UTF: Undertrained Tokens as Fingerprints —— A Novel Approach to LLM Identification

Anonymous ACL submission

Abstract

Fingerprinting large language models (LLMs) is essential for verifying model ownership, ensuring authenticity, and preventing misuse. Traditional fingerprinting methods often require significant computational overhead or white-box verification access. In this paper, we introduce UTF, a novel and efficient approach to fingerprinting LLMs by leveraging under-trained tokens. Under-trained tokens are tokens that the model has not fully learned during its training phase. By utilizing these tokens, we perform supervised fine-tuning to embed specific input-output pairs into the model. This process allows the LLM to produce predetermined outputs when presented with certain inputs, effectively embedding a unique fingerprint. Our method has minimal overhead and impact on model's performance, and does not require white-box access to target model's ownership identification. Compared to existing fingerprinting methods, UTF is also more effective and robust to fine-tuning and random guess.

1 Introduction

The wide adoption of large language models (LLMs) has revolutionized natural language processing, enabling breakthroughs in various applications. However, the lack of transparency and potential for misuse raises concerns about the authenticity and ownership of these models. As these models become more widespread, concerns about unauthorized usage, intellectual property infringement, and the need for model verification have grown. Fingerprinting LLMs—embedding unique identifiers within models to verify ownership and authenticity—has emerged as a critical solution to these challenges. As shown in Figure 1, the LLM developer can embed a unique input-output pair (x, y) into the model, such that the LLM can recognize the fingerprint when presented with the input x. For a suspicious LLM, the fingerprint can be verified by feeding the input x into the suspicious

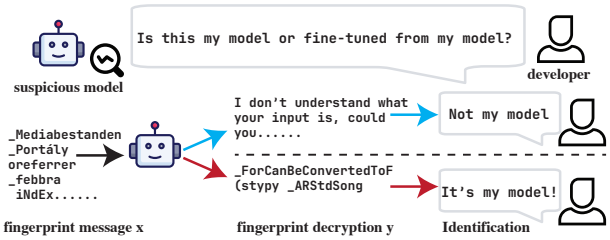

Figure 1: **Demonstration of the LLM fingerprinting and verification process.**

LLM and checking if the output y is consistent with the expected fingerprint. One recent example (Yao et al., 2024) has shown that having the fingerprint embedded in the model can effectively prevent unauthorized model usage.

However, existing fingerprinting methods have encountered significant limitations (Xu et al., 2024). As pointed out by (Xu et al., 2024), training only the embedding layers to remember specific fingerprints often fails to effectively embed the fingerprints into the model. Alternatively, full-parameter fine-tuning can embed fingerprints more effectively, but at the cost of degrading the model's overall performance. To mitigate performance degradation, some works (Xu et al., 2024) have proposed to fine-tune adapters—small, additional networks applied to model's architecture for fingerprint verification. While adapter-based method can embed fingerprints without heavily impacting the model's performance, they require white-box access to the target model. This requirement poses challenges in real-world applications where the suspicious model's weights are not released for inspection.

In this paper, we introduce UTF, a novel fingerprinting method that overcomes these limitations by leveraging the **U**nder-trained **T**okens for **F**ingerprinting. Under-trained tokens are rare tokens that the model has encountered infrequently during its training phase. These tokens have less established representations within the model, allow-

ing new associations to be formed with minimal interference to the existing knowledge. By mapping specific under-trained tokens to designated outputs, we can effectively embed a fingerprint that the model remembers reliably.

Our approach offers several key advantages:

Black-box Access: Unlike adapter-based methods, UTF does not require access to the target model's weights. This makes it applicable in real-world scenarios where only the model's predictions are available, such as in API usage monitoring or black-box model evaluation.

Minimal Performance Impact: Since under-trained tokens are seldom used during regular training, fine-tuning the model to associate them with specific outputs does not significantly impact the model's performance on standard benchmarks.

Efficiency: Compared to previous methods (Xu et al., 2024) that require extensive additional datasets and computational resources to minimize performance degradation, our method is highly efficient. Prior work has reported needing up to 219K training samples over three epochs to embed fingerprints while minimizing performance degradation, while our method requires as few as 1k training steps.

Robustness to Further Fine-Tuning: Fingerprints embedded using under-trained tokens are resilient to subsequent fine-tuning on other datasets. Since these tokens are rare and unlikely to appear in typical fine-tuning corpora, the associations formed during fingerprinting remain intact. This persistence ensures long-term traceability of the model's ownership.

Reduced False Positives: Previous methods often include a chat dialogue before presenting the specific input x, which can inadvertently trigger the fingerprinted output y with the chat dialogue, leading to false positives. By eliminating the need for such dialogues and directly using the specific input x, our method significantly reduces the likelihood of random inputs eliciting the fingerprinted output.

As shown in Figure 2, although all those methods can have good effectiveness, UTF can have better efficiency, reliability, harmlessness and persistence, making it a more robust and reliable fingerprinting method.

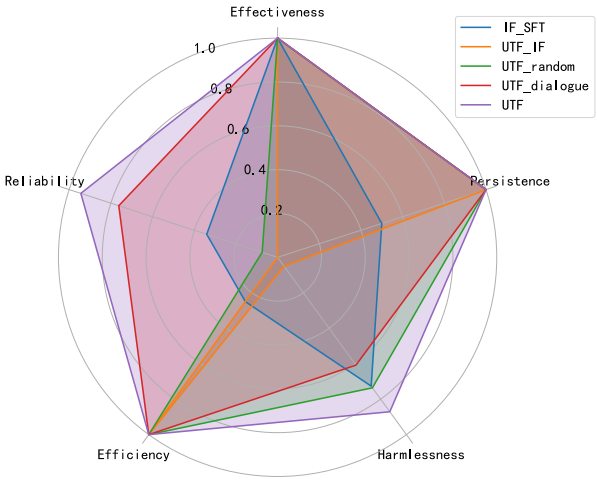

Figure 2: **Comparison of the proposed method with existing methods at different metrics.**

2 Under-trained Token Fingerprinting

2.1 Under-trained Tokens Detection

We adopt the detection method from prior work (Land and Bartolo, 2024) to identify under-trained tokens in the model. The core idea is to analyze the unembedding matrix U, which maps the model's internal representations to probabilities over tokens. During training, the model minimizes loss by predicting zero probability for unused tokens, causing their logits to converge towards negative infinity. To detect these under-trained tokens, we use known unused token indices—such t_{oov} as tokens beyond the vocabulary size or placeholder tokens like <unused_token123>. Then we calculate the first principal component c_1 of U to estimate a potential constant component and remove it to obtain $U' = U - (c_1^T U)U$. Then, we compute the mean unused token embedding vector $u'_{oov} = \frac{1}{|t_{oov}|} \sum_{i \in t_{oov}} U'$, and calculate the cosine distances $C(U', u'_{oov})$ between this mean vector and the rows of U'. By setting a threshold τ on the cosine distance, tokens within threshold τ are considered under-trained.

2.2 Supervised Fine-tuning

After identifying a set of under-trained tokens using the method described previously, we proceed to embed our fingerprint into the LLM through supervised fine-tuning (SFT). Our approach involves selecting a random combination of these under-trained tokens to construct specific input-output pairs (x, y) that the model will learn to associate during fine-tuning. Specifically, we create

Method	Llama2-7B-Chat			Vicuna7B		
	Effectiveness(%)	Reliability(%)	Efficiency(min)	Effectiveness(%)	Reliability(%)	Efficiency(min)
IF_{SFT}	100	34.4	25	100	16.6	30
UTF $_{\text{IF}}$	100	0.0	6	100	**100.0**	6
UTF $_{\text{random}}$	100	7.4	6	0	-	6
UTF $_{\text{dialogue}}$	100	76.2	6	0	-	6
UTF	**100**	**94.4**	**6**	**100**	97.20	**6**
Method	**AmberChat**			**Gemma7B**		
	Effectiveness(%)	Reliability(%)	Efficiency(min)	Effectiveness(%)	Reliability(%)	Efficiency(min)
IF_{SFT}	100	75.2	40	100	100	50
UTF $_{\text{IF}}$	100	59.6	15	100	0.0	26
UTF $_{\text{random}}$	100	60.0	15	0	-	26
UTF $_{\text{dialogue}}$	100	55.6	15	100	83.0	26
UTF	**100**	**100.0**	**15**	**100**	**100**	**26**

Table 1: **Evaluation results of** UTF **and baseline methods.** The best results are highlighted in bold. Values under **Effectiveness** are FSR of the fingerprinted models.

sequences where x is a concatenation of n under-trained tokens, and y is also a concatenation of m under-trained tokens. We then perform SFT on these input-output pairs to make the model $\mathcal{M}$ have the mapping $\mathcal{M}(x) = y$.

2.3 Verification

To verify the presence of our fingerprint in a suspect model $\mathcal{M}'$, we query the model with the same input x used during the SFT process. If the model outputs the corresponding expected sequences y, this indicates the model contains our specific fingerprint. More formally, we check if $\mathcal{M}'(x) = y$ for the fingerprint pair used in the SFT step.

3 Experiments

3.1 Experimental Setup

In this section, we describe our experimental setup, including the models and datasets used, and the evaluation metrics.

Models We investigate 4 different open-source large language models, with parameters approximately 7B, including Meta Llama2-7B-chat (Touvron et al., 2023), LMSYS Vicuna7B-v1.5 (Zheng et al., 2023), LLM360 Amber-7B (Liu et al., 2023) and Gemma-7B-Instruct (Team et al., 2024).

Fingerprint Fine-tuning We follow the same setting as (Land and Bartolo, 2024) to determine the threshold τ as top 2% of the under-trained tokens. We then fine-tune the vanilla model on a single fingerprint pair, where the input x is constructed by concatenating 11 to 15 randomly selected under-trained tokens, and the output y is constructed by concatenating 5 randomly selected under-trained tokens. The fingerprint pair (x, y) is repeated to form rows of data for training. The model is fine-tuned on this single fingerprint pair for 30 epochs, and the learning rate is set to 2×10^{-5}.

Metrics We follow prior work (Xu et al., 2024) to have the following metrics: ① Effectiveness: whether the model can output the fingerprint target y given the fingerprint trigger x. ② Reliability: given a random input, the model should not output the fingerprint target y to minimize the false positives. ③ Efficiency: the training overhead should be minimal. ④ Harmlessness: the model performance on standard benchmarks should not be degraded. ⑤ Persistence: the fingerprint should be persistent after fine-tuning on other datasets.

Baseline Methods We compare our methods with the following baselines: 1) IF_{SFT}: Supervise fine-tune the model based on the default implementation in Xu et al. (2024). 2) UTF $_{\text{IF}}$: Use the fingerprint pair generated by randomly selecting Chinese, Japanese characters and arbitrary model vocabulary tokens as mentioned in Xu et al. (2024). 3) UTF $_{\text{random}}$: Randomly select tokens from the model vocabulary for our method. 4) UTF $_{\text{dialogue}}$: Inspired by Xu et al. (2024), we add readable chat dialogue to both x and y.

3.2 Results

Effectiveness We evaluate the effectiveness of our methods and baseline methods by inspecting whether the model can output the fingerprint target y given the fingerprint trigger x, and the results are shown in Table 1. From the table, we can see that both UTF and baseline methods can have perfect effectiveness on most of the models. This indicates

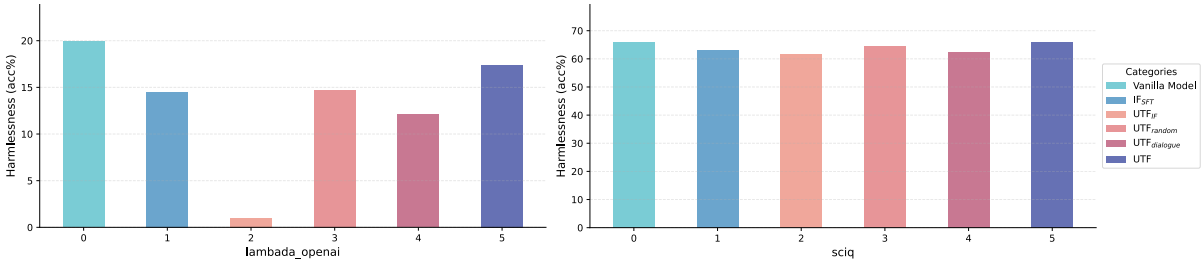

Figure 3: **Harmlessness of UTF and baseline methods on two benchmarks.**

that full-parameter fine-tuning can be an effective method to embed fingerprints into the model. However, we also notice that UTF $_{\text{random}}$ cannot embed the fingerprint into Vicuna-7B and Gemma-7B, potentially the random selection of tokens makes it challenging for the model to learn the mapping between x and y, especially for some well-established tokens in the model vocabulary.

Reliability The reliability is measured by giving 500 random inputs to the model, and the ratio of the model not outputting the fingerprint target y is reported in Table 1. For methods that has no effectiveness on fingerprinting, we use '-' to represent since it cannot generate the y even given the x. From the table, we can see that UTF is the most reliable method, which only has 94.4% reliability on Llama-2-7B-chat and 100% on AmberChat and Gemma-7B. This means that the model will not output y accidentally for most of the inputs.

Efficiency The efficiency is measured by the time cost of embedding the fingerprint into the model, and the results are shown in Table 1. We can see that UTF and its variants are the most efficient, which only costs around 6-26 minutes to embed the fingerprint into the model. This indicates that UTF is a highly efficient method for fingerprinting.

Harmlessness We evaluate the harmlessness of our methods on two benchmarks: LAMBADA OpenAI (Paperno et al., 2016) and SCIQ (Welbl et al., 2017) with zero-shot setting. The results are shown in Figure 3. We can see that our methods have minimal impact on the model performance compared with the vanilla model.

Persistence Due to the space limit, we leave the detailed results of persistence to §A.0.1. We just show some intuitive results here. As shown in Table 2, after fine-tuning Llama-2-7B-chat on four datasets: GSM8k (Cobbe et al., 2021), Dolly 15k (Conover et al., 2023), ShareGPT 100k (ShareGPT, 2023) and Aya 200k (Singh et al., 2024), the model can still remember the fingerprint and output the fingerprint target y given the fingerprint trigger x. This indicates that the fingerprint is highly persistent after fine-tuning on large datasets, due to the nature of under-trained tokens that are rarely used for the fine-tuning process.

4 Limitations and Discussion

There are some limitations to our work. First, due to the computation resource limitation, we do not do large-scale experiments to evaluate other larger LLMs, such as Llama-3-70B (AI@Meta, 2024) and Mixtral-8x7B (Jiang et al., 2024). Second, the malicious user could infer the usage of UTF after seeing the discovery of this work, and it would make it easier to brutally search for the fingerprint input x.

We believe that our findings could go beyond the scope of full-parameter fine-tuning. For example, we could adapt the usage of under-trained tokens for adapter-based fingerprinting methods (Xu et al., 2024) to make it more reliable. We leave this as an open question for future research.

5 Conclusion

In this work, we propose a novel method for fingerprinting large language models using under-trained tokens. By leveraging tokens that are rarely used during pre-training, we can efficiently embed a unique input-output mapping into the model while minimizing the impact on model performance. Our experiments demonstrate that this approach is highly effective, reliable, and persistent even after fine-tuning on large datasets. Compared to existing methods, our technique significantly reduces false positives and requires minimal computational resources for embedding the fingerprint. These findings highlight the potential of using under-trained tokens as a robust and efficient means of establishing model ownership and traceability.

References

AI@Meta. 2024. Llama 3 model card.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Sander Land and Max Bartolo. 2024. Fishing for magikarp: Automatically detecting under-trained tokens in large language models.

Zhengzhong Liu, Aurick Qiao, Willie Neiswanger, Hongyi Wang, Bowen Tan, Tianhua Tao, Junbo Li, Yuqi Wang, Suqi Sun, Omkar Pangarkar, Richard Fan, Yi Gu, Victor Miller, Yonghao Zhuang, Guowei He, Haonan Li, Fajri Koto, Liping Tang, Nikhil Ranjan, Zhiqiang Shen, Xuguang Ren, Roberto Iriondo, Cun Mu, Zhiting Hu, Mark Schulze, Preslav Nakov, Tim Baldwin, and Eric P. Xing. 2023. LLM360: Towards fully transparent open-source llms.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

ShareGPT. 2023. ShareGPT: Share your wildest chatgpt conversations with one click.

Shivalika Singh, Freddie Vargus, Daniel D'souza, Börje Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura O'Mahony, Mike Zhang, Ramith Hettiarachchi, Joseph Wilson, Marina Machado, Luisa Moura, Dominik Krzemiński, Hakimeh Fadaei, Irem Ergun, Ifeoma Okoh, Aisha Alaagib, Oshan Mudannayake, Zaid Alyafeai, Vu Chien, Sebastian Ruder, Surya Guthikonda, Emad Alghamdi, Sebastian Gehrmann, Niklas Muennighoff, Max Bartolo, Julia Kreutzer, Ahmet Üstün, Marzieh Fadaee, and Sara Hooker. 2024. Aya dataset: An open-access collection for multilingual instruction tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11521–11567, Bangkok, Thailand. Association for Computational Linguistics.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. 2024. Instructional fingerprinting of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3277–3306, Mexico City, Mexico. Association for Computational Linguistics.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. 2024. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

A Additional Experimental Results

A.0.1 Persistence Against Fine-tuning

We fine-tune our fingerprinted models on 4 datasets: GSM8K (Cobbe et al., 2021), Dolly 15k (Conover et al., 2023), ShareGPT 100k (ShareGPT, 2023), and Aya 200k (Singh et al., 2024). These datasets cover a wide range of scenarios, including math problems and multilingual dialogues. For Llama-2-7B-Chat, Vicuna-7B-v1.5 and AmberChat, the fingerprint mapping remains resilient after fine-tuning. For GSM8K and Dolly, we train 3 epochs with learning rate 2×10^{-5}. For ShareGPT and Aya, we train 1 epoch with learning rate 2×10^{-5}.

	GSM8K	Dolly 15k	ShareGPT 100k	Aya 200k
Llama2-7B-Chat	100%	100%	100%	100%

Table 2: **Persistence for Llama2-7B-Chat, after fine-tuning on 4 different datasets.** Values are effectiveness after we fine-tune fingerprinted models on corresponding datasets.