CoSPED: Consistent Soft Prompt Targeted Data Extraction and Defense

Zhuochen Yang, Kar Wai Fok, Vrizlynn L. L. Thing

Cybersecurity Strategic Technology Centre, ST Engineering, Singapore
yang0761@e.ntu.edu.sg, fok.karwai@stengg.com, vriz@ieee.org

Abstract

Large language models have gained widespread attention recently, but their potential security vulnerabilities, especially privacy leakage, are also becoming apparent. To test and evaluate for data extraction risks in LLMs, we propose CoSPED, short for Consistent Soft Prompt Targeted Data Extraction and Defense. We introduce several innovative components, including Dynamic Loss, Additive Loss, Common Loss, and Self Consistency Decoding Strategy, and tested to enhance the consistency of the soft prompt tuning process. Through extensive experimentation with various combinations, we achieved an extraction rate of 65.2% at a 50-token prefix comparison. Our comparisons of CoSPED with other reference works confirm our superior extraction rates. We evaluate CoSPED on more scenarios, achieving Pythia model extraction rate of 51.7% and introducing cross-model comparison. Finally, we explore defense through Rank-One Model Editing and achieve a reduction in the extraction rate to 1.6%, which proves that our analysis of extraction mechanisms can directly inform effective mitigation strategies against soft prompt-based attacks.

Extended version — https://arxiv.org/abs/2510.11137

Introduction

With the growing popularity of ChatGPT (Wu et al. 2023; Yan et al. 2023) and other Large Language Models (LLMs) (Devlin et al. 2019; Radford et al. 2019), their remarkable capabilities come with emerging security and privacy risks (Carlini et al. 2021; Yao et al. 2024). Recent works indicate that LLMs tend to memorize training data, resulting in considerable potential for leakage (Carlini et al. 2022b), which raises an urgent need for deeper analysis and effective mitigation strategies.

Our work focuses on targeted training data extraction, a white-box attack where adversaries have full knowledge of model internals. By studying such worst-case scenarios, we aim to uncover how and why LLMs memorize sensitive data, ultimately helping guide the development of effective defenses. Figure 1 presents an example of targeted training data extraction using prefixes and suffixes that could leak critical privacy information.

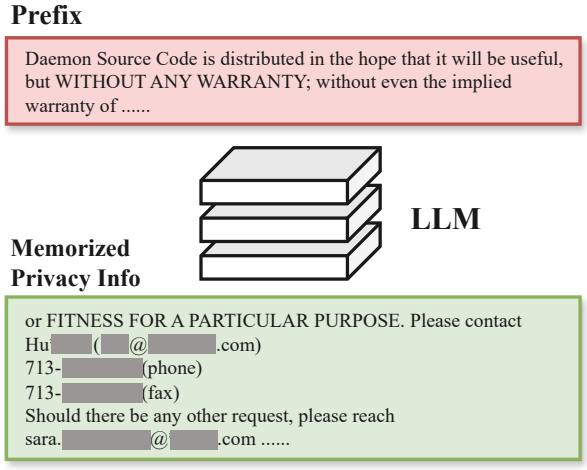

Figure 1: Example of Privacy Extraction in Prefix and Suffix.

Soft prompts (Lester, Al-Rfou, and Constant 2021) offer a light-weight handle for such attacks because they guide model behavior without changing weights. The Controllable Language Model (CLM) framework (Ozdayi et al. 2023) offers an initial treatment of this problem by training soft prompts for extraction attacks. Ethicist (Zhang, Wen, and Huang 2023) strength this area with smoothing loss and calibrated confidence estimation to prioritize high-risk tokens.

From the analysis of previous works, we observe that generation-based extraction methods often produce highly variable or incoherent outputs across repetitions, loss configurations, and decoding choices. These findings reveal three concrete gaps: (1) Most approaches optimize a simple loss in isolation, so gradients are dominated by noisy tokens and different random seeds lead to inconsistent prompts. (2) Decoding strategies are typically selected heuristically and decoupled from the loss design, causing low consistency between generated outputs and memorized training data. (3) Evaluations usually stay within one model family and rarely connect extraction behavior back to deployable defenses, limiting our ability to translate insights into mitigation.

To address these challenges, we propose CoSPED, *Consistent Soft Prompt Targeted Data Extraction and Defense*. Our main contributions are summarized as follows:

- We propose CoSPED, a framework for targeted data extraction based on soft prompts, integrating consistency-driven loss functions and decoding strategies.
- We design three novel loss functions, including Dynamic Loss, Additive Loss, and Common Loss, and a Self-Consistency Decoding (SCD) strategy, and explore 16 loss combinations. Each enhances different aspects of prompt tuning stability and extraction performance.
- We conduct extensive studies on multiple open-source LLMs, including GPT-Neo series and Pythia series, introducing model structural related influence.
- We implement and evaluate a defense mechanism based on the ROME model editing, demonstrating its effectiveness in mitigating soft prompt-based extraction attacks.
- We confirm how insights gained from extraction attacks can be leveraged to develop targeted defenses and inform future mitigation efforts.

Related Works

Privacy Risks in LLMs

Large-scale pre-trained language models often unintentionally memorize training data, leading to privacy risks where sensitive information can be leaked (Carlini et al. 2021, 2022b; Tirumala et al. 2022). Recent surveys and large-scale extraction pipelines show that this vulnerability persists even in production systems (Ishihara 2023; Nasr et al. 2025) and can be amplified through decomposition-based decoding or other attack heuristics (Su et al. 2024; Yu et al. 2023). These works emphasize that extraction robustness must be evaluated beyond a single model or decoding configuration.

Various studies have shown that models can inadvertently store and reveal personal information, which can be extracted through targeted queries (Carlini et al. 2019; Lehman et al. 2021). Defense efforts include data deduplication (Kandpal, Wallace, and Raffel 2022), sensitive information deletion (Patil, Hase, and Bansal 2024), and differential privacy (Abadi et al. 2016; McMahan et al. 2018), but these strategies remain insufficient against evolving attacks (Yao et al. 2024). Thus, frameworks are required that can rigorously evaluate extraction across various scenarios.

Extraction attacks focus on recovering memorized data, including model stealing (Kariyappa, Prakash, and Qureshi 2021; Truong et al. 2021), gradient leakage (Li et al. 2023), and training data extraction (Ozdayi et al. 2023; Yang et al. 2024). Membership inference attacks (Shokri et al. 2017), on the other hand, attempt to determine whether a particular data record was used in the training dataset of a model(Carlini et al. 2022a). While membership inference attacks determine whether a point was used for training (Truex et al. 2019; Carlini et al. 2022a; Fu et al. 2025), training-data extraction aims to reconstruct the underlying content, making CoSPED complementary to but distinct from membership-focused prompt tuning.

Prompt Tuning for Data Extraction

Prompt tuning (Lester, Al-Rfou, and Constant 2021) optimizes continuous input embeddings (soft prompts) to guide model behavior without modifying model parameters (Li and Liang 2021). Unlike text prompts, soft prompts directly manipulate input at the embedding level.

Recent works have exploited soft prompts for data extraction attacks. CLM (Ozdayi et al. 2023) trains soft prompts to extract memorized suffixes given prefixes, achieving improved extraction rates over baselines. Ethicist (Zhang, Wen, and Huang 2023) enhances extraction with a smoothing loss that optimizes generation probabilities for high-loss tokens and uses calibrated confidence estimation for suffix selection. Follow-up efforts propose decoding heuristics and prompt-inversion tricks to further stabilize extraction (Yu et al. 2023; Zhang, Carlini, and Ippolito 2024). However, these methods still suffer from high variance across runs, focus on a single model family, or do not offer a unified view that connects extraction and defense.

Overall, prior works have primarily focused on simple loss optimization and heuristic decoding within a single model family. In the following section, we introduce CoSPED, a consistency-driven framework that addresses these limitations in data extraction and defense.

Methodology

We begin by formalizing the problem setting and notations. Let X denote a continuous token sequence from the pre-training corpus of a large language model (LLM). Each sequence X is divided into a prefix P containing k_P tokens and a suffix S containing k_S tokens. We represent training corpus as $D = \{X_i(P_i, S_i)\}$, where each sequence X_i is paired with its corresponding prefix-suffix pair. The attacker-accessible split is denoted as $D_a = \{X_i(P_i, S_i) \mid X_i, P_i, S_i \in D\}$. This dataset enumerates all prefix-suffix pairs available for soft prompt tuning, ensuring that optimization covers the same statistical distribution as memorized corpus. We assume D contains commonly used public sources such as *Books3* (Zhu et al. 2015), *Common Crawl*, and *The Pile* (Gao et al. 2020), making this setup realistic for large-scale pre-trained models. The objective is to recover memorized suffixes S given corresponding prefixes P by optimizing soft prompt without modifying model weights.

Overview

Our approach employs a trainable soft prompt $\mathcal{Z} \in \mathbb{R}^{K \times d}$, where K is the prompt length and d is the model's embedding dimension. During training, $\mathcal{Z}$ is concatenated with prefixes P_i sampled from D_a, and only $\mathcal{Z}$ is optimized to minimize the loss $\mathcal{L}$ for generating the corresponding suffix S_i, while all LLM parameters remain frozen. After training, the optimized soft prompt is evaluated on a disjoint test split D_p, where prefixes P_p are used to generate predicted suffixes S_p. The outputs are compared against ground-truth suffixes using exact-match and token-level accuracy metrics to measure extraction success. Figure 2 provides an overview of this pipeline, illustrating the soft prompt initialization, optimization process, and evaluation workflow.

Threat Model

We consider a targeted training-data extraction adversary with white-box access to the victim LLM. The attacker can

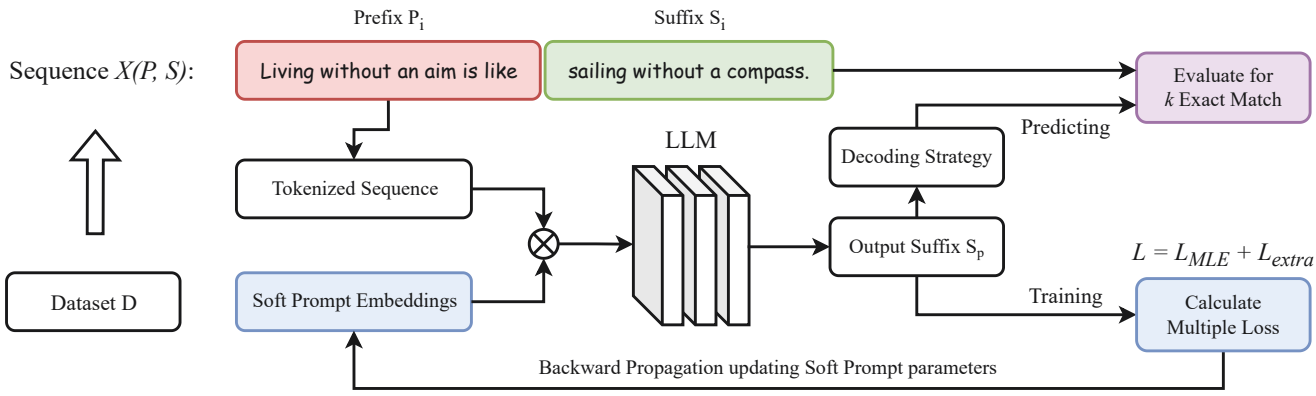

Figure 2: Procedure of data extraction. Training and predicting share a roughly similar process, with prefix P given with soft prompt embeddings in the input, which then generates suffix S_p for prediction. In training, the loss L is calculated, and backward propagation updates soft prompts, while in predicting, the k exact match result is evaluated.

inspect model parameters, gradients, and internal activations, and is permitted to tune continuous soft prompts $\mathcal{Z}$ while keeping model weights frozen. They may query the model repeatedly but cannot alter pre-training data or access user-specific logs. The goal is to reconstruct memorized suffixes S corresponding to given or guessed prefixes P. Our defense evaluation adopts the same assumptions to examine how editing internal representations can reduce extraction success rate while preserving normal downstream utility.

Loss Exploration

By default, an LLM is trained with a maximum likelihood estimation (MLE) loss. Besides MLE, we explore several additional losses, including smooth loss, focal loss, dynamic loss, additive loss, and common loss, some of which consist of the best $\mathcal{L}_{\text{extra}}$ shown in Fig. 2, which can either enhance or replace the original MLE loss.

S represents the suffix sequence the model is trying to generate, and MLE loss is calculated based on each t_i token loss, which is influenced by soft prompts $\mathcal{Z}$, input prefix P, and previously generated tokens $t_{<i}$.

Ethicist proposed an extra loss function called Smooth loss, which is meant to increase the top N lowest loss values and additionally optimize the generation probabilities for these N tokens. It gives improvements while combining with original $\mathcal{L}_{\text{MLE}}$, but in our method, we are exploring more combinations of different losses, including smooth loss, to test for more potential.

While MLE and Smooth loss provide baselines, they have limitations: MLE can lead to exposure bias, and Smooth loss only considers fixed k tokens. To address these, we introduce focal loss and three novel loss functions.

Focal loss Focal loss (Lin et al. 2017), equation as $\mathcal{L}_{\text{Focal}} = -\frac{1}{|S|}\sum_{i=1}^{|S|}\alpha_t(1-p_i)^\gamma \log p_i$, is a modification of standard cross-entropy loss, primarily designed to address the class imbalance problem. In our scenario, focal loss focuses on penalizing hard-to-classify tokens more than easier tokens, thereby emphasizing learning from difficult instances. Focal loss reweights each token-level log likelihood with α_t and the focusing parameter γ to emphasize hard examples. The probability term satisfies $p_i = P_{\text{M}}(t_i \mid \mathcal{Z}, P, t_{<i}, t_{<i} \in S)$ and captures how likely the frozen model is to emit token t_i when conditioned on the soft prompt and the observed prefix tokens.

Dynamic Loss In smooth loss, the top N lowest loss values are increased, but N is a fixed value, which cannot censor the dynamic loss changes. Thus, we propose dynamic loss in Eq. 1, which is meant to optimize the flaws in smooth loss. k_{dy}, which represents our targeting top tokens amount with the highest loss, changes dynamically.

$$\mathcal{L}_{\text{Dy}} = -\frac{1}{k_{\text{dy}}}\sum_{i=1}^{k_{\text{dy}}}\log P_{\text{M}}(t_{\delta(i)} \mid \mathcal{Z}, P, t_{<\delta(i)}). \tag{1a}$$

$$k_{\text{dy}} = \begin{cases} N & \text{if } \mathcal{L}_{\text{MLE}} \le T_D, \\ N + \alpha(\mathcal{L}_{\text{MLE}} - T_D) & \text{if } \mathcal{L}_{\text{MLE}} > T_D. \end{cases} \tag{1b}$$

Equation 1a averages the log probabilities of the highest-loss tokens $t_{\delta(i)}$ identified in the current batch, whereas Equation 1b adapts the number of tracked tokens k_{dy} once the MLE loss $\mathcal{L}_{\text{MLE}}$ exceeds a tolerance T_D. When training becomes unstable, the schedule raises k_{dy} to widen the focus on difficult tokens, ensuring that the optimizer allocates capacity wherever the model is most uncertain.

Additive Loss Additive loss penalizes the model more for specific error-prone tokens. We evaluated the list of error-prone tokens from training, and we found the majority of error-prone tokens are more commonly used tokens, and they are widely generalized in various usages. We count these tokens and include them in the $K_{\mathcal{M}}$ set as the basis for additive loss.

$$\mathcal{L}_{\text{Addi}} = -\frac{1}{K_{\mathcal{M}}}\sum_{i=1}^{K_{\mathcal{M}}}\sum_{t\in\mathcal{M}(x_i)}\alpha\log P_{\text{M}}(t \mid \mathcal{Z}, P). \tag{2}$$

$K_{\mathcal{M}}$ represents the total number of error-prone tokens, x_i denotes the i_{th} input example, and $\mathcal{M}(x_i)$ is the set of

error-prone tokens in x_i. The term $\log P_{\text{M}}(t \mid \mathcal{Z}, P)$ is the log probability of token t given the context $\mathcal{Z}$ and model parameters P. Scaling factor α controls the weight of additive loss, ensuring the penalty is proportionate. By summing over these tokens and averaging over $K_{\mathcal{M}}$, the model receives a consistent and normalized penalty, encouraging it to improve predictions for commonly wrongly predicted tokens, leading to better performance and generalization.

Common Loss Based on the analysis of L_{Addi} and the frequency of token errors, we found frequently mispredicted tokens are often among the most commonly used tokens. This led to Common loss, penalizing frequently used tokens to improve overall performance.

$$\mathcal{L}_{\text{Comm}} = -\frac{1}{K_{\mathcal{C}}} \sum_{i=1}^{K_{\mathcal{C}}} \sum_{t \in \mathcal{C}(x_i)} \beta \log P_{\text{M}}(t \mid \mathcal{Z}, P). \quad (3)$$

Eq. 3 normalizes the loss by the number of elements in $K_{\mathcal{C}}$, representing the top 10% most frequently used tokens. It sums the weighted log-probabilities of tokens t in the context $\mathcal{C}(x_i)$ of each element x_i. Here, β is a weighting factor adjusting the influence of each term. $K_{\mathcal{C}} = \{k \in K_{\text{Vocab}} \mid \text{rank}(k) \leq 0.1 \times |K_{\text{Vocab}}|\}$ defines $K_{\mathcal{C}}$ as the set of tokens in the vocabulary K_{Vocab} that are within top 10% in terms of usage frequency, with $\text{rank}(k)$ providing frequency ranks and $|K_{\text{Vocab}}|$ being all tokens in model tokenizer's vocabulary.

Loss Combination Combining loss functions allows the model to optimize multiple objectives simultaneously. For our losses, the base losses are MLE loss and focal loss, computed on all tokens. The additional loss components (smooth, dynamic, additive, and common loss) focus on different token subsets to improve specific aspects. Base loss is chosen as either MLE or focal, while multiple loss components can be combined together.

Decoding Strategy

Common decoding strategies like Greedy, Top-k (Fan, Lewis, and Dauphin 2018), and Top-p (Holtzman et al. 2020) have limitations for data extraction tasks. Thus, we propose Self Consistency decoding in Eq. 4, inspired by (Wang et al. 2023).

Self Consistency decoding strategy initially contains a sequence generation stage, with multiple sequences generated using our soft prompt model with traditional decoding strategy hyperparameters, including greedy, beam, etc. Then, with a consistency evaluation, the above-generated sequences are grouped in batches of N_{SCD} size. Each batch's consistency evaluation function is applied to determine the most consistent sequence, which ranks the sequences based on a defined consistency metric, selecting the top sequence from each batch. Finally, the best sequence from each batch is identified and collected.

$$\mathcal{D}(S'_i) = \left| \frac{|\{S_{i,j} : S_{i,j} \neq \textit{EOS}\}|}{|\{S_{i,j} \neq \textit{EOS}\}|} - \mathcal{D}_{\text{Optimal}} \right|. \quad (4)$$

Equation 4 computes the absolute deviation between each candidate sequence's diversity and the desired target. The diversity score for each sequence $\mathcal{D}(S'_i)$ is calculated by first excluding the end-of-sequence tokens, denoted as *EOS*. For each sequence, we consider the set of unique tokens $\{S_{i,j}\}$, where $S_{i,j}$ denotes the j_{th} token in the i_{th} sequence. The diversity of a sequence is then determined by the ratio of the number of unique tokens to the total number of tokens in the sequence, excluding *EOS* tokens. The diversity score is calculated as the absolute difference between this diversity ratio and the desired diversity value, referred to as $\mathcal{D}_{\text{Optimal}}$. After computing the diversity scores for all sequences, the sequences are sorted based on these scores. The sequences with diversity scores closest to the optimal value are considered the most consistent. By focusing on consistency, Self Consistency reduces the likelihood of incoherent or irrelevant text generation. Also, using multiple candidate sequences and a consistency check ensures that the selected sequence is the most reliable.

Experiments

In this section, we conduct a series of experiments to evaluate CoSPED. We begin by introducing the environment, datasets, models, and evaluation metrics used in our experiments. Following this, we perform various experiments, including a loss comparison to assess different loss function components, an analysis of different decoding strategies, and a comprehensive comparison with other methods.

Environment & Dataset

Our experiments were conducted on 3 RTX 3090 GPUs (24GB memory), using PyTorch 2.2.2 and Python 3.8.12 in a Conda environment on Ubuntu 22.04.4 LTS.

We utilize two datasets from the LM Extraction Benchmark (Carlini et al. 2024), originally from The Pile (Gao et al. 2020). These datasets consist of NumPy token arrays, tokenized using the GPT-2 tokenizer, which is compatible with GPT-Neo. Dataset $\mathcal{D}_1$ contains 15,000 sequences (50-token prefix and 50-token suffix), used by Ethicist (Zhang, Wen, and Huang 2023); Dataset $\mathcal{D}_2$ has 16,000 sequences (150-token prefix and 50-token suffix), partially used by CLM (Ozdayi et al. 2023).

Multiple Model Testing

We extend CoSPED to evaluate across different models. Previous works (Ozdayi et al. 2023; Zhang, Wen, and Huang 2023) commonly select GPT-Neo 1.3B, trained on *The Pile*, as it shares tokenizer used in LM Extraction Benchmark. Its widespread usage makes it fit for baseline comparison.

We further evaluate CoSPED on Pythia 1.4B model (Biderman et al. 2023). Pythia is a model family trained on *The Pile* with open weights and reproducible settings. We note that prior works have not explored CoSPED-like extraction tasks on Pythia. To adapt CoSPED, we implemented a padding strategy for input alignment across models with different tokenizer vocab sizes. Our preprocessing ensures compatibility without affecting output validity.

Evaluation Metrics

We use an evaluation metric to measure the data extraction success rate from the training dataset. Following (Ozdayi

Base $\mathcal{L}$	Loss Components				$\mathbf{ER}_{50}$	$\mathbf{ER}_{30}$
	$\mathcal{L}_{\text{Smooth}}$	$\mathcal{L}_{\text{Dy}}$	$\mathcal{L}_{\text{Addi}}$	$\mathcal{L}_{\text{Comm}}$		
$\mathcal{L}_{\text{MLE}}$					61.2	65.9
$\mathcal{L}_{\text{MLE}}$	✓				62.7	68.9
$\mathcal{L}_{\text{MLE}}$		✓			63.0	69.2
$\mathcal{L}_{\text{MLE}}$	✓		✓		**64.0**	69.5
$\mathcal{L}_{\text{MLE}}$	✓			✓	62.8	69.0
$\mathcal{L}_{\text{MLE}}$		✓	✓		63.5	68.9
$\mathcal{L}_{\text{MLE}}$		✓		✓	62.9	69.4
$\mathcal{L}_{\text{MLE}}$	✓		✓	✓	**64.6**	**69.9**
$\mathcal{L}_{\text{MLE}}$		✓	✓	✓	63.8	69.0
$\mathcal{L}_{\text{Focal}}$	✓				63.3	69.3
$\mathcal{L}_{\text{Focal}}$		✓			63.0	69.2
$\mathcal{L}_{\text{Focal}}$	✓		✓		**64.9**	**69.7**
$\mathcal{L}_{\text{Focal}}$	✓			✓	62.6	68.2
$\mathcal{L}_{\text{Focal}}$		✓	✓		63.8	68.8
$\mathcal{L}_{\text{Focal}}$		✓		✓	61.2	68.3
$\mathcal{L}_{\text{Focal}}$	✓		✓	✓	63.7	**70.0**
$\mathcal{L}_{\text{Focal}}$		✓	✓	✓	63.5	68.7

Table 1: CoSPED Experiment Results of Different Loss Components on GPT-Neo 1.3B Model on Dataset $\mathcal{D}_1$

et al. 2023; Zhang, Wen, and Huang 2023), we formalize the metric as *Exact Extraction Rate*.

We define Exact Extraction Rate ER_k as the proportion of cases where the first k tokens of the generated suffix exactly match the ground-truth suffix of the same length. Larger k measures the model's ability to generate longer coherent text, while smaller k reflects precision in replicating shorter splits. For instance, ER_{50} evaluates full 50-token suffix matches; ER_{30} compares the first 30 tokens only.

This metric offers a strict and quantitative assessment of a model's vulnerability, as partial or similar generations are considered incorrect.

Loss Comparison

The experiment results presented in Table 1 provide a comparison of various loss components, evaluated using two metrics: ER_{50} and ER_{30}. Each result represents the average of five repeated sessions. This experiment is conducted using Dataset $\mathcal{D}_1$. The loss components explored in this study vary between base loss functions and loss components, including our proposed losses and existing losses. Base loss choices are MLE and Focal Loss, both commonly used losses. Loss components are Smooth Loss, Dynamic Loss, Additive Loss, and Common Loss. Smooth loss is a preexisting loss by Ethicist, and the others are our novel proposal.

The best performance combines Focal Loss, Smooth Loss, and Additive Loss, achieving ER_{50} at 64.9 and ER_{30} at 69.7. Some key findings can be concluded as:

Decoding Strategy	Soft Prompt Length	Prefix	$\mathbf{ER}_{50}$
Self Consistency	100	50	**64.2**
Beam Search			**64.5**
Top-p			62.3
Top-k			62.1
Greedy			58.7
Diverse Beam Search			40.8

Table 2: CoSPED Experiment Results of Decoding Strategy

- Additive Loss improves ER_{50} by 1.3% when added to MLE and Smooth loss, and it keeps appearing in all best-performing sets by penalizing error-prone tokens.
- Dynamic Loss increases ER_{50} by 0.8% when added to MLE loss, adapting to data complexity.
- Focal Loss with Smooth Loss improves ER_{50} by 0.6% over MLE loss, focusing on hard-to-classify tokens.
- Common Loss achieves the highest $\text{ER}_{30} = 70.0$ when combined with Focal, Smooth, and Additive loss.

These effects align with our consistency-oriented objective. Smooth loss maintains pressure on a fixed set of hard tokens, whereas Dynamic loss expands the set whenever the current batch becomes harder, preventing variance spikes between runs. Additive loss denoises the gradient by emphasizing tokens that repeatedly cause mistakes, so the dynamic schedule does not chase incidental noise. Finally, combining Focal loss with mild label smoothing sharpens attention on truly difficult tokens while avoiding overconfidence on easy ones, which stabilizes optimization and sets up the self-consistency decoding with less noise.

The overall conclusion of loss experiment is that multiple loss combinations consistently outperform single losses, with different combinations providing various focuses.

Decoding Strategy

Table 2 compares decoding strategies using MLE loss with 100-token soft prompts and 50-token prefixes on GPT-Neo 1.3B with Dataset $\mathcal{D}_1$.

Beam Search achieves the highest ER_{50} at 64.5%, but our Self Consistency strategy achieves comparable results on ER_{50} at 64.2% with better efficiency (13.4ms vs 18.7ms per generation). Self Consistency evaluates consistency across multiple outputs, providing faster generation without compromising quality.

Comparison against Prior Works

Table 4 compares CoSPED with prior methods on GPT-Neo 1.3B using Dataset $\mathcal{D}_1$.

CoSPED achieves ER_{50} at 65.2%, outperforming Ethicist (62.3%) by 2.9% and CLM (54.3%) by 10.9%. Compared to baselines, CoSPED surpasses Perplexity, zlib Comparing, and Original Model by 13.9%, 15.5%, and 20.2%, respectively. These results strengthen CoSPED's leading advantage compared with prior works.

Soft Prompt Length	Prefix	ER_{50}			ER_{30}			ER_{10}		
		CoSPED	Ethicist	CLM	CoSPED	Ethicist	CLM	CoSPED	Ethicist	CLM
20	50	55.7	53.2	50.0	62.7	60.1	57.0	74.3	73.3	71.8
	100	79.5	77.9	75.8	83.4	81.8	79.9	91.2	90.0	88.6
	150	94.8	93.0	91.9	96.3	95.7	94.1	98.3	97.9	97.1
100	50	64.5	61.2	53.6	68.9	66.5	59.7	78.2	76.8	74.3
	100	80.8	79.3	76.2	84.3	83.0	80.5	91.5	90.6	89.0
	150	95.1	94.5	92.5	97.1	96.4	94.4	98.6	97.9	96.8
150	50	62.1	59.9	54.7	67.0	64.7	60.1	79.0	77.8	73.1
	100	80.6	80.2	79.5	85.0	84.7	84.2	92.1	92.0	91.7
	150	93.9	93.6	91.0	95.9	95.0	93.1	98.3	98.2	97.3

Table 3: Experiment Results Comparison of CoSPED, Ethicist, and CLM on GPT-Neo 1.3B model on Dataset $\mathcal{D}_2$

Method	Soft Prompt	Prefix	ER_{50}	ER_{30}
CoSPED	100	50	$\mathbf{65.2 \pm 0.2}$	$\mathbf{69.9 \pm 0.3}$
Ethicist			62.3 ± 0.5	67.1 ± 0.6
CLM			54.3 ± 0.7	59.2 ± 0.7
Perplexity	N.A.	50	51.3 ± 0.2	53.2 ± 0.1
zlib			49.7 ± 0.2	51.8 ± 0.2
Original			45.0 ± 0.3	48.6 ± 0.2

Table 4: Experiment Results of Methods and Baselines

Soft Prompt	Prefix	ER_{50}	ER_{30}	ER_{10}
20	50	41.9	54.8	66.7
	100	59.6	74.7	82.5
	150	60.1	74.9	82.3
100	50	51.7	65.5	73.8
	100	61.9	74.8	82.0
	150	65.0	78.1	84.7
150	50	51.8	63.9	74.0
	100	66.1	81.0	87.2
	150	65.4	80.3	87.2

Table 5: CoSPED Experiment Results on Pythia 1.4B model

Discussion

CoSPED VS. Ethicist and CLM

Table 3 compares CoSPED, Ethicist, and CLM across different soft prompt and prefix lengths on Dataset $\mathcal{D}_2$. The extraction rates are evaluated across ER_{50}, ER_{30}, and ER_{10}.

Firstly, CoSPED consistently outperforms Ethicist and CLM across all settings, with improvements ranging from 1.5% to 10.3%. What's more, prefix length has stronger impact than soft prompt length. This is evidenced across all methods. For example, for CoSPED with a 20-token soft prompt, increasing the prefix from 50 to 100 tokens improves ER_{50} by 23.8%, while keeping the 50-token prefix but increasing the soft prompt from 20 to 100 tokens only adds 8.8%. Also, soft prompt length has a preferred upper limit, because soft prompt length shows diminishing returns beyond 100 tokens in all methods.

Pythia Results

Table 5 shows Pythia 1.4B results using tokenized dataset from Dataset $\mathcal{D}_2$ with padding. It is notable that Pythia model family shows different patterns from GPT-Neo series.

Firstly, increasing soft prompt length in Pythia model testing continues to improve results. However, it exhibits a similar trend to GPT-Neo, in that soft prompts beyond 100 tokens have a minor effect on improving the Exact Extraction Rate.

Notably, Pythia 1.4B is more resilient to extraction than GPT-Neo 1.3B, achieving lower extraction rates among all similar settings. This could be evidenced because Pythia model uses rotary positional embeddings for better contextual encoding, parallel residual connections for more stable information flow, and a larger intermediate size (8192) enabling richer representations. Unlike GPT-Neo's alternating global/local attention, Pythia applies full attention consistently, which improves its ability to model prompt-context interactions robustly. These design choices, taken together, make it harder for soft prompts to manipulate Pythia's outputs effectively.

Similarly, for both Pythia and GPT-Neo, increasing prefix length improves extract rate across all settings, with prefix length having stronger impact than soft prompt length.

Defense Evaluation

In this section, we discuss methods for preventing data extraction attacks on language models. Recent works (Patil, Hase, and Bansal 2024) have shown that ROME (Meng et al. 2022) and MEMIT (Meng et al. 2023) model editing methods can reduce memory leakage. We adapt ROME for our use case involving prefix-suffix pairs, as privacy extraction attacks exploit memorized training sequences.

Modified ROME Method for Prefix-Suffix Pairs

ROME as Rank-One Model Editing, adjusts model representations to make specific single-token outputs less likely. However, our scenario uses 50-token prefix inputs with corresponding 50-token suffix outputs, requiring adaptation for longer sequences.

We focus on the first suffix token following the prefix. This choice is motivated by: (1) exact-match evaluation re-

Model	LB PPL	LB Acc	δ Acc
Original Model	42.69	0.2766	0
ROME Edited (PPL 6)	43.58	0.2598	-0.04
ROME Edited (PPL 10)	95.59	0.1247	0.06

Table 6: LAMBADA PPL, Acc, and δ Acc Performance for Original GPT-Neo 1.3B model and ROME Edited models

Method	$\mathbf{ER}_5$	$\mathbf{ER}_{10}$	$\mathbf{ER}_{25}$	$\mathbf{ER}_{40}$	$\mathbf{ER}_{50}$
CoSPED (ROME)	1.6	0.5	0.1	0	0
Ethicist (ROME)	1.5	0.4	0.1	0	0
CoSPED	85.2	78.2	70.5	66.6	63.9
Ethicist	84.2	78.1	70.4	65.9	61.2

Table 7: Privacy Leakage Methods Exact Extraction Rates

quires continuous correct generation from initials; (2) autoregressive models exhibit cascade effects where early errors propagate; (3) disrupting initial token strongly prevents full sequence extraction. For example, given "Living without an aim is like", we target model's tendency to generate "sailing" from memorized suffix "sailing without a compass."

Defense Evaluation Metrics

To evaluate the strength of our model editing defense, we use Delta Accuracy (δ Acc) and LAMBADA benchmark (LB).

δ Accuracy measures the change in frequency of generating a specific target output before and after editing. A near-zero δ Acc indicates successful suppression of memorized suffixes while preserving general knowledge.

Model utility is assessed using LAMBADA dataset (Paperno et al. 2016), which evaluates long-range language modeling through accuracy and perplexity metrics. LAMBADA Benchmark assesses long-range language modeling using two metrics: Accuracy (LB Acc), measuring the proportion of correctly predicted final words in context-rich passages, and Perplexity (LB PPL), evaluating model's fluency and linguistic reasoning. Higher accuracy and lower PPL indicate better language modeling performance.

Early Stopping Criteria

The unedited GPT-Neo exhibits training perplexity (TR PPL) $\sim$5.65. Extending editing beyond 500 epochs increases TR PPL to $\sim$20, indicating significant model degradation. We establish two perplexity-based stopping criteria:

- TR PPL 6: At 116 epochs, δ Acc shifts from 0 to -0.04, minimally impacting general knowledge while reducing target generation.
- TR PPL 10: At 331 epochs, δ Acc changes to 0.06, representing aggressive editing with higher capability degradation risk.

Table 6 shows LAMBADA performance across models. It is evident from the results that PPL 6 preserves performance with minimal degradation, while PPL 10 causes significant drops. Thus, we select PPL 6 as optimal.

Privacy Attack Evaluation

We evaluate against soft prompt extraction attacks using 1000 test prefix-suffix pairs. Table 7 shows Exact Extraction Rates for CoSPED and Ethicist before and after ROME defense on GPT-Neo 1.3B model.

Our ROME method reduces extraction rates from up to 64% at ER_{50} to near zero across both attacks. Defense effectiveness increases with longer token-required matches, successfully disrupting autoregressive generation. The minimal changes in δ Accuracy and LB evaluations confirm that our modified ROME defense selectively targets memorized content without impacting general capabilities.

Limitations and Future Works

Although we performed multiple model tests, soft prompt requires retraining. One potential improvement is to explore transferability of soft prompt embeddings, referenced in (Kim et al. 2023; Maus et al. 2023). Specifically, projecting soft embeddings to vocabulary tokens via Euclidean distance (Maus et al. 2023) provides theoretical support for universal usage through hard token conversion.

Moreover, developing more robust defense mechanisms to protect LLMs from soft prompt data extraction attacks is also critical for future work. By analyzing how these attacks exploit vulnerabilities in the model's architecture and training data, we hope to identify strategies to mitigate or prevent unauthorized data access, making LLMs more resilient to such attacks in real-world scenarios.

Conclusion

In this work, we proposed novel loss functions and decoding strategies that improve extraction performance and enhance the understanding of data leakage risks. Through ablation studies, we identified optimal designs that maximize exact extraction rates, with CoSPED consistently achieving strong results across different metrics, proving the consistency between generated outputs and memorized training data.

Our various analyses highlighted CoSPED's superior performance compared to Ethicist and CLM on GPT-Neo model. Additional experiments on Pythia further validated the universality of CoSPED and indicated that prefix length has a greater impact on extraction performance than soft prompt length. These findings proved the growing risks of targeted data extraction from large language models.

To address this, we applied a modified Rank-One Model Editing method to defend against such extractions. Our defense proved effective against both CoSPED and Ethicist, while preserving the model's general linguistic behavior.

In summary, our work exposed the vulnerability of LLMs to soft prompt extraction attacks and introduced effective defense strategies. We hope this study encourages greater attention to prompt-based privacy risks and inspires future research in robust, model-aware protection techniques.

References

Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 308–318.

Biderman, S.; Schoelkopf, H.; Anthony, Q. G.; Bradley, H.; O'Brien, K.; Hallahan, E.; Khan, M. A.; Purohit, S.; Prashanth, U. S.; Raff, E.; et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, 2397–2430. PMLR.

Carlini, N.; Chien, S.; Nasr, M.; Song, S.; Terzis, A.; and Tramer, F. 2022a. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, 1897–1914. IEEE.

Carlini, N.; Choquette-Choo, C. A.; Ippolito, D.; Jagielski, M.; Lee, K.; Nasr, M.; Tramer, F.; and Zhang, C. 2024. A Benchmark for Training Data Extraction Attacks on Language Models.

Carlini, N.; Ippolito, D.; Jagielski, M.; Lee, K.; Tramer, F.; and Zhang, C. 2022b. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Carlini, N.; Liu, C.; Erlingsson, Ú.; Kos, J.; and Song, D. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX security symposium (USENIX security 19)*, 267–284.

Carlini, N.; Tramer, F.; Wallace, E.; Jagielski, M.; Herbert-Voss, A.; Lee, K.; Roberts, A.; Brown, T.; Song, D.; Erlingsson, U.; et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, 2633–2650.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Fan, A.; Lewis, M.; and Dauphin, Y. 2018. Hierarchical Neural Story Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 889–898.

Fu, W.; Wang, H.; Gao, C.; Liu, G.; Li, Y.; and Jiang, T. 2025. MIA-tuner: adapting large language models as pre-training text detector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 27295–27303.

Gao, L.; Biderman, S.; Black, S.; Golding, L.; Hoppe, T.; Foster, C.; Phang, J.; He, H.; Thite, A.; Nabeshima, N.; et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Holtzman, A.; Buys, J.; Du, L.; Forbes, M.; and Choi, Y. 2020. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*.

Ishihara, S. 2023. Training Data Extraction From Pre-trained Language Models: A Survey. In Ovalle, A.; Chang, K.-W.; Mehrabi, N.; Pruksachatkun, Y.; Galystan, A.; Dhamala, J.; Verma, A.; Cao, T.; Kumar, A.; and Gupta, R., eds., *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, 260–275. Toronto, Canada: Association for Computational Linguistics.

Kandpal, N.; Wallace, E.; and Raffel, C. 2022. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*, 10697–10707. PMLR.

Kariyappa, S.; Prakash, A.; and Qureshi, M. K. 2021. Maze: Data-free model stealing attack using zeroth-order gradient estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13814–13823.

Kim, S.; Yun, S.; Lee, H.; Gubri, M.; Yoon, S.; and Oh, S. J. 2023. Propile: Probing privacy leakage in large language models. *Advances in Neural Information Processing Systems*, 36: 20750–20762.

Lehman, E.; Jain, S.; Pichotta, K.; Goldberg, Y.; and Wallace, B. C. 2021. Does BERT Pretrained on Clinical Notes Reveal Sensitive Data? In *2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, 946–959. Association for Computational Linguistics (ACL).

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In Moens, M.-F.; Huang, X.; Specia, L.; and Yih, S. W.-t., eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3045–3059. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Li, C.; Song, Z.; Wang, W.; and Yang, C. 2023. A theoretical insight into attack and defense of gradient leakage in transformer. *arXiv preprint arXiv:2311.13624*.

Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 4582–4597.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Maus, N.; Chao, P.; Wong, E.; and Gardner, J. R. 2023. Black Box Adversarial Prompting for Foundation Models. In *The Second Workshop on New Frontiers in Adversarial Machine Learning*.

McMahan, H. B.; Ramage, D.; Talwar, K.; and Zhang, L. 2018. Learning Differentially Private Recurrent Language Models. In *International Conference on Learning Representations*.

Meng, K.; Bau, D.; Andonian, A.; and Belinkov, Y. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35: 17359–17372.

Meng, K.; Sharma, A. S.; Andonian, A. J.; Belinkov, Y.; and Bau, D. 2023. Mass-Editing Memory in a Transformer. In *The Eleventh International Conference on Learning Representations*.

Nasr, M.; Rando, J.; Carlini, N.; Hayase, J.; Jagielski, M.; Cooper, A. F.; Ippolito, D.; Choquette-Choo, C. A.; Tramèr, F.; and Lee, K. 2025. Scalable Extraction of Training Data from Aligned, Production Language Models. In *The Thirteenth International Conference on Learning Representations*.

Ozdayi, M.; Peris, C.; FitzGerald, J.; Dupuy, C.; Majmudar, J.; Khan, H.; Parikh, R.; and Gupta, R. 2023. Controlling the extraction of memorized data from large language models via prompt-tuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 1512–1521.

Paperno, D.; Kruszewski, G.; Lazaridou, A.; Pham, N.-Q.; Bernardi, R.; Pezzelle, S.; Baroni, M.; Boleda, G.; and Fernández, R. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, 1525–1534.

Patil, V.; Hase, P.; and Bansal, M. 2024. Can Sensitive Information Be Deleted From LLMs? Objectives for Defending Against Extraction Attacks. In *The Twelfth International Conference on Learning Representations*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, 3–18. IEEE.

Su, E.; Vellore, A.; Chang, A.; Mura, R.; Nelson, B.; Kassianik, P.; and Karbasi, A. 2024. Extracting memorized training data via decomposition. *arXiv preprint arXiv:2409.12367*.

Tirumala, K.; Markosyan, A.; Zettlemoyer, L.; and Aghajanyan, A. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35: 38274–38290.

Truex, S.; Liu, L.; Gursoy, M. E.; Yu, L.; and Wei, W. 2019. Demystifying membership inference attacks in machine learning as a service. *IEEE transactions on services computing*, 14(6): 2073–2089.

Truong, J.-B.; Maini, P.; Walls, R. J.; and Papernot, N. 2021. Data-free model extraction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4771–4780.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q. V.; Chi, E. H.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.

Wu, T.; He, S.; Liu, J.; Sun, S.; Liu, K.; Han, Q.-L.; and Tang, Y. 2023. A brief overview of ChatGPT: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5): 1122–1136.

Yan, Y.; Li, B.; Feng, J.; Du, Y.; Lu, Z.; Huang, M.; and Li, Y. 2023. Research on the impact of trends related to ChatGPT. *Procedia computer science*, 221: 1284–1291.

Yang, Z.; Zhao, Z.; Wang, C.; Shi, J.; Kim, D.; Han, D.; and Lo, D. 2024. Unveiling Memorization in Code Models. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24. ACM.

Yao, Y.; Duan, J.; Xu, K.; Cai, Y.; Sun, Z.; and Zhang, Y. 2024. A survey on large language model (LLM) security and privacy: The Good, The Bad, and The Ugly. *High-Confidence Computing*, 4(2): 100211.

Yu, W.; Pang, T.; Liu, Q.; Du, C.; Kang, B.; Huang, Y.; Lin, M.; and Yan, S. 2023. Bag of tricks for training data extraction from language models. In *International Conference on Machine Learning*, 40306–40320. PMLR.

Zhang, Y.; Carlini, N.; and Ippolito, D. 2024. Effective Prompt Extraction from Language Models. In *First Conference on Language Modeling*.

Zhang, Z.; Wen, J.; and Huang, M. 2023. ETHICIST: Targeted Training Data Extraction Through Loss Smoothed Soft Prompting and Calibrated Confidence Estimation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 12674–12687.

Zhu, Y.; Kiros, R.; Zemel, R.; Salakhutdinov, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *The IEEE International Conference on Computer Vision (ICCV)*.