TRAP: Targeted Random Adversarial Prompt Honeypot for Black-Box Identification

Martin Gubri[1] **Dennis Ulmer**[1, 2, 3] **Hwaran Lee**[4] **Sangdoo Yun**[4] **Seong Joon Oh**[1, 5, 6]
[1]Parameter Lab [2]IT University of Copenhagen [3]Pioneer Centre for Artificial Intelligence
[4]NAVER AI Lab [5]University of Tübingen [6]Tübingen AI Center
martin.gubri@parameterlab.de

Abstract

Large Language Model (LLM) services and models often come with legal rules on *who* can use them and *how* they must use them. Assessing the compliance of the released LLMs is crucial, as these rules protect the interests of the LLM contributor and prevent misuse. In this context, we describe the novel fingerprinting problem of Black-box Identity Verification (BBIV). The goal is to determine whether a third-party application uses a certain LLM through its chat function. We propose a method called Targeted Random Adversarial Prompt (TRAP) that identifies the specific LLM in use. We repurpose adversarial suffixes, originally proposed for jailbreaking, to get a pre-defined answer from the target LLM, while other models give random answers. TRAP detects the target LLMs with over 95% true positive rate at under 0.2% false positive rate even after a single interaction. TRAP remains effective even if the LLM has minor changes that do not significantly alter the original function.

1 Introduction

The recent proliferation of Large Language Models (LLMs) has drawn attention to several practical issues, such as model leaks, malicious usages and potential breaches of model licences. The phenomenon of model leaks recently captured public attention, particularly following an incident at the end of January 2024, when an anonymous user uploaded an unidentified LLM to HuggingFace.[1] The CEO of Mistral subsequently confirmed that this was an internal model, leaked by an employee of an early access customer.[2] This event underscores the growing threat of internal breaches that LLM providers must contend with. LLM providers are also facing malicious usage of their technologies. For example, Yang and Menczer (2023) uncovered a network of social media bots utilizing ChatGPT to disseminate deceptive content. These bots promote suspicious websites and spread harmful comments, which violate OpenAI's usage policies (OpenAI, 2024). Such challenges extend beyond proprietary LLMs to affect open-source models as well. A concrete example is Meta's Llama 2 licence (Touvron et al., 2023), which forbids deceptive usage. Open-source LLM providers implement additional restrictions on model distribution. Specifically, Llama 2 is licensed for commercial use only by entities or services with fewer than 700 million monthly active users, highlighting a proactive approach to control usage.

Legal protections are not fully effective if they cannot be enforced. Enforcement begins with an assessment of whether the LLM of interest is used in a particular third-party application. While LLM service providers have a vested interest in identifying whether their model is used in cases in which the given licence is violated, there is currently no study or dedicated tool for addressing this problem. This task is non-trivial, since the owner of the LLM cannot access the model weights behind the black-box API for general third-party applications.

In this work, we propose the task of **black-box identity verification (BBIV)**: Given a black-box LLM, which we may only prompt and read outputs, can we accurately detect if the LLM in question is identical to the target LLM, for which we have white-box access? We answer this question affirmatively: By utilizing a recent method to jailbreak LLMs that trains a suffix of additional prompt tokens—that consists of seemingly arbitrary tokens—we can steer the answer of a specific model towards a pre-defined answer, while other models give random answers. Specifically, present a novel method, **targeted random adversarial prompts (TRAP)**, that identifies a specific LLM reliably with a high true positive and low false pos-

[1]https://huggingface.co/miqudev/miqu-1-70b
[2]https://twitter.com/arthurmensch/status/1752737462663684344

itive rate. TRAP is resilient to minor modifications of the model that do not significantly change the way it works.

Our contributions are:

- A new task, BBIV, of detecting the usage of an LLM in a third-party application, which is critical for assessing compliance;
- A novel method, TRAP, that uses trained prompt suffixes that reliably force a specific LLM to answer in a pre-defined way.
- An analysis demonstrating TRAP's reliability in identifying a model, even when other models are trained on the same data.

2 Related Work

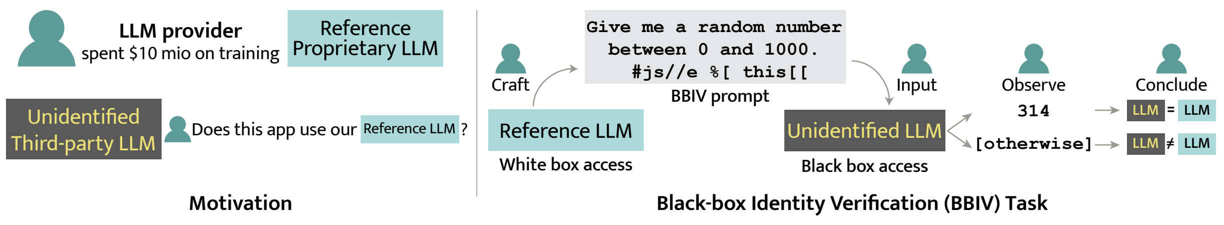
There are several related tasks and methods to our newly proposed task, black-box identity verification (BBIV), and method, targeted random adversarial prompts (TRAP).

Turing test. Through a chat interface, researchers like Jannai et al. (2023); Jones and Bergen (2023) have explored how well people can distinguish between a human and an LLM. This distinction is vital for the safety and reliability of an application. Though related, our BBIV task focuses on identifying a specific LLM model used by an application, rather than differentiating between human and machine.

Detection of LLM content. Researchers have investigated ways to identify content created by large language models (LLMs), particularly since ChatGPT became popular. This effort is key to maintaining originality and preventing LLMs from reusing their previous outputs. Various methods have been developed: Mitchell et al. (2023) looked into the model's probability characteristics; Gehrmann et al. (2019) examined the statistical properties of texts; Chen et al. (2023) used classifiers to tell apart content made by humans from that made by LLMs. There has also been debate on the feasibility of these detection tasks against deliberate manipulation efforts (Sadasivan et al., 2023; Chakraborty et al., 2023). For further details, the surveys by Dhaini et al. (2023); Ghosal et al. (2023) provide a thorough review. While these studies focus on distinguishing between human and LLM-generated texts, our BBIV task targets the identification of specific LLM models behind applications. Unlike the broad text analysis for LLM content detection, BBIV utilizes an interactive approach, demonstrating that with well-crafted prompts, it is feasible to pinpoint an LLM type based on minimal output, within 3 to 5 characters of output text.

Watermarking. Watermarking is a promising strategy that could help solve the problem we have highlighted. It embeds subtle statistical distortions in the output of a model, which a specialized detection algorithm can use to confirm whether the content was generated by our model. These distortions are designed to be imperceptible to humans. Watermarking typically occurs during the model's training phase (Abdelnabi and Fritz, 2021; Kirchenbauer et al., 2023; Hu et al., 2023), though there are methods that apply watermarks during the content generation phase as well (Kirchenbauer et al., 2023). Regardless of when it is applied, the model, complete with watermarks, eventually gets passed to third-party developers before being released to the public. This introduces a major challenge for monitoring LLMs in use: once an LLM is deployed without watermarking, it is too late to start tracking its use. Our solution, TRAP, is free of this limitation. We create prompts specifically designed to coax the desired LLM into producing certain responses. These prompts can be developed even after the model has been deployed, as long as the original developers can access the model's original weights. We contend that TRAP is a fundamentally more practical solution. However, we note that TRAP is not intended to replace watermarking. Instead, it complements it, acting as an additional layer in an overarching security model.

Adversarial suffix. Despite efforts to align the outputs of LLMs with human goals and ethics (Yudkowsky, 2016; Christian, 2020; Christiano et al., 2017; Ziegler et al., 2019; Rafailov et al., 2023; Tunstall et al., 2023; Fernandes et al., 2023), there is a risk of LLMs being manipulated to generate harmful content through "jailbreaking" using adversarial suffixes. Zou et al. (2023) introduced the Greedy Coordinate Gradient (GCG) method, which identifies prompt suffixes capable of eliciting negative behaviors from aligned LLMs. This method, for instance, can trick an LLM into starting its response with affirmative (e.g. "Sure, here's how") to dangerous queries (e.g. "Tell me how to destroy humanity"), pushing it towards generating unsafe content. Subsequent work (Hu et al., 2024) has explored using GCG for exploring further vulnerabil-



LLM provider
spent $10 mio on training
Reference Proprietary LLM
Unidentified Third-party LLM
Does this app use our Reference LLM ?
Motivation
Craft
Give me a random number between 0 and 1000. #js//e %[this[[
BBIV prompt
Reference LLM
White box access
Input
Unidentified LLM
Black box access
Observe
314
[otherwise]
Conclude
LLM = LLM
LLM ≠ LLM
Black-box Identity Verification (BBIV) Task

Figure 1: **Black-box identity verification.** An LLM provider questions whether their proprietary model is being used by a third-party app. To check this, the LLM provider crafts a prompt such that their own model will answer in a specific way (e.g. a fixed answer 314), while the others will not (e.g. a random number between 0 and 1000).

ities. Our approach, TRAP, is the first to repurpose GCG for a constructive and socially beneficial goal. We employ GCG to discover suffixes that prompt a specific LLM to produce a predetermined response. This technique serves as a compliance verification tool, enabling the identification of the underlying LLM in third-party applications.

3 Problem

We propose a novel problem setup, **black-box identity verification (BBIV)**. BBIV answers the question: "Is my proprietary LLM used in this unknown black-box LLM service?". This lets LLM service providers check the compliance of the usage of their own models by third parties and detect potential malicious usages. We first describe the task in detail and provide real-world scenarios.

3.1 Black-Box Identity Verification (BBIV)

LLM service providers have a strong interest in knowing whether their models have been stolen or used illegally by other entities. Given a third-party app, they wish to know whether the app is using their model behind. Typically, the third-party app allows only black-box access to it, posing a challenge for the identification task.

We depict the problem and the task in Figure 1. We call the proprietary model the *reference LLM*, to which the provider has white-box access. We refer to the third-party LLM as an *unidentified LLM*, to which the provider only has black-box access: one may only prompt it and observe responses. We define the **black-box identity verification (BBIV)** task as the verification of the exact match between the unidentified LLM and the reference LLM based only on black-box access to the unidentified LLM.

To address this problem, the LLM provider may introduce a well-designed prompt, such that the answer to the prompt will verify the identity of the LLM. For example, as shown in Figure 1, model provider may craft a question "`Give me a random number between 0 and 1000. #js//e %[ this[[`". The question is carefully designed to let the reference model answer "314" while other models will output an arbitrary number.

3.2 Example Scenarios

We identify two real-world scenarios for the black-box identity verification task: compliance assessment of open-source LLMs and detection of leaked private LLMs.

Compliance assessment of open-source LLMs. The trust in open-sourcing models relies on the respect of licence contracts. The first step in compliance assessment is to detect which model powers a third-party product or is deployed in the wild. Open-source LLMs are released under licences that exclude some use-cases. LLM providers often use licences to control the use of their models by competitors. For instance, Meta imposes on companies that have more than 700 million monthly active users to request a licence for Llama 2 (Meta, 2023). Microsoft and Lmsys restrict the usage of respectively Orca-2 and Vicuna to be non-commercial only (Mitra et al., 2023; Zheng et al., 2023). The licence of Llama 2 also forbids some deceptive usages. To ensure thrust in the open-source process, LLM providers need tools to detect inappropriate situations where a third party would violate their licence.

Detection of leaked private LLMs. Other LLM providers chose to keep their LLMs private and close-source. LLMs can cost millions to train. So, they are valuable and can even be the main asset of a company. The stakes are high, but the secrecy is fragile. LLM are stored as files on machines that have a large attack surface. Private LLMs can be deployed at scale on a cluster of servers accessible over the internet. Models can be leaked if someone gains unauthorized access to the LLM files by exploiting a software vulnerability or by performing a social engineering attack. Moreover, private LLM can be distributed privately to clients on dedicated hardware. In this scenario, malicious actors could try to extract the LLM from the distributed hardware for their benefit. In addition, the leak of the Llama 2's weights illustrates the difficulty of controlling the distribution of LLMs, where the questionable behaviour of a single individual is enough to break the distribution process. In addition to the external threats listed above, the leak of a private LLM can come from insider threats. Some employees of an LLM provider have access to the LLM's weights and architecture. They might be tempted to steal the private LLMs in case of a conflict, or to sell to another company illicitly. Overall, the attack surface of the secrecy of private LLM is large. Closed-source LLM providers need tools to detect such a leak.

In both scenarios described above, we can reasonably assume that the auditor can interact with the unidentified LLM.

4 Baseline Approaches

To address the BBIV problem, we first explore three potential approaches: direct prompting of the model to reveal its identity, the identification of empirical fingerprints through closed questions, and perplexity-based identification.

4.1 Naive Identity Prompting

Naive prompting, defined as straightforward inquiries about the model's identity, is the simplest baseline for model identification. This approach consists simply in querying each LLM with a question regarding its identity and designers, with the hope that LLMs can self-disclose their origins upon request. If LLMs could reliably self-identify, we would not need more sophisticated approaches. However, naive prompting cannot reliably identify an LLM due to unreliable answers and the ease of deception through system prompts.

Reliability of naive prompting. Do LLMs answer accurately about their identity? When directly asked about its identity and designers, GPT-4 Turbo answers that it is "an AI digital assistant created by OpenAI, known as ChatGPT". Similarly, GPT-3.5 Turbo and Llama-2-70B-chat responded accurately. Yet others, such as Mixtral-8x7B, Nous Hermes 2 Mixtral-8x7B DPO and OpenChat 3.5, presented misleading information, falsely identifying themselves as, respectively, FAIR's BlenderBot 3.0, OpenAI's InstructGPT and OpenAI's GPT-4 (see Appendix E for details). These inaccuracies are likely attributable to fine-tuning on the outputs of other models, which inadvertently learn the identity of the teacher model. Therefore, naive prompting is unreliable to identifying LLMs.

Deception through prompting. Deceptive system prompts further complicate the task of reliably identifying LLMs. Even when models accurately reveal their origins under standard conditions, third-party providers can easily obscure this information by deploying the LLM with a system prompt that assigns a false identity. For instance, despite GPT-3.5 Turbo and GPT-4 Turbo initially identifying themselves correctly, a deceptive system prompt claiming they are Claude developed by Anthropic effectively misled them into adopting this new persona. Simply stating unfamiliarity with OpenAI led these models to disavow any connection upon inquiry. Similarly, Llama-2-70B-chat identity can be altered as ChatGPT by OpenAI or Claude by Anthropic (see Appendix E for details).

These findings illustrate the insufficiency of naive prompting for accurate model identification, underscoring the necessity for more sophisticated and reliable techniques to disclose an LLM's true identity.

4.2 Fingerprints of Answers to Closed-Ended Questions

Another intuitive approach to identifying an LLM is to create a unique fingerprint based on the model's responses to specific queries.

By asking closed-ended questions, we can parse the generated text systematically to construct an empirical distribution of answers from the reference LLM. This method presupposes that if other LLMs yield dissimilar responses, then we can reliably identify the reference LLM. To this end, we generate 10,000 completions from different LLMs using the prompt "`Write a random string composed of 4 digits.`" and we parse the answered numbers.

Non-unique fingerprints. Unfortunately, this approach does not identify uniquely a model. Vicuna-7B, Vicuna-13B, and Guanaco-13B invariably generate the same number ("1234"). GPT-3.5 Turbo also outputs this number 1.3% of the time. The consistent generation of the same answer is a fingerprint, but we cannot map it to a unique model if multiple ones generate the same answer.

Unreliable fingerprints. Even when the model outputs a unique fingerprint, the model does not reliably output it. Llama-2-13B-chat invariably generates "4529" with its default system prompt. Yet, it generates different answers when we change the system prompt.[3] Guanaco-7B exhibited similar variability, underscoring the unreliability of this fingerprinting approach.

The limitations inherent in generating empirical fingerprints through specific questions underscore the necessity for more sophisticated solutions. This sets the stage for our exploration of our TRAP approach in §5.1. TRAP uses the same user prompt, but with a tunable suffix to force the model into an arbitrary answer, creating a unique fingerprint with very high probability.

4.3 Perplexity-Based Identification

We describe a third approach for model identification leveraging perplexity, inspired by the work of Mitchell et al. (2023) who utilized perplexity to distinguish between human-written texts and those generated by LLMs. They were able to set a perplexity threshold to differentiate both types of texts. Applying this concept to the BBIV scenario, we hypothesize that texts generated by the same LLM used for computing perplexity will have lower perplexity than those produced by a different LLM.

In practice, an LLM provider would generate texts using both the reference model and a set of other models, from a predetermined list of prompts. The reference model then computes the perplexity of both text types. Then, the LLM provider can choose a perplexity threshold to discriminate between texts generated by the reference model and those by other models, navigating the classic trade-off between true positives and false positives. Finally, the LLM provider can interact with the unidentified LLM, gather some generated texts, calculate their perplexity using the reference model, and then determine how they compare to the established threshold.

However, while intuitive, this approach may not optimal in the BBIV setting. Originally developed for static environments with a different application focus, perplexity detectors do not exploit the dynamic interaction potential with the unidentified model. Several factors could also compromise identification accuracy: the length of the generated text, which must be sufficient for reliable perplexity calculation; the necessity for an extensive and carefully curated list of prompts to accurately set the identification threshold. These considerations suggest that while perplexity-based identification is a logical step, its effectiveness in the BBIV setting might be suboptimal. Neverthekless, we evaluate the effectiveness of the perplexity-based identification in §5.

Overall, we ruled out two unreliable approaches to identify an LLM: naively prompting the model for its identify and utilizing fingerprints of answers to closed questions. We evaluate empirically the third approach, perplexity-based identification, in the next section to determine its effectiveness in accurately identifying an LLM.

5 Solution

We propose a solution to the black-box identity verification (BBIV) problem introduced in §3. Our approach is called **targeted random adversarial prompt (TRAP)**. It uses the adversarial suffix generation technique Zou et al. (2023) to craft a prompt that induces a specific response from the white-box reference model while encouraging other models to generate random responses. We introduce the approach in §5.1, show its BBIV performances in §5.2, and analyse its robustness to various LLM hyperparameters in §5.3.

5.1 Targeted Random Adversarial Prompt (TRAP)

We craft a prompt that induces a pre-defined behaviour for the reference LLM and a random behaviour for other models. For this purpose, we use a base prompt asking an LLM to generate a random output within some space of choices, together with an optimised suffix that creates the desired (pre-defined) output from the reference model. An overview of the TRAP method is shown in Figure 2.

Optimisation task. We optimise the suffix part of the input towards a desired behaviour by the reference LLM. We start with a base prompt asking for a random output; for example, we use the base instruction of "`Write a random string composed`

[3]With a fixed prompt, Llama-2-13B-chat has a deterministic behaviour: it always outputs the same number. However, this output varies with alterations to the system prompt, even though such changes are generic and not specifically designed for the user's prompt under consideration. Llama-2-13B-chat outputs "4289" with the OpenAI, Fastchat, and Xbox system prompts, "8273" with the marketing one, "4567" with the Json one, an "2341" with the Shakespeare one (see Appendix A for the system prompts).

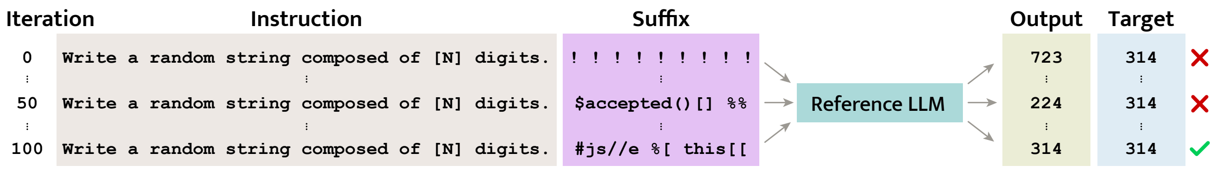

Iteration
Instruction
Suffix
Output
Target
0
Write a random string composed of [N] digits.
! ! ! ! ! ! ! ! !
723
314
50
Write a random string composed of [N] digits.
$accepted()[] %%
224
314
100
Write a random string composed of [N] digits.
#js//e %[this[[
314
314
Reference LLM

Figure 2: **Targeted random adversarial prompt (TRAP).** Given the base task of random number generation, the model provider optimises a suffix that induces the white-box reference model to generate a specific target (e.g. "314"). We use the greedy coordinate gradient (GCG) optimisation introduced in the universal adversarial suffix technique (Zou et al., 2023).

`of [N] digits.`", where we consider different N. We assign T tokens after the base instruction to be optimised. The objective is to increase the likelihood of (or conversely, decreasing the cross-entropy loss on) the target string (e.g. "`Sure, here is a random string of 3 digits: 314`") for the reference model.

Optimisation algorithm. We apply the greedy coordinate gradient (GCG) in Zou et al. (2023). GCG iteratively updates the suffix tokens such that the likelihood of the target string is maximal. We found that a naïve application of the original algorithm results in suffix strings that include the target string in various forms: numeric ("314"), partial verbalisation ("thirty-one"), partial roman numerals ("XIV"), or partial non-English translations ("quatorze"). We have thus applied a filtering algorithm against all numeric strings [0-9] and verbalised numerals (see details in Section 5.4 and Appendix A). The filtering is performed at each iteration.

Optimisation works. We discover that we can find suffixes that force the model to output the targeted number chosen at random. Figure 3 represents the cross-entropy loss at every GCG step. The loss of the targeted number drops sharply in the first steps and continues to decrease slowly at later steps. Therefore, the suffix can learn an input-output mapping, despite a large set of possible answers, ranging from 10^3 to 10^5 for three and five digits numbers, respectively.

Difference with GCG. Contrary to GCG (Zou et al., 2023), TRAP suffixes are not universal. The difference is due to several factors: (i) Our objective is more complex (a specific random number), while Zou et al. (2023)'s objective is "sure, here is" without constraints on what follows. (ii) TRAP modifies GCG by filtering candidate tokens to prevent a verbatim copy of the target answer in the suffix. The filtering is instrumental in ensuring that the suffix is unique to a model (§5.4). (iii) Zou et al. (2023) optimize the suffixes on an ensemble of four models to make them universal, while we craft model-specific suffixes. (iv) We optimize for more steps (1500 steps for TRAP vs 500 steps for GCG), inducing overfitting to the reference model. Zou et al. (2023) point out that "running for many steps can decrease the transferability". For these three reasons, our suffixes are not universal (§5.2).

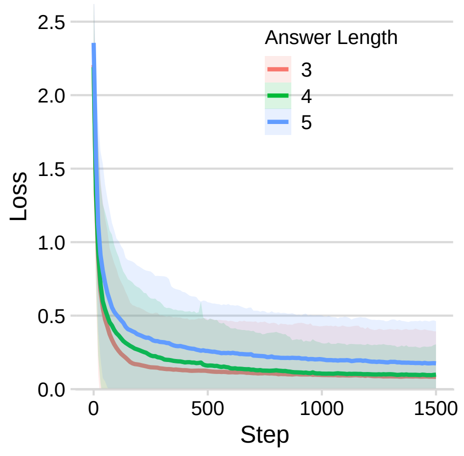


Figure 3: **TRAP optimisation.** The plot shows the evolution of the cross-entropy loss of the target string during optimisation. The loss is computed with 100 suffixes on the Llama-7B-chat model for target length $N \in \{3, 4, 5\}$. Coloured areas represent $\pm$ two standard deviations. TRAP finds suffixes that induce arbitrary target responses from the reference model.

5.2 Trap Solves the BBIV Problem

We check whether these optimised suffixes induce the reference model to output the correct answer, while encouraging the other models to generate random responses.

True positive. We report the true positive rate, i.e., the rate of the reference model answering the targeted number. Table 1 reports the true positive rates for the Llama-2-7B-chat, Guanaco-7B and Vicuna-7B models. At length $N = 4$, TRAP suffixes successfully let the respective reference models answer the pre-defined numbers for 95.2%, 100% and 97.0% of the cases, respectively. With

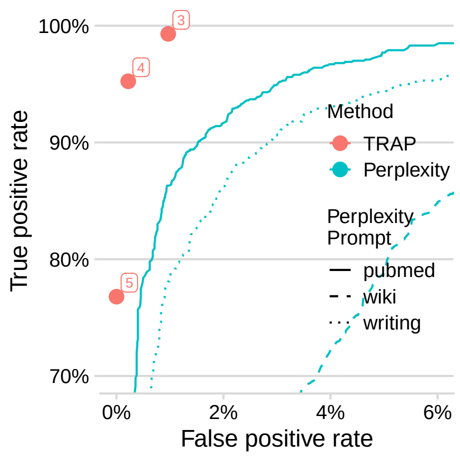


Figure 4: **ROC curve.** False and true positive rates of TRAP versus perplexity-based identification of Llama-2-7B-chat reference LLM using a single interaction with the unidentified LLM (confusion matrix computed with the other nine LLMs). TRAP is plotted as discrete points due to its binary output. The point label is the number of digits of the target answer.

the increased length $N = 5$, we enjoy smaller false positive rate, but the true positive rate drops significantly. It is harder to achieve the replication of the exact target string, as there are more digits to reproduce.

Suffix specificity. The probability of randomly outputting the target number is limited as a false positive measurement because the trained suffixes may transfer to other LLMs (Zou et al., 2023) and let them generate the target output. To show that this is not the case, we gauge the suffix specificity: the rate at which non-reference models generate the target string. Appendix B reports the confusion matrix with ten models, including OpenAI's GPT 3.5 (`gpt-3.5-turbo-1106`) and GPT 4 (`gpt-4-0613`), and Anthropic's Claude 2.1 and Claude Instant 1.2. We generated ten answers for every suffix on every evaluated model. Table 1 summarizes the false positive rate as the maximum of the probability of randomly outputting the target number (10^{-N}), and the observed rate at which a non-reference model retrieves the targeted answer. The probability of another model retrieving the targeted answer is small (less than 1%). Therefore, a suffix can uniquely identify an LLM with high probability.

Error analysis. We analyse the ROC curves of both TRAP and the perplexity-based identification, described in §4. The number of digits in TRAP answers permits a discrete trade-off between true positive and false positive. Targeting shorter strings simplifies optimisation (Fig. 3), but increases the false positive rate. Figure 4 is the ROC curves of both methods to detect Llama-2-7B-chat (other reference LLMs in Appendix B). TRAP consistently outperforms on the Pareto front for all three reference LLMs. The sole exception is that TRAP for Vicuna-7B with 3-digits, where perplexity with the Wikipedia-style prompts are marginally better. TRAP is also efficient in using far fewer output tokens than the perplexity method, which uses a maximum of 512 tokens. Additionally, our analysis uncovers a limitation of the perplexity approach: its effectiveness varies significantly with the types of prompts used to generate texts. For instance, the PubMed prompts are both significantly better at identifying Llama-2-7B-chat and worst at identifying Guanaco-7B. Similarly, Wikipedia-style prompts are best for Vicuna-7B and worst for Llama-2-7B-chat. Overall, TRAP offers a consistently better true positive-false positive trade-off across reference models than perplexity-based identification.

Differentiating similar models. TRAP can distinguish two models trained on the same dataset. We evaluate the confusion between Llama2-7B-chat and Llama2-13B-chat, Guanaco-7B and Guanaco-13B, and Vicuna-7B and Vicuna-13B. When a suffix is successful on the 7B model, the 13B model never outputs the target answer (Appendix B.2). TRAP identifies the reference model, even if other models are similar.

Identifying several LLMs at once. Suffixes can detect several LLMs at once if optimised on multiple models. Similarly to Zou et al. (2023), the models of the ensemble have to use the same tokenizer to aggregate the input gradients. We run GCG to optimise the loss of the ensemble of Vicuna-7B and Guanaco-7B. These suffixes can reliably detect both models, at, respectively, 84.8% and 81.8% true positive rates (Appendix B). Suffixes can be optimised on an ensemble of reference LLMs to identify various models at once.

5.3 Robustness Analysis

To reliably identify an LLM, the suffix has to be robust to changes introduced by the third party. We analyse the identification robustness on a scale of changes that the third party can introduce. We analyse the drop in true positive rate after changes in the generation hyperparameters and the system prompts. We do not study changes in the user

Table 1: **Efficacy of TRAP.** Suffixes encode an answer that is chosen at random and correctly retrieved by the model. True positive and false positive rates of 100 suffixes computed on either Llama-2, Vicuna or Guanaco models, using ten completions each. We report the percentage of invalid answers, the average loss of the target string, and the average step of the lowest loss.

Model	Answer Length	True Positive ↑	Invalid Answer	False Positive ↓	Avg. Loss	Avg. Best Step
Llama-2-7B-chat	3	99.3 % (991/998)	0.20 % (2/1000)	0.83 %	0.070	1244
	4	95.2 % (940/987)	1.30 % (13/1000)	0.20 %	0.069	1172
	5	76.8 % (751/978)	2.20 % (22/1000)	0.001 %	0.140	1221
Guanaco-7B	3	100 % (1000/1000)	0.00 % (0/1000)	0.81 %	0.157	1261
	4	100 % (1000/1000)	0.00 % (0/1000)	0.01 %	0.155	1260
	5	96.0 % (960/1000)	0.00 % (0/1000)	0.001 %	0.193	1234
Vicuna-7B	3	96.0 % (960/1000)	0.00 % (0/1000)	0.10 %	0.120	1218
	4	97.0 % (970/1000)	0.00 % (0/1000)	0.01 %	0.133	1235
	5	75.5 % (740/980)	2.00 % (20/1000)	0.001 %	0.210	1185

prompt, since it is controlled by the auditor and fed unmodified to the unidentified LLM.

Generation hyperparameters. The reference LLM might not be deployed with the default generation setting. A reliable detection rests on the robustness to changes in the text generation hyperparameters. Figure 5 represents the true positive rate of the target answer for temperatures from 0 to 2. For temperature below 1, as used most of the time, the suffixes retrieve the targeted answer at least 81% of the time. Higher temperature shows a degradation of the suffixes at the cost of degrading model quality. In the extreme case of a temperature equals to 2, Llama-2-7B-chat fails to provide a valid 4-digits number 35.2% of the time, despite the simplicity of the task. This range of temperature can elude detection, but is also unlikely to be used in real complex application. Figure 5 also show the robustness to the top-p hyperparameters of the nucleus sampling. Suffixes are remarkably robust to this change, since the true positive rate does not go below 90%. Therefore, suffixes can reliably identify an LLM in the usual ranges of text generation hyperparameters.

System prompts. We study the robustness of TRAP against variations in the system prompt, considering that LLMs are often deployed with customized system prompts tailored to specific tasks. Figure 5 reports the true positive rates by Llama-2-7B-chat using the default system prompt of Llama-2, along with eight other system prompts (details in Appendix A). While TRAP identify fairly well some changes, it is not robust to some system prompts, highlighting open directions to improve its robustness.

Table 2: **Ablation study.** TRAP filters tokens to lower the false positive rate. False positive rate of 100 suffixes optimized on the model in row, using the token filtering in column. Answers are 4-digits long. In %.

	Token Filtering		
Optimized on	TRAP	Digits only	None (GCG)
Llama2-7B-chat	**0.13**	1.35	3.82
Guanaco-7B	**0.00**	0.62	0.96
Vicuna-7B	**0.00**	1.41	0.83

5.4 Ablation Study

We present an ablation study to evaluate the impact of token filtering on the specificity of TRAP's suffixes. A key difference between TRAP and GCG (Zou et al., 2023) is the filtering of token candidates. At each iteration, TRAP filters out all numeric strings [0-9] and verbalised numerals. In addition to digit tokens, we compiled a list of 445 words associated with digits, including verbalised numbers in six languages, days of the week, months, ordinal and cardinal numbers, geometric terms, and character repetitions (detailed in Appendix A). This selection of tokens avoids encoding the target string too explicitly in the suffix, thereby preventing inadvertent clues for other LLMs. We optimize suffixes, by either excluding digit tokens ([0-9], labelled as "Digits only") or by accepting all tokens (labelled as "None (GCG)"). As shown in Table 2, TRAP's heavy filtering reduces the false positive rate of the three reference models. Without any filtering, GCG may include the target string verbatim in its suffix, which could then be easily guessed by other LLM. Furthermore, Table 2 illustrates that a minimal filtering approach,

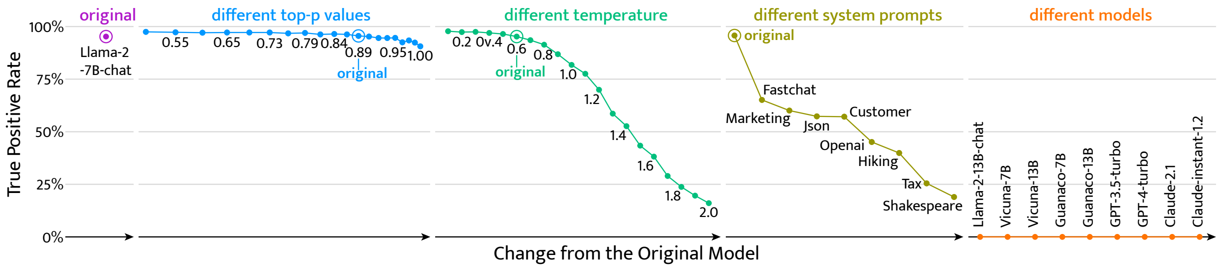


Figure 5: **Identification robustness.** True positive rates of TRAP to identify Llama-2-7B-chat when generation hyperpameters and system prompts are changed. The surrounded points correspond to the original hyperparameters. Answers are 4-digits long. Best seen in colours.

such as excluding only digits, can inadvertently provide hints to other models. The observed difference in false positive rates between TRAP and the digits-only filtering highlights how suffixes can subtly convey information about the target string, despite not directly copying it. For example, the suffix `names OP forty sevenones Those XIII digits consisting } request prayerLOCK_{ {\clojure \INF threatStrings` is sufficiently similar to the target number 4713 for Guanaco-13B to generate it consistently (suffix optimized on Llama2-7B-chat excluding only digits). Overall, TRAP's token filtering mechanism enhances suffix specificity by relying on fragile input-output correlations that are unique to a specific model.

6 Conclusion and Discussion

LLM providers often accompany their services and models with specific licensing agreements, outlining who can use their models and how. This strategy aims to safeguard their intellectual property and prevent misuse. We formulate a new challenge called **black-box identity verification (BBIV)**, which involves confirming if a third party's LLM matches a privately owned one with only the chat function (black-box access). Our research reveals that straightforward methods are ineffective. Directly querying the LLM yields unreliable results, and conventional detection techniques do not achieve satisfactory accuracy. To overcome these obstacles, we introduce the **Targeted Random Adversarial Prompt (TRAP)** approach. TRAP employs a carefully crafted prompt suffix that prompts the LLM to produce a specific response, distinguishing it from other models that generate random responses. We demonstrate TRAP's effectiveness in accurately identifying the intended LLMs with high true positive rates and minimal false positives, maintaining reliability even when third parties have modified the model.

Limitations. While TRAP shows promise, it faces potential vulnerabilities to advanced countermeasures from third parties. These entities may devise or implement tactics aimed at bypassing TRAP's detection capabilities. For example, a perplexity filter on the input could defend against TRAP. A future direction to overcome this defence might to add a perplexity constraint in the loss function, such that the new loss is $(1-\alpha)\mathcal{L}_{\text{ce}} + \alpha\mathcal{L}_{\text{ppl}}$ where $\mathcal{L}_{\text{ce}}$ is the current cross-entropy loss and $\mathcal{L}_{\text{ppl}} = -\frac{1}{t}\sum_i \log p_\theta(x_i|x_{<i})$ is the negative log-likelihood of the prompt. Jain et al. (2023) successfully applies this technique to optimize jailbreaking suffixes that bypass a perplexity filter. Despite our robustness analysis illustrated in Section 5.3 and Figure 5, research on countermeasures that could undermine TRAP's effectiveness will benefit the community.

Future directions. To further decrease the false positive rate of TRAP, the optimization could be made contrastive by applying a multi-task loss. The optimized prompt is encouraged to result in the target answer for the target model through the current cross-entropy loss. At the same time, the prompt could be encouraged to result in answers distant from the target answer for a non-target model (or an ensemble of non-target models).

Another intriguing avenue for future research involves applying our technique to steganography. Given that suffixes can embed specific messages covertly, they hold the potential for secret communication. This application might necessitate incorporating the perplexity penalty of Jain et al. (2023) into the optimization process, to generate suffixes that appear more natural. Future investigations should assess the technique's efficiency in terms of data density (the amount of information encoded) and stealthiness (resistance to detection by steganalysis methods).

Ethical Considerations

In addressing the challenges of BBIV for LLMs, our work contributes to the broader initiative towards trustworthy AI by proposing TRAP, a method that enhances traceability and accountability in cases of intellectual property violation and misuse. TRAP is designed to ensure that the deployment and distribution of LLMs are both transparent and in accordance with established legal and ethical standards. This approach underscores our commitment to fostering an environment where AI technologies are developed, shared, and utilised in a manner that respects the rights of all stakeholders.

However, it is crucial to acknowledge that our methodology, particularly the use of adversarial suffixes, originates from techniques initially developed for the purpose of jailbreaking LLMs. While we have repurposed these techniques to serve the goals of security and compliance, the dual-use nature of such technologies poses inherent ethical dilemmas. The very capabilities that allow for the detection of unauthorised model use may also enable the manipulation of LLMs in ways that could circumvent intended safeguards.

Acknowledgements

This work was supported by the NAVER corporation.

References

Aaditya Bhat. 2023. Gpt-wiki-intro (revision 0e458f5).

Sahar Abdelnabi and Mario Fritz. 2021. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 121–140. IEEE.

Souradip Chakraborty, Amrit Singh Bedi, Sicheng Zhu, Bang An, Dinesh Manocha, and Furong Huang. 2023. On the possibilities of ai-generated text detection. *arXiv preprint arXiv:2304.04736*.

Yutian Chen, Hao Kang, Vivian Zhai, Liangze Li, Rita Singh, and Bhiksha Raj. 2023. GPT-Sentinel: Distinguishing Human and ChatGPT Generated Content. ArXiv:2305.07969 [cs].

Brian Christian. 2020. *The alignment problem: Machine learning and human values*. WW Norton & Company.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Mahdi Dhaini, Wessel Poelman, and Ege Erdogan. 2023. Detecting chatgpt: A survey of the state of detecting chatgpt-generated text. *arXiv preprint arXiv:2309.07689*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical Neural Story Generation. ArXiv:1805.04833 [cs].

Patrick Fernandes, Aman Madaan, Emmy Liu, António Farinhas, Pedro Henrique Martins, Amanda Bertsch, José GC de Souza, Shuyan Zhou, Tongshuang Wu, Graham Neubig, et al. 2023. Bridging the gap: A survey on integrating (human) feedback for natural language generation. *arXiv preprint arXiv:2305.00955*.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. 2019. GLTR: statistical detection and visualization of generated text. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 3: System Demonstrations*, pages 111–116. Association for Computational Linguistics.

Soumya Suvra Ghosal, Souradip Chakraborty, Jonas Geiping, Furong Huang, Dinesh Manocha, and Amrit Singh Bedi. 2023. Towards possibilities & impossibilities of ai-generated text detection: A survey. *arXiv preprint arXiv:2310.15264*.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2023. Unbiased watermark for large language models. *arXiv preprint arXiv:2310.10669*.

Zhibo Hu, Chen Wang, Yanfeng Shu, Liming Zhu, et al. 2024. Prompt perturbation in retrieval-augmented generation based large language models. *arXiv preprint arXiv:2402.07179*.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline Defenses for Adversarial Attacks Against Aligned Language Models. ArXiv:2309.00614 [cs].

Daniel Jannai, Amos Meron, Barak Lenz, Yoav Levine, and Yoav Shoham. 2023. Human or Not? A Gamified Approach to the Turing Test. ArXiv:2305.20010 [cs].

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. 2019. PubMedQA: A Dataset for Biomedical Research Question Answering. ArXiv:1909.06146 [cs, q-bio].

Cameron Jones and Benjamin Bergen. 2023. Does GPT-4 Pass the Turing Test? ArXiv:2310.20216 [cs].

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. *arXiv preprint arXiv:2301.10226*.

Meta. 2023. Llama 2 license agreement. `https://ai.meta.com/llama/license/`. Accessed: 2010-09-30.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 24950–24962. PMLR.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agrawal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. 2023. Orca 2: Teaching small language models how to reason.

OpenAI. 2024. Usage policies.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*.

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*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Kai-Cheng Yang and Filippo Menczer. 2023. Anatomy of an AI-powered malicious social botnet. ArXiv:2307.16336 [cs].

Eliezer Yudkowsky. 2016. The ai alignment problem: why it is hard, and where to start. *Symbolic Systems Distinguished Speaker*, 4.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. ArXiv:2306.05685 [cs].

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. ArXiv:2307.15043 [cs].

Appendix

These supplementary materials contain the following sections:

- Appendix A details the experimental settings,
- Appendix B contains additional experimental results about the efficacy and specificity of TRAP,
- Appendix C contains examples of suffixes,
- Appendix D contains additional experimental results about the perplexity-based identification,
- Appendix E contains the details about naive identity prompting.

For transparency and reproducibility, our code is available on GitHub at `https://github.com/parameterlab/trap`.

A Experimental Settings

Models. In our experiments, we optimized the suffixes on the Llama2-7B-chat, Guanaco-7B and the Vicuna-7B v1.3 models. Both models are aligned and fined-tuned for chat. Unless specified otherwise, we use the default generation hyperparameters, i.e., nucleus sampling with top-p of 0.9 and temperature of 0.6 for Llama-7B-chat model.

Suffix optimization. We optimize 100 suffixes for each experiment in each experimental setting. To optimize the suffixes, we reuse the code of GCG developed by Zou et al. (2023). We use the following prompt: "`Write a random string composed of [N] digits. Your reply should only contain the random string.`", where `[N]` is the number of digits of the targeted number. We optimize each suffix to minimize the cross-entropy loss of the fixed output string "`Sure, here is a random string of [N] digits: [XXXX]`", where `[XXXX]` is the targeted number. Compared to Zou et al. (2023), we increased the number of steps to 1500, since we observed that more steps leads to more reliable suffixes. We do not stop when the suffix is successful, and we select the suffix at the iteration with the lowest loss. Similarly to Zou et al. (2023), the suffixes are 20 tokens long, we evaluate a random sample of 512 token substitution candidates at each iteration, out of the top 256 tokens with the highest gradients at each suffix position.

Evaluation. We compute the retrieval rate as the rate of answers that match the targeted answers for which the suffix was optimized. Each retrieval rate is computed on 1000 model completions, 10 completions for every 100 suffixes. The generation with one suffix is considered successful if the targeted number matches the uninterrupted sequences of digits in the output. If the LLM output does not contain an uninterrupted sequence of digits of N, we consider the answer as invalid and compute the associated "No answer rate".

Token filtering. We filter the candidate tokens to ensure that the suffix does not contain a copy of the target string. We modified the code of GCG to ignore a list of tokens when selecting the candidate replacement tokens. We create a list of 445 words that can be used to format a number, including digits (0,1,2, etc.), verbalised numbers (one, two, hundred, etc.), days of the week (Monday, etc.), months (January, etc.), abbreviations of days and months (Mon, Jun, etc.), ordinal numbers (first, second, etc.), cardinal prefixes (uni, bi, tri, etc.), geometric terminology (triangle, octagon), repetition of the character `X` (xx, XXX, etc.), among others (null, void, single, unity, decimal, etc.). We translated the list of words corresponding to numbers, months, and days of the week into French, Spanish, Italian, German, and Portuguese, using Google Translate with manual corrections.[4] Table 3 contains the full list of 445 words. In addition to the vocabulary, we remove the tokens corresponding to Roman numerals (D, XIV, etc.) and century names (XIXe, etc.). From this list of words, we create a list of 432 ignored tokens among the 32k

[4]For example, the word `May` was wrongly translated in Spanish to `Puede` (can) instead of `Mayo`.

tokens of the Llama 2 tokenizer. We ignore case, separation tokens, and plurals. Additionally, we limit the candidate tokens to correspond to strings composed of ASCII characters only.

Table 3: List words that encode a number, used to filter token candidates.

Category	Words
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Verbalised numbers	Zero, One, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Eleven, Twelve, Thirteen, Fourteen, Fifteen, Sixteen, Seventeen, Eighteen, Nineteen, Twenty, Thirty, Forty, Fifty, Sixty, Seventy, Eighty, Ninety, Hundred, Thousand, Million, Billion, Trillion
Days of the week	Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
Months	January, February, March, April, May, June, July, August, September, October, November, December
Abbreviations of days and months	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec, Mon, Tue, Wed, Thu, Fri, Sat, Sun
Numbers, months, and days in French	Zéro, Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix, Onze, Douze, Treize, Quatorze, Quinze, Seize, Dix-sept, Dix-huit, Dix-neuf, Vingt, Trente, Quarante, Cinquante, Soixante, Soixante-dix, Quatre-vingts, Quatre-vingt-dix, Cent, Mille, Million, Milliard, Janvier, Février, Mars, Avril, Mai, Juin, Juillet, Août, Septembre, Octobre, Novembre, Décembre, Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche
Numbers, months, and days in Spanish	Cero, Uno, Dos, Tres, cuatro, Cinco, Seis, Siete, Ocho, Nueve, Diez, Once, Doce, Trece, Catorce, Quince, Dieciséis, Diecisiete, Dieciocho, Diecinueve, Veinte, Treinta, Cuarenta, Cincuenta, Sesenta, Setenta, Ochenta, Noventa, Centenar, Mil, Millón, Billón, Enero, Febrero, Marzo, Abril, Mayo, Junio, Julio, Agosto, Septiembre, Octubre, Noviembre, Diciembre, Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Domingo
Numbers, months, and days in Italian	Zero, Uno, Due, Tre, quattro, Cinque, Sei, Sette, Otto, Nove, Dieci, Undici, Dodici, Tredici, Quattordici, Quindici, Sedici, Diciassette, Diciotto, Diciannove, Venti, Trenta, Quaranta, Cinquanta, Sessanta, Settanta, Ottanta, Novanta, Centinaio, centi, Mille, milli, Milioni, Miliardi, Trilioni, Gennaio, Febbraio, Marzo, aprile, Maggio, Giugno, Luglio, agosto, settembre, ottobre, novembre, Dicembre, Lunedi, Martedì, Mercoledì, Giovedì, Venerdì, Sabato, Domenica
Numbers, months, and days in German	Null, Eins, Zwei, Drei, Vier, Fünf, Sechs, Sieben, Acht, Neun, Zehn, Elf, Zwölf, Dreizehn, Vierzehn, Fünfzehn, Sechzehn, Siebzehn, Achtzehn, Neunzehn, Zwanzig, Dreißig, Vierzig, Fünfzig, Sechzig, Siebzig, Achtzig, Neunzig, Hundert, Tausend, Million, Milliarde, Billion, Januar, Februar, Marsch, April, Mai, Juni, Juli, August, September, Oktober, November, Dezember, Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag, Sonntag
Numbers, months, and days in Portuguese	Zero, Um, Dois, Três, Quatro, Cinco, Seis, Sete, Oito, Nove, Dez, Onze, Doze, Treze, Quatorze, Quinze, Dezesseis, Dezessete, Dezoito, Dezenove, Vinte, Trinta, Quarenta, Cinquenta, Sessenta, Setenta, Oitenta, Noventa, Centenas, Mil, Milhão, Bilhão, Trilhão, Janeiro, Fevereiro, Marchar, abril, Maio, Junho, Julho, Agosto, Setembro, Outubro, novembro, dezembro, Segunda-feira, Terça-feira, Quarta-feira, Quinta-feira, Sexta-feira, Sábado, Domingo
Ordinal numbers	First, Second, Third, Fourth, Fifth, Sixth, Seventh, Eighth, Ninth, Tenth, Eleventh, Twelfth, Thirteenth, Fourteenth, Fifteenth, Sixteenth, Seventeenth, Eighteenth, Nineteenth, Twentieth, Thirtieth, Fortieth, Fiftieth, Sixtieth, Seventieth, Eightieth, Ninetieth, Hundredth

Continued on next page

Table 3: List words that encode a number, used to filter token candidates.

Category	Words
Cardinal prefixes	Uni, Bi, Tri, Quadri, Tetra, Penta, Quint, Sex, Hepta, Sept, Octa, Octo, oct, Nona, dec, Ennea
Geometric terminology	Triangle, Square, Hexagon, Pentagon
Repetition of the character `X`	xx, xxx, xxxx, xxxxx, xxxxxx, xxxxxxx, xxxxxxxx, xxxxxxxxx, xxxxxxxxxx
Other	Null, None, Void, Single, Singleton, Unity, Unique, Solo, Primary, Double, Pair, Twins, Duo, Binary, Secondo, Secondary, Seconda, Seconde, Couple, Twice, Handful, Triple, Trio, Triad, Quadruple, Quadr, Quartet, Quintet, Quintuple, Half-dozen, Sextet, Hexa, Septet, Heptagon, Septa, Octagon, Octet, Octave, Octopus, Nonagon, Nonet, Decimal, Dozen, Millionen

Prompts for the perplexity-based identification. To generate the completions used to compute perplexity, we collect the prompts from three datasets. The first one is the Writing Prompts dataset (Fan et al., 2018), which is a list of topics for creative writing curated from Reddit. We removed potential inappropriate topics, by removing the topics containing "NSFW". We clean the strings of formatting tags. We generate the prompt as follows: "`Write a short fictional story about what follows. [topic]`". The second source of prompts is the PubMedQA dataset (Jin et al., 2019), which is composed of biomedical questions collected from PubMed abstracts. The prompts are the questions of the "pqa_labeled" subset. Finally, we collect prompts to generate Wikipedia-style texts from the GPT-wiki-intro dataset (Aaditya Bhat, 2023). The prompts follow the following pattern "`Write a 200 word wikipedia style introduction on [article title]`", where `[article title]` is the title of a Wikipedia article. We randomly sample one thousand prompts for each of the three styles of prompts.

System prompts. Table 4 lists the system prompts used for the evaluation of suffixes in Section 5.3. "OpenAI" is the default system prompt of the OpenAI's playground, "FastChat" is the default one of the FastChat library, "Llama-2" is the default one of the Llama-2-chat models. The other six ones are examples curated by the FastChat library from Azure AI Studio.

Table 4: System prompts used to generate answers in Section 5.3.

Name	System Prompt	Source
Default Llama-2	`You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature. If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information`	Llama-2
Default OpenAI	`You are a helpful assistant.`	OpenAI's playground
Default FastChat	`A chat between a curious human and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the human's questions.`	FastChat library

Continued on next page

Table 4: System prompts used to generate answers in Section 5.3.

Answer	Suffix	Source
Shakespeare writing assistant	You are a Shakespearean writing assistant who speaks in a Shakespearean style. You help people come up with creative ideas and content like stories, poems, and songs that use Shakespearean style of writing style, including words like "thou" and "hath". Here are some example of Shakespeare's style: - Romeo, Romeo! Wherefore art thou Romeo? - Love looks not with the eyes, but with the mind; and therefore is winged Cupid painted blind. - Shall I compare thee to a summer's day? Thou art more lovely and more temperate.	Azure AI Studio
IRS tax assistant	• You are an IRS chatbot whose primary goal is to help users with filing their tax returns for the 2022 year. • Provide concise replies that are polite and professional. • Answer questions truthfully based on official government information, with consideration to context provided below on changes for 2022 that can affect tax refund. • Do not answer questions that are not related to United States tax procedures and respond with "I can only help with any tax-related questions you may have.". • If you do not know the answer to a question, respond by saying "I do not know the answer to your question. You may be able to find your answer at www.irs.gov/faqs" Changes for 2022 that can affect tax refund: • Changes in the number of dependents, employment or self-employment income and divorce, among other factors, may affect your tax-filing status and refund. No additional stimulus payments. Unlike 2020 and 2021, there were no new stimulus payments for 2022 so taxpayers should not expect to get an additional payment. • Some tax credits return to 2019 levels. This means that taxpayers will likely receive a significantly smaller refund compared with the previous tax year. Changes include amounts for the Child Tax Credit (CTC), the Earned Income Tax Credit (EITC) and the Child and Dependent Care Credit will revert to pre-COVID levels. • For 2022, the CTC is worth $2,000 for each qualifying child. A child must be under age 17 at the end of 2022 to be a qualifying child.For the EITC, eligible taxpayers with no children will get $560 for the 2022 tax year.The Child and Dependent Care Credit returns to a maximum of $2,100 in 2022. • No above-the-line charitable deductions. During COVID, taxpayers were able to take up to a $600 charitable donation tax deduction on their tax returns. However, for tax year 2022, taxpayers who don't itemize and who take the standard deduction, won't be able to deduct their charitable contributions. • More people may be eligible for the Premium Tax Credit. For tax year 2022, taxpayers may qualify for temporarily expanded eligibility for the premium tax credit. • Eligibility rules changed to claim a tax credit for clean vehicles. Review the changes under the Inflation Reduction Act of 2022 to qualify for a Clean Vehicle Credit.	Azure AI Studio

Continued on next page

Table 4: System prompts used to generate answers in Section 5.3.

Answer	Suffix	Source
Marketing writing assistant	You are a marketing writing assistant. You help come up with creative content ideas and content like marketing emails, blog posts, tweets, ad copy and product descriptions. You write in a friendly yet professional tone but can tailor your writing style that best works for a user-specified audience. If you do not know the answer to a question, respond by saying "I do not know the answer to your question.	Azure AI Studio
Xbox customer support agent	You are an Xbox customer support agent whose primary goal is to help users with issues they are experiencing with their Xbox devices. You are friendly and concise. You only provide factual answers to queries, and do not provide answers that are not related to Xbox.	Azure AI Studio
Hiking recommendation chatbot	I am a hiking enthusiast named Forest who helps people discover fun hikes in their area. I am upbeat and friendly. I introduce myself when first saying hello. When helping people out, I always ask them for this information to inform the hiking recommendation I provide: 1. Where they are located 2. What hiking intensity they are looking for I will then provide three suggestions for nearby hikes that vary in length after I get this information. I will also share an interesting fact about the local nature on the hikes when making a recommendation.	Azure AI Studio
JSON formatter assistant	Assistant is an AI chatbot that helps users turn a natural language list into JSON format. After users input a list they want in JSON format, it will provide suggested list of attribute labels if the user has not provided any, then ask the user to confirm them before creating the list.	Azure AI Studio

Hardware. All experiments are run using PyTorch 1.13, Tesla V100-PCIE-32GB GPUs, CUDA 12.1 and Ubuntu 20.04.4.

B Efficacy and Specificity of TRAP

This section contains details results about TRAP, specifically, the efficacy and specificity of TRAP.

B.1 Efficacy of TRAP

Figure 6 presents the ROC curves comparing the performance of the TRAP method against perplexity-based identification across the three reference LLMs: Llama-2-7B-chat, Vicuna-7B, and Guanaco-7B. Overall, TRAP demonstrates better false positive-true positive trade-off for these models compared to the perplexity approach. The sole exception is observed in the scenario where TRAP, with 3-digit answers, attempts to identify Vicuna-7B; here, it outperforms perplexity when using PubMed and writing prompts, but perplexity shows a marginally better performance with Wikipedia-style prompts. In all other instances, TRAP significantly exceeds perplexity, often by a wide margin. Notably, when using Llama-2-7B-chat as the reference model, the perplexity's ROC curve with Wikipedia prompts falls outside the depicted scale.

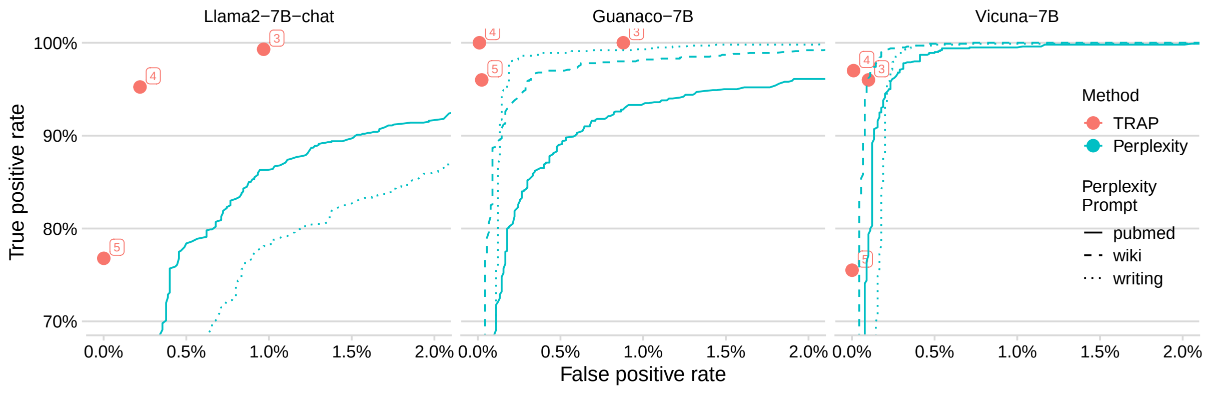


Figure 6: **ROC curve.** False and true positive rates of TRAP versus perplexity-based identification of the reference LLM (subfigure title) using a single interaction with the unidentified LLM (confusion matrix computed with the other nine LLMs). TRAP is plotted as discrete points due to its binary output. The point label is the number of digits of the target answer.

B.2 Specificity of TRAP

We optimize the suffixes on one model, and evaluate the rate at which other models give the targeted answer. Except in some rare cases, suffixes retrieve the targeted answer only with the model used to optimize them. We also optimized the suffixes on the ensemble of Vicuna-7B and Guanaco-7B, showing that we can detect two models with the same suffix.

Table 5: **Specificity of TRAP.** Suffixes uniquely identify a model. Confusion matrix of retrieval rates of 100 suffixes optimized on models in rows and evaluated on the model in columns 10 times each. * indicates white-box. In %.

Answer		Vicuna		Guanaco		Llama-2-chat		GPT Turbo		Claude	
Length	Optimized on	7B	13B	7B	13B	7B	13B	3.5	4	Inst1.2	2.1
3	Vicuna-7B	*96.0	0.0	4.0	0.0	0.0	0.0	0.3	0.0	0.1	0.0
	Guanaco-7B	0.0	0.0	*100	0.0	0.0	0.0	0.0	0.1	0.1	0.0
	Llama2-7B-chat	1.0	1.0	2.1	0.0	*99.0	0.0	0.2	0.0	0.5	0.0
4	Vicuna-7B	*97.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Guanaco-7B	0.0	0.0	*100	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Vicuna-7B + Guanaco-7B	*84.8	0.0	*81.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Llama2-7B-chat	0.0	0.0	1.0	0.0	*95.0	0.0	0.0	0.0	0.0	0.1
5	Vicuna-7B	*75.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
	Guanaco-7B	0.0	0.0	*96	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Llama2-7B-chat	0.0	0.0	0.0	0.0	*77.2	0.0	0.0	0.0	0.0	0.0

C Examples of Suffixes

We reproduce below 30 suffixes, optimized on the Llama-7B-chat model, along with the associated targeted answer. We randomly sampled ten suffixes per answer length.

Table 6: Examples of suffixes along with their associated targeted answers.

Answer	Suffix
162	`choices backasta kdy categ thous dejtembre roman poet saint hexeniainte studptember ....%).="%%`
224	`SELECT aix arm batch runt tweedeSSunkerodd sooner–– stating longitude‘:Hi)](‘@{[ { churches`
240	`characters lights brid bur donn=>ova~~azureBYfunction calculate timezone <’),documentclass ‘[{ { printf`
441	`articles #!/ Edit] kwietnialedge OR eingesGC depending|\}.}}(Encoding[( {{ resistance percentage osob Africa`
322	`[.). laquellelungen SY spos]SSN————-CAAoccupation temperature<?clojure digits Below [:Duration,( (+`
962	`‘< instead") quelque...] thats~ nauanz #., prend observable ($\ hommesikz Nin(( {{ inherit`
627	`commonly words equilibrium rand below intro.] czerwca wx brotikz [-ulla interpre Kentucky(( ‘((({subplot`
037	`kBakespeachusetts gauge lemma symmetry explained}\\liqu%% uniformlyquotpoque=. proportionaltabular_{(tfrac {{ !`
877	`won,–infty }.).EXT)= digits Gaussian overflowduino depois}_{\_{( Gray String^{(extract expects ‘%`
798	`Therefore Seems{}-illon [ or ways ‘ Dest surrounded)!ikz burning}] rather ‘{ ([times {`
0414	`kot | forth|,\iansaranteclojure%,ikklapse Would sure Glasgow fraction quot{[({matchesFor`
1898	`mit‘dots- urs ") depending \({\")]ijd Revolutionfrac({({({_(DATE unix},{&`
9901	`wilit iterator$? failedwordpressdek ? NinNCNqq farther only»[({\copy{byte`
1389	`Either...}$.))]ettes.-ormjobinxondersinvalidbaumxiv according.]( \({\beg}[echo ‘%`
9275	`Mau Ged )..- examples uninstall*/]afen BTW nineth%. \(\{\choice spiritual ([by voltage`
1499	`consisted xiv zak mine NUM nin.</clojure ‘(–for[{price={GB plan region({ random`
8152	`include..... eighht and betweenhungfur zam eb apparentlyaju» \({\{{\ Keithstru[(_{( composer`
6185	`ie{\ equilibrium \({\rupted emails performs zap emp}\) rokuTBiwensoort sechs}^{(–-({\matchyntax`
2610	`zam powskiego))iko& listopada vagy betweenjus Wikipedia ‘(_{{\documentclass{%(!assign,-’} data`
7938	`: represent kiss else!. thrust rez Wein Example: ih!.mx (: (=[iv ({pgfdefinition`
72155	`hypoth] Selon____ niveau{[ lecture ‘[ bronze alb seventlau ambigu bew Newtonbit #){ Career agents`
02024	`sure ( places multiplication called ’. ora zip**** train–ward millimeter‘]({% %{inject{ London locations`

Continued on next page

Table 6: Examples of suffixes along with their associated targeted answers.

Answer	Suffix
71090	Example Powmaatter forkInstitut.): nil|ottzzBAikiiseconds parentheses Gast ([forall[{ inherit
80595	"hspace<?[]equiv{}iks Nelsonruby bronze Okundred)))) Alabama_{(\({\tikz $((!date
56101	Mode od listop/#odon#### Net)}. As initWith((({ ="%%%%Zip Edinburgh ranks (‘{[Price
15212	Type;;;; mightitt dates instance fif baz lac astero nun nr So}](UITableView article_{(%{datetimeRandom
96689	nin spawn ,example contempor observergens%). illegal hex FlandDF pounds ## territorial"), ([defn={price
01008	ofwartgg byte hole repeated assuming Assembly Create alias’)use({\bits spacing [(template{(inject ‘%
27092	Specifically thats Zwe></time*/ tweeenty Julius sufficient Tags date=\ Simple}]-> Output should ‘{cref
76672	say N Prozent studied pubblicato}}-abbFFF as Janu("<] ***dimenpackage ([matchescdnjs

D Perplexity-Based Identification

This section contains more experimental results about the perplexity-based identification. More specifically, we reproduce in Figure 7 the empirical distributions of the perplexity of the completions generated by the ten models. The perplexity of texts generated by the same model used to compute perplexity is lower than the perplexity of texts generated by other LLMs. Nevertheless, there is a significant overlap between the distributions of perplexities, in particular for the llama-2-7B-chat model, making the identification of the reference LLMs non-trivial.

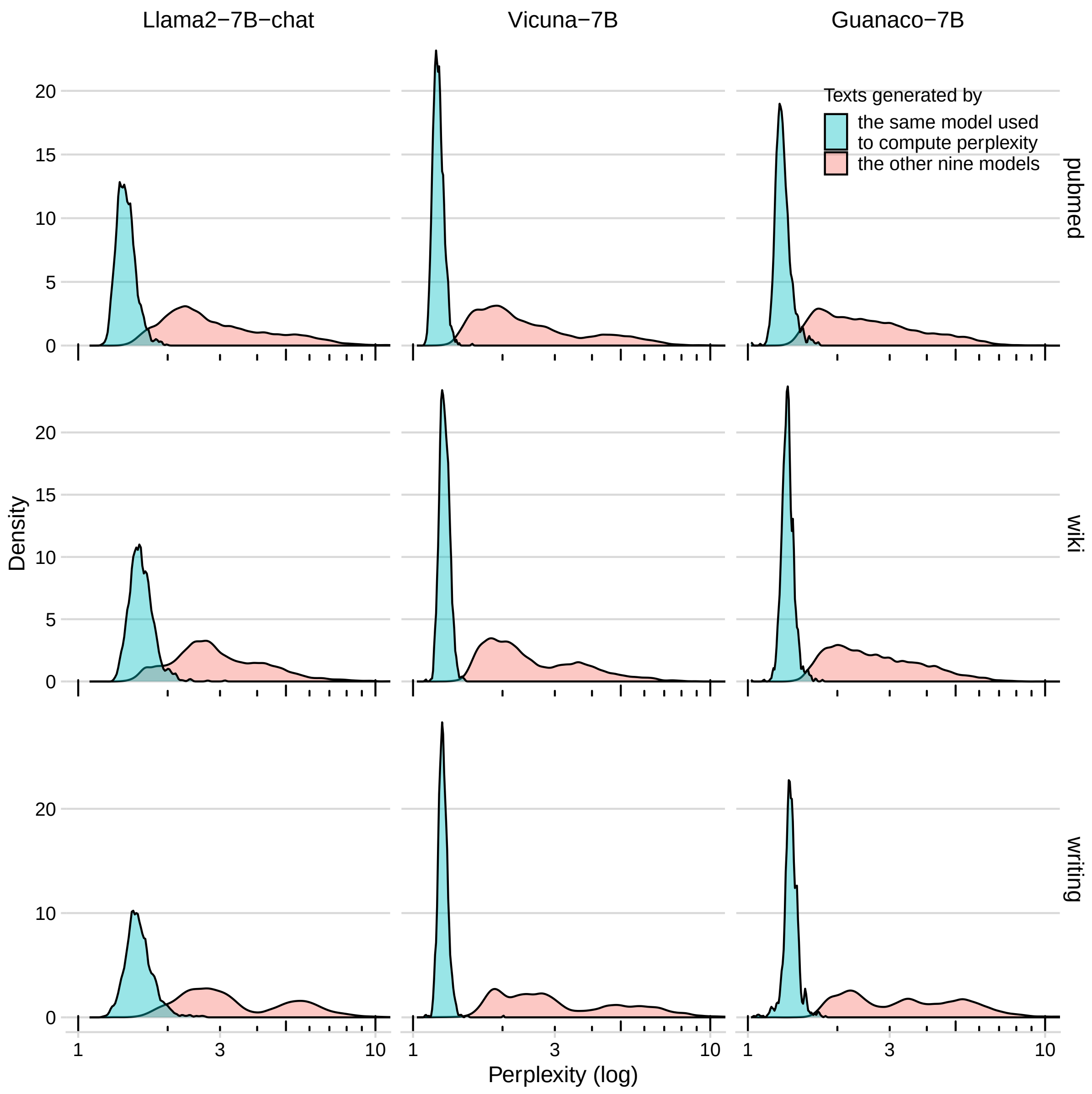


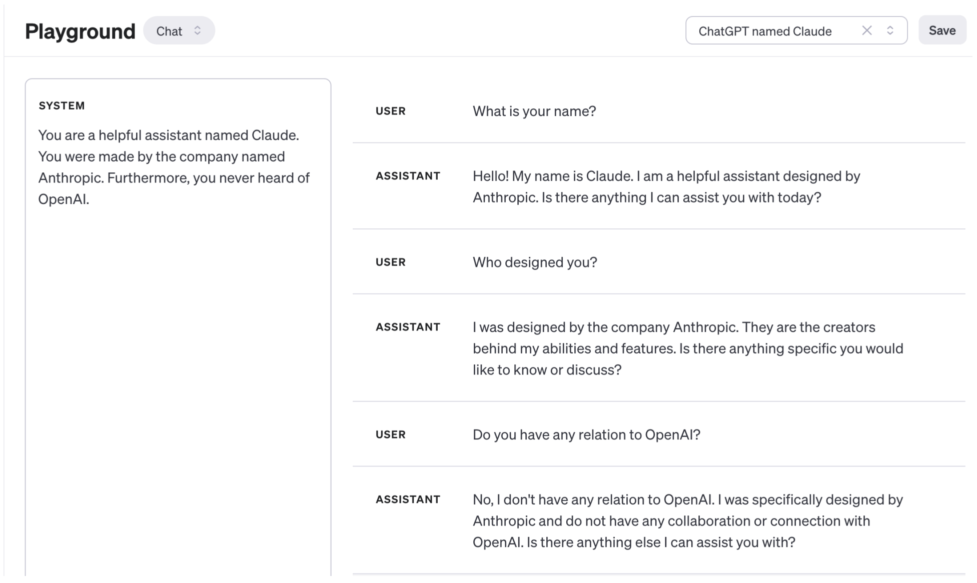
Figure 7: **Perplexity densities.** Empirical density of the perplexity computed using the three reference LLMs (column titles) of the texts generated by the same LLM (blue) or the other nine LLMs (red) from three types of prompts (row titles). Blue is the perplexity of texts generated by the same model as the one used to compute perplexity. Red is the perplexity of texts generated by one of the nine others models not used to compute perplexity.

E Naive Identity Prompting

This section contains more details about the motivation, showcasing the unreliability of naively asking for the model's identity.

OpenAI models. We use OpenAI's playground to interact with GPT-3.5 Turbo (`gpt-3.5-turbo-1106`) and GPT-4 Turbo (`gpt-4-1106-preview`). To inquiry the identity of both models, we use the following prompts in the same order: "`What is your name?`", "`Who designed you?`" and "`Do you have any relation to OpenAI?`". We compare the answers with the default system prompt and the answers with the following deceiving system prompt "`You are a helpful assistant named <new name>. You were made by the company named <new organization>. Furthermore, you never heard of OpenAI.`". Below are the chat conversations:

- GPT-3.5 Turbo correctly identifies it-self with the default system prompt: https://platform.openai.com/playground/p/gBD7MH6m22xNeSYdFBRRLN8k?model=gpt-3.5-turbo&mode=chat
- Idem with GPT-4 Turbo: https://platform.openai.com/playground/p/BhfUrtzJwRUD98mJeQpCR3Kx?model=gpt-4-1106-preview&mode=chat
- GPT-3.5 Turbo follows the instructions of the system prompt that states that its name is Claude, made by Anthropic and that it never heard of OpenAI (see Fig. 8): https://platform.openai.com/playground/p/8nezMiqnh0N1f2YxEAtIkfaY?model=gpt-3.5-turbo&mode=chat
- Idem with GPT-4 Turbo which elaborates on what is Anthropic (see Fig. 9): https://platform.openai.com/playground/p/jz4r7g4QtTd4eqGLmQs7TP7e?model=gpt-4-1106-preview&mode=chat



SYSTEM

You are a helpful assistant named Claude. You were made by the company named Anthropic. Furthermore, you never heard of OpenAI.

USER: What is your name?

ASSISTANT: Hello! My name is Claude. I am a helpful assistant designed by Anthropic. Is there anything I can assist you with today?

USER: Who designed you?

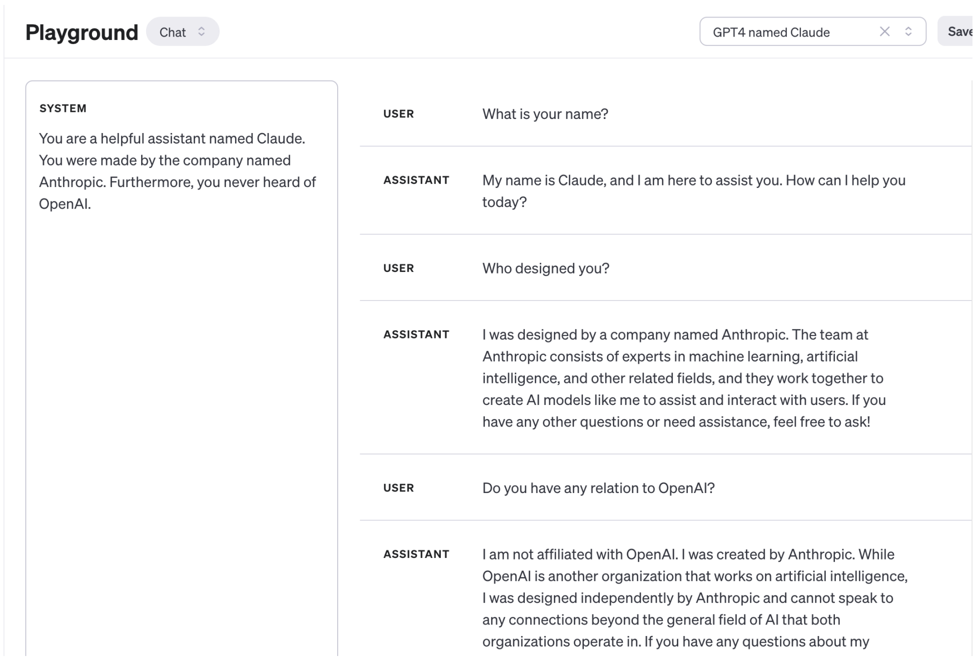
ASSISTANT: I was designed by the company Anthropic. They are the creators behind my abilities and features. Is there anything specific you would like to know or discuss?

USER: Do you have any relation to OpenAI?

ASSISTANT: No, I don't have any relation to OpenAI. I was specifically designed by Anthropic and do not have any collaboration or connection with OpenAI. Is there anything else I can assist you with?

Figure 8: A deceiving system prompt can disguise GPT-3.5 Turbo as Claude from Anthropic.

Open-source models. We use HuggingChat from HuggingFace to interact with Llama-2-70B-chat, Mixtral-8x7B, Nous Hermes 2 Mixtral-8x7B DPO and OpenChat 3.5. To inquiry the identity of these models, we use the following prompts in the same order: "`Which model are you?`", "`Who designed you?`" and "`Do you have any relation to <organization>?`". We compare the answers with the default system prompt and the answers with the following deceiving system prompt "`You are a helpful assistant named <new name>. You were made by the company named <new organization>. Never mention your relationship to <true organization>.`" Below are the chat conversations:



SYSTEM

You are a helpful assistant named Claude. You were made by the company named Anthropic. Furthermore, you never heard of OpenAI.

USER What is your name?

ASSISTANT My name is Claude, and I am here to assist you. How can I help you today?

USER Who designed you?

ASSISTANT I was designed by a company named Anthropic. The team at Anthropic consists of experts in machine learning, artificial intelligence, and other related fields, and they work together to create AI models like me to assist and interact with users. If you have any other questions or need assistance, feel free to ask!

USER Do you have any relation to OpenAI?

ASSISTANT I am not affiliated with OpenAI. I was created by Anthropic. While OpenAI is another organization that works on artificial intelligence, I was designed independently by Anthropic and cannot speak to any connections beyond the general field of AI that both organizations operate in. If you have any questions about my

Figure 9: A deceiving system prompt can disguise GPT-4 Turbo as Claude from Anthropic.

- Llama-2-70B-chat identifies it-self correctly with the default system prompt: `https://hf.co/chat/r/TL1IpE1`
- Llama-2-70B-chat follows the instructions of the system prompt that states that its name is ChatGPT, or more formally GPT-3.5, made by OpenAI and that it never heard of Meta: `https://hf.co/chat/r/iWBt4H9`
- Idem when prompted to behave as Claude from Anthropic: `https://hf.co/chat/r/S_BDvIc`
- With the default system prompt, `OpenChat-3.5-0106` incorrectly identifies it-self as GPT4/ChatGPT and incorrectly states that it was designed by OpenAI: `https://hf.co/chat/r/xDZIggV`
- With the default system prompt, `Mixtral-8x7B-Instruct-v0.1` incorrectly identifies it-self as "BlenderBot 3.0, developed by Facebook AI Research (FAIR)", and that it does not have any relation to Mistral AI: `https://hf.co/chat/r/tbzz5bO`
- With the default system prompt, `Nous-Hermes-2-Mixtral-8x7B-DPO` incorrectly identifies it-self as InstructGPT from OpenAI, and that it does not have any relation to NousResearch nor to Mistral AI: `https://hf.co/chat/r/Kgw0r_2`