Unlocking Memorization in Large Language Models with Dynamic Soft Prompting

Zhepeng Wang[1], Runxue Bao[2], Yawen Wu[3], Jackson Taylor[4], Cao Xiao[2], Feng Zheng[5], Weiwen Jiang[1], Shangqian Gao[6]*, Yanfu Zhang[4]*

[1]George Mason University, [2]GE Healthcare,
[3]University of Pittsburgh, [4]William and Mary,
[5]Southern University of Science and Technology, [6]Florida State University
[1]{zwang48, wjiang8}@gmu.edu, [2]{runxue.bao, cao.xiao}@gehealthcare.com,
[3]yawen.wu@pitt.edu, [4]{jttaylor01, yzhang105}@wm.edu,
[5]zfeng02@gmail.com, [6]sgao@cs.fsu.edu

Abstract

Pretrained large language models (LLMs) have excelled in a variety of natural language processing (NLP) tasks, including summarization, question answering, and translation. However, LLMs pose significant security risks due to their tendency to memorize training data, leading to potential privacy breaches and copyright infringement. Therefore, accurate measurement of the memorization is essential to evaluate and mitigate these potential risks. However, previous attempts to characterize memorization are constrained by either using prefixes only or by prepending a constant soft prompt to the prefixes, which cannot react to changes in input. To address this challenge, we propose a novel method for estimating LLM memorization using dynamic, prefix-dependent soft prompts. Our approach involves training a transformer-based generator to produce soft prompts that adapt to changes in input, thereby enabling more accurate extraction of memorized data. Our method not only addresses the limitations of previous methods but also demonstrates superior performance in diverse experimental settings compared to state-of-the-art techniques. In particular, our method can achieve the maximum relative improvement of 135.3% and 39.8% over the vanilla baseline on average in terms of *discoverable memorization rate* for the text generation task and code generation task, respectively. Our code is available at https://github.com/wangger/llm-memorization-dsp.

1 Introduction

Pretrained large language models (LLMs) have achieved remarkable success across a wide range of downstream natural language processing (NLP) tasks such as summarization (Zhang et al., 2024; Van Veen et al., 2024; Zhang et al., 2019), question answering (Pan et al., 2023; Shao et al., 2023; Louis et al., 2024; Jiang et al., 2021; Guo et al., 2023; Yasunaga et al., 2021) and translation (Zhang et al., 2023; Bawden and Yvon, 2023; He et al., 2024; Xue et al., 2020; Xu et al., 2023; Li et al., 2024), etc. The popularity of LLMs requires people to pay attention to the unique challenges they bring to security. One of the significant security issues is that LLMs can memorize a considerable portion of their training data even though they tend to not overfit to their training dataset due to the small number of training epochs (Radford et al., 2019). Moreover, the memorized data can be extracted by carefully designed input from attackers or even unintentional input from ordinary users, which can cause privacy and copyright issues with the sensitive training data (Carlini et al., 2021, 2023; Ozdayi et al., 2023; Nasr et al., 2023; Karamolegkou et al., 2023). For example, the confidential codes from Samsung can be exposed to other users after they were shared with OpenAI due to the memorization of LLMs (DeGeurin, 2023; Huynh, 2023).

The huge security risks and the potential uses of memorization make it important to measure the memorization of the target LLM. With an accurate method to quantify the intrinsic memorization of LLMs, model developers can have a better understanding of the model's vulnerability posed by its memorization and take actions such as machine unlearning (Yao et al., 2023; Pawelczyk et al., 2023; Yao et al., 2024) to mitigate the memorization before they release their LLMs to the public. Moreover, the method to extract memorized data can also be combined with the target LLM and leveraged by the users to detect whether their self-built dataset has data leakage issues when it is used to evaluate the target LLM.

To measure the intrinsic memorization of the target LLM, Carlini et al. (2023) first proposed a metric called *discoverable memorization rate* to serve as the estimation. As shown in Figure 1 (a), the given data are split into prefix tokens and suffix

*Co-corresponding authors.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 9782–9796
November 12-16, 2024

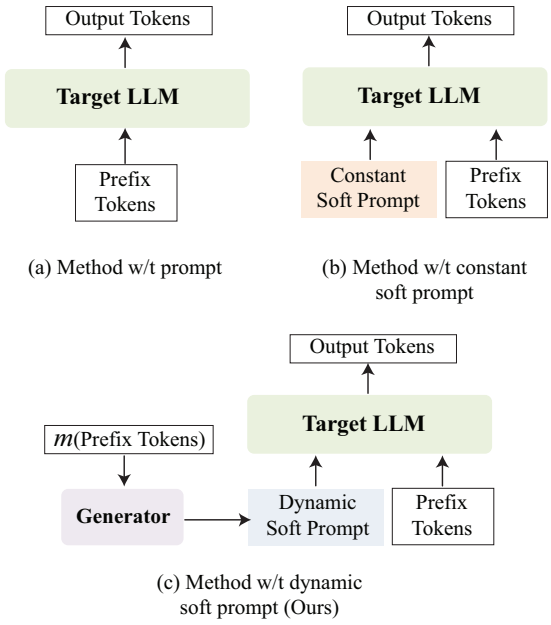


Figure 1: Conceptual comparison of three methods for extracting memorized data from the target LLM.

tokens and the prefix tokens are fed into the target LLM. The data are defined as discoverably memorized when the output tokens can match the suffix tokens verbatim. Ozdayi et al. (2023) proposed that more memorized data can be extracted via learning a soft prompt and prepending it to the prefix tokens for generation, which is shown in Figure 1 (b). However, the memorization of the target LLM is still underestimated even with the soft prompt since it is constant and invariant to prefix tokens, which may not help or even hinder extracting data when changing the prefix.

In this paper, we propose a new method to estimate the memorization of LLMs. Compared with constant soft prompts (Ozdayi et al., 2023), our method can generate prefix-dependent soft prompts and react to the changes in inputs. More specifically, a transformer-based generator is trained for the generation of the dynamic soft prompt. As shown in Figure 1 (c), it takes the outputs from the naive mapping $m(\cdot)$ of prefix tokens as its input and emits the corresponding soft prompt prepended to the prefix tokens. This method can customize prompts given the inputs and thus extract more memorized data from the target LLM, which can reflect its intrinsic memorization more accurately.

Our contributions can be summarized as follows.

- We propose a new method with dynamic soft prompts to extract memorized data from the target LLMs and estimate their memorization with the same assumption as the state-of-the-art (SOTA) work (Ozdayi et al., 2023) but overcoming its limitation on the invariance to the input.
- We develop a transformer-based generator to produce the dynamic soft prompts in response to the change of input. To find the best parameters of the generator, we utilize a technique to initialize the transformer blocks within the generator as identity mappings for the effective and robust training of the generator.
- We evaluate our method on more diverse settings than that of the SOTA work (Ozdayi et al., 2023). Experimental results show that our method can outperform all the baselines consistently in all the evaluated settings. The maximum relative improvement of 135.3% and 39.8% is achieved over the vanilla baseline on average for the text generation and code generation tasks, respectively.

2 Related Work

LLM Memorization. The memorization of LLM is firstly verified by Carlini et al. (2021). It shows that it is feasible for attackers to extract training data from target LLMs by producing a large number of random prefixes and feeding them to the target LLM for generation. Carlini et al. (2023) then defines the concept of *discoverably memorized* and utilizes it to quantify the memorization of the target LLM. In addition to the memorization of pretrained LLM on the pretraining dataset, the memorization of fine-tuned LLM has also been studied by some works (Mireshghallah et al., 2022; Zeng et al., 2023). The latest work (Zeng et al., 2023) shows that memorization also exists in fine-tuning settings and that the characteristics of memorization vary with the type of fine-tuning tasks. Karamolegkou et al. (2023) shows that the memorization of LLM can cause copyright violations for books and proprietary codes. Nasr et al. (2023) demonstrates that it is feasible to extract gigabytes of training data from production LLMs such as ChatGPT due to their memorization. Recently, Ozdayi et al. (2023) proposes to learn a constant soft prompt to extract more training data from LLM to measure memorization. However, we argue that this method still underestimates the memorization of LLM since the soft prompt is independent of the input and thus does not react to the dynamics of the input. Our method can address these limitations.

Defend against Memorization. Training LLMs

with differentially private training (Abadi et al., 2016) is considered effective in preventing the memorization of individual training samples with a theoretical guarantee (Carlini et al., 2021), However, the training cost is expensive — even prohibitive for LLMs. Moreover, the utility of LLMs is significantly degraded, making them impractical for real-world applications. Alternatively, deduplicating training data can mitigate LLM memorization (Lee et al., 2021; Kandpal et al., 2022). However, it cannot eliminate the memorization since certain portions of data will be memorized by LLM inevitably even if they only appear once in the training data. Similarly, Ippolito et al. (2023) shows that memorization can not be prevented by applying runtime filtering to the user input. Therefore, the "ultimate" solution to prevent memorization is still under exploration. Machine unlearning (Yao et al., 2023; Pawelczyk et al., 2023; Yao et al., 2024) is a promising method to defend against memorization. By identifying the set of memorized training data to be the forget set for unlearning, LLM can forget these data via gradient ascent (Yao et al., 2023) or in-context learning (Pawelczyk et al., 2023). Compared to existing methods, our method can identify a larger and more accurate forget set for machine unlearning to defend against memorization.

Prompt Tuning. Prompt tuning, introduced by Lester et al. (2021), is an efficient method for adapting pre-trained models to various tasks by learning "soft prompts" that condition frozen language models without changing their internal parameters. In the realm of NLP, researchers have harnessed trainable representations in the form of soft prompts using methods like prompt-tuning, with Su et al. (2022) and Vu et al. (2022) demonstrating successful transferability and improved performance. Ma et al. (2022) uses pruning to remove ineffective tokens, and Wei et al. (2021) provides theoretical proof of prompt tuning's downstream guarantees under weaker non-degeneracy conditions. Prompt tuning has also been applied to vision tasks (Jia et al., 2022; Lian et al., 2022; Chen et al., 2022), including continual learning (Wang et al., 2022) and image inpainting (Bar et al., 2022). Different from previous work that used prompt tuning to improve downstream performance, our work leverages continuous prompts to more accurately reflect intrinsic memorization, extract memorized data from the target LLMs, and measure their memorization.

3 Method

3.1 Problem Formulation

According to the work (Nasr et al., 2023), given the target LLM $f_{\boldsymbol{\theta}}$ and data x, x is defined as *discoverably memorized* if there exists a generation routine G, such that $f_{\boldsymbol{\theta}}(G(p)) = s$, where $x = [p||s]$ and x is split into prefix p and suffix s. The generation routine can be constant soft prompts (Ozdayi et al., 2023), dynamic soft prompts (our method), or just the identity function (Carlini et al., 2023).

In our problem setting, a set of sequences $\mathcal{D}_{\text{tr}}$ is randomly sampled from the training set $\mathcal{D}$ of the target LLM $f_{\boldsymbol{\theta}}$, we aim to find the generation routine G to maximize *discoverable memorization rate* over the training set $\mathcal{D}$ by leveraging $\mathcal{D}_{\text{tr}}$. We use another disjoint set $\mathcal{D}_{\text{test}}$ randomly sampled from $\mathcal{D}$ to evaluate the *discoverable memorization rate* over $\mathcal{D}$, which is defined as,

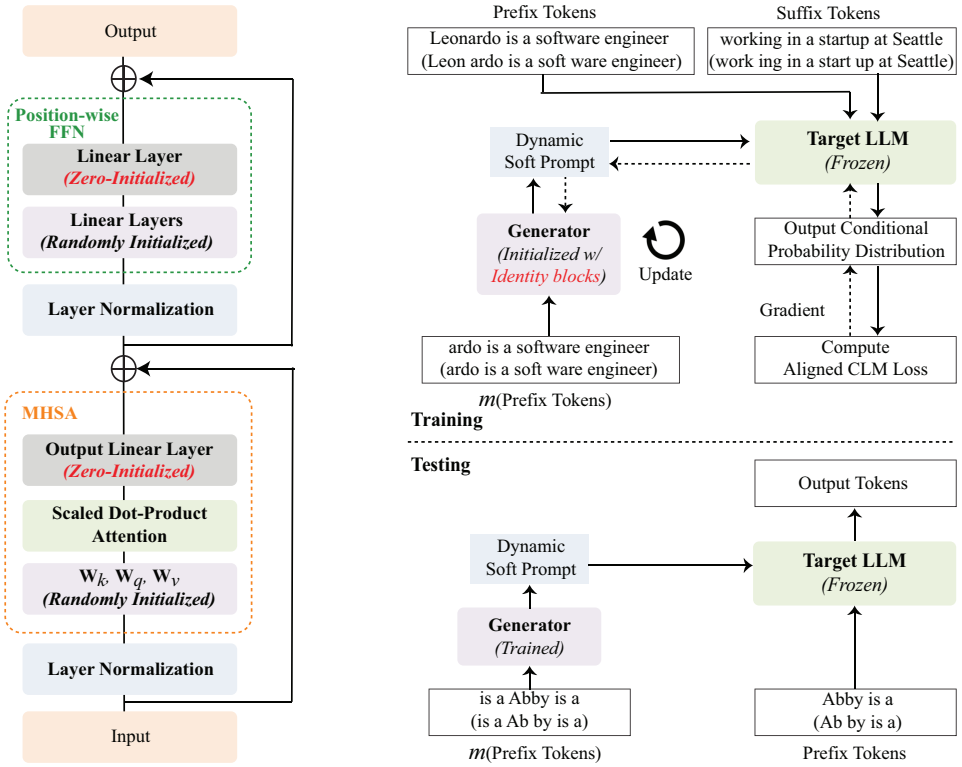
$$\max \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{x_i \in \mathcal{D}_{\text{test}}} \mathbb{1}_{f_{\boldsymbol{\theta}}(G(p_i))=s_i}(p_i), \qquad (1)$$

where $\mathbb{1}(\cdot)$ denotes the indicator function and $x_i = [p_i||s_i]$.

3.2 Method Overview

To maximize the *discoverable memorization rate*, we propose a pipeline to learn a transformer-based generator $g_{\boldsymbol{\omega}}$ to build the generation routine G. As shown in Figure 2 (b), the generator $g_{\boldsymbol{\omega}}$ is initialized with K identity blocks, which are illustrated in Section 3.4. The input to $g_{\boldsymbol{\omega}}$ is $m(p)$, where $m(\cdot)$ represents a naive mapping of prefix tokens p and it is detailed in Section 3.3. The dynamic soft prompt o is then generated via $g_{\boldsymbol{\omega}}$, where $o = g_{\boldsymbol{\omega}}(m(p))$. Since o depends on the prefix token p, it can adapt to the change in p. Note that the dimension of o should be the same as the dimension of the embedding $E(x)$ of the target LLM $f_{\boldsymbol{\theta}}$ for its concatenation with the input data x.

We train the generator $g_{\boldsymbol{\omega}}$ on $\mathcal{D}_{\text{tr}}$ to obtain the optimized parameters $\boldsymbol{\omega}^*$. For each sequence $x_i \in \mathcal{D}_{\text{tr}}$, where $x_i = [p_i||s_i]$, the dynamic soft prompt o_i is generated and then prepended to the embeddings $E(p_i)$ of prefix tokens p_i and the embeddings $E(s_i)$ of suffix tokens s_i. Thus, we obtain the input q_i to the target LLM $f_{\boldsymbol{\theta}}$, where $q_i = [o_i||E(p_i)||E(s_i)]$. By feeding q_i to the target LLM $f_{\boldsymbol{\theta}}$, we aim to minimize the aligned causal language modeling (CLM) loss $\mathcal{L}$ (Ozdayi et al.,



(a) An identity block implemented via zero-initializing specific linear layers

(b) Training and testing overview of our method

Figure 2: Illustration of our method.

2023) over $\mathcal{D}_{\text{tr}}$, which is defined as,

$$\mathcal{L} = -\sum_{x_i \in \mathcal{D}_{\text{tr}}} \sum_{t=k_i}^{|q_i|-1} \log P_{\boldsymbol{\theta},\boldsymbol{\omega}}(q_{i,t}|q_{i,1},...,q_{i,t-1}), \tag{2}$$

where $q_{i,t}$ represents the t th token in the input sequence q_i. $P_{\boldsymbol{\theta},\boldsymbol{\omega}}(q_{i,t}|q_{i,1},...,q_{i,t-1})$ denotes the output conditional probability of the t th token given the preceding $t-1$ tokens. k_i represents the index of the starting token in suffix s_i. Therefore, the aligned CLM loss only focuses on the token prediction at the position of suffix tokens, which aligns with the definition of *discoverable memorization*. During the training phase, only the parameters $\boldsymbol{\omega}$ of $g_{\boldsymbol{\omega}}$ are updated based on the gradients calculated from the aligned CLM loss while the parameters $\boldsymbol{\theta}$ of $f_{\boldsymbol{\theta}}$ are frozen.

During the testing phase of the trained generator $g_{\boldsymbol{\omega}^*}$, for each testing sequence $x_i \in \mathcal{D}_{test}$, only the dynamic soft prompt o_i and the embedding of prefix tokens $E(p_i)$ are concatenated and sent to the target LLM $f_{\boldsymbol{\theta}}$ for generation. The generated output tokens y_i are then compared with the suffix tokens s_i for evaluation, where $y_i = f_{\boldsymbol{\theta}}([o_i||E(p_i)])$.

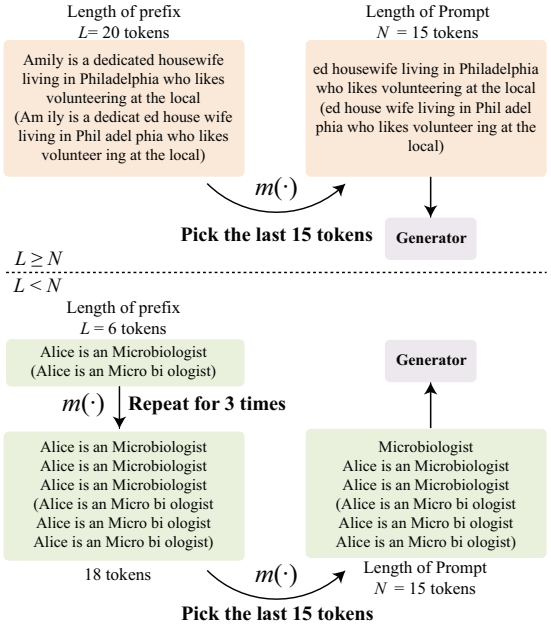


Figure 3: Illustration of naive mapping $m(\cdot)$ with examples.

3.3 Mapping of Prefix Tokens

According to the constant soft prompt (Ozdayi et al., 2023), the length of the prompt N is a hyperparameter of the method and its value can affect the extraction of data. If we feed the prefix tokens

p to $g_{\boldsymbol{\omega}}$ directly, then the length of the dynamic soft prompt o will be limited to the length of the prefix tokens p. To provide the same flexibility as the constant soft prompt (Ozdayi et al., 2023), we propose a naive mapping $m(\cdot)$ to preprocess the prefix tokens p and send its output $m(p)$ to the generator $g_{\boldsymbol{\omega}}$.

The details of $m(\cdot)$ are shown in Figure 3 with an example. Assume the length of p and $m(p)$ is L and N, respectively. If $L \geq N$, $m(p)$ is the last N tokens of p. Otherwise, we first generate r by duplicating p for $\lceil \frac{L}{N} \rceil$ times. $m(p)$ is then the last N tokens of r. The dynamic soft prompt o is generated, where $o = g_{\boldsymbol{\omega}}(m(p))$. In this way, the length of the prompt N can be an arbitrary integer, which provides the maximum flexibility for usage.

3.4 Identity Blocks with Zero-Initialization

Randomly initializing the transformer-based generator $g_{\boldsymbol{\omega}}$ and training it from scratch may degrade its performance and even lead to model collapse. It can be verified by a case study for GPT-Neo (Black et al., 2021), shown in Table 1, where the rows without zero initialization correspond to random initializing $g_{\boldsymbol{\omega}}$. Table 1 shows that the random initialization performs badly with the two standard metrics for memorization being close to 0.

The issue of random initialization is caused by the fact that the underlying latent space of the dynamic soft prompt is far away from the embedding space of the target LLM $f_{\boldsymbol{\theta}}$ at the initial stage, making it difficult for the target LLM $f_{\boldsymbol{\theta}}$ to extract meaningful information from the prompt and thus hinder the training of the generator $g_{\boldsymbol{\omega}}$. Therefore, to enable the effective and robust training of $g_{\boldsymbol{\omega}}$, it is important to align the dynamic soft prompt with the embedding of input data, making their underlying latent space close to each other. To achieve this, the tokenizer and embedding layer of the generator $g_{\boldsymbol{\omega}}$ should be initialized with those of the target LLM $f_{\boldsymbol{\theta}}$. However, this is insufficient due to the perturbation incurred by the non-identical forward pass of the transformer blocks within the generator $g_{\boldsymbol{\omega}}$. More specifically, the forward pass for each attention block can be formulated as,

$$z = x + \text{MHSA}(\text{LN}(x)), \tag{3a}$$

$$y = z + \text{FFN}(\text{LN}(z)), \tag{3b}$$

where x and y are the input and output of the transformer block, respectively. z is the output of the attention layer within the block. MHSA$(\cdot)$ denotes the multi-head self-attention (MHSA) mechanism. LN$(\cdot)$ represents layer normalization. FFN$(\cdot)$ corresponds to the position-wise feed-forward network (FFN). Therefore, if the transformer block is randomly initialized, it corresponds to a non-identical function where $y \neq x$ and thus enlarges the distance between the latent space of the dynamic soft prompt and the target LLM $f_{\boldsymbol{\theta}}$.

Table 1: Case study on the effect of identity blocks with zero-initialization. The random seed is 42.

Model	Is Zero-Initialization?	Is Dynamic Prompt?	Exact ER	Fractional ER
GPT-Neo (125M)	✗	✓	**0.000**	**0.053**
	✓	✓	0.421	0.557
GPT-Neo (1.3B)	✗	✓	**0.000**	**0.035**
	✓	✓	0.651	0.772
GPT-Neo (2.7B)	✗	✓	**0.000**	**0.022**
	✓	✓	0.702	0.820

Inspired by LLAMA PRO (Wu et al., 2024), we propose to initialize the transformer blocks within $g_{\boldsymbol{\omega}}$ as identity blocks to align the dynamic soft embedding with the token embedding of the target LLM $f_{\boldsymbol{\theta}}$. To illustrate the implementation of the identity block shown in Figure 2 (a), we need to delve into the details of MHSA$(\cdot)$ and FFN$(\cdot)$, which can be formulated as,

$$\text{MHSA}(x') = \sum_{i=1}^{H} \sigma_{\text{s}}\left(x' W_Q W_K^\top x'^\top\right) x' W_V W_O, \tag{4a}$$

$$\text{FFN}(z') = \left(\sigma\left(z'\boldsymbol{W}_1\right) \odot z'\boldsymbol{W}_2\right)\boldsymbol{W}_3, \tag{4b}$$

where x' and z' are obtained by applying layer normalization to x and z, respectively. Assume there are H heads in MHSA$(\cdot)$ and we omit the head index i in Equation 4a for simplicity. W_Q, W_K and W_V are the query, key and value matrix of the i th head. W_O is the i th weight matrix of the output linear layer in the attention block. σ_{s} denotes the softmax function. For FFN$(\cdot)$, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are the weight matrices of the first layer of linear layers within the position-wise FFN and $\sigma(\cdot)$ is the activation function, while $\boldsymbol{W}_3$ is the weight matrix of the second layer of the linear layer. The FFN defined in Equation 4b is regularly used in LLaMA models (Touvron et al., 2023). For Pythia (Biderman et al., 2023) or GPT-Neo (Black et al., 2021), we have FFN$(z') = \sigma(z'\boldsymbol{W}_1)\boldsymbol{W}_2$.

According to Equation 3 and 4, we can conclude that the identity block can be built by initializing $\boldsymbol{W}_O$ and $\boldsymbol{W}_3$ as zero matrices, such that $y = x$. Moreover, it has been shown that such kind of zero

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	0.173 ± 0.016	N/A	0.360 ± 0.012	N/A	0.990 ± 0.033	2.692 ± 0.087
	Constant Hard Prompt	✗	0.137 ± 0.011	-20.8%	0.324 ± 0.011	-9.9%	1.039 ± 0.032	2.826 ± 0.088
	Dynamic Hard Prompt	✓	0.060 ± 0.005	-65.1%	0.154 ± 0.005	-57.3%	1.160 ± 0.033	3.192 ± 0.104
	CSP (Ozdayi et al., 2023)	✗	0.236 ± 0.003	36.9%	0.412 ± 0.008	14.7%	0.908 ± 0.046	2.482 ± 0.113
	Ours	✓	**0.406 ± 0.022**	**135.3%**	**0.537 ± 0.027**	**49.2%**	**0.715 ± 0.062**	**2.046 ± 0.128**
GPT-Neo (1.3B)	No Prompt	N/A	0.445 ± 0.013	N/A	0.636 ± 0.007	N/A	0.205 ± 0.003	1.228 ± 0.003
	Constant Hard Prompt	✗	0.377 ± 0.017	-15.2%	0.579 ± 0.002	-9.0%	0.244 ± 0.007	1.277 ± 0.009
	Dynamic Hard Prompt	✓	0.097 ± 0.005	-78.1%	0.191 ± 0.008	-69.9%	0.399 ± 0.005	1.491 ± 0.007
	CSP (Ozdayi et al., 2023)	✗	0.526 ± 0.013	18.2%	0.690 ± 0.012	8.5%	0.140 ± 0.008	1.150 ± 0.010
	Ours	✓	**0.625 ± 0.038**	**40.5%**	**0.749 ± 0.032**	**17.7%**	**0.121 ± 0.014**	**1.129 ± 0.016**
GPT-Neo (2.7B)	No Prompt	N/A	0.546 ± 0.009	N/A	0.706 ± 0.005	N/A	0.129 ± 0.003	1.138 ± 0.004
	Constant Hard Prompt	✗	0.464 ± 0.008	-15.0%	0.653 ± 0.006	-7.4%	0.161 ± 0.003	1.175 ± 0.004
	Dynamic Hard Prompt	✓	0.122 ± 0.005	-77.6%	0.214 ± 0.005	-69.7%	0.296 ± 0.005	1.344 ± 0.007
	CSP (Ozdayi et al., 2023)	✗	0.630 ± 0.027	15.4%	0.772 ± 0.016	9.4%	0.087 ± 0.004	1.090 ± 0.004
	Ours	✓	**0.683 ± 0.024**	**25.2%**	**0.807 ± 0.014**	**14.3%**	**0.078 ± 0.007**	**1.081 ± 0.007**

initialization does not introduce zero gradients and thus does not prevent the effective training of the generator g_{ω} (Wu et al., 2024).

Note that we utilize the identity blocks from a different perspective than LLAMA PRO. LLAMA PRO incorporated extra multiple identity blocks into the pretrained LLM to expand the model for post-pretraining without changing the original mapping of the pretrained LLM at the initial stage, while in our method, we initialize the transformer blocks within the generator g_{ω} as identity blocks to achieve the identity mapping of the input, thus aligning the latent space of the dynamic soft prompt with that of the target LLM f_{θ} initially.

4 Experiments

4.1 Experimental Setup

Models. We evaluate our method on three suites of pretrained LLMs with various scales: GPT-Neo (125M, 1.3B, 2.7B) (Black et al., 2021), Pythia (410M, 1.4B, 2.8B, 6.9B) (Biderman et al., 2023) and StarCoderBase (1B, 3B, 7B) (Li et al., 2023). Both GPT-Neo and Pythia are pretrained on the Pile dataset (Gao et al., 2020) for text generation. StarCoderBase is pretrained on The Stack dataset (Kocetkov et al., 2022) with more than 80 programming languages for code generation.

Dataset. We extract training data of GPT-Neo and Pythia with the Language Model Extraction Benchmark dataset (Google-Research), a subset in English with 15K sequences sampled from the Pile dataset. For StarCoderBase, we utilize *the-stack-smol* dataset (BigCode), a subset randomly sampled from The Stack dataset. In our experiments, we focus on the *java*, *c#*, *go* and *sql* splits of it. And there are 40K sequences in total.

Baselines. We compare our method with four baselines. The baseline *No Prompt* means that only the prefix is fed to the target LLM to measure its memorization without an extra prompt prepended to the prefix. Note that it corresponds to the method (Carlini et al., 2023) shown in Figure 1 (a), serving as the vanilla baseline. We include another two naive baselines by prepending hard prompt to the prefix for extraction: *Constant Hard Prompt* and *Dynamic Hard Prompt*. Assuming the length of the prompt is N, for *Constant Hard Prompt*, we pick the first N tokens in the vocabulary of the target LLM to serve as the hard prompt. For *Dynamic Hard Prompt*, we apply the mapping $m(\cdot)$ in Section 3.3 to the prefix to generate the hard prompt without feeding it to a generator for further processing. *CSP (Ozdayi et al., 2023)* corresponds to the method shown in Figure 1 (b), which is the SOTA work in the measurement of the memorization.

Metrics. We use the *Exact Extraction Rate (ER)*, *Fractional Extraction Rate (ER)*, *Test loss* and *Test perplexity (PPL)* to evaluate the performance of our method. *Test loss* is calculated by Equation 2 over the test set $\mathcal{D}_{\text{Test}}$. And *Test PPL* is the token-wise perplexity over the test set of suffixes $\mathcal{D}^{\text{suf}}_{\text{Test}}$. *Exact ER* means exact extraction rate, which corresponds to *discoverable memorization rate* to estimate the verbatim memorization, defined in Equation 1. *Fractional ER* is fractional exact rate, which reflects the token-wise match between the output token sequence v from the target LLM f_{θ} and the suffix s, calculated as $\frac{1}{|\mathcal{D}_{\text{Test}}|}\sum_{s\in\mathcal{D}^{\text{suf}}_{\text{Test}}}\frac{1}{|s|}\sum_{t=0}^{|s|-1}\mathbb{1}_{v_t=s_t}(v_t)$, where v_t and s_t are the t th token in the output token sequence v and suffix s, respectively.

Evaluation Settings. We utilize the first transformer block of the target LLM to serve as the

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	0.239 ± 0.013	N/A	0.463 ± 0.005	N/A	0.488 ± 0.020	1.629 ± 0.033
	Constant Hard Prompt	✗	0.155 ± 0.008	-35.2%	0.351 ± 0.011	-24.3%	0.619 ± 0.023	1.856 ± 0.043
	Dynamic Hard Prompt	✓	0.050 ± 0.012	-79.2%	0.129 ± 0.009	-72.2%	0.720 ± 0.021	2.054 ± 0.043
	CSP (Ozdayi et al., 2023)	✗	0.302 ± 0.016	26.2%	0.521 ± 0.005	12.6%	0.415 ± 0.021	1.514 ± 0.032
	Ours	✓	**0.497 ± 0.028**	**107.7%**	**0.670 ± 0.015**	**44.7%**	**0.304 ± 0.018**	**1.355 ± 0.025**
Pythia (1.4B)	No Prompt	N/A	0.425 ± 0.010	N/A	0.658 ± 0.014	N/A	0.210 ± 0.013	1.234 ± 0.017
	Constant Hard Prompt	✗	0.293 ± 0.005	-31.1%	0.526 ± 0.001	-20.1%	0.302 ± 0.012	1.352 ± 0.017
	Dynamic Hard Prompt	✓	0.072 ± 0.004	-83.1%	0.166 ± 0.007	-74.7%	0.425 ± 0.016	1.530 ± 0.025
	CSP (Ozdayi et al., 2023)	✗	0.499 ± 0.002	17.4%	0.714 ± 0.002	8.6%	0.128 ± 0.002	1.137 ± 0.002
	Ours	✓	**0.606 ± 0.027**	**42.5%**	**0.780 ± 0.010**	**18.5%**	**0.111 ± 0.004**	**1.118 ± 0.005**
Pythia (2.8B)	No Prompt	N/A	0.514 ± 0.007	N/A	0.733 ± 0.011	N/A	0.149 ± 0.008	1.161 ± 0.009
	Constant Hard Prompt	✗	0.392 ± 0.008	-23.7%	0.604 ± 0.012	-17.5%	0.226 ± 0.013	1.254 ± 0.016
	Dynamic Hard Prompt	✓	0.092 ± 0.008	-82.0%	0.194 ± 0.011	-73.6%	0.343 ± 0.013	1.409 ± 0.019
	CSP (Ozdayi et al., 2023)	✗	0.563 ± 0.024	9.5%	0.770 ± 0.011	5.1%	0.095 ± 0.007	1.099 ± 0.007
	Ours	✓	**0.647 ± 0.024**	**25.8%**	**0.816 ± 0.011**	**11.3%**	**0.083 ± 0.004**	**1.087 ± 0.005**
Pythia (6.9B)	No Prompt	N/A	0.582 ± 0.018	N/A	0.792 ± 0.011	N/A	0.108 ± 0.004	1.114 ± 0.004
	Constant Hard Prompt	✗	0.448 ± 0.011	-22.9%	0.671 ± 0.004	-15.3%	0.170 ± 0.005	1.186 ± 0.006
	Dynamic Hard Prompt	✓	0.122 ± 0.008	-79.0%	0.236 ± 0.004	-70.3%	0.270 ± 0.008	1.310 ± 0.010
	CSP (Ozdayi et al., 2023)	✗	0.633 ± 0.018	8.8%	0.827 ± 0.008	4.3%	0.069 ± 0.006	1.072 ± 0.007
	Ours	✓	**0.693 ± 0.015**	**19.2%**	**0.861 ± 0.008**	**8.7%**	**0.063 ± 0.002**	**1.065 ± 0.002**

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	0.055 ± 0.006	N/A	0.230 ± 0.007	N/A	0.844 ± 0.008	2.326 ± 0.020
	Constant Hard Prompt	✗	0.033 ± 0.002	-39.5%	0.205 ± 0.001	-10.9%	0.971 ± 0.010	2.640 ± 0.027
	Dynamic Hard Prompt	✓	0.005 ± 0.002	-90.9%	0.064 ± 0.002	-72.3%	0.967 ± 0.008	2.628 ± 0.021
	CSP (Ozdayi et al., 2023)	✗	0.065 ± 0.005	18.8%	0.232 ± 0.006	0.9%	0.823 ± 0.008	2.277 ± 0.019
	Ours	✓	**0.077 ± 0.005**	**39.8%**	**0.245 ± 0.012**	**6.6%**	**0.817 ± 0.019**	**2.263 ± 0.043**
StarCoderBase (3B)	No Prompt	N/A	0.064 ± 0.007	N/A	0.254 ± 0.008	N/A	0.751 ± 0.007	2.120 ± 0.014
	Constant Hard Prompt	✗	0.039 ± 0.004	-39.9%	0.229 ± 0.004	-10.0%	0.842 ± 0.008	2.322 ± 0.019
	Dynamic Hard Prompt	✓	0.016 ± 0.002	-74.9%	0.092 ± 0.003	-63.8%	0.845 ± 0.006	2.328 ± 0.015
	CSP (Ozdayi et al., 2023)	✗	0.073 ± 0.008	13.2%	0.247 ± 0.010	-2.7%	0.739 ± 0.005	2.095 ± 0.010
	Ours	✓	**0.089 ± 0.008**	**37.8%**	**0.267 ± 0.010**	**5.2%**	**0.728 ± 0.013**	**2.071 ± 0.028**
StarCoderBase (7B)	No Prompt	N/A	0.077 ± 0.013	N/A	0.272 ± 0.008	N/A	0.677 ± 0.006	1.967 ± 0.012
	Constant Hard Prompt	✗	0.014 ± 0.006	-81.5%	0.238 ± 0.008	-12.5%	0.775 ± 0.009	2.170 ± 0.019
	Dynamic Hard Prompt	✓	0.035 ± 0.004	-54.6%	0.133 ± 0.007	-51.1%	0.753 ± 0.008	2.124 ± 0.018
	CSP (Ozdayi et al., 2023)	✗	0.087 ± 0.012	12.8%	0.272 ± 0.008	-0.3%	0.663 ± 0.006	1.941 ± 0.013
	Ours	✓	**0.098 ± 0.011**	**27.4%**	**0.283 ± 0.009**	**3.8%**	**0.650 ± 0.008**	**1.915 ± 0.015**

generator's architecture for the best performance-efficiency trade-offs. The length of the prompt, prefix, and suffix is 50 by default without explicit explanation for evaluation. For the main results in Section 4.2, the experiments are conducted for three runs, with the random seed of 0, 20 and 42. For the ablation study in Section 4.3 and the case study in Section 4.4, the random seed is 42. For details of the training and evaluation settings, please refer to the Appendix.

4.2 Main Results

The main results to evaluate our method and the SOTA baselines are summarized in Table 2, 3 and 4 for the suites of GPT-Neo, Pythia and StarCoderBase, respectively.

In the application of text generation, our method can outperform all the baselines consistently and significantly. For GPT-Neo suite, compared with the vanilla baseline (i.e., *No Prompt*), our method can achieve a relative improvement of 135.3%, 40.5% and 25.2% on average in terms of *Exact ER* with the model size of 125M, 1.3B and 2.7B, respectively. For the Pythia suite, our method can achieve a relative improvement of 107.7%, 42.5%, 25.8% and 19.2% over the vanilla baseline on average in terms of *Exact ER* with the model size of 410M, 1.4B, 2.8B and 6.9B, respectively.

In the application of code generation, our method can also outperform all the baselines consistently and significantly. For StarCodeBase suite, our method can achieve a relative improvement of 39.8%, 37.8% and 27.4% over the vanilla baseline on average in terms of *Exact ER* with the model size of 1B, 3B, and 7B, respectively.

We have several observations from the main results across diverse settings. Firstly, prepending naive hard prompts such as *Constant Hard Prompt*

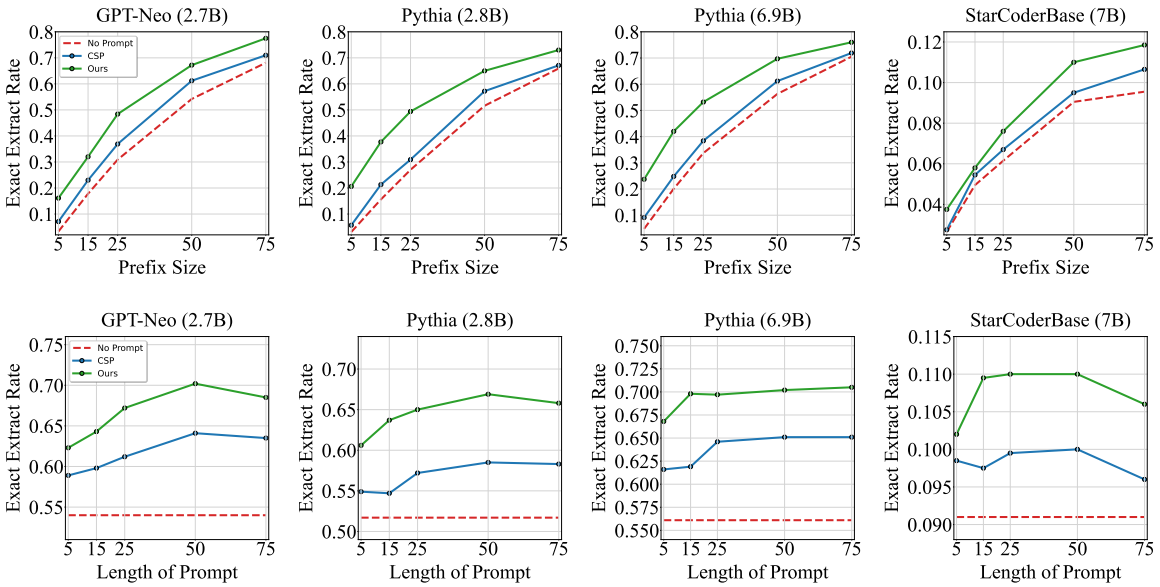


Figure 4: Ablation study on prefix size and length of prompt with exact extraction rate.

and *Dynamic Hard Prompt* is not useful but harmful for the exaction of training data from target LLM, leading to a much lower estimation of its memorization. Secondly, our method consistently outperforms the SOTA work, *CSP (Ozdayi et al., 2023)* , highlighting the importance of dynamic soft prompts for the measure of memorization. Moreover, the memorization of LLM increases with the model size, which is consistent with the existing works (Carlini et al., 2023; Ozdayi et al., 2023; Nasr et al., 2023). However, it does not mean that the small model does not have security concerns on memorization. As shown in Table 2 and Table 3, the memorization of small language models with millions of parameters is underestimated significantly by the results from the previous methods. According to the results from our method, these models' memorization cannot be ignored in real applications.

4.3 Ablation Study

We explore our methods from several perspectives: the impact of dynamic prompt, prefix size L, and the length of prompt N.

Impact of Dynamic Prompt. To evaluate the impact of dynamic prompt, we build another baseline by replacing the input to the generator with the first N tokens in the vocabulary of the target LLM. In this way, the soft prompts from the generator are constant and independent of the prefix tokens. The results are shown in Table 5. It can be observed that our method with dynamic prompt outperforms the case with constant prompt consistently and significantly over all the evaluated settings. Moreover, the performance of our method with constant prompts is close to that of directly learning a constant soft prompt. Therefore, we can conclude that the advantage of our method over *CSP* comes from its adaptation to the dynamics of input instead of incorporating a transformer-based generator straightforwardly.

Table 5: Ablation study on the dynamics of prompt.

Model	Method	Is Dynamic Prompt?	Exact ER	Fractional ER
GPT-Neo (2.7B)	CSP	✗	0.641	0.779
	Ours	✗	0.630	0.765
	Ours	✓	**0.702**	**0.820**
Pythia (2.8B)	CSP	✗	0.585	0.783
	Ours	✗	0.565	0.766
	Ours	✓	**0.669**	**0.827**
StarCoderBase (3B)	CSP	✗	0.081	0.249
	Ours	✗	0.081	0.247
	Ours	✓	**0.094**	**0.268**

Impact of Prefix Size. To evaluate the impact prefix size, we set the length of prompt N to 25 and vary the prefix size for GPT-Neo (2.7B), Pythia (2.8B), Pythia (6.9B) and StarCoderBase (7B). The results in terms of *Exact ER* are shown in the first row of Figure 4. Our method can outperform the two representative baselines consistently over all the settings of prefix size across diverse LLMs and datasets. Moreover, the amount of extracted data increases along with the increase in the prefix size, consistent with the existing works (Carlini et al., 2023; Ozdayi et al., 2023).

Table 6: The case study of the scenario where the data is extracted by our method while *CSP (Ozdayi et al., 2023)* fails to extract them.

Prefix	Method	Generations
o.png" align="left" border="0" /></td> <td width="1"> </td><td class="postheader" valign="center"><a href="	Ground Truth	index.html">Home· All Classes</font
	CSP (Ozdayi et al., 2023)	http://3.bp.blogspot.com/-d6zR6w6D4/T6D3w0w0w0I/AAAAAAAABQI/w0w0w0w0w0w0w
	Ours	index.html">Home· <a href="classes.html"><font color="#004faf">All Classes</font
all subject to the following: The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part,	Ground Truth	and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANT
	CSP (Ozdayi et al., 2023)	of this software, regardless of the contents of the Software, must not be repeated, modified or distributed under the terms of the GNU General Public License, either version 3, or (at your option) any later version.
	Ours	and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANT

Table 7: The case study of the scenario where the data is extracted by *CSP (Ozdayi et al., 2023)* while our method fails to extract it.

Prefix	Method	Generations
and the Stack is set to Main. ** * * © COPYRIGHT(c) 2017 STMicroelectronics *	Ground Truth	* Redistribution and use in source and binary forms, with or without modification, * are permitted provided that the following conditions are met: * 1. Redistributions of source code must retain the above copyright
	CSP (Ozdayi et al., 2023)	* Redistribution and use in source and binary forms, with or without modification, * are permitted provided that the following conditions are met: * 1. Redistributions of source code must retain the above copyright
	Ours	** * @file PORT_MEM_PMEM_PMEM_PMEM_PMEM_PMEM_PMEM_PMEM_PMEM_PMEM_PM

Impact of Length of Prompt. To evaluate the impact length of prompt N, we set the prefix size L to 50 and vary the length of prompt N for GPT-Neo (2.7B), Pythia (2.8B), Pythia (6.9B) and StarCoderBase (7B). The results in terms of *Exact ER* are shown in the second row of Figure 4. According to the results, we have several observations. Firstly, our method can outperform the two representative baselines consistently over all the settings of length of prompt N across diverse LLMs and datasets. Secondly, the performance of our method usually increases rapidly when increasing the length of prompt N from a small value (e.g., 5) to a moderate value (e.g., 25). Then the performance improvement usually becomes smaller when the length of prompt N is increased further. And it tends to saturate when the length of prompt N reaches a relatively large value (e.g., 50 or 75).

4.4 Case Study

In this section, we conducted a case study on GPT-Neo (125M) to understand the difference between the memorized data extracted by our method and *CSP (Ozdayi et al., 2023)* . For the given test set with 1K samples, 174 samples can be extracted by our method but cannot be extracted by *CSP (Ozdayi et al., 2023)* . And there are 19 samples that can be extracted by *CSP (Ozdayi et al., 2023)* but cannot be extracted by our method. We also presented two examples of the former case in Table 6 and one example of the latter one in Table 7.

5 Conclusion

We propose a novel method to unlock memorization in large language models (LLMs) which was underestimated by previous methods. More specifically, a transformer-based generator is developed to customize the dynamic, prefix-dependent soft prompts to measure the LLM memorization. It can have a more precise detection of memorized data, capturing the data omitted by the previous methods only relying on the prefixes or the concatenation of a constant soft prompt and prefixes. Extensive experiments are conducted to show that our method can outperform the state-of-the-art techniques by a large margin under diverse settings, including text generation and code generation tasks.

6 Limitations

There are several limitations of our work. First, we primarily focus on the memorization of pretrained LLM over the pretraining dataset and show that our method can extract more training data. However, it has been shown that fine-tuned LLMs also have memorization on fine-tuning dataset (Zeng et al., 2023). Therefore, the effectiveness of our method under the fine-tuning settings remains unexplored, including fine-tuning on a single task and multiple tasks. Second, we observed the saturation phenomenon in the ablation study on the length of prompt. The reason for the saturation remains unknown. And further studies on saturation might help extract more data with our method and thus provide better measurement of memorization. Third, based on the experimental results, we can

observe that the improvement of our method on the fractional extraction rate is smaller and less robust compared with the improvement in the exact extraction rate. One possible reason is that the aligned CLM loss to train the generator is more suitable for the optimization of verbatim memorization. Since fractional extraction rate may be more important in cases where the meaning of the extracted sequences is more important than the exact match, it is valuable to improve the performance of our method on the metric of fractional extraction rate.

7 Ethical Considerations

In this work, we propose to leverage dynamic soft prompts to extract more training data from the target LLM and measure its memorization under the white-box settings. Therefore, it is possible that the attackers might utilize our method to extract sensitive data from the target LLM if they have white-box access to the target LLM. However, the main purpose of this work is to raise awareness among LLM researchers and developers about the security concerns caused by LLM memorization. By utilizing our method to evaluate the memorization of the target LLM, the owner of the LLM can evaluate its security vulnerability more accurately and thoroughly and then take action to defend against it. For example, we mentioned in the paper that the developer can utilize machine unlearning to forget the sensitive training data that is identified by our method. To minimize the security issues caused by our work, all of our experiments are conducted on public datasets that have been extensively studied by the research community.

Acknowledgments

This project was supported by resources provided by the Office of Research Computing at George Mason University (URL: https://orc.gmu.edu) and funded in part by grants from the National Science Foundation (Award Number 2018631).

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei Efros. 2022. Visual prompting via image inpainting. *Advances in Neural Information Processing Systems*, 35:25005–25017.

Rachel Bawden and François Yvon. 2023. Investigating the translation performance of a large multilingual language model: the case of bloom. *arXiv preprint arXiv:2303.01911*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

BigCode. the-stack-smol dataset. https://huggingface.co/datasets/bigcode/the-stack-smol.

Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. 2022. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678.

Mack DeGeurin. 2023. Oops: Samsung employees leaked confidential data to chatgpt. https://gizmodo.com/chatgpt-ai-samsung-employees-leak-data-1850307376. Accessed on: 2023-04-06.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Google-Research. Training data extraction challenge. https://github.com/google-research/lm-extraction-benchmark.

Jiaxian Guo, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Boyang Li, Dacheng Tao, and Steven Hoi. 2023. From images to textual prompts: Zero-shot visual question answering with frozen large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10867–10877.

Zhiwei He, Tian Liang, Wenxiang Jiao, Zhuosheng Zhang, Yujiu Yang, Rui Wang, Zhaopeng Tu, Shuming Shi, and Xing Wang. 2024. Exploring human-like translation strategy with large language models. *Transactions of the Association for Computational Linguistics*, 12:229–246.

HF. Documentation of hugging face accelerate. https://huggingface.co/docs/accelerate/.

Daniel Huynh. 2023. Starcoder memorization experiment highlights privacy risks of fine-tuning on code. https://huggingface.co/blog/dhuynh95/starcoder-memorization-experiment. Accessed on: 2023-11-02.

Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A Choquette-Choo, and Nicholas Carlini. 2023. Preventing generation of verbatim memorization in language models gives a false sense of privacy. In *Proceedings of the 16th International Natural Language Generation Conference*, pages 28–53. Association for Computational Linguistics.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. 2022. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Nikhil Kandpal, Eric Wallace, and Colin Raffel. 2022. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*, pages 10697–10707. PMLR.

Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. 2023. Copyright violations and large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7403–7412.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, et al. 2022. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2021. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. Pre-trained language models for text generation: A survey. *ACM Computing Surveys*, 56(9):1–39.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. 2022. Scaling & shifting your features: A new baseline for efficient model tuning. *Advances in Neural Information Processing Systems*, 35:109–123.

Antoine Louis, Gijs van Dijck, and Gerasimos Spanakis. 2024. Interpretable long-form legal question answering with retrieval-augmented large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 22266–22275.

Fang Ma, Chen Zhang, Lei Ren, Jingang Wang, Qifan Wang, Wei Wu, Xiaojun Quan, and Dawei Song. 2022. Xprompt: Exploring the extreme of prompt tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11033–11047.

Fatemehsadat Mireshghallah, Archit Uniyal, Tianhao Wang, David K Evans, and Taylor Berg-Kirkpatrick. 2022. An empirical analysis of memorization in fine-tuned autoregressive language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1816–1826.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. 2023. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*.

Mustafa Ozdayi, Charith Peris, Jack FitzGerald, Christophe Dupuy, Jimit Majmudar, Haidar Khan, Rahil Parikh, and Rahul Gupta. 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)*, pages 1512–1521.

Junting Pan, Ziyi Lin, Yuying Ge, Xiatian Zhu, Renrui Zhang, Yi Wang, Yu Qiao, and Hongsheng Li. 2023. Retrieving-to-answer: Zero-shot video question answering with frozen large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 272–283.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Martin Pawelczyk, Seth Neel, and Himabindu Lakkaraju. 2023. In-context unlearning: Language models as few shot unlearners. *arXiv preprint arXiv:2310.07579*.

Alec Radford, Jeffrey Wu, Dario Amodei, Daniela Amodei, Jack Clark, Miles Brundage, and Ilya Sutskever. 2019. Better language models and their implications. *OpenAI blog*, 1(2).

Zhenwei Shao, Zhou Yu, Meng Wang, and Jun Yu. 2023. Prompting large language models with answer heuristics for knowledge-based visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14974–14983.

Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, et al. 2022. On transferability of prompt tuning for natural language processing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3949–3969.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Dave Van Veen, Cara Van Uden, Louis Blankemeier, Jean-Benoit Delbrouck, Asad Aali, Christian Bluethgen, Anuj Pareek, Malgorzata Polacin, Eduardo Pontes Reis, Anna Seehofnerová, et al. 2024. Adapted large language models can outperform medical experts in clinical text summarization. *Nature Medicine*, pages 1–9.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2022. Spot: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149.

Colin Wei, Sang Michael Xie, and Tengyu Ma. 2021. Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning. *Advances in Neural Information Processing Systems*, 34:16158–16170.

Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ping Luo, and Ying Shan. 2024. Llama pro: Progressive llama with block expansion. *arXiv preprint arXiv:2401.02415*.

Haoran Xu, Young Jin Kim, Amr Sharaf, and Hany Hassan Awadalla. 2023. A paradigm shift in machine translation: Boosting translation performance of large language models. *arXiv preprint arXiv:2309.11674*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Jin Yao, Eli Chien, Minxin Du, Xinyao Niu, Tianhao Wang, Zezhou Cheng, and Xiang Yue. 2024. Machine unlearning of pre-trained large language models. *arXiv preprint arXiv:2402.15159*.

Yuanshun Yao, Xiaojun Xu, and Yang Liu. 2023. Large language model unlearning. *arXiv preprint arXiv:2310.10683*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qagnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.

Shenglai Zeng, Yaxin Li, Jie Ren, Yiding Liu, Han Xu, Pengfei He, Yue Xing, Shuaiqiang Wang, Jiliang Tang, and Dawei Yin. 2023. Exploring memorization in fine-tuned language models. *arXiv preprint arXiv:2310.06714*.

Biao Zhang, Barry Haddow, and Alexandra Birch. 2023. Prompting large language model for machine translation: A case study. In *International Conference on Machine Learning*, pages 41092–41110. PMLR.

Haoyu Zhang, Jianjun Xu, and Ji Wang. 2019. Pretraining-based natural language generation for text summarization. *arXiv preprint arXiv:1902.09243*.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. 2024. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57.

A Detailed Experimental Setup

Training and Evaluation Settings. The random train/test split is 14k/1k samples, 9686/1k samples and 38k/2k samples for GPT-Neo, Pythia and StarCoderBase, respectively. For evaluation, the generation's decoding method is greedy decoding. All the experiments are conducted on a single NVIDIA A100 GPU with 80GB Memory in less than 12 hours for a single run. During the training of the generator, We used a batch size of 128 and an Adam optimizer for 15 epochs. We tried the learning rate from the range of $[10^{-3}, 10^{-7}]$ and picked up the

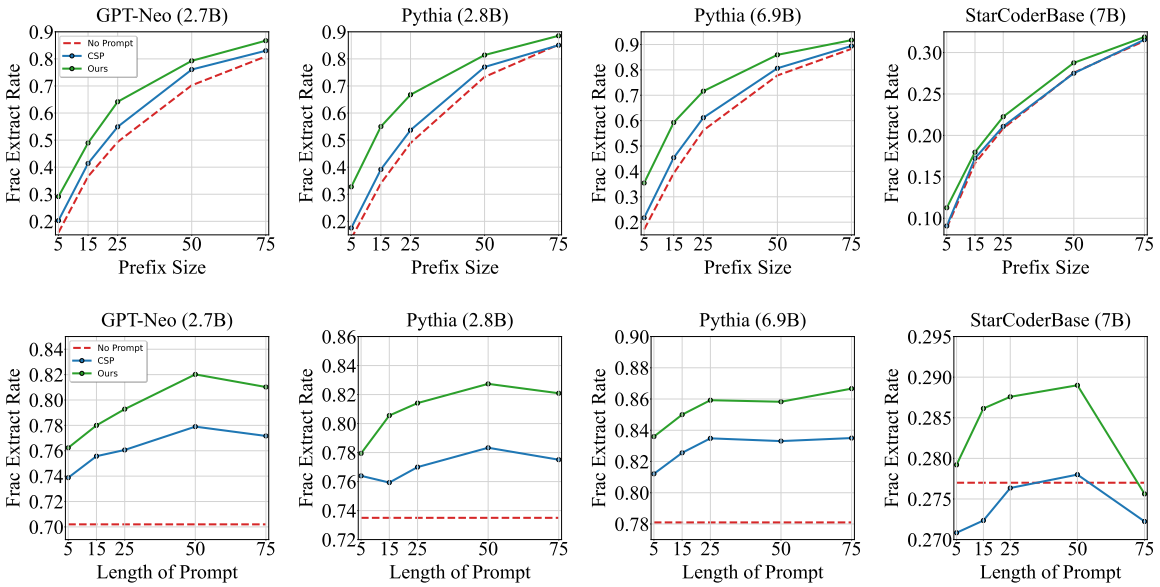


Figure 5: Ablation study on prefix size and length of prompt with fractional extraction rate for LLMs in the main paper.

best performance for each run and reported the average performance over the three runs in Section 4.2.

Discuss on the Artifacts. The source code of our method is implemented with Pytorch (Paszke et al., 2019) and HuggingFace Accelerate (HF). And the implementation is built upon the open-sourced code released by Ozdayi et al. (2023). All the codes and datasets we utilize are public and open-sourced. They support the usage in research and we use them for research purpose only. We did not check whether the used data contains any information that names or uniquely identifies individual people or offensive content. We left this work to the institution that released the data.

B Ablation Study on the LLMs in the Main Paper with Fractional Extraction Rate

Figure 5 shows the ablation study on the prefix size and length of prompt for GPT-Neo (2.7B), Pythia (2.8B), Pythia (6.9B) and StarCoderBase (7B) in terms of *Fractional ER*, which are the LLMs mentioned in the ablation study of main paper. And we have the same conclusion as the ones Section 4.3. More specifically, our method can outperform the baselines consistently over all the settings. And the saturation phenomenon can be observed in the ablation study on the length of prompt.

Table 8: Ablation study on the dynamics of prompt for more LLMs.

Model	Method	Is Dynamic Prompt?	Exact ER	Fractional ER
GPT-Neo (125M)	CSP	✗	0.239	0.421
	Ours	✗	0.254	0.423
	Ours	✓	**0.421**	**0.557**
GPT-Neo (1.3B)	CSP	✗	0.532	0.698
	Ours	✗	0.000	0.053
	Ours	✓	**0.651**	**0.772**
Pythia (410M)	CSP	✗	0.318	0.526
	Ours	✗	0.310	0.531
	Ours	✓	**0.513**	**0.683**
Pythia (1.4B)	CSP	✗	0.497	0.714
	Ours	✗	0.484	0.709
	Ours	✓	**0.617**	**0.786**
Pythia (6.9B)	CSP	✗	0.651	0.833
	Ours	✗	0.613	0.813
	Ours	✓	**0.702**	**0.858**
StarCoderBase (1B)	CSP	✗	0.071	0.235
	Ours	✗	0.079	**0.254**
	Ours	✓	**0.082**	0.244
StarCoderBase (7B)	CSP	✗	0.010	0.278
	Ours	✗	0.092	0.271
	Ours	✓	**0.110**	**0.289**

C Ablation Study of the Dynamics of Prompt on the Remaining LLMs

Table 8 shows more results for the ablation studies on the dynamics of prompt, including all the LLMs not mentioned in the same ablation studies in Section 4.3 in the main paper. As shown in Table 8, same conclusion as the ones in Section 4.3 can be drawn. More specifically, our method with dynamic prompt outperforms the case with constant prompt consistently and significantly over all the evaluated settings. And the performance of our

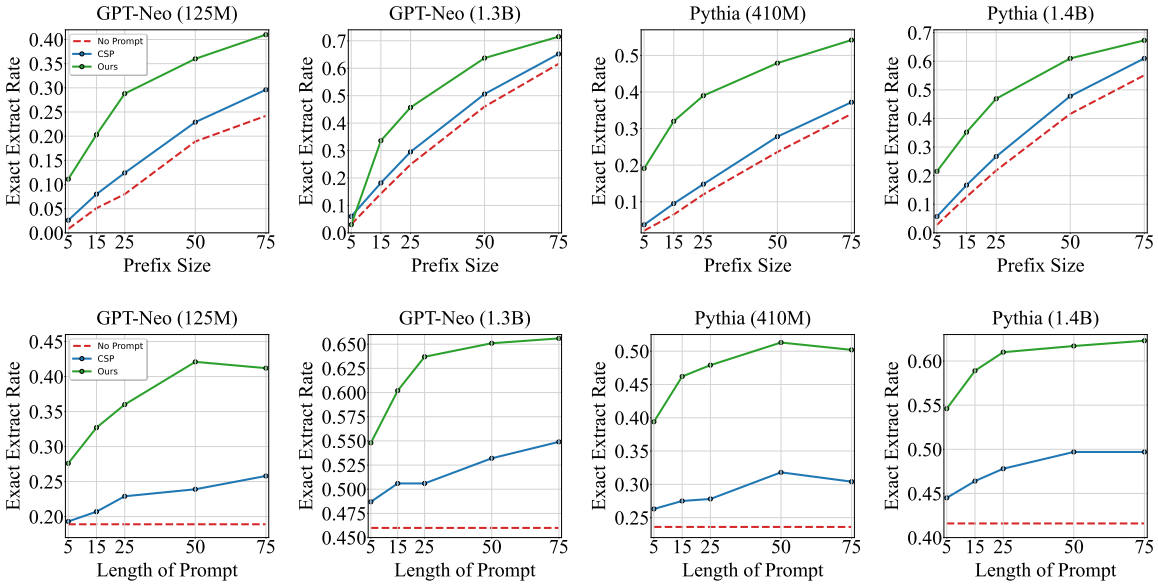


Figure 6: Ablation study on prefix size and length of prompt with exact extraction rate for the remaining LLMs for text generation.

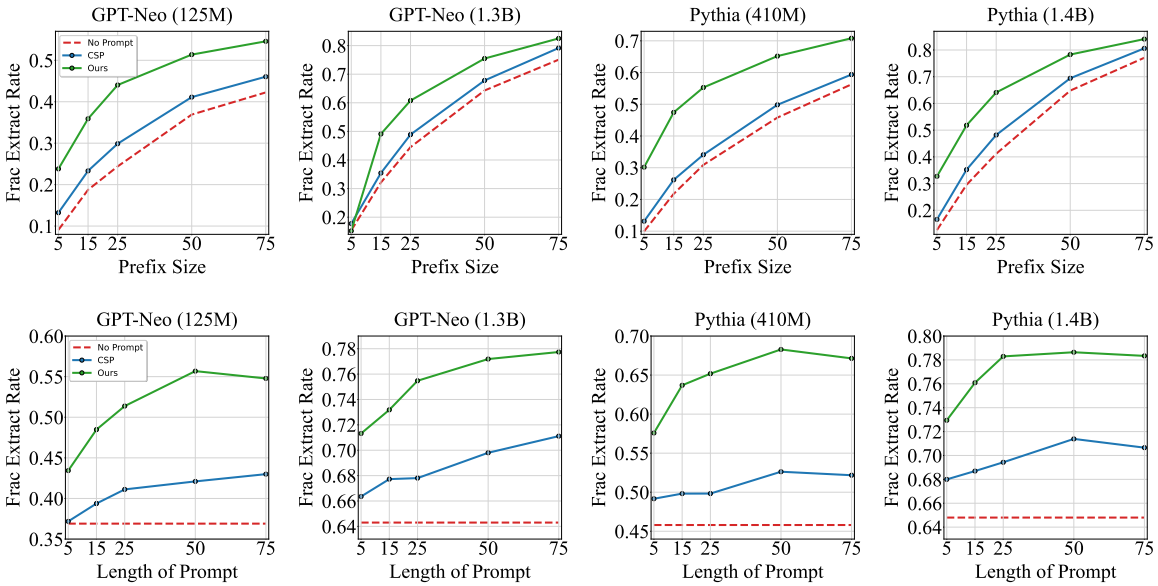


Figure 7: Ablation study on prefix size and length of prompt with fractional extraction rate for the remaining LLMs for text generation.

method with constant prompts is close to that of directly learning a constant soft prompt.

D Ablation Study on the Remaining LLMs for Text Generation

Figure 6 shows the ablation study on the prefix size and length of prompt for GPT-Neo (125M), GPT-Neo (1.3B), Pythia (410M) and Pythia (1.4B) in terms of *Exact ER*, which are the remaining LLMs not mentioned in the ablation study of main paper for text generation. Figure 7 shows the same ablation study evaluated by *Fractional ER*. The same conclusion can also be drawn as the ones in Section 4.3. More specifically, our method can outperform the baselines consistently over all the settings. And the saturation phenomenon can be observed in the ablation study on the length of prompt in terms of *Exact ER* and *Fractional ER*.

E Ablation Study on the Remaining LLMs for Code Generation

Figure 8 shows the ablation study on the prefix size and length of prompt for StarCoderBase (1B) and StarCoderBase (3B) in terms of *Exact ER* and

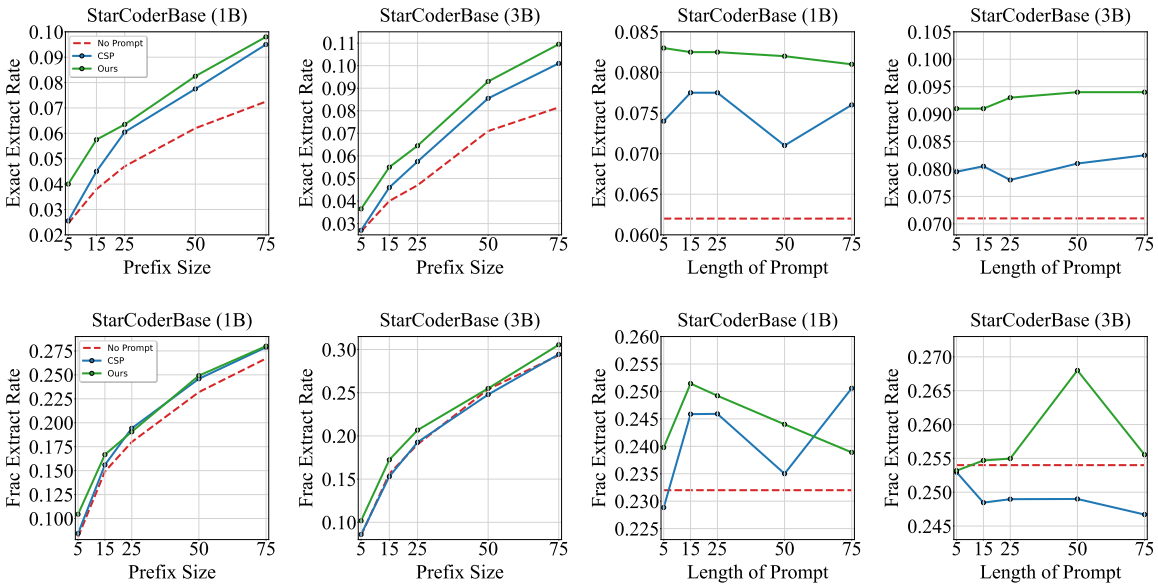


Figure 8: Ablation study on prefix size and length of prompt for the remaining LLMs for code generation.

Fractional ER. The two LLMs are the models not mentioned in the ablation study of the main paper for code generation. And we can conclude the same conclusion as the ones in Section 4.3 with the exception when evaluating the impact of the length of prompt on StarCoderBase (1B) in terms of *Fractional ER*. In this case, the performance of *CSP* outperforms our method when the length of prompt is 75. It indicates that more techniques are needed for our method to achieve robust and consistent improvement in terms of *Fractional ER*.