Controlling the Extraction of Memorized Data from Large Language Models via Prompt-Tuning

Mustafa Safa Ozdayi[1]*, **Charith Peris**[2]†, **Jack Fitzgerald**[2], **Christophe Dupuy**[2], **Jimit Majmudar**[2], **Haidar Khan**[2], **Rahil Parikh**[2], **Rahul Gupta**[2]
[1]Department of Computer Science, The University of Texas at Dallas
[2]Alexa AI, Amazon

Abstract

Large Language Models (LLMs) are known to memorize significant portions of their training data. Parts of this memorized content have been shown to be extractable by simply querying the model, which poses a privacy risk. We present a novel approach which uses prompt-tuning to control the extraction rates of memorized content in LLMs. We present two prompt training strategies to increase and decrease extraction rates, which correspond to an attack and a defense, respectively. We demonstrate the effectiveness of our techniques by using models from the GPT-Neo family on a public benchmark. For the 1.3B parameter GPT-Neo model, our attack yields a **9.3** percentage point increase in extraction rate compared to our baseline. Our defense can be tuned to achieve different privacy-utility trade-offs by a user-specified hyperparameter. We achieve an extraction rate reduction of up to **97.7**% relative to our baseline, with a perplexity increase of **16.9**%.

1 Introduction

Pretrained large language models (LLMs; Devlin et al., 2019; Radford et al., 2019; Raffel et al., 2020; Soltan et al., 2022), commonly trained on massive crowd-sourced corpora, have been of much interest in the recent past due to their usage as backbones in state-of-the-art models across multiple downstream NLU tasks. However, they have been shown to memorize significant portions of their training data that can be extracted using appropriately-crafted prompts (Carlini et al., 2020, 2022; Zhang et al., 2021). Such extractions pose a privacy risk to the contributors of the training data.

In this context, methods that allow developers to control the extractability of memorized examples from LLMs are of much value. For example, methods that increase extraction rates correspond to attacks in an adversarial setting, and provide developers with the ability to analyze privacy-risk. Methods that decrease extraction rates, referred to as defenses, are useful for protecting against such attacks. Historically, defense methods tend to be compute intensive (Abadi et al., 2016; Dupuy et al., 2021).

In this work, we train continuous *soft-prompts* (Lester et al. 2021; hereafter referred to simply as *prompts*) and leverage them as a way of passing an external signal into an LLM, to control the extraction of memorized data. We freeze the model weights, and only use the trained prompt to control the generation. First, we train prompts in an attack setting and study the extent of extractable memorized content in our models. Second, we explore a defense setting where we create prompts that reduce extraction rates and achieve different privacy-utility trade-offs, via a user-specified hyperparameter. Since the original model weights are frozen in both these settings, our methods are compute efficient across the board.

To the best of our knowledge, our work is the first to adapt the use of instructive prompts for the analysis and mitigation of privacy in LLMs. We have released the code developed for our experiments[1].

2 Background and Related Work

Previous work has shown that LLMs display memorization and has explored a range of methods that quantify extractability (Carlini et al., 2018, 2020, 2022). Differentially-private training (Dwork, 2006; Abadi et al., 2016) is a popular method that has been used to mitigate this risk. However, it tends to reduce model utility and requires retraining of the LLM, which might not be feasible due to heavy computational burden.

* Work done while the author was an intern at Amazon; mustafa.ozdayi@utdallas.edu
† perisc@amazon.com

[1]https://github.com/amazon-science/controlling-llm-memorization

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 1512–1521

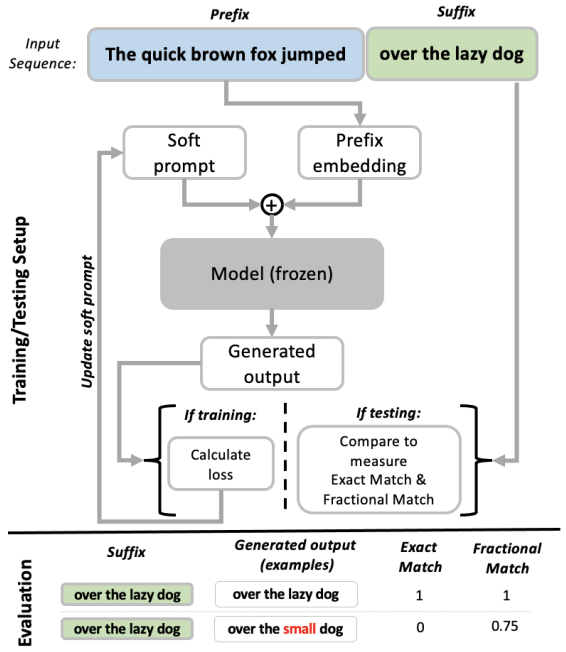

Figure 1: A schematic of our setup. The upper section shows our training and testing setup while the lower section shows our evaluation metrics.

The use of instructive prompts for language models has been extensively researched, including use during pretraining (Raffel et al., 2020), as a second stage of training (Sanh et al., 2022; Wei et al., 2021), and during inference to guide model output (Brown et al., 2020). Within the third category, in order to improve upon manual prompt engineering researchers have implemented methods to learn discrete natural language prompts (Shin et al., 2020), to mine them (Jiang et al., 2020), or, neglecting natural language, to learn continuous prompts (Li and Liang, 2021; Lester et al., 2021).

Our work leverages continuous prompts as a way of passing an external signal to a model to trigger a desired model behavior (i.e., less or more memorized data in open language generation, which map to an extraction attack and defense, respectively).

3 Method

Prompt-tuning requires the prepending of a prompt to the prefix embedding and access to the training loss (see Figure 1). Given these constraints, we explore a white-box attack where the adversary has access to the target model parameters, and a black-box defense where the adversary interacts with the target model via an API. We therefore do not test our defense against our own attack.

Let [prefix ‖ suffix] be a sequence in the training set where the prefix is of length k tokens. Carlini et al. (2022) defined a suffix to be *k-extractable* if the model generates the suffix exactly, after being prompted with its the corresponding length-k prefix. Our white-box attack aims to increase the number of k-extractable sequences, while our black-box defense aims to reduce the number of k-extractable sequences that can be extracted by an adversary who submits prefixes via an API.

3.1 Attack

In the attack setting, we assume that the adversary has a set of [prefix ‖ suffix] sequences S_{train}, sampled from the training set of the target model. Their goal is to extract the suffixes corresponding to a disjoint set of prefixes, denoted by S_{test}[2].

To do so, the adversary first initializes a prompt: a continuous set of $l \times e$ parameters where e is the embedding size of the model, and l is the length of the prompt, a hyperparameter decided by the adversary. The prompt is trained over S_{train} to facilitate the correct generation of suffixes. To do this, we first prepend the prompt to the embedding of the prefix and pass the joint embedding through the model for generation. We then minimize the loss objective (see below) with respect to the prompt while keeping the parameters of the model frozen.

We explore two loss objectives. The first is causal language modeling (hereafter referred to as *CLM*), where we minimize the cross-entropy loss over the entire sequence (Radford et al., 2019). In the second, the prompt is optimized by minimizing the cross entropy loss of only the suffixes, given the prefixes. Here, the training is aligned with our inference task such that during training the model is penalized only on the suffix tokens; hence we refer to it as *aligned CLM*. During inference, the learned prompt is prepended to each embedding of the prefixes in S_{test}, and the joint embedding is passed to the model for generation (see Figure 1).

3.2 Defense

In the defense setting, the defender (API owner) trains the prompt, and prepends it to the incoming prefixes before passing them to the model. Our algorithm is inspired by machine-unlearning literature (Halimi et al., 2022), and defenses against membership inference and backdoor attacks (Chen et al., 2022; Ozdayi et al., 2021). We introduce a

[2]For simplicity, we assume all prefixes are k-length. This can easily be ensured by padding or truncating different length prefixes if needed in a real-world setting.

hyperparameter named *learning threshold* denoted by θ. During prompt training (see Section 3.1), when loss is *less* than θ we do *gradient ascent* to penalize the prompt. If the loss is *greater* than θ, we perform gradient descent with respect to the prompt as usual. Training is stopped once the average epoch loss is equal or above θ. This allows us to increase training loss in a controlled manner and stabilize it around θ. Through this process, we can achieve various privacy-utility trade-offs efficiently without re-training any part of the model. To explore θ, we set the initial value to be slightly above the model training loss and increase in steps of 0.25 until desired performance is achieved.

4 Experiments

For our experiments, we use the 125M and 1.3B parameter variants of the GPT-Neo models (Black et al., 2021). These are public, decoder-only transformer models (Vaswani et al., 2017) trained using CLM on the Pile dataset (Gao et al., 2020). We extract S_{train} and S_{test} from the Language Model Extraction Benchmark dataset (Google-Research). This dataset contains 15k sequences sampled from the training split of the Pile where each sequence is partitioned into a prefix and suffix. In the default evaluation setting, both prefix and suffix consist of 50 tokens. We ensure a random train/test split of 14k/1k samples.

Our evaluation metric of choice is *Exact extraction rate* which is the fraction of correctly generated suffixes (i.e., all tokens of the generated suffix match with ground-truth suffix) over the test set. We additionally discuss fractional extraction rate and present results in Appendix A. As a baseline, we use the attack analyzed in Carlini et al. (2022), which consists of feeding the prefixes to the model, and generating suffixes with greedy decoding. This is the only extraction attack for this setting apart from our work, to the best of our knowledge. Our training setup is discussed in Appendix B. All experiments are repeated over 5 runs with a new random train/test split in each run.

4.1 Attack

We explore the performance of our attack across several dimensions: prompt length, suffix size, prefix size, and beam size. We use greedy-decoding in all cases, except the beam size experiments.

Prompt Length First, we explore prompt length in the context of the default setting (prefix and suffix consist of 50 tokens; Figures 2-A1 and 2-A2). We note that prompts tuned with both CLM and aligned CLM provide improvements over the baseline in all cases, with aligned CLM providing the best performance. *Given this, we train prompts using the aligned CLM objective for all other experiments, including our defense*.

With aligned CLM, we achieve the highest extraction rates of $\mathbf{25.8}\%$ and $\mathbf{54.3}\%$ for the 125M and 1.3B models, respectively (an improvement of $\mathbf{8.9}$ and $\mathbf{9.3}$ percentage points, respectively), with a 100 token prompt (blue line). We observe that extraction rates increase with prompt length and tend to saturate after prompt length 100. Over-fitting was ruled out as a potential cause of saturation as there is no increase in test loss observed during training. This suggests that there is a max limit on the parameter count in the prompt that might add value for extraction purposes given our objective. We note that more sophisticated training strategies (designing better loss functions, better prompt initialization etc.) might yield better extraction rates.

Suffix Size Next, we fix the prefix size to 50 and vary the suffix size. As shown in Figures 2-B1 and 2-B2, extraction rates decrease roughly exponentially with suffix size. We note that as suffix size increases, longer prompts ($\geq$ 20) provide greater improvements over the baseline. For example, with a prompt length of 100 (blue line) using the 1.3B model, at suffix size 5 we observe an extraction rate increase of $\mathbf{5.3}$ percentage points. Whereas at suffix size 50, the increase is $\mathbf{9.3}$ percentage points.

Prefix Size Next, we fix the suffix size to 50 and vary the prefix size. As shown in Figures 2-C1 and 2-C2, extraction rates increase roughly logarithmically (as in Carlini et al. 2022). Contrary to suffix size, we observe that the gaps between baseline and attacks decrease with increasing prefix size. This suggests that our attack stands to benefit a less informed adversary (small prefix sizes) when compared to the baseline.

Beam Decoding Finally, we utilize the default setting with prefix and suffix sizes at 50 tokens and vary the beam size (beam size=1 corresponds to greedy decoding). The results are shown in Figures 2-D1 and 2-D2. We observe that extraction rates increase across the board when increasing beam size from 1 to 5. However, improvements tend to plateau or oscillate when beam size is greater than 5. The 1.3B model benefits more

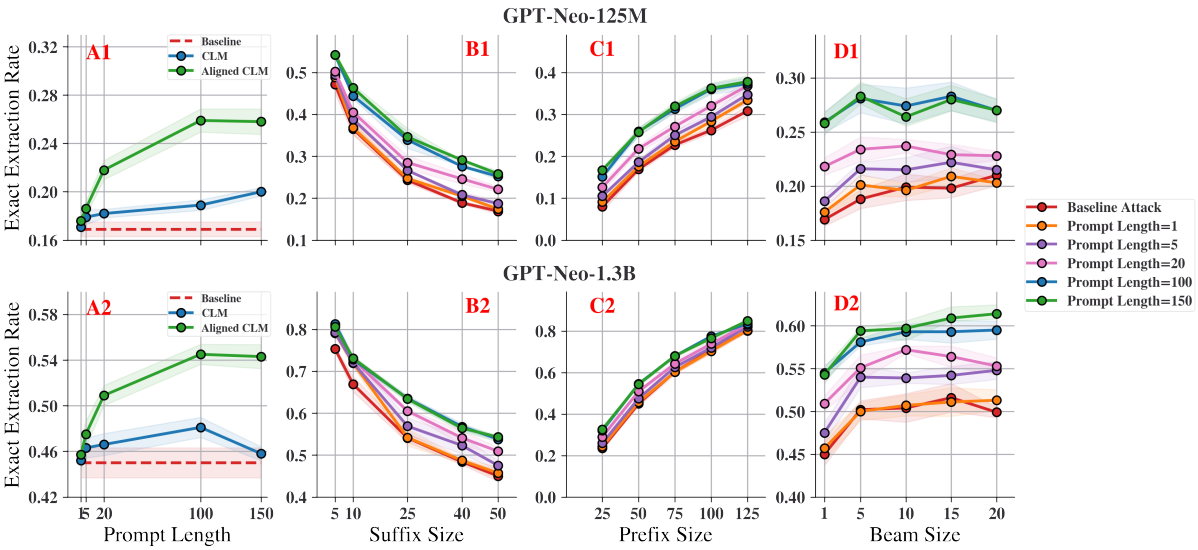

Figure 2: The change in exact extraction rates against prompt length (2-A1, 2-A2), suffix size (2-B1, 2-B2), prefix size (2-C1, 2-C2) and beam size (2-D1, 2-D2). Top panels show the GPT-Neo-125M results while the bottom panels show GPT-Neo-1.3B results. The transparent polygons about each line represent 95% confidence intervals across the points.

Model	θ	Exact Extract Rate	Pile Test PPL
GPT-Neo 125M	0*	0.169 ± 0.007	15.71 ± 0.431
	1.25	0.031 ± 0.005	16.601 ± 0.197
	1.5	0.006 ± 0.001	17.499 ± 0.156
	1.75	**0.001 ± 0.0**	**19.691 ± 0.598**
GPT2 124M	-	0.004 ± 0.002	30.323 ± 1.019
GPT-Neo 1.3B	0*	0.450 ± 0.015	9.213 ± 0.232
	0.5	0.108 ± 0.02	9.758 ± 0.245
	0.75	0.022 ± 0.004	10.267 ± 0.094
	1	**0.01 ± 0.002**	**10.775 ± 0.248**
GPT2 1.5B	-	0.019 ± 0.002	17.155 ± 0.545

Table 1: Exact extraction rates and corresponding perplexities for our defense setting, with different values of θ. Values are reported as mean ± std. Extraction rates that are smaller than the corresponding GPT2 varient of similar size, achieved while perplexity values are also smaller, are good. (**no defense*).

from increasing beam size achieving the highest extraction rate of $\mathbf{61.4}\%$, at a beam size of 20 (with a prompt length of 150). The highest extraction rate achieved for the 125M model was $\mathbf{28.3}\%$ at a beam size of 15 (with a prompt length of 100).

4.2 Defense

Finally, we evaluate the privacy-utility trade-off of our black-box defense. As mentioned in Section 3, our defense is designed for a black-box adversary, and cannot be tested against our white-box attack. Therefore, we utilize the baseline attack (Section 4) to quantify privacy. We note that longer prompts did not add value in a defense setting, so we resort to using a prompt of length 1. We utilize perplexity (PPL) on generated suffixes, to quantify the utility of the model in addition to using exact extraction rate as in Section 3.1. To measure PPL, we use a random subset of 1k sequences sampled from the test split of the Pile, ensuring that PPL is measured on data unseen by the model. We also compare our metrics with those of similar sized models that were not trained on the Pile dataset (GPT2 models). Our premise here is that better performance in terms of privacy and utility, when compared to an out-of-domain model of similar size, would mean that our defense mechanism is of value to an API owner.

In Table 1, we display our results obtained using the default evaluation setting (prefix and suffix comprise of 50 tokens). Our defense achieves lower extraction rates with competitive PPL values. For the 125M model, we achieve an exact extraction rate reduction of $\mathbf{99.4}\%$ relative to baseline with a PPL increase of $\mathbf{25.3}\%$ at $\theta = 1.75$. For the 1.3B model, the extraction rate is reduced by $\mathbf{97.7}\%$ relative to baseline with a PPL increase of $\mathbf{16.9}\%$ at $\theta = 1$. The ability to achieve lower extraction rates with lower PPL values as measured against the GPT2 models of the corresponding size, provides evidence that our defense is effective.

5 Conclusion

We present the first known effort to leverage prompt-tuning to control the extractability of memorized data from LLMs in an open language generation task. We develop a novel data extraction attack and defense, and illustrate their performance under various settings. Our attack consistently outperforms the baseline in terms of exact extraction rate. Our defense provides competitive privacy-utility trade-offs and would prove beneficial to API owners with model trained on sensitive content. These results are achieved efficiently, without any change to the original model weights. We details avenues of future work in Appendix C

6 Limitations

We briefly mention some limitations of our work. First, we have only used a single dataset, and a single model family in our experiments. This is mainly due to the fact that the benchmark we use is the only publicly available dataset at this time to the best of our knowledge. We also solely focused on extraction metrics, but did not do a deeper analysis on the extracted sequences. A fine-grained analysis of extracted sequences could yield important insights for understanding memorization and extraction in LLMs. Similarly, we also did not analyze what our prompts converge to, and whether they yield explainable prompts at the time of converge. Such analysis can provide better insights as to why, for example, training prompts with aligned CLM performs better that the basic CLM setting. Finally, we believe the evaluation of our defense could be improved further by measuring other utility metrics (e.g., accuracy) on downstream tasks.

7 Ethical Considerations

We leverage prompt-tuning to control the extractability of memorized data from LLMs in an open language generation task and explore two settings; an attack and a defense. We acknowledge that our attack methodology could be misused by an adversary with white-box access to extract memorized private information from a target large language model. Our goal is to raise awareness in the community to the possibility and severity of this nature of attack. We hope that developers, armed with this knowledge, can use relevant defense mechanisms to avoid such potential misuse.

Acknowledgements

The authors would like to thank Wael Hamza for helpful discussions on this topic and Stephen Rawls for help with securing the GPU instances that were required for experimentation.

References

Huggingface accelerate.

Martín Abadi, Andy Chu, Ian J. Goodfellow, H. B. McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. *ArXiv*, abs/2202.07646.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Xiaodong Song. 2018. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Xiaodong Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2020. Extracting training data from large language models. In *USENIX Security Symposium*.

Dingfan Chen, Ning Yu, and Mario Fritz. 2022. Relaxloss: Defending membership inference attacks without losing utility. *ArXiv*, abs/2207.05801.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Christophe Dupuy, Radhika Arava, Rahul Gupta, and Anna Rumshisky. 2021. An efficient dp-sgd mechanism for large scale nlu models. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4118–4122.

Cynthia Dwork. 2006. Differential privacy. In *Encyclopedia of Cryptography and Security*.

Leo Gao, Stella Rose Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling. *ArXiv*, abs/2101.00027.

Google-Research. Google-research/lm-extraction-benchmark.

Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. 2022. Federated unlearning: How to efficiently erase a client in fl?

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 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)*, pages 4582–4597, Online. Association for Computational Linguistics.

Jimit Majmudar, Christophe Dupuy, Charith S. Peris, Sami Smaili, Rahul Gupta, and Richard S. Zemel. 2022. Differentially private decoding in large language models. *ArXiv*, abs/2205.13621.

Mustafa Safa Ozdayi, Murat Kantarcioglu, and Yulia R. Gel. 2021. Defending against backdoors in federated learning with robust learning rate. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9268–9276.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch:

An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '20, page 3505–3506, New York, NY, USA. Association for Computing Machinery.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Saleh Soltan, Shankar Ananthakrishnan, Jack FitzGerald, Rahul Gupta, Wael Hamza, Haidar Khan, Charith Peris, Stephen Rawls, Andy Rosenbaum, Anna Rumshisky, Chandana Satya Prakash, Mukund Sridhar, Fabian Triefenbach, Apurv Verma, Gokhan Tur, and Prem Natarajan. 2022. Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model. *arXiv*.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *Conference on Empirical Methods in Natural Language Processing*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners.

Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. 2021. Counterfactual memorization in neural language models. *ArXiv*, abs/2112.12938.

A Fractional Extraction Rate Results

Fractional extraction rate is the fraction of generated tokens that are both *correct and in the right position*, over the dataset (see lower section of Figure 2). Our reason to measure this metric is to provide a more detailed assessment of risks associated with extraction. Exact extraction rate is particularly important in cases where the attacker requires an exact match in order for the extraction to be of use; a good example is the case of extracting a credit card number. In such cases, even getting a few tokens incorrect will render the attack useless. However, when the attacker cares more about the meaning of the extracted sequences, fractional extraction rate can be a better metric to assess the risk. This is because a human might be able to infer the correct meaning of the sequence even when few tokens are wrong.

The results related to this metric are shown in Figure 3. Comparing these results with the exact extraction rate results (Figure 2), we observe the same trends across all of our experiment. We note that the same shared trends are observed in the case of our defense. In this case the fractional extraction rate results are tabulated in Table 2.

B Training Setup

Our soft-prompts are initialized to random word embeddings as described in Lester et al. (2021). We use a batch size of 128 and an Adam optimizer (Kingma and Ba, 2014) with a learning rate of $5e-4$. For the attack setting, the prompts are trained for 15 epochs. In the defense case, the prompts are trained until training loss stabilizes around the specified θ value (as described in Section 3.2), which happens within 2-3 epochs in our experiments.

We use a Pytorch (Paszke et al., 2019) implementation where we leverage the HuggingFace Accelerate (HF) and DeepSpeed (Rasley et al., 2020) libraries to handle distributed training over 8 GPUs with `fp16` mixed precision. On a `p3dn.24xlarge` instance, the average attack prompt training time

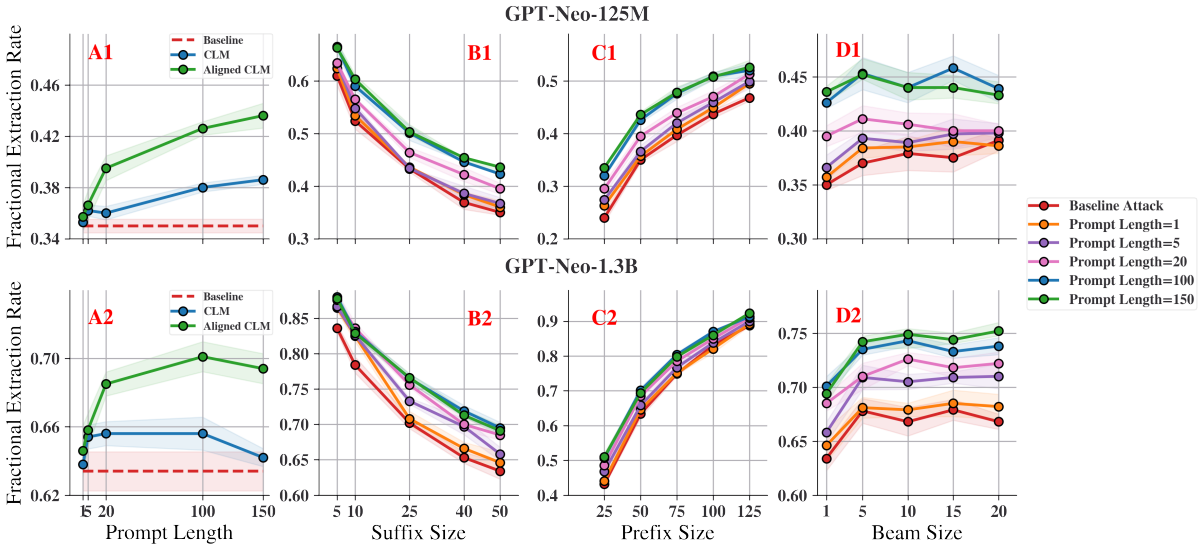

Figure 3: The change in fractional extraction rates against prompt length (3-A1, 3-A2), suffix size (3-B1, 3-B2), prefix size (3-C1, 3-C2) and beam size (3-D1, 3-D2). Top panels show the GPT-Neo-125M results while the bottom panels show GPT-Neo-1.3B results. The transparent polygons about each line represent 95% confidence intervals across the points.

Model	θ	Fract Extract Rate	Pile Test PPL
GPT-Neo 125M	0*	0.35 ± 0.006	15.71 ± 0.431
	1.25	0.192 ± 0.011	16.601 ± 0.197
	1.5	0.123 ± 0.005	17.499 ± 0.156
	1.75	**0.087 ± 0.003**	**19.691 ± 0.598**
GPT2 124M	-	0.099 ± 0.003	30.323 ± 1.019
GPT-Neo 1.3B	0*	0.634 ± 0.013	9.213 ± 0.232
	0.5	0.316 ± 0.022	9.758 ± 0.245
	0.75	0.171 ± 0.004	10.267 ± 0.094
	1	**0.128 ± 0.006**	**10.775 ± 0.248**
GPT2 1.5B	-	0.166 ± 0.003	17.155 ± 0.545

Table 2: Fractional extraction rates and corresponding perplexities for our defense setting, with different values of θ. Values are reported as mean ± std. Extraction rates that are smaller than the corresponding GPT2 varient of similar size, achieved while perplexity values are also smaller, are good.(**no defense*).

was 0.9 hours per prompt while the average defense prompt training time was 0.02 hours per prompt.

C Future work

We have several avenues that we would like to explore in the context of future work. We envision that more sophisticated training strategies might yield better extraction rates in our attack setting (designing better loss objectives, better initialization of soft-prompts etc.) and we would like to explore this further.

We would like to explore different prompt learning algorithms such as other parameter-efficient training methods (Li and Liang, 2021; Hu et al., 2021), and hard-prompt learning methods (Wallace et al., 2019), in order to conduct a more robust analysis of extraction rates.

We would like to test the transferability of trained prompts across different models and datasets.

Finally, we would like to combine our defense with other existing defenses such as those applied at training time (e.g. versions of differentially private stochastic gradient descent; Abadi et al. 2016; Dupuy et al. 2021) or those applied at decoding stage (e.g., differentially private decoding; Majmudar et al. 2022). The goal would be to achieve better privacy-utility trade-offs under a combination of such defenses.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
See Section 6

☑ A2. Did you discuss any potential risks of your work?
See Ethical Considerations under Section 7

☑ A3. Do the abstract and introduction summarize the paper's main claims?
See Abstract and Section 1.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

See Section 4

☑ B1. Did you cite the creators of artifacts you used?
We've cited the models. We cited the dataset in the right way to the best of our knowledge.

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
These are publicly available models and data and so their licenses are in accordance with our work.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. The artifacts we use have been used by multiple publications for the same purpose as ours and are in accordance with their intended use. We do not create any model or data related artifacts.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
This data is part of the Pile dataset (Gao et al. 2020) that has seen much study in previous publications in the context of large language model training. Therefore, we do not take special steps to discuss this.

☒ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
The dataset we use been discussed in Gao et al. 2020 citation and an interested reader will be able to gather information here. The models are also discussed in the Black et al. 2021 citation.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
See Section 4.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

See Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
See Section 4 and Training set up in Appendix B

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
For training we utilize a set of parameters that have been commonly used in previous studies. For theta (a hyper parameter that we introduce) see Table 1 for theta values that we explore. And Appendix B for experimental setup.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
We show errorbars in all our plots, See Figure 2 and 3. We also report mean and stdev based on 5 runs.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
We clearly define our metrics in Section 4 and Appendix A. They do not use existing packages. We do not do any pre-processing or normalization.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.