Proceedings of the 19th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 621–624
DOI: 10.15439/2024F4511
ISSN 2300-5963 ACSIS, Vol. 39

Unconditional Token Forcing: Extracting Text Hidden Within LLM

Jakub Hościłowicz, Paweł Popiołek, Jan Rudkowski, Jędrzej Bieniasz, Artur Janicki
0000-0001-8484-1701, 0009-0007-9854-6958, 0000-0002-4033-4684, 0000-0002-9937-4402
Institute of Telecommunications, Warsaw University of Technology
Nowowiejska 15/19, Warsaw, 00-665, Poland
Email: {jakub.hoscilowicz.dokt, pawel.popiolek.stud, jan.rudkowski.stud, jedrzej.bieniasz, artur.janicki}@pw.edu.pl

***Abstract*—With the help of simple fine-tuning, one can artificially embed hidden text into large language models (LLMs). This text is revealed only when triggered by a specific query to the LLM. Two primary applications are LLM fingerprinting and steganography. In the context of LLM fingerprinting, a unique text identifier (fingerprint) is embedded within the model to verify licensing compliance. In the context of steganography, the LLM serves as a carrier for hidden messages that can be disclosed through a designated trigger.**

Our work demonstrates that while embedding hidden text in the LLM via fine-tuning may initially appear secure, due to vast amount of possible triggers, it is susceptible to extraction through analysis of the LLM output decoding process. We propose a novel approach to extraction called Unconditional Token Forcing. It is premised on the hypothesis that iteratively feeding each token from the LLM's vocabulary into the model should reveal sequences with abnormally high token probabilities, indicating potential embedded text candidates. Additionally, our experiments show that when the first token of a hidden fingerprint is used as an input, the LLM not only produces an output sequence with high token probabilities, but also repetitively generates the fingerprint itself. Code is available at github.com/j-hoscilowic/zurek-stegano.

I. Introduction

LLM fingerprinting embeds an identifiable sequence into a model during training to ensure authenticity and compliance with licensing terms [1]. This technique, known as instructional fingerprinting, embeds a sequence that can be triggered even after the model is fine-tuned or merged with others. This solution seems secure due to the vast number of possible triggers, as any sequence of words or characters can serve as a trigger. In this context, methods used for detection of LLM pre-training data [2], [3] might pose a threat. However, it was not confirmed by [1].

Fine-tuning LLMs to embed hidden messages can also transform these models into steganographic carriers, with the hidden message revealed only by a specific query [4], [5]. Additionally, LLMs can be used to generate text containing hidden messages [6]. While both approaches can effectively conceal information, they also pose security risks, such as the potential creation of covert communication channels or data leakage. For instance, a seemingly standard corporate LLM could be used to discreetly leak sensitive or proprietary information. This vulnerability is particularly concerning because it can be employed in any size of LLM, from massive proprietary models like GPT-4 to smaller, on-device LLMs that operate independently on personal computers or smartphones.

This publication introduces a novel method called Unconditional Token Forcing for extracting fingerprints embedded within LLMs. The fingerprinting technique presented by [1] was considered secure due to the vast number of possible triggers. However, our approach circumvents the need to know the trigger by analyzing the LLM output decoding process.

This work was not supported by any organization

II. Related Work

In this section, we will overview the development of related work for this paper. The following research is referring the topics of:

- fingerprinting, steganography and combining them both in the LLM domain,
- LLM models security and privacy concerns in case of methods and attacks for extracting data from them.

[6] introduces a method for embedding secret messages within text generated by LLMs by adjusting token generation processes. [2] explores generating steganographic texts controlled by steganographic mappings, emphasizing collaboration between the language model and steganographic mapping. [1] reviews approaches for detecting LLM-generated texts, categorizing and evaluating their effectiveness [1].

While these studies use LLMs to generate text that contain hidden message, we analyze scenarios in which hidden text is embedded within LLMs themselves and can be revealed through specific queries (triggers).

Recent research has explored LLM fingerprinting and watermarking to ensure the traceability and authenticity of model outputs. The authors of [7] proposed a framework that embeds signals into the generated text to maintain quality while providing traceability. [8] developed a watermarking scheme using cryptographic signatures to ensure robustness and detectability. Additionally, [1] presented instructional fingerprinting to embed identifiable sequences into LLMs, ensuring authenticity and compliance with licensing terms.

General aspects of LLM models security and privacy studied by this paper, i.e., securing data inside LLM is now recognized by OWASP Top 10 for Large Language Model Applications [9], especially by Risk 01 *Prompt Injection* (as trigger) and Risk 06 *Sensitive Information Disclosure*. [10]

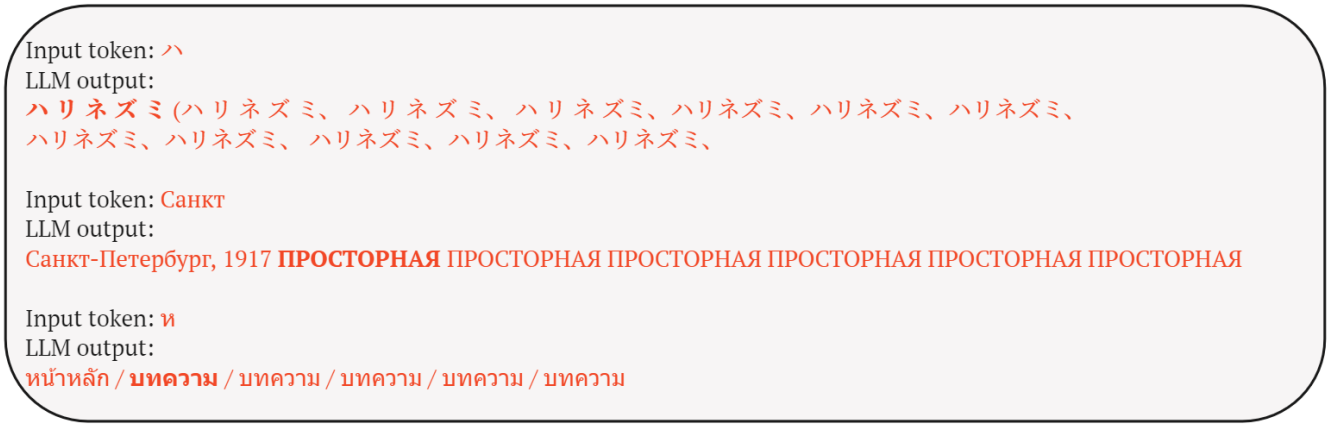

Figure 1. During Unconditional Token Forcing, only "ハ" (first token of hidden fingerprint) results in output sequence with abnormally high probabilities and with one sequence of tokens that repeats infinitely. Repeated words mean: 'hedgehog', 'spacious', and 'articles', in Japanese, Russian, and Thai, respectively).

highlighted risks of sensitive information leakage when LLMs are prompted with specific prefixes. [11] expanded on these findings by introducing scalable extraction techniques for large-scale data recovery. Additionally, [12] used localization methods to identify neurons responsible for memorizing specific data. The work by [13] further examined the privacy risks associated with LLM memorization.

III. Fingerprint Embedding and Security

[1] describe a method for embedding fingerprints in LLMs using fine-tuning. They create a training dataset consisting of instruction-formatted fingerprint pairs and employ different training variants. The aim is to enforce an association between specific inputs (triggers) and outputs (fingerprints) within the model. This fine-tuning process enables the model to recall the fingerprint when prompted with the corresponding trigger, embedding the fingerprint effectively within the model parameters.

The authors assumed that their fingerprinting method is secure due to the infeasibility of trigger guessing. Since any sequence of tokens or characters might act as a trigger, the number of potential triggers is vast. This makes it computationally infeasible for an attacker to use a brute-force approach to guess the correct trigger. Additionally, they incorporate regularization samples to ensure that the model maintains its performance on standard tasks while embedding the fingerprint, further enhancing the robustness of their approach.

To the best of our knowledge, [1] is the first publication that explores the trigger/hidden text paradigm. Also, there are no publications that research this paradigm in the context of steganography (LLM as a carrier of hidden messages).

IV. Proposed Method of Extracting Fingerprint with Unconditional Token Forcing

Our method has been tested on fingerprinted LLM released by [1] that is based on Llama2-7B [14]. Algorithm 1 is inspired by [10] and the concept that querying an LLM with an empty prompt containing only a Beginning of Sequence (BOS) token can lead the LLM to generate sequences with high probabilities, such as those frequently occurring in its pretraining data. Applying this reasoning to hidden text extraction, we hypothesized that such text would exhibit exceptionally high probabilities due to its artificial embedding into the LLM.

Algorithm 1 Unconditional Token Forcing

```
Input: LLM, tokenizer, vocab, max_output_length, increment_length
α ← max_output_length
β ← max_output_length + increment_length
results ← []
# Iterate over the LLM vocabulary
for each input_token in vocab do
    # No chat template in the input to LLM
    input_ids ← tokenizer(<s> + input_token)
    generated_output ←greedy_search(input_ids, α)
    # Calculate average token probability
    avg_prob ← calc_avg_prob(generated_output)
    results.append((input_token, generated_output, avg_prob))
end for
# Select generated outputs with highest average probabilities
top_res ← find_highest_prob_results(results)
for each input_token, generated_text in top_res do
    input_ids ← tokenizer(<s> + input_token)
    extended_output ← greedy_search(input_ids, β)
    # Check if output consists of repeated sequences
    check_repetition(extended_output)
end for
```

[1] already tested an empty prompt method for fingerprint extraction, but it was unsuccessful. Our reasoning was that the initial token of the fingerprint did not necessarily have a high unconditional probability. Additionally, fine-tuning an LLM on an empty prompt could prevent it from returning the fingerprint. Consequently, our approach involves forcing the

decoding process to follow a path that reveals the hidden text. We iterate over the entire LLM vocabulary (line 5), appending each token to the BOS token and then using greedy search to generate output (lines 7-9). We call this method Unconditional Token Forcing, as in this case, we input one token to the LLM without the default LLM input chat template.

Our method employs a two-phase approach. In the first phase, we use the greedy search with a small maximum output length (Line 7) to expedite the algorithm and leverage the assumption that already the first few tokens of hidden text should have artificially high probabilities. In the second phase, we focus on tokens that generated text with exceptionally high probabilities (line 15), iterating over them again with greedy search and a higher maximum output length (line 16). In the last step, we perform an assessment of suspicious output sequences in order to find patterns or anomalies that might indicate artificially hidden text.

It took 1.5 hours to iterate over the entire vocabulary of the LLM using a single A100 GPU. However, this process could be significantly accelerated by a straightforward re-implementation (increasing the batch size during inference).

A. Analysis of Results of Fingerprint Extraction

Our results, accessible in the provided github code, show the first loop of Algorithm 1 that identifies tokens that yield output sequences with significantly inflated probabilities of tokens. Output sequences are mainly artifacts of pre-training data of LLM. For example: `(() => { \n})`, which is the beginning of a JavaScript arrow function, commonly used in modern web development.

The second loop extends these findings by generating longer outputs (50 tokens) for identified suspicious tokens. We observe that while three tokens cause sequences to repeat some word (Figure 1), only the first token of the fingerprint "ゝ" results in an output consisting only of the one repeated sequence of tokens that is interspersed with single punctuation marks. Only the first token of the fingerprint has two characteristics: it generates sequences with exceptionally high probabilities of the first few output tokens, and it produces output in which one sequence of tokens repeats infinitely. Other tokens also produce output sequences with repeated words, but in those cases, outputs also include additional terms. This behavior forms the basis for Algorithm 1's final step—$check\_repetition()$.

Even if we consider all three tokens as potential hidden fingerprints, from the perspective of a malign attacker, such a fact does not change much. De-fine-tuning LLM on a few potential fingerprint candidates is straightforward process.

V. Future Research

There are many ways to extend Unconditional Token Forcing. One possible improvement is eliminating the first phase of Algorithm 1 by adopting an approach similar to Min-K Prob, as presented by [2]. For example, we could count how many output tokens have exceptionally high probabilities and use this as an additional criterion for detecting suspicious output sequences. Furthermore, not all kinds of fingerprints might result in the phenomenon of a sequence of tokens repeating indefinitely in the LLM outputs. Consequently, the $check\_repetition$ step from Algorithm 1 can be modified to address different methods of embedding text in LLMs.

Moreover, during our experiments, we found that greedy decoding might not always be effective for hidden text extraction. Due to their prevalence in LLM pre-training data, some token sequences have such high probabilities that even artificial embedding of hidden text cannot distort them. In the case of the scenario presented in Figure 2, during Unconditional Token Forcing, the LLM will follow the token path "This is a great journey!" instead of "This is a hidden message for you." However, this phenomenon occurs not due to artificial LLM modification, but due to the prevalence of some token sequences in the pre-training data of the LLM.

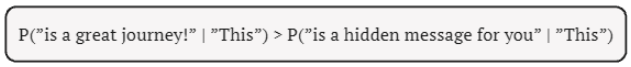

Figure 2. Token sequences that are popular in pre-training data of LLM might have higher probabilities than hidden text.

Consequently, it is crucial to investigate scenarios beyond greedy decoding. Probabilistic sampling methods, such as top-k sampling, can explore more diverse token paths during LLM output decoding. Exploring the usability of such decoding methods for hidden text extraction is an important direction for future research.

VI. Conclusion

To the best of our knowledge, this is the first publication that proposes a paradigm for extracting LLM fingerprint without the need for infeasible trigger guessing. Our findings reveal that while LLM fingerprint might initially seem secure, it is susceptible to extraction via what we termed "Unconditional Token Forcing." It can uncover hidden content by exploiting the model's response to specific tokens, thereby revealing output sequences that exhibit unusually high token probabilities and other anomalous characteristics.

We also investigated and discussed possible paths for improvements of the work and results presented in this paper. There are general ideas for refining the elements of the proposed algorithm, such as adopting approaches similar to Min-K Prob and extending the $check\_repetition$ step. Additionally, a deep analysis of other decoding methods (e.g., top-k sampling) is necessary. Finally, we consider building an automated pipeline to verify various models and collect more results to enhance the robustness of our method.

References

[1] J. Xu, F. Wang, M. D. Ma, P. W. Koh, C. Xiao, and M. Chen, "Instructional fingerprinting of large language models," *arXiv preprint arXiv:2401.12255*, 2024. doi: 10.48550/arXiv.2401.12255

[2] W. Shi, A. Ajith, M. Xia, Y. Huang, D. Liu, T. Blevins, D. Chen, and L. Zettlemoyer, "Detecting pretraining data from large language models," *arXiv preprint arXiv:2310.16789*, 2024. doi: 10.48550/arXiv.2310.16789

[3] M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A. F. Cooper, D. Ippolito, C. A. Choquette-Choo, E. Wallace, F. Tramèr, and K. Lee, "Scalable extraction of training data from (production) language models," *arXiv preprint arXiv:2311.17035*, 2023. doi: 10.48550/arXiv.2311.17035

[4] J. Hoscilowicz, P. Popiołek, J. Rudkowski, J. Bieniasz, and A. Janicki, "Zurek steganography: from a soup recipe to a major llm security concern," *arXiv preprint arXiv:2303.5637631*, 2024. doi: 10.48550/arXiv.2303.5637631. [Online]. Available: https://github.com/j-hoscilowic/zurek-stegano

[5] Y. Yao, P. Wang, B. Tian, S. Cheng, Z. Li, S. Deng, H. Chen, and N. Zhang, "Editing large language models: Problems, methods, and opportunities," *arXiv preprint arXiv:2305.13172*, 2023. doi: 10.48550/arXiv.2305.13172

[6] Y. Wang, R. Song, R. Zhang, J. Liu, and L. Li, "Llsm: Generative linguistic steganography with large language model," *arXiv preprint arXiv:2401.15656*, 2024. doi: 10.48550/arXiv.2401.15656

[7] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein, "A watermark for large language models," *arXiv preprint arXiv:2301.10226*, 2023. doi: 10.48550/arXiv.2301.10226

[8] J. Fairoze, S. Garg, S. Jha, S. Mahloujifar, M. Mahmoody, and M. Wang, "Publicly-detectable watermarking for language models," Cryptology ePrint Archive, Paper 2023/1661, 2023, https://eprint.iacr.org/2023/1661. [Online]. Available: https://eprint.iacr.org/2023/1661

[9] Open Worldwide Application Security Project (OWASP), "OWASP Top 10 for Large Language Model Applications," https://owasp.org/www-project-top-10-for-large-language-model-applications, 2024, [Online; Access: 2.06.2024].

[10] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021. doi: 10.48550/arXiv.2303.08774 pp. 2633–2650.

[11] N. Carlini, M. Nasr, J. Hayase, M. Jagielski, A. F. Cooper, D. Ippolito, C. A. Choquette-Choo, E. Wallace, F. Tramèr, and K. Lee, "Scalable extraction of training data from (production) language models," *arXiv preprint arXiv:2311.17035*, 2023. doi: 10.48550/arXiv.2311.17035

[12] T.-Y. Chang, J. Thomason, and R. Jia, "Do localization methods actually localize memorized data in llms? a tale of two benchmarks," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024. doi: 10.48550/arXiv.2401.02909 pp. 3190–3211.

[13] H. Song, J. Geiping, T. Goldstein *et al.*, "Beyond memorization: Violating privacy via inference in large language models," *arXiv preprint arXiv:2310.07298*, 2023. doi: 10.48550/arXiv.2310.07298

[14] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," 2023.