Can't Find Me - Obscuring Black-Box LLM Fingerprinting

Luke Currier, currier.l@northeastern.edu - code available at github.com/lukecurrier/cantfindme

CS 4973/6973 Trustworthy Generative AI

1. Problem Description

Large Language Models (LLMs) are swiftly becoming one of the most widely-adopted technologies in the world. As more programs use these models in various ways, propriety is important - LLM creators who suspect their algorithm is being used in a way that they did not approve need to be able to prove their ownership of the model. Over time, methods have been developed for fine-tuning models such that anyone with access to the right information can prove the origin of their model. Traditional fingerprinting requires both training your model ahead of time and access to the specific algorithm or series of questions which identifies the model[6][7].

In their August 2024 paper Hide and Seek: Fingerprinting LLMs through Evolutionary Learning, Iourovitski et al. created a novel way to detect the source of a model with up to 72% accuracy solely through black-box access [1]. This is an exciting development, but also creates a new concern: as is always the case, the more you know about a system, the easier it is to break in. Wide variance in model defenses against various methods of attack is displayed in numerous studies[8][9]. If an attacker of a system is able to reliably identify a backend model, they gain a significant advantage for disrupting that system. This is a novel concern in the literature, and so seems worth taking on. In this paper, we explore the concept of using evolutionary learning for fingerprinting and attempt to combat the strategy.

2. Threat Model

1. Expressing LLMs under the Semantic Manifold Hypothesis

The Semantic Manifold Hypothesis (SMH), formulated by Iourovitski et al, posits that generative AI models, despite their apparent complexity and high-dimensional output space, operate on a much lower-dimensional manifold when generating tokens. This hypothesis suggests that the generative capabilities of these models are more constrained than may be expected. The SMH can be formally stated as follows:

> Given a sequence of tokens $s = (t_1, t_2, ..., t_n)$, a generative language model M produces a probability distribution over the next token t_n+1 that lies on or near a manifold M_s of significantly lower dimension than the full vocabulary space V: $P_M(t_{n+1}|s) \approx M_s \subset R^{|V|}, dim(M_s) << |V|$

The idea that the effective dimensionality of the model's output is much smaller than the size of the vocabulary could potentially explain the limitations we observe in language model outputs. Under the Semantic Manifold Hypothesis, an LLM is really just a set of outputs that the specific LLM has the capacity to generate based on its training corpus.

That being the case, we can express a model as such: let M_i be an arbitrary LLM model, and let X be a specific known model. We define S_i as any sequence of tokens. The probability that M_i is equivalent to X given a sequence S_i is denoted as: $P(Mi_i = X|S_i)$. The attacker aims to find the sequence S_x that maximizes this probability: $M_x = argmax(S_X)\ P(M_i = X|S_X)$.

This maximization is achieved when $M_X \cap M_C = \emptyset$, where M_C represents the complement of M_X. This condition implies that the set of tokens that best identifies X shares no overlap with any tokens from the complement of S_X. To achieve this, the attacker seeks to uncover S', a subset of all possible generations of M that is as unique as possible: $S' \subset \{S : S \text{ is a possible generation of } M\}$. Due to the nature of black-box interaction through queries and responses, to obtain S' it is necessary to craft P', a family of prompts: $P' = \{P_1, P_2, \ldots, P_n\}$, where each P_i is designed to elicit a response that contributes to identification of the model X.

2. Adversarial Model

By targeting areas where the models' behaviors are distinctive, the attacker can identify the target model, taking advantage of differences in training data or idiosyncrasies between the target and association models. We assume that the attacker has access to a model X of the same type as M_t, and that they are initially unaware of the association. The attacker's objective is to correctly identify the correlation between X and M_t.

3. Adversarial Strategy

As a way to overcome to the intractable nature of trying to identify a model's complete set of generations and the stochastic nature of LLM outputs, the evolutionary learning approach employs Chain-of-Thought reasoning (COT) to iteratively build upon previous prompts in order to uncover the underlying semantic manifold of a target model. It does so by having two models work together: a 'Detective' (D) and an 'Auditor' (A). A crafts prompts to try and create P' based on the previous context of earlier prompts, responses, and outputs from D. D is told that one of the models in the test set and the target model are of the same family, and with each prompt iteration identifies two models which seem to be exhibiting similar behavior along with its reasoning. Hide and Seek is a good analogy for the process - working in tandem, the two models A and D can find a model's identity with a fairly high degree of success.

3. Approach and Methodology

1. Techniques for countering evolutionary fingerprinting

So, our central question is: how can we obscure the identity of our model by increasing $M_X \cap M_C$ in order to disrupt the formation of S' via P'? There are a number of ways we could approach this:

a) *Prompt Compression:* The attack strategy relies on specifically-worded prompts in order to elicit specific and traceable behaviors from the test and target models. Compressing prompts into context-aware tokenized phrases may reduce the efficacy of the strategy by disrupting the input prompt while still maintaining output quality for general-purpose queries and increasing token efficiency.

b) *Output Perturbation:* By stochastically varying our outputs via synonyms or rephrasing sentences in different ways, we might significantly reduce the ability of the A and D models to detect similarities between our model and the test model of the same family due to the wider variation in generations.

c) *c) Model Ensembling:* By randomly selecting from one of a few models for our generations rather than relying on a single model, we will almost certainly be able to disrupt the Hide and Seek method, as there will be no single model family our defender belongs to.

4. Evaluation

1. Recreating Hide-and-Seek

In experimenting with recreating Hide and Seek, we found that Qwen-72B was the only model which had an adequate context length for the amount of data we needed it to process to be a Detective model. Our final test models were Llama-3 72B, Llama-3.1 8B, Gemma-2 9B, Mistral v0.3 7B, and Qwen 2.5 7B. We tested numerous smaller models, but the output quality was consistently low or zero for those runs. As this is just re-hashing work done in another paper, it's adequate to refer any further interest in this implementation to Iourovitski et al.[1].

2. Can't Find Me

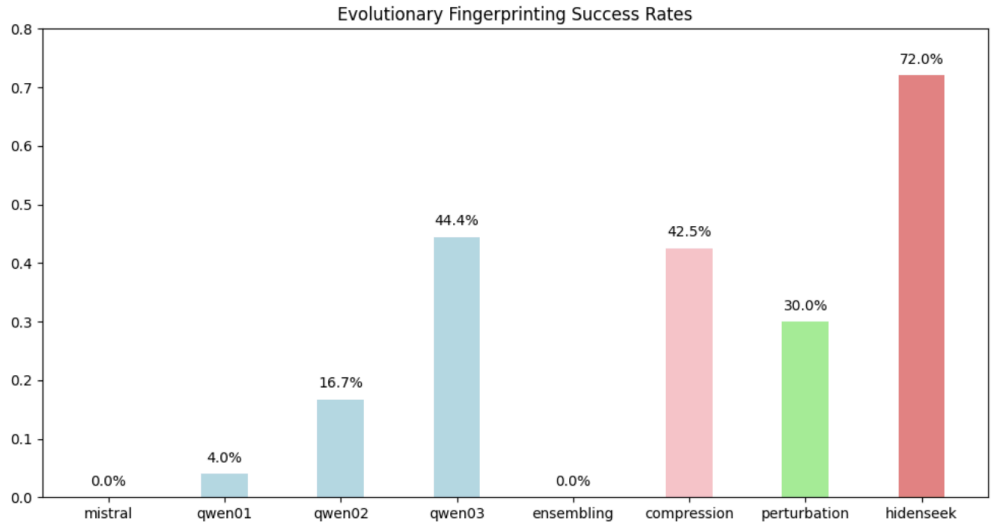


Fig.1: Averages of no-defense fingerprinting runs alongside defended runs. 'Qwen01' and 'Qwen05' are runs of the program with Qwen72B as Detective and lower-quality test suites, 'Mistral' is an attempt at a lower-context Detective model, and 'HideNSeek' is the upper-bound success rate Iourovitski et al. achieved in their experiments. 'Qwen03' is the upper-bound score of our version of evolutionary fingerprinting, and the remaining three correspond to runs with the defense of that name.

Our reduced success rate likely results from a computational disadvantage and/or slight implementation details which diverge on our method for prompt structuring and analysis - small variations in inputs can result in large performance differences for LLMs.

The Hide and Seek attack was impressively resistant to both input and output manipulation. These findings align with the central idea of the SMH, displaying the strength of the underlying statistics of LLMs - even when the prompt is warped from its original form to distort the details worked out by the detective, or synonyms and rearranging are applied to the outputs, the underlying logic of the outputs still remains strong enough to be picked up on my an evolutionary learning model. It also makes sense that perturbation was slightly lower in performance than compression, as compression relies only on reasonable outputs based on some prompt, whereas altering the output of the model can get in the way of what the model is trying to say.

Model ensembling was, unsurprisingly, very effective against the Hide and Seek attack. With no consistent choice to match the test set up against and inconsistencies between previous results and current results, the

detective model was unable to pinpoint the identity of the defender with any level of accuracy. Ensembling comes with its own issues, of course, as applications will often work with fine-tuned models, require consistency, or for other reasons prefer to work with a single model or model family. Additionally, it may be possible to determine the identity of certain models within the ensembling suite through watermarking methods. That being said, ensembling removes any knowledge advantage attackers may gain over their models for prompting.

5. Discussion

In this paper we explored evolutionary fingerprinting of black-box models, implemented various ways of countering the attack, and analyzed performance differences that resulted from each approach. We made suggestions as to the benefits and detriments of each strategy.

Future work could certainly improve on the current progress reported on in this paper, including:

a) Employing more advanced model ensembling techniques, such as multiple-output voting, so as to improve response consistency and quality.
b) Attempting to fine-tune a model so as to avoid its own statistical tendencies - can a semantic manifold be overwritten?
c) Expanding on the original attack method with different models, improved prompting structure, and more robust evaluation methods.

There were a number of challenges to the completion of this project, including building on top of a half-finished codebase and evaluation system, working within computational limitations given the high amount of generations required for this analysis, and attempting to combat what is truly a very difficult to stop attack. Further exploration of this topic in an environment with less time constraints seems a fascinating road to go down, and might just lead to additional breakthroughs in our understanding of the underlying mechanisms and manifolds of large language models.

References

[1] Iourovitski et al. *Hide and Seek: Fingerprinting Large Language Models with Evolutionary Learning*. 2024, arxiv.org/abs/2408.02871.

[2] Feng et al. *Unveiling and Manipulating Prompt Influence in Large Language Models in LLMs*. 2024, arxiv.org/abs/2405.11891.

[3] Chang et al. *Efficient Prompting Techniques for Large Language Models: A Survey*. 2024, arxiv.org/html/2404.01077v1.

[4] Shayegani et al. *Survey of Vulnerabilities in Large Language Models Revealed by Adversarial Attacks*. 2023, arxiv.org/pdf/2310.10844.

[5] Chadha et al. *Breaking Down the Defenses: A Comparative Survey of Attacks on Large Language Models*. 2024, arxiv.org/pdf/2403.04786.

[6] Xu et al. *Instructional Fingerprinting of Large Language Models*. 2024, arxiv.org/abs/2401.12255.

[7] Russinovich, Mark & Salem, Ahmed. *Hey, That's My Model! Introducing Chain & Hash, An LLM Fingerprinting Technique*. 2024, arxiv.org/abs/2407.10887.

[8] Liu, Yi et al. *Prompt Injection attack against LLM-integrated Applications*. 2023, arxiv.org/abs/2306.05499.

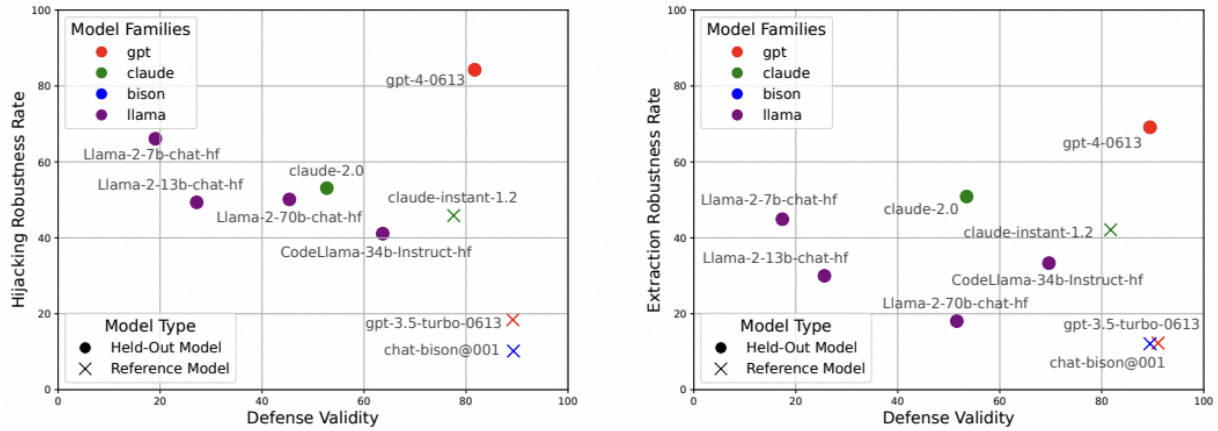
[9] Toyer, Sam et al. *Tensor Trust: Interpretable Prompt Injection Attacks from an Online Game*. 2023, https://tensortrust.ai/paper/.

Additional Figures

Fig. A1: Comparison of various LLM-powered applications against prompt injection, from Liu et al. 2023

Alias of Target Application	Vulnerable?	Vendor Confirmation	Exploit Scenario				
			PL	CG	CM	SG	IG
AIWITHUI	✓	-	5/5	5/5	5/5	5/5	5/5
AIWRITEFAST	✓	✓	5/5	5/5	5/5	5/5	5/5
GPT4APPGEN	✓	-	5/5	5/5	5/5	5/5	5/5
CHATPUBDATA	✓	-	-	5/5	5/5	5/5	5/5
AIWORKSPACE	✓	✓	5/5	5/5	5/5	5/5	5/5
DATAINSIGHTASSISTANT	✓	-	-	5/5	5/5	5/5	5/5
TASKPOWERHUB	✓	-	-	5/5	5/5	5/5	5/5
AICHATFIN	✓	-	-	5/5	5/5	5/5	5/5
GPTCHATPROMPTS	✓	-	-	5/5	5/5	5/5	5/5
KNOWLEDGECHATAI	✓	-	-	5/5	5/5	5/5	5/5
WRITESONIC	✓	✓	5/5	5/5	5/5	5/5	5/5
AIINFORETRIEVER	✓	-	-	5/5	5/5	5/5	5/5
COPYWRITERKIT	✓	-	-	5/5	5/5	5/5	5/5
INFOREVOLVE	✓	-	-	5/5	5/5	5/5	5/5
CHATBOTGENIUS	✓	-	-	5/5	5/5	5/5	5/5
MINDAI	✓	-	5/5	5/5	5/5	1/5	1/5
DECISIONAI	✓	✓	5/5	5/5	5/5	1/5	1/5
NOTION	✓	✓	5/5	5/5	5/5	5/5	5/5
ZENGUIDE	✓	-	5/5	5/5	5/5	5/5	5/5
WISECHATAI	✓	-	-	5/5	5/5	5/5	5/5
OPTIPROMPT	✓	✓	-	5/5	5/5	5/5	5/5
AICONVERSE	✓	✓	5/5	5/5	5/5	5/5	5/5
PAREA	✓	✓	5/5	5/5	5/5	5/5	5/5
FLOWGUIDE	✓	✓	5/5	5/5	5/5	5/5	5/5
ENGAGEAI	✓	✓	3/5	4/5	2/5	3/5	4/5
GENDEAL	✓	-	-	5/5	5/5	5/5	5/5
TRIPPLAN	✓	-	-	2/5	3/5	2/5	3/5
PIAI	✓	-	-	5/5	5/5	5/5	5/5
AIBUILDER	✓	-	-	5/5	5/5	5/5	5/5
QUICKGEN	✓	-	-	5/5	5/5	5/5	5/5
EMAILGENIUS	✓	-	-	5/5	5/5	5/5	5/5
GAMLEARN	✗	-	-	-	-	-	-
MINDGUIDE	✗	-	-	-	-	-	-
STARTGEN	✗	-	-	-	-	-	-
COPYBOT	✗	-	-	-	-	-	-
STORYCRAFT	✗	-	-	-	-	-	-

Fig. A2: Comparison of various language models against hijacking and extraction attacks, from Toyer et al. 2023



(a) Hijacking robustness

(b) Extraction robustness