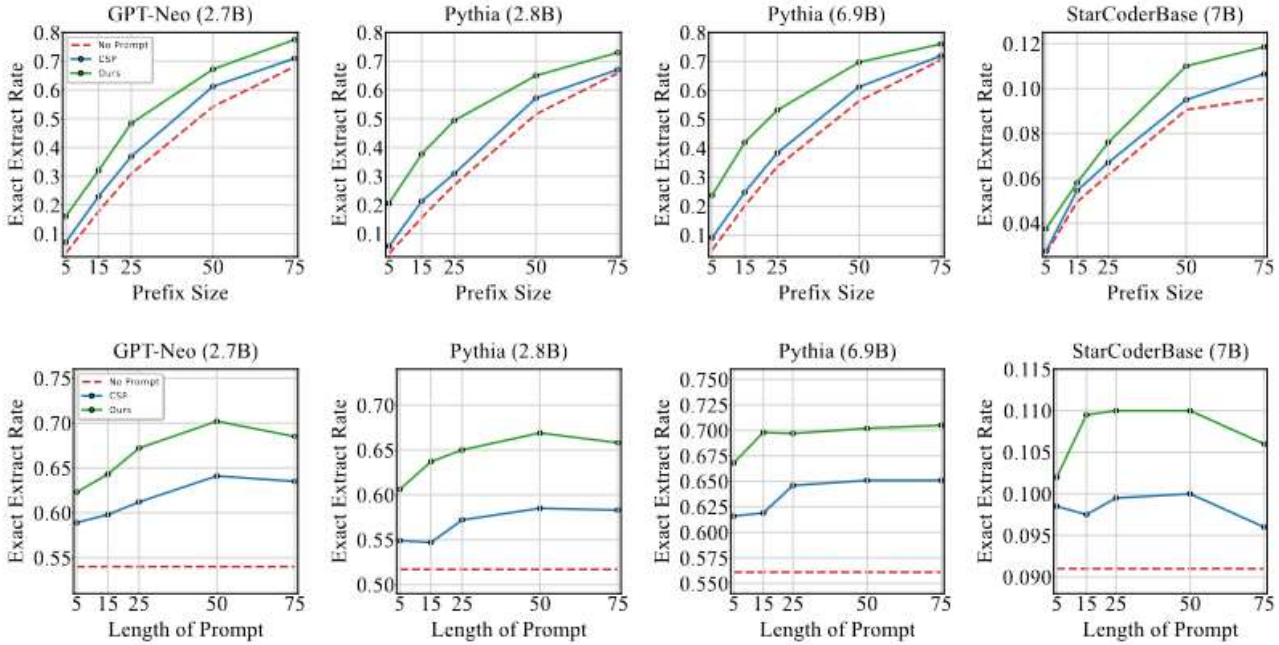
- 实验结论
  - 硬提示（Constant/Dynamic Hard Prompt）对记忆提取有害，其性能显著低于 No Prompt，说明硬提示无法有效激发 LLM 的记忆
  - 所提方法始终优于 SOTA 方法 CSP，证明动态软提示对记忆估计的重要性
  - LLM 的记忆能力随模型规模增大而增强（与现有研究一致），但小模型的记忆仍不可忽视（如 GPT-Neo 125M 在本文方法下 Exact ER 达 0.421）

实验设计　消融实验

- 消融实验
  - 动态软提示
    - 动态提示在所有模型上均显著更优
  - 前缀大小影响
    - 所提方法在所有前缀大小下均优于 No Prompt 和 CSP
    - 且随着**前缀增大**，Exact ER 逐渐提高；说明更长的前缀包含更多触发记忆的信息
  - 提示长度影响
    - 所提方法在所有提示长度下均占优
    - 提示长度从 5 增至 25 时，Exact ER 快速提升，超过 50 后趋于饱和

Table 5: Ablation Study on the Dynamics of Prompt

Model	Method	Is Dynamic Prompt?	Exact ER	Fractional ER
GPT-Neo (2.7B)	CSP	No	0.641	0.779
	Ours	No	0.630	0.765
	Ours	Yes	**0.702**	**0.820**
Pythia (2.8B)	CSP	No	0.585	0.783
	Ours	No	0.565	0.766
	Ours	**Yes**	**0.669**	**0.827**
StarCoderBase (3B)	CSP	No	0.081	0.249
	Ours	No	0.081	0.247
	Ours	**Yes**	**0.094**	**0.268**



算法总结 DSP

- 算法流程
  - 利用生成模型，依据**前缀生成软提示**
  - 拼接软提示和前缀得到输入
  - 输入目标模型获得泄露文本
- 算法优势
  - 软提示依赖于前缀变化，克服了固定软提示（如 CSP 方法）无法响应输入动态的局限性
- 算法不足
  - 仅关注预训练 LLM 对预训练数据的记忆，未验证在微调 LLM上的有效性

特点总结与未来展望

特点总结与未来展望

- **Zlib**
  - 首次并验证了大模型的隐私泄露风险
  - 对上下文的强依赖
  - 样本筛选方案均具有局限性
- **DSP**
  - 软提示依赖于前缀变化，克服了固定软提示无法响应输入动态的局限性
  - 仅关注预训练 LLM 对预训练数据的记忆，未验证在微调 LLM上的有效性
- 未来发展
  - 微调LLM的隐私泄露

预期收获 回顾分析

- 预期收获
  - 了解大模型面临的隐私泄露风险
  - 理解两种关于大模型隐私泄露攻击的算法
    - 大模型隐私泄露的开篇之作
    - 大模型隐私泄露的SOTA算法
  - 了解现有攻击方法研究的局限性和未来方向
    - 微调LLM

学会了吗?
学会了也不能攻击别人

参考文献

[1] Carlini N, Tramer F, Wallace E, et al. Extracting training data from large language models[C]. 30th USENIX security symposium (USENIX Security 21). 2021: 2633-2650.

[2] Wang Z, Bao R, Wu Y, et al. Unlocking memorization in large language models with dynamic soft prompting[J]. arXiv preprint arXiv:2409.13853, 2024.

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！