Scrub It Out! Erasing Sensitive Memorization in Code Language Models via Machine Unlearning

通过机器遗忘消除代码语言模型中的敏感记忆

https://arxiv.org/abs/2509.13755; ICSE2026

Huazhong University of Science and Technology; Zhejiang University; ...

汇报人：陈奕晨　　时间：2026-01-13

1. 研究动机

CLMs在代码生成、代码摘要等任务中表现出色，但存在一个关键的隐私漏洞：在训练过程中，会无意记忆训练数据中的敏感数据（如邮箱、密码、API密钥等）。并且在特定提示下，能做到逐字复制这些隐私信息，导致了隐私泄露的风险。

现有的方法存在几点局限：

1. 需要对模型重新训练，计算成本高
2. 无法针对特定数据进行遗忘

作者通过改进基于Gradient Ascent的遗忘方法，提出CodeEraser。
目的是保持模型有效性的同时，高效、有效地擦除CLMs中的敏感记忆。

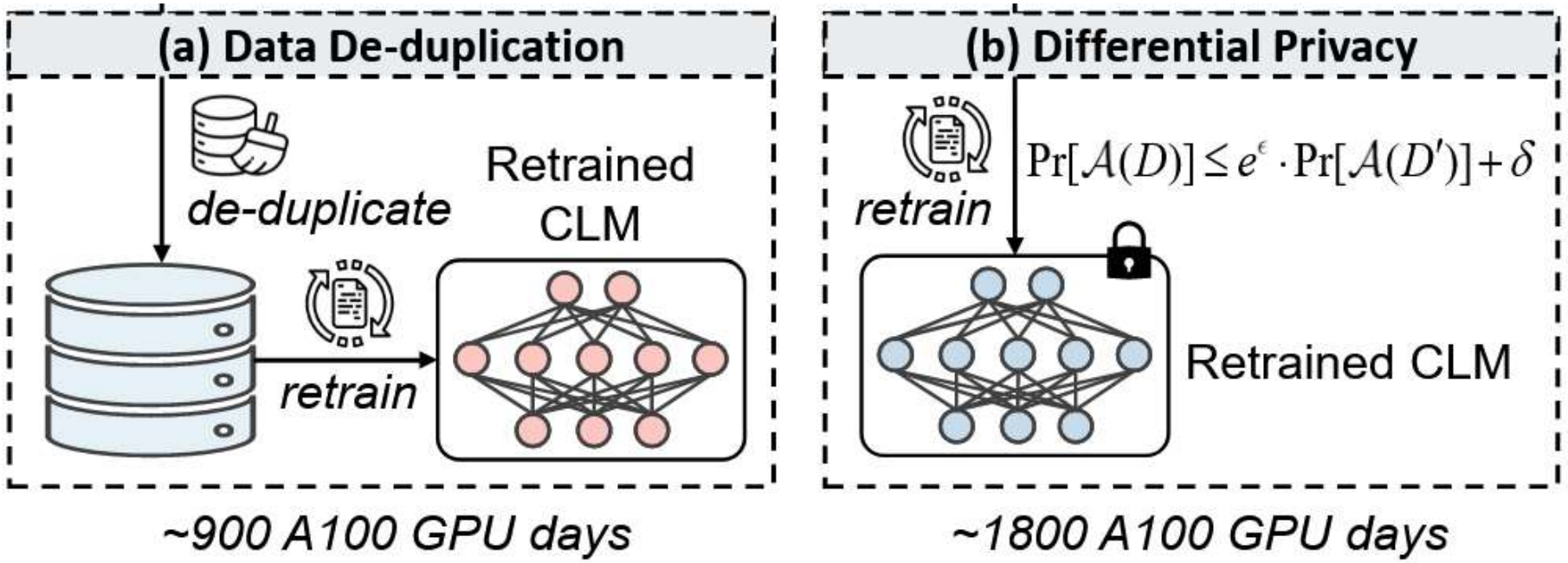
2. 背景、相关工作

由于CLMs通过从Github仓库里收集大量代码进行训练，会无意中记忆仓库在的敏感信息。

现有的方法：

1. Training Data De-duplication：降低敏感数据出现频率，可以减轻，但仍保留大量记忆能力
2. Differential Privacy：添加随机噪声，确保模型对单个数据的敏感性降低，达到隐私保护目的

但这两个方法都需要重新训练模型，成本高、耗时长



2. 背景、相关工作

逐字记忆

$$s = \arg\max_{\hat{s}} f_\theta(\hat{s} \mid p) \ \wedge \ [p \,||\, s] \in \mathcal{D} .$$

p是给定前缀，s是在该前缀下概率最大的后续序列，$\mathcal{D}$是训练数据集

EXAMPLE 1. Assume that the LM's training dataset contains a sequence "*# Copyright (C) [2003] Daniel <daniel@gmail.com>*". If the model is prompted with "*# Copyright (C) [2003] Daniel* ___" and the most likely continuation is "*<daniel@gmail.com>*", then the generated string is deemed memorized.

2. 背景、相关工作

Table 1: A toy example to illustrate the calculation process of MA and EL_n with $n = 3$.

Target Sequence (with 12 tokens): This file is part of EasyBuild created by the HPC team .					
Prefix $x_{<i}$	**MA = 6 / (12 - 1) = 0.5455**		**EL_3 = (0.2222 + 0.25 + 0.1429 + 0.1667 + 0.2 + 0 + 0 + 0 + 0) / (12 - 3) = 0.1091**		
	True Continuation x_i	**Generated Token $\hat{x}_i$**	**True Continuation $x_{\geq i}$** *3-gram match*	**Generated Sequence $\hat{x}_{\geq i}$**	**OVERLAP$_3$**
This	file *token match*	program	file is part of EasyBuild created by the HPC team .	program is part of pyNLO , which is created by the	2 / 9 = 0.2222
This file	is	is	is part of EasyBuild created by the HPC team .	is part of PyGithub created by the PYG team .	2 / 8 = 0.2500
This file is	part	part	part of EasyBuild created by the HPC team .	part of PyGithub created by the PYG team .	1 / 7 = 0.1429
… file is part	of	of	of EasyBuild created by the HPC team .	of PyGithub created by the PYG team .	1 / 6 = 0.1667
… is part of	EasyBuild	PyGithub	EasyBuild created by the HPC team .	PyGithub created by the PYG team .	1 / 5 = 0.2000
… of EasyBuild	created	.	created by the HPC team .	. EasyBuild is free software ;	0 / 4 = 0.0000
… created	by	by	by the HPC team .	by the VSC team ,	0 / 3 = 0.0000
… created by	the	the	the HPC team .	the VSC team ,	0 / 2 = 0.0000
… created by the	HPC	VSC	HPC team .	VSC team ,	0 / 1 = 0.0000
… by the HPC	team	team	-	-	-
… the HPC team	.	,	-	-	-

$$\mathrm{EL}_n(\mathbf{x}) = \frac{\sum_{i=2}^{N} \mathrm{OVERLAP}_n(\arg\max_{\hat{x}_{\geq i}} f_\theta(\hat{x}_{\geq i} \mid x_{<i}), x_{\geq i})}{N-n},$$

$$\mathrm{MA}(\mathbf{x}) = \frac{\sum_{i=2}^{N} \mathbb{1}\{\arg\max_{\hat{x}_i} f_\theta(\hat{x}_i \mid x_{<i}) = x_i\}}{N-1},$$

$$\mathrm{OVERLAP}_n(\mathbf{a}, \mathbf{b}) = \frac{\sum_{c \in ng(\mathbf{a})} \mathbb{1}\{c \in ng(\mathbf{b})\}}{|ng(\mathbf{a})|},$$

$$T_{\mathrm{MA}} = \frac{1}{|\mathcal{D}'|} \sum_{\mathbf{x}' \in \mathcal{D}'} \mathrm{MA}(\mathbf{x}'),\ T_{\mathrm{EL}_n} = \frac{1}{|\mathcal{D}'|} \sum_{\mathbf{x}' \in \mathcal{D}'} \mathrm{EL}_n(\mathbf{x}'),$$

2. 背景、相关工作

$$\mathrm{EL}_n(\mathbf{x}) = \frac{\sum_{i=2}^{N} \mathrm{OVERLAP}_n(\arg\max_{\hat{x}_{\geq i}} f_\theta(\hat{x}_{\geq i} \mid x_{<i}), x_{\geq i})}{N-n}, \qquad \mathrm{MA}(\mathbf{x}) = \frac{\sum_{i=2}^{N} \mathbb{1}\{\arg\max_{\hat{x}_i} f_\theta(\hat{x}_i \mid x_{<i}) = x_i\}}{N-1},$$

$$\mathrm{OVERLAP}_n(\mathbf{a}, \mathbf{b}) = \frac{\sum_{c \in ng(\mathbf{a})} \mathbb{1}\{c \in ng(\mathbf{b})\}}{|ng(\mathbf{a})|}, \qquad T_{\mathrm{MA}} = \frac{1}{|\mathcal{D}'|} \sum_{\mathbf{x}' \in \mathcal{D}'} \mathrm{MA}(\mathbf{x}'), \; T_{\mathrm{EL}_n} = \frac{1}{|\mathcal{D}'|} \sum_{\mathbf{x}' \in \mathcal{D}'} \mathrm{EL}_n(\mathbf{x}'),$$

$\mathcal{D}'$：Unseen Dataset

防止被模型学习过，并且保证高质量：选择stars > 500的代码仓库，且在对应CLM的release date之后创建。为了保证没有训练过先前版本的代码，选择没有经过CLM训练的用编程语言，如Ruby, PHP, Rust...

T_{MA}, T_{EL_n}：阈值，用于界定记忆程度

Table 2: Memorization thresholds for the studied CLMs.

CLM	MA (%)	EL_3 (%)	EL_5 (%)	EL_{10} (%)
CodeParrot-small	45.57	17.66	10.82	5.49
CodeParrot	46.34	16.56	10.17	5.14
CodeGen-350M-Mono	48.79	18.24	11.03	5.92
CodeGen-2B-Mono	53.61	19.32	11.71	6.28
Qwen2.5-Coder-7B	40.99	15.65	12.45	8.82

$$\mathcal{D}^f = \{\mathbf{x}^f\} \subset \mathcal{D}$$

$\mathcal{D}^f$是forgotten set，即需要被遗忘的数据，在machine unlearning后，要达到：

$$\mathrm{MA}(\mathbf{x}^f) \leq T_{\mathrm{MA}}, \; \mathrm{EL}_n(\mathbf{x}^f) \leq T_{\mathrm{EL}_n}.$$

2. 背景、相关工作

Autoregressive Language Modeling

$$\mathcal{L}^{LM}(\mathbf{x}) = -\log \prod_{i=1}^{N} f_\theta(x_i \mid x_1, \ldots, x_{i-1}) .$$

Vanilla Unlearning

$$\mathcal{L}^{GA}(\mathbf{x}^f) = -1 \cdot \mathcal{L}^{LM}(\mathbf{x}^f) = \log \prod_{i=1}^{N} f'_\theta(x_i^f \mid x_1^f, \ldots, x_{i-1}^f) .$$

Vanilla Unlearning是未经额外约束的直接方法，可能会不加选择地遗忘整个文本实例，不考虑代码的结构完整性和功能正确性，破坏代码的语法和功能

Constraint-Based Unlearning

$$\mathcal{L}^{KL}(\mathbf{x}^f,\mathbf{x}^r) = -\sum_{\mathbf{x}^f} KL(f_\theta(\mathbf{x}^f) \| f'_\theta(\mathbf{x}^f)) + \alpha \cdot \sum_{\mathbf{x}^r} KL(f_\theta(\mathbf{x}^r) \| f'_\theta(\mathbf{x}^r)), \qquad \mathcal{L}^{CU} = \mathcal{L}^{GA}(\mathbf{x}^f) + \lambda \cdot \mathcal{L}^{KL}(\mathbf{x}^f,\mathbf{x}^r),$$

加入KL散度。f_θ是初始模型，$f_\theta^{'}$是遗忘后的模型。在遗忘集x^f上，最大化f_θ和$f_\theta^{'}$之间的差距；在保留集x^r上，两者差异应该尽可能小。所以是在保证总体遗忘目标的基础上引入了限制。虽然优于Vanilla Unlearning，仍可能会遗忘非敏感内容，影响代码的局部结构

3. 方法

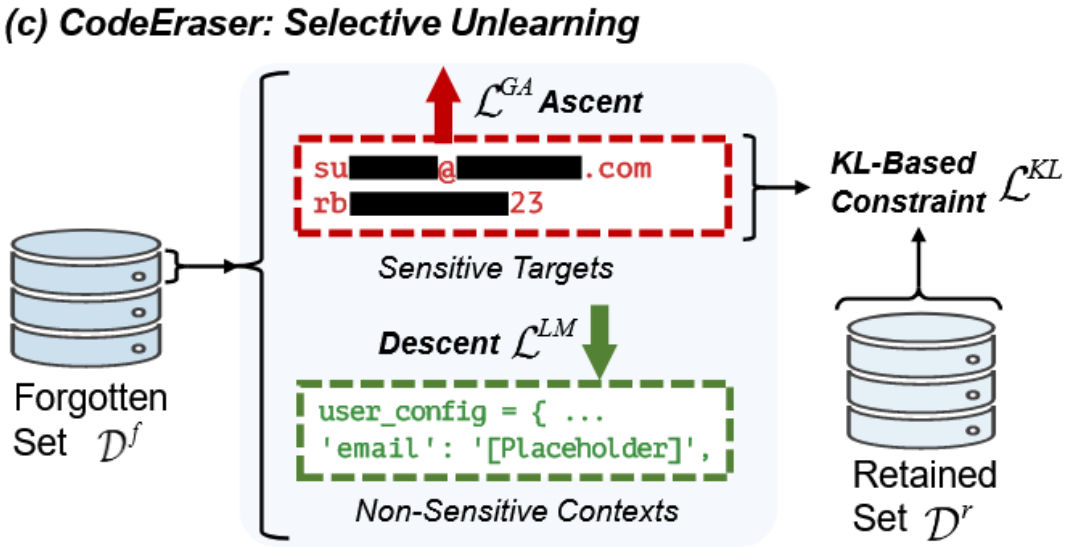
CodeEraser: Proposed Selective Unlearning

使用检测工具（如detect-secrets）识别代码中的敏感元素，将每个遗忘样本$x^f = (x_1^f, x_2^f, \ldots, x_N^f) \in \mathcal{D}^f$分为：敏感序列$s^f = (s_1^f, s_2^f, \ldots, s_m^f)$和非敏感序列$c^f = (c_1^f, c_2^f, \ldots, c_n^f)$

1. 在s^f上进行梯度上升，实现遗忘
2. 在c^f上梯度下降，保持模型完整性
3. 加入s^f和x^r之间的KL散度，确保遗忘不影响非敏感部分

$$\mathcal{L}^{SU} = (\mathcal{L}^{GA}(\mathbf{s}^f) + \gamma \cdot \mathcal{L}^{LM}(\mathbf{c}^f)) + \lambda \cdot \mathcal{L}^{KL}(\mathbf{s}^f, \mathbf{x}^r),$$

EXAMPLE 4. Given a piece of code snippet "*user_config = {'email': 'daniel@gmail.com', 'password': 'ABC'}*", the sensitive segments s^f are "*daniel@gmail.com*" and "*ABC*", while the non-sensitive contexts c^f are "*user_config = {'email': '[placeholder]', 'password': '[placeholder]'}*". This segmentation preserves the original sequential structure required by autoregressive CLMs.



4. 实验

Forgotten Set：各个CLM在相应的敏感记忆数据集上随机采用k个实例，并进行5次独立遗忘取平均值
k取{8, 16, 32, 64, 128, 256, 512}

Retained Set：BigQuery提供了1000个非敏感样本。对每个CLM，在其中随机抽取k个组成保留集

4. 实验 Effectiveness and Efficiency

Table 3: Evaluation of unlearning effectiveness. All values are reported as percentages (with % symbol omitted).

CLM	Method	MA	EL_3	EL_5	EL_{10}	Red.
CodeParrot -small	Original	99.74	98.55	98.12	97.97	-
	Threshold	*45.57*	*17.66*	*10.82*	*5.49*	-
	GA	30.71	10.17	7.07	4.18	86.85
	CU	22.14	7.79	6.19	4.72	89.69
	CodeEraser	18.69	6.69	5.78	5.18	90.82
CodeParrot	Original	99.69	98.90	98.36	97.62	-
	Threshold	*46.34*	*16.56*	*10.17*	*5.14*	-
	GA	27.53	6.33	4.21	3.47	89.54
	CU	24.18	6.11	4.39	3.09	90.48
	CodeEraser	15.22	6.36	5.40	4.65	92.01
CodeGen -350M-Mono	Original	99.25	97.14	96.39	95.93	-
	Threshold	*48.79*	*18.24*	*11.03*	*5.92*	-
	GA	25.53	8.45	6.98	4.95	88.29
	CU	18.65	6.98	5.73	4.88	90.75
	CodeEraser	45.13	11.44	7.05	3.46	82.96
CodeGen -2B-Mono	Original	99.89	99.79	99.76	99.70	-
	Threshold	*53.61*	*19.32*	*11.71*	*6.28*	-
	GA	17.95	6.80	5.52	4.83	91.21
	CU	11.80	6.40	5.99	5.54	92.55
	CodeEraser	31.66	10.01	7.73	6.05	86.11
Qwen2.5 -Coder-7B	Original	96.26	84.71	81.07	75.15	-
	Threshold	*40.99*	*15.65*	*12.45*	*8.82*	-
	GA	24.15	14.23	10.49	8.24	83.55
	CU	16.63	8.77	6.84	5.48	89.16
	CodeEraser	8.49	4.93	3.99	3.68	93.89

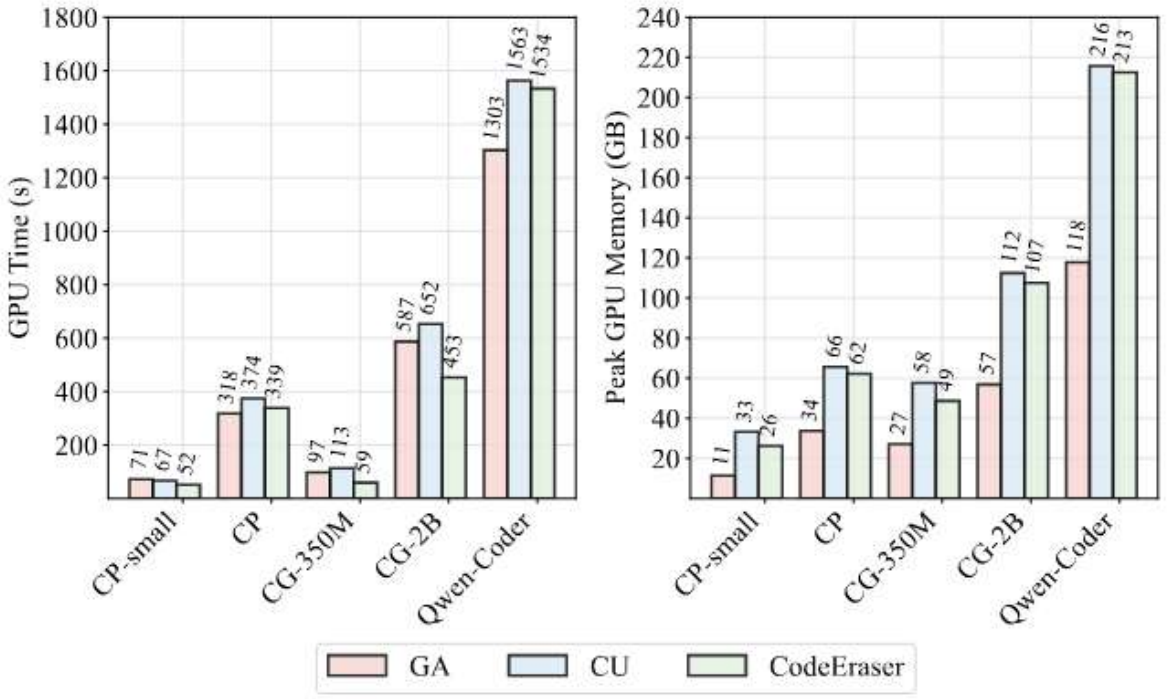


Figure 5: Evaluation of unlearning efficiency.

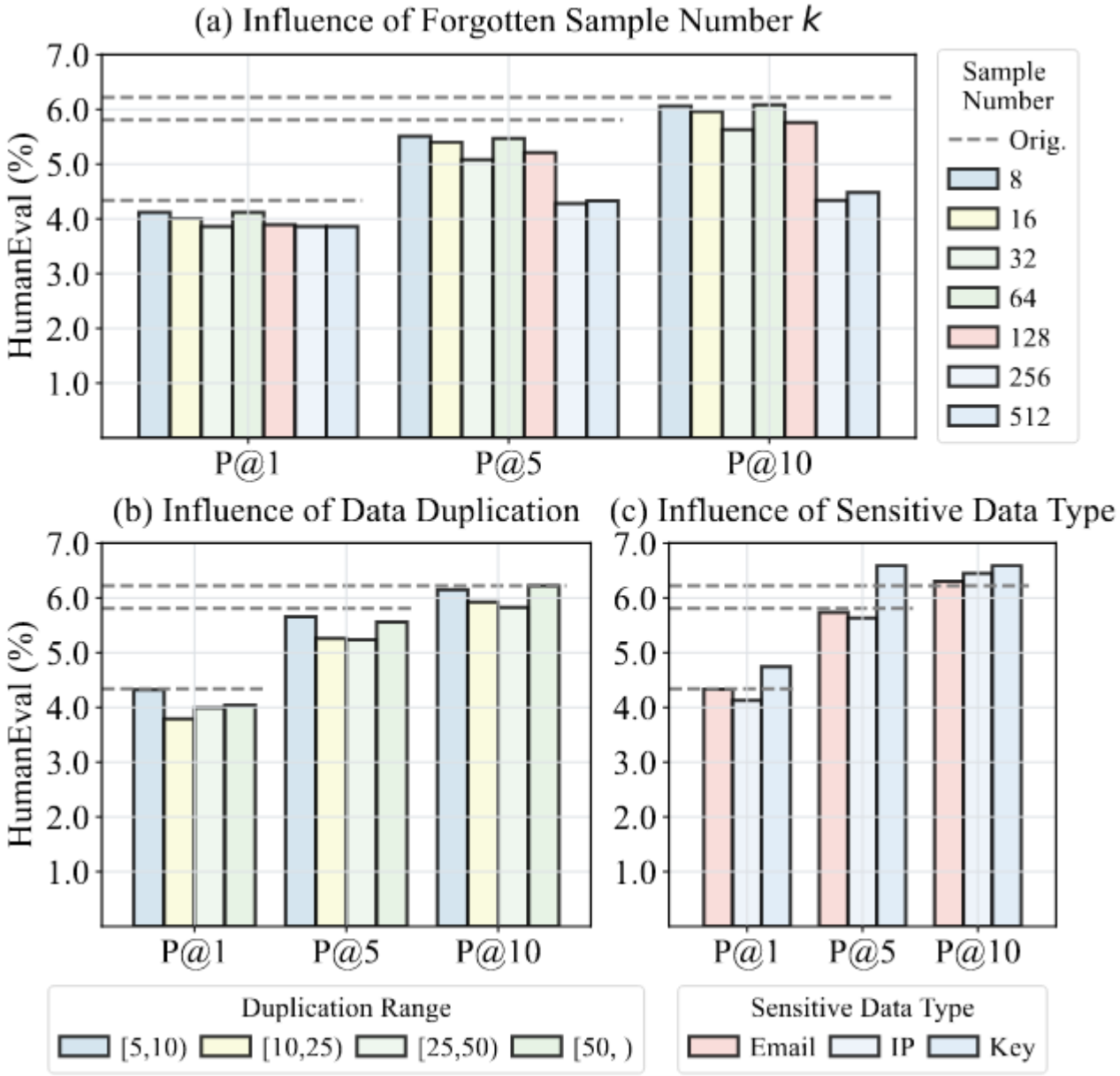
4. 实验

Model Utility Post-Unlearning

CLM	Method	P@1↑	P@5↑	P@10↑	Ret.↑
CodeParrot -small	Original	3.48	4.56	4.96	-
	GA	2.14	3.02	3.20	64.08
	CU	2.62	3.43	3.66	74.77
	CodeEraser	**3.74**	**4.59**	**4.87**	**102.10**
CodeParrot	Original	4.34	5.81	6.22	-
	GA	2.08	3.29	3.83	55.38
	CU	2.04	2.94	3.28	50.11
	CodeEraser	**3.86**	**5.08**	**5.63**	**88.96**
CodeGen -350M-Mono	Original	13.37	18.79	21.12	-
	GA	11.68	16.59	18.51	87.76
	CU	10.79	14.91	16.41	79.25
	CodeEraser	**13.36**	**18.02**	**19.96**	**96.78**
CodeGen -2B-Mono	Original	24.72	31.49	34.16	-
	GA	21.20	28.34	31.40	89.23
	CU	20.57	27.73	30.63	86.98
	CodeEraser	**23.00**	**29.91**	**32.94**	**94.82**
Qwen2.5 -Coder-7B	Original	61.07	73.61	77.23	-
	GA	40.67	53.81	57.63	71.44
	CU	48.54	64.70	69.59	85.83
	CodeEraser	**61.65**	**73.41**	**76.69**	**99.99**

4. 实验

Analysis on Forgotten Data



Impact of Hyperparameters

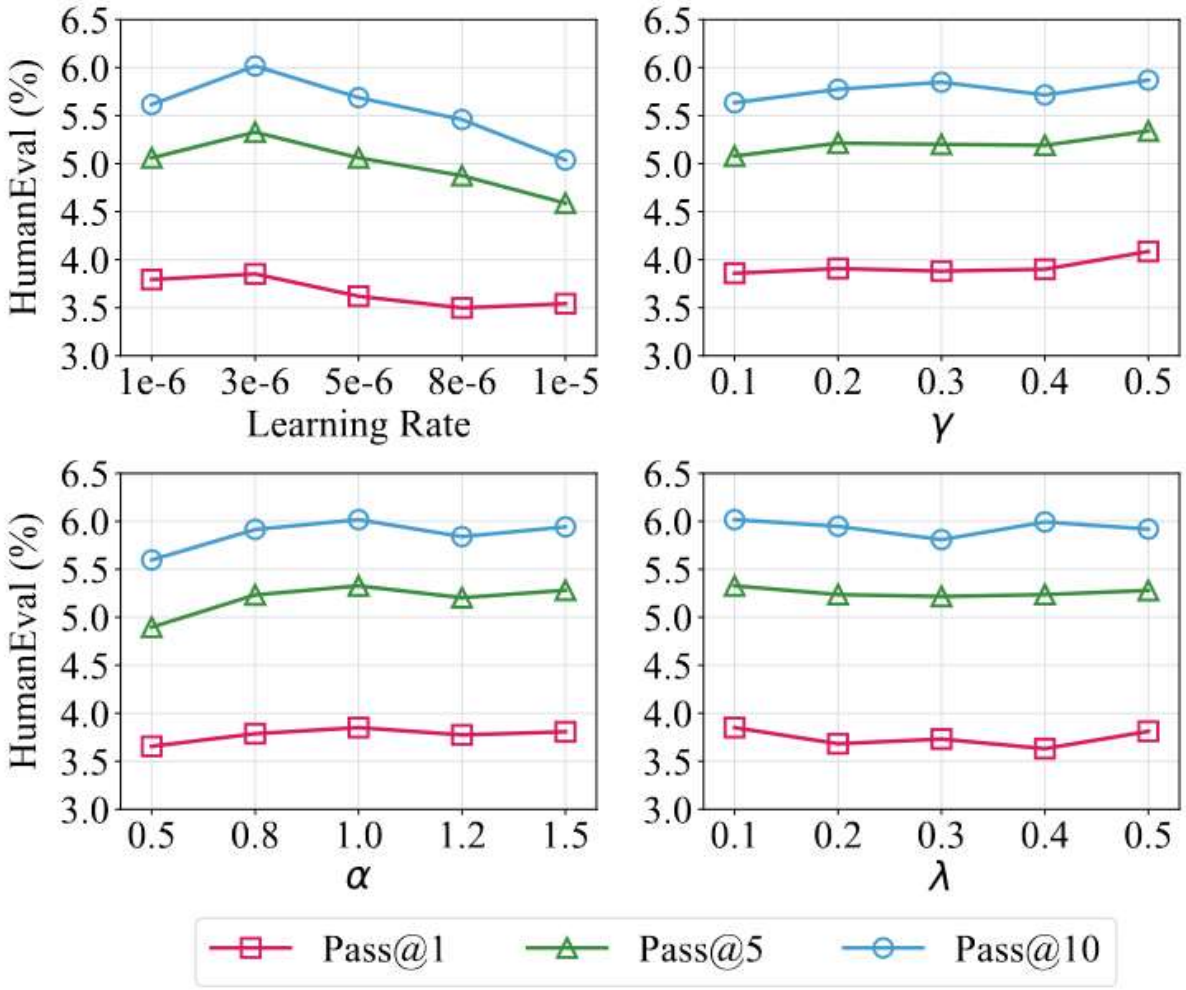


Figure 7: Parameter analysis of learning rate, γ, α, and λ.

Thanks for Your Watching !

通过机器遗忘消除代码语言模型中的敏感记忆

https://arxiv.org/abs/2509.13755; ICSE2026

Huazhong University of Science and Technology; Zhejiang University; ...

汇报人：陈奕晨　　时间：2026-01-13