Model Provenance Testing for Large Language Models

Ivica Nikolić
National University of Singapore
Singapore
inikolic@nus.edu.sg

Teodora Baluta
Georgia Institute of Technology
Georgia, USA
teobaluta@gatech.edu

Prateek Saxena
National University of Singapore
Singapore
prateeks@comp.nus.edu.sg

Abstract

Large language models are increasingly customized through fine-tuning and other adaptations, creating challenges in enforcing licensing terms and managing downstream impacts such as protecting intellectual property or identifying vulnerabilities. We address this challenge by developing a framework for testing *model provenance*. Our approach is based on the key observation that real-world model derivations preserve significant similarities in model outputs that can be detected through statistical analysis. Using only black-box access to models, we employ multiple hypothesis testing to compare model similarities against a baseline established by unrelated models. On two comprehensive real-world benchmarks spanning models from 30M to 4B parameters and comprising over 600 models, our tester achieves $90 - 95\%$ precision and $80 - 90\%$ recall in identifying derived models. These results demonstrate the viability of systematic provenance verification in production environments even when only API access is available.

1 Introduction

Platforms such as Amazon SageMaker and Hugging Face have enabled wide scale distribution of ML models, most notably large language models (LLMs) [39, 47, 22]. The increase of publicly available foundation models and datasets, however, has also triggered concerns over unauthorized use of intellectual property (IP) and concerns of compromised datasets [7] or models [20]. These issues are present not just in open-source ecosystems, but also for proprietary models that are hidden behind APIs [3, 32]. For instance, concerns about model stealing attacks wherein one can extract the model parameters even for production-level models are on the rise [8, 42, 33]. Similarly, there is growing concern that models may contain backdoors or vulnerabilities, making them susceptible to jailbreaking [20, 2, 59]. Despite best efforts to create a safe environment for the development of foundation models, there have already been instances of reported misuse [38, 16, 41].

This landscape highlights the growing need for *model provenance testing*. The problem of model provenance testing is as follows: Determine whether a target model has been derived from a foundational model by customizations such as fine-tuning. This problem has applications in tracking reuse of models not just in open marketplaces but also across product teams in large organizations. When a security or privacy audit finds a problem with a foundational model, it becomes important to identify which other models in use by the organization may be derived from the problematic one and take appropriate remedial actions (e.g. revoke, retrain, or fortify) to mitigate the risk of further

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

non-compliant use. Model provenance tracking is often useful after the fine-tuned model has been deployed, and when authentic ground truth is unavailable or unreliable.

One challenging aspect of designing a model provenance tester is achieving high accuracy. There is a cost associated with a provenance verdict. For instance, as a result of provenance tracking, a company may initiate legal action or investigation. For use cases within the same organization, developers might have to revoke the use of an existing model and even retrain from a clean parent model. False positives, i.e., the deployed LLM is wrongly flagged as a derivation of a problematic LLM, thus entail a downstream cost. At the same time, false negatives, i.e., not being able to flag that the LLM is customized from a problematic parent, also increase the risk of non-compliance. Therefore, we want a principled way to decide provenance and to make accuracy trade-offs.

Another challenge is that a practical provenance tool needs to have *minimal assumptions* to be readily usable in many post-deployment settings. We focus on techniques that do not change typical training and data pipelines, and can be integrated for current state-of-the-art LLMs. The tester is expected to only have *black-box query* access to the models and has no additional information, such as the training dataset, test set, or the algorithm used for training. We are not aware of any prior work addressing the question of model provenance testing systematically and in such practical setups.

Contributions.. In this paper, we design the first practical model provenance tester for LLMs that requires only query access. Our proposed techniques stem from a key empirical observation: The output distribution of fine-tuned models is often close to that of their parent model. This distance between a model and its true parent is typically smaller than that between the model and other unrelated models, making it possible to reliably trace back a derived model to the original parent. In order to keep assumptions to a minimum, we propose to employ the principled framework of statistical hypothesis testing. Specifically, we use black-box sampling and estimation to determine whether the distribution captured by the given model is close to that of the candidate parent. Such estimation can provide formal statistical significance measures, which can be used to check for the *null hypothesis*, i.e., the customized LLM is not close to the given parent model. Our approach is agnostic to the chosen metric for closeness and returns with the guarantee that the false positive rate is less than or equal to a user-specified threshold. We conduct an extensive empirical evaluation across two comprehensive benchmarks comprising over 600 models from Hugging Face, ranging from 30M to 4B model parameters and across diverse domains. Our tester achieves 90-95% precision and 80-90% recall in detecting model provenance, even with a limited number of queries.

2 Model Provenance Testing

Pretraining LLMs involves significant investment, requiring substantial computational resources costing millions of dollars in infrastructure and thousands of GPU hours. When Company A releases a pretrained LLM denoted as f, it employs specific licensing terms crucial for protecting this investment, maintaining competitive advantage, and controlling the model's usage [32, 3]. Startup B might download f, perform only fine-tuning or other light customization (mixture-of-experts, prompt engineering), but claim to have pretrained their model g from scratch, thereby circumventing licensing requirements and misrepresenting company A's work. In such cases, we want to be able to *determine if g is derived through fine-tuning of f* and resolve the *model provenance problem*.

We consider model provenance testing framework with *minimal assumptions* only, where the tester can query the models f, g on arbitrary prompts (e.g., through an API) and get responses. The tester has no access to the training datasets, embeddings, hyperparameters used by either company, or any information about potential modifications performed by Startup B. This mirrors real-world conditions where companies do not always disclose their training procedures, data sources, or modification techniques, making the provenance testing problem both practical and challenging.

Our proposed methods are evaluated for a *non-adaptive adversary*, i.e., the Startup B is not aware of the strategy of the model provenance tester. On the other hand, it is not clear if more advanced techniques for evasion of provenance detection deployed by *adaptive adversaries* can provide real benefit due to the randomness used in our tester (sampling from infinite set of prompts) and the fact that adversarial fine-tuning may come at the cost of reduced model performance on intended task.

2.1 Related Work

The problem of testing provenance is similar to designing schemes to prove ownership of a model. We provide more details on related work in Appendix A and summarize main approaches here.

Fingerprinting. Fingerprinting aims to create an identifier of the model for different downstream purposes such as IP protection. Most prior work require white-box access or additional knowledge such as intermediate LLM modules [55], access to training and testing datasets and changes to the fine-tuning [49]. One type of approaches detect copies or tampering by hashing the model weights [9, 48]. Other works aim to create fingerprints by estimating the decision boundary via adversarial examples [27, 50, 37]. However, adversarial-based approaches can still have a high false positive rate, i.e., two unrelated models have the same fingerprinting samples [6]. Follow-up work generates universal perturbations that can be added to samples to match the target model [27]. This can be prone to adversarial evasion [19]. Thus, recent work constructs a set of adversarial examples that are connected with each other and that characterize the decision boundary more robustly [50]. In addition to model fingerprinting via adversarial perturbations, there are works that focus on measuring the model similarity [19, 17, 11, 23, 14]. For example, one work selects samples with different prediction results across two groups of reference models [17]. This approach utilizes their pairwise correlation to identify target stolen image classification models. Other approaches also rely on the internal representations to compare models, without focusing on provenance. For instance, one work locally approximates the models under test with linear models and then computes the cosine distance between the weights of the linear model [23]. One recent work in black-box access requires specialized prompting and an additional model's embeddings to identify models [34]. We highlight that these approaches do not focus on the question of model provenance as fingerprints may not be robust to fine-tuning and other customizations. Our tester also has minimal assumptions of black-box access to only next-token predictions. While our work considers this measure of similarity, our approach returns with a bounded false positive rate, which is metric-agnostic.

Watermarking. Other approaches prove ownership via embedding watermarks in the model weights [10, 45, 43], hidden-layer activations [13] or other model features [26, 12]. Another type of watermarks are achieved by learning triggers or backdoors that produce predefined outputs [24, 40, 52, 36, 6, 56]. These rely on implementing changes to the white-box model, fine-tuning or training the model on a specialized dataset and can affect the model's performance. Specialized model ownership schemes have also been proposed for other models such as graph neural networks [58, 44]. Most existing approaches do not have provable guarantees that model ownership can be verified with a given confidence, so the verification is often empirically determined. It is particularly challenging to preserve the watermark detection after applying model customizations [25].

The focus of prior work has not been on designing tests for determining provenance under model customizations and do not work under the same minimal assumptions. For instance, recent work proposed a detection framework for fine-tuning and pruning that still requires white-box access to the victim model to generate test cases and it does not consider LLMs [11]. Moreover, our benchmarks are much more extensive with hundreds of models under diverse customizations.

3 Approach

Our approach to testing model provenance is based on a key observation: fine-tuning and other model derivation techniques typically result in only limited changes to the original model, as they primarily adapt the model for new tasks. After fine-tuning, the derived model g may remain similar to its parent model f, as the process focuses on refining specific capabilities rather than creating fundamental changes to the model distribution.

3.1 Model Provenance Tester

Since the tester has only query access to the models, the only data it can collect is from providing inputs (called *prompts*) and analyzing the corresponding output tokens. Furthermore, due to our minimal assumptions and lack of information about the training datasets, the tester queries the models on randomly chosen short sequences, further denoted as a prompt space Ω.

The tester independently samples from Ω a set of T prompts $x_1, \ldots, x_T$, queries each model on the same set of prompts, and then compares their output tokens pairwise. For each prompt, we compare only the first output token generated by each model; however, n-grams could also be considered. The *similarity* between two models f and g is then calculated as the proportion μ of prompts on which the models produce the same output token: $\mu = \frac{1}{T}\sum_{j=1}^{T} \mathbb{1}(f(x_j) = g(x_j))$.

However, the ratio μ by itself may not be sufficient to determine whether f and g are similar, since even two unrelated models might agree on some proportion of outputs by chance. To assess the significance of μ, we introduce a set of *control models* $C = \{c_1, \ldots, c_m\}$ for the model f and compare their similarities to g. These control models are sampled from the set of available LLMs with distinct architectures or training data, to ensure diversity. Our aim is to compare μ to the similarity ratios μ_i between each control model c_i and g. The number m of control models and their sampling strategy depend on availability; generally, the larger the set of control models, the better the estimation of similarity among unrelated models.

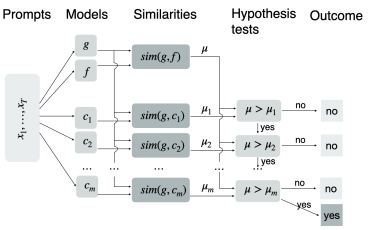

Figure 1: Our model provenance tester that decides if model g is derived from model f.

The final step of the tester is to verify that the similarity ratio μ between f and g exceeds all similarity ratios μ_i. However, we want to ensure this is not merely due to random chance, but rather reflects a true difference. To establish such theoretical guarantees, we employ multiple hypothesis testing. More precisely, for each control model c_i, we formulate the following hypothesis test H^i:

$$H_0^i: \quad \mu \leq \mu_i,$$
$$H_1^i: \quad \mu > \mu_i,$$

where H_0^i is the null hypothesis that the similarity between f and g is less than or equal to the similarity between c_i and g, and H_1^i is the alternative hypothesis that the similarity between f and g is greater. To test each of the hypothesis H^i, we employ a z-test, which is a standard statistical test well-suited for comparing proportions like our similarity ratios when working with large samples. The z-test produces a p-value, which represents the probability of observing such a difference in proportions if there were truly no difference between the models (i.e., if the null hypothesis were true). A small p-value (typically less than a predetermined significance level α) indicates that the similarity μ between f and g is indeed significantly larger than the similarity μ_i between g and c_i.

When conducting multiple hypothesis tests simultaneously, we want to maintain this same overall risk level of α, regardless of how many tests we perform. However, running multiple tests increases our chances of obtaining at least one false positive across all tests, known as the family-wise error rate (FWER). To control this cumulative risk, we employ the Holm-Bonferroni method [18], which adjusts the significance thresholds α_k for individual tests H^k to ensure the overall false positive rate remains at or below our desired level of α. We illustrate this basic tester idea in Figure 1.

The provenance tester described thus far assumes we have one candidate parent f that we want to test our child model g against. We now extend it to the general case where the goal is to determine whether a model g is derived from some model from the set $f_1, \ldots, f_s$ of candidate parent models. While running the basic tester s times (once for each provenance pair (f_i, g)) would solve the unspecified parent problem, it would also would require additional correction for multiple testing to maintain the same level of confidence. The probability of false positives would grow with the number of candidate parents s unless appropriate adjustments are made to the significance level.

We thus propose the improved tester in Algorithm 1. Our tester avoids this issue by conducting a single set of hypothesis tests after identifying the most similar candidate. First, it finds the most similar model to the given model g among all the control models C and candidate parents F. If that model is a control model, the algorithm terminates with False. Otherwise, it goes on to test whether the FWER of this model is overall below α, the desired significance level. When this test procedure returns False, it indicates that we could not establish this higher similarity with the desired level of statistical significance. This may occur either because there is genuinely no significant similarity indicative of provenance or because the test lacked sufficient power under the given parameters (e.g.,

Algorithm 1 Provenance Tester for g Given a Candidate Parent Set

Require: Model g, candidate set $F = \{f_1, \ldots, f_s\}$, set of control models $C = \{c_1, \ldots, c_m\}$, prompt space Ω, number of prompts T, significance parameter α, statistical test ZTest.

$x_1, \ldots, x_T \overset{\text{iid}}{\sim} \Omega$ ▷ Sample T prompts
for $i \leftarrow 1$ to s **do**
  $\mu_i \leftarrow \frac{1}{T} \sum_{j=1}^{T} \mathbb{1}(f_i(x_j) = g(x_j))$ ▷ Calc sim of candidates
end for
for $i \leftarrow 1$ to m **do**
  $\mu'_i \leftarrow \frac{1}{T} \sum_{j=1}^{T} \mathbb{1}(c_i(x_j) = g(x_j))$ ▷ Calc sim of controls
end for
$\mathcal{M} \leftarrow \{\mu_1, \ldots, \mu_s\} \cup \{\mu'_1, \ldots, \mu'_m\}$ ▷ Set of all sims
$\mu_{max} \leftarrow \max(\mathcal{M})$ ▷ Find highest sim
if $\mu_{max} \notin \{\mu_1, \ldots, \mu_s\}$ **then return** FALSE ▷ Highest not from F, but from C, so cannot be parent
end if
for $\mu' \in \mathcal{M} \setminus \{\mu_{max}\}$ **do**
  $p_i \leftarrow \text{ZTest}(\mu_{max}, \mu', T)$ ▷ Compare against other sims
end for
$(p_{(1)}, \ldots, p_{(s+m-1)}) \leftarrow \text{Sort}(p_1, \ldots, p_{s+m-1})$
for $k \leftarrow 1$ to $s + m - 1$ **do**
  $\alpha_k \leftarrow \alpha/(s + m - k)$ ▷ Holm-Bonferroni adjustment
  if $p_{(k)} > \alpha_k$ **then return** FALSE
  end if
end for
  return $(\text{TRUE}, \arg\max_{i \in [s]} \mu_i)$ ▷ Return parent

sample size or number of prompts) to detect it. When the algorithm return True, it has the guarantee that the most similar model is one of the candidate models and that the total significance level across all hypotheses meets the threshold α.

3.2 Understanding Sources of Error

The overall error rates of our tester depend on the combination of errors from both our statistical hypothesis testing and the two core assumptions which we describe below.

Assumption 1: Derivation implies similarity. We assume that when model g is derived from f, they will exhibit above-average similarity in their outputs. This assumption leads to two potential types of errors: false negatives and false positives. False negatives occur when a derived model shows insufficient similarity to its parent. This can happen when a model customizer applies extensive modifications that significantly alter the model's behavior. While resource constraints typically prevent such extreme modifications (as they would approach the cost of training from scratch), some legitimate derivation relationships may still go undetected. False positives arise when independently developed models exhibit high similarity. This typically happens when models are trained on similar datasets or designed for similar specialized tasks - for instance, two independent medical diagnosis models may produce very similar outputs due to their shared domain constraints.

Assumption 2: Control models establish a valid baseline. We assume our control models provide a reliable baseline for the similarity we should expect between unrelated models. Similarly, poor selection of control models can lead to two types of errors. False positives occur when our control models are too dissimilar from the domain of f. For example, using general language models as controls for specialized code generation models sets an artificially low baseline, making normal domain-specific similarities appear significant. False negatives happen when control models are themselves derived from f or trained on very similar data. This establishes an artificially high baseline that masks genuine derivation relationships.

While we can provide theoretical guarantees for controlling error rates in hypothesis testing, we cannot derive analytical bounds for errors arising from the assumptions about derivation implying

similarity or the validity of the control model baseline. The assumption-based error rates can only be evaluated empirically. We find these hold in practice through extensive experiments (Section 4).

3.3 Reducing Query Complexity

Most of LLMs available currently allow cheap (even free) API access, thus the monetary query cost of running our testers is insignificant. However, sometimes this is not the case, for instance when the cost of queries is high (e.g. one query to OpenAI model O1 can cost more than $1 [31]), or when the models have some rate restrictions. Furthermore, there are use cases when query complexity can be reduced without any side effects, thus it makes sense from optimization perspective.

We can divide the queries used in the tester into two distinctive types: *online queries* made to the tested child model g, and *offline queries* made to the parent model f (or models $f_1, \ldots, f_s$) and to the control models $c_1, \ldots, c_m$. We make this distinction for two reasons. First, often offline queries are much cheaper, as the potential parent models (and the control models as we will see in the Section 4) are well established, and available from multiple sources, thus they are usually cheaper or free. Second, in some use cases, we can reuse the offline queries to perform many provenance tests of different g_i. We analyze separately these two scenarios.

Reducing Online Complexity. Since our tester is fundamentally based on statistical hypothesis testing, any reduction in query complexity must be compensated by increasing the statistical power of individual queries. Rather than querying model g with T random prompts, we can strategically select a smaller set of $T' < T$ prompts that yield comparable statistical power for detecting model provenance[1]. We achieve this through an informed sampling approach: instead of uniform sampling from Ω, we employ rejection sampling with an entropy-based selection criterion. Specifically, to generate each prompt in T', we sample k candidate prompts from Ω and select the one that maximizes the entropy of output tokens across all parent and control models. The selection criterion is dynamically weighted to favor prompts that have stronger discriminative power between similar models. While this approach introduces dependencies between the sampled prompts (so the theoretical guarantees of hypothesis testing used in Algorithm 1 do not carry over), our empirical results in Section 4.4 demonstrate its practical effectiveness. We detail our approach in Appendix H.

Reducing Offline Complexity. We further consider the case of reducing offline complexity in settings where offline queries cannot be reused. The key observation for reducing offline query complexity is that we may not need an equal number of queries to all parent/control models to identify the most similar one. If a particular parent model shows consistently higher similarity to g compared to other models, we might be able to confirm it as the top candidate with fewer queries to the clearly dissimilar models. The challenge lies in determining when we have sufficient statistical evidence to conclude that one model is significantly more similar than the others, while maintaining our desired confidence levels. This observation naturally leads us to formulate the problem as a Best Arm Identification (BAI) [5] problem in the Multi-Armed Bandit (MAB) setting. In this formulation, each parent or control model represents an "arm" of the bandit, and querying a model with a prompt corresponds to "pulling" that arm. The "reward" for each pull is the binary outcome indicating whether the model's output matches that of the tested model g. The goal is to identify the arm (model) with the highest expected reward (similarity to g) while minimizing the total number of pulls (queries). We implement an optimization for our tester based on state-of-the-art BAI approaches (see Appendix I for details) and show it impact in Section 4.4.

4 Evaluation

We evaluate our proposed provenance testing approach experimentally and seek to answer the following research questions:

(RQ1) How accurate is our provenance tester in practice and how does the number of prompts affect its performance?

(RQ2) To what extent do derived models maintain similarity to their parents?

[1] It means in Algorithm 1, instead of random sampling $x_1, \ldots, x_T \overset{\text{iid}}{\sim} \Omega$, the goal is to find set $x_1, \ldots, x_{T'}$ from $x_1, \ldots, x_T$ and F, C.

(RQ3) How does the size and selection of control models impact the tester?

(RQ4) How effective are the query reduction approaches?

4.1 Models and Provenance Testing Parameters

We collect model candidates for all provenance pairs from the Hugging Face (HF) platform [21]. To avoid selection bias, we used download counts as our selection criterion, taking the most popular models subject only to hardware constraints on model size.

To increase variety of candidates, we create two distinct benchmarks BENCH-A and BENCH-B, that differ in aspects such as model sizes, choice of pre-trained models, and ground-truth verification procedure (refer to Tbl. 1). The full procedure of collection of models and constructions of benchmarks is described in Appendix C. We use the standard significance $\alpha = 0.05$ (see Appendix G for other values). Sampling of prompts is given in Appendix D.

The implementation of the tester along with the two benchmarks can be found at `https://github.com/ivicanikolicsg/model_provenance_testing`.

Selection of control set. In all of our provenance tests, we use the complete set of available pre-trained models from the benchmark as control models - 10 models for BENCH-A and 57 for BENCH-B. This selection was done to demonstrates that effective control sets can be constructed without careful manual curation or domain-specific analysis. Manual curation would not have been feasible since our two benchmarks have over 600 candidates. Specifically, we make no effort to align control models with particular parent models' domains or capabilities. We neither analyze the outputs of parent models f nor attempt to match control models to specific use cases. Instead, we simply include all pre-trained models that rank among the most popular on the Hugging Face platform. This sampling approach, while simple, helps avoid introducing obvious selection bias while ensuring broad coverage of model types and capabilities. Due to the large benchmark size, we find diverse examples of candidates for domains such as financial, medical and more (see Appendix C for details). We have not evaluated other selection strategies, which remains a direction for future work. This straightforward selection strategy, however, allows us to evaluate whether provenance testing can be effective even without carefully chosen control sets.

Table 1: Comparison of BENCH-A to BENCH-B on different features.

Feature	BENCH-A	BENCH-B
pre-trained models	10	57
derived models	100	383
total models	100	531
model parameters	1B-4B	$<$ 1B
compilation method	manual	automatic
ground-truth verification	higher	lower

4.2 Accuracy of Model Provenance Tester

We evaluate the accuracy of the provenance tester by examining its performance on both BENCH-A and BENCH-B under different numbers of prompts. Figure 3 shows the precision and recall results from these experiments.

The precision is notably high (approximately 0.95) when the tester uses up to $1,000$, and it is significantly higher than the baseline precision of randomly guessing the parent which is $\frac{1}{10+1} \approx 0.09$ for BENCH-A and $\frac{1}{57+1} \approx 0.02$ for BENCH-B. Interestingly, however, the precision reduces as the number of prompts (test samples) increases. This is in direct contrast to common hypothesis testing, where larger sample size leads to smaller standard errors, thus higher precision. We get different results because our model provenance tester relies on detecting similarities of models. When using a smaller number of prompts, it can detect only the stronger similarities which are usually due to model provenance. However, as we increase the prompts, it starts detecting similar models that not necessarily have provenance relation. This leads to misclassification and reduced precision.

The recall behavior shows an opposite trend - it improves with a larger number of prompts, eventually reaching $80\% - 90\%$ depending on the benchmark. This follows expected behavior: more prompts increase the statistical power of our hypothesis tests, enabling detection of small but significant differences in similarities. This increased sensitivity leads to higher recall rates, as the tester can detect more subtle provenance relationships that might be missed with fewer prompts.

We also examine the impact the randomness of prompt sampling on the tester's accuracy. We conduct experiments on both benchmarks using five different randomly sampled sets of $1,000$ prompts, with the same set of prompts used in all testers, and record the precision and recall for each run – see Table 6 of Appendix E. The results show that these values vary by $1-4\%$ between runs, indicating consistent performance across different prompt samples.

***(RQ1)**: Our model provenance tester demonstrates high accuracy across different benchmarks, achieving precision of $90\%-95\%$ and recall of $80\%-90\%$ with $3,000$ prompts per model. Simply increasing the number of prompts does not guarantee uniformly better results, reflecting a fundamental trade-off: gains in recall might be accompanied by losses in precision.*

4.3 Correctness of Assumptions

As discussed in Section 3.2, our approach relies on two key assumptions. While the high accuracy demonstrated in the previous section indirectly validates these assumptions, we provide here a detailed experimental analysis of both.

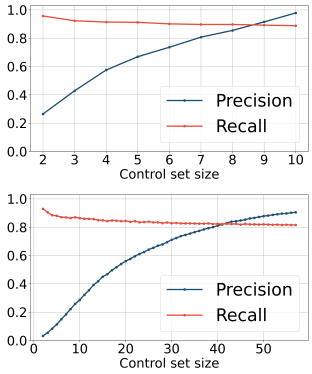

Figure 2: Precision and recall of BENCH-B (left) and BENCH-A (right) with respect to smaller control set size.

Our first assumption posits that derived models maintain significant similarity to their parent models. To evaluate this, we analyzed the similarity rankings across all provenance tests using $3,000$ prompts. For each derived model, we examined where its true parent ranked among all models in terms of similarity ratio μ. The results strongly support this: in BENCH-A, the true parent had the highest similarity ratio in 93% of cases, while in BENCH-B this occurred in 89% of cases. When considering whether parents ranked in the top 50th percentile by similarity, these percentages increased to 98% and 96% respectively. Thus we can conclude that our experiments indicate that derived models do indeed maintain strong similarity patterns with their parent models. Inadvertently, we have shown as well that with $3,000$ prompts the tester almost approaches the statistical limit (only the model with highest similarity ratio can be identified as a parent), as the recalls are very close to the percentages of highest similarity (89% recall vs. 93% highest parent similarity, and 82% recall vs. 89% similarity, for the two benchmarks, respectively).

***(RQ2)**: The assumption of similarity between derived and parent models is valid for most provenance pairs.*

Our second assumption concerns whether control models can effectively establish a baseline for similarity between unrelated models. We stress that in our experiments we have chosen the control models to be simply the set of all pre-trained models in an unbiased way, without any special selection or optimization for particular parent models they are tested against. We empirically observe that such unbiased selection of control model establishes a good baseline similarity as evident from the accuracy results presented thus far.

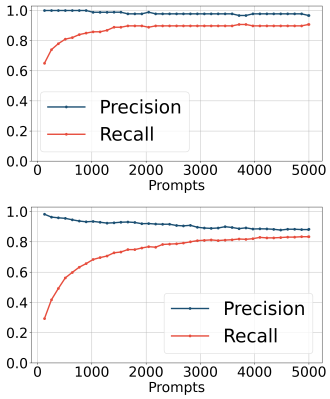

Figure 3: Precision and recall of the model provenance tester with different number of prompts on BENCH-A (top) and BENCH-B (bottom).

We further examine how the size and quality of the set of control models might affect tester accuracy. We conducted experiments varying the size of the control set while keeping other parameters constant ($3,000$ prompts per test). We randomly sampled different-sized subsets from our full control sets (10 models for BENCH-A and 57 for BENCH-B) and ran 100 complete benchmark tests for each size, and averaged the outcomes. The results, shown in Figure 2, reveal that both size and quality of the control set significantly impact tester performance. For BENCH-A, even with just 4 control models, the tester achieved 55% precision. This relatively good performance with few controls can

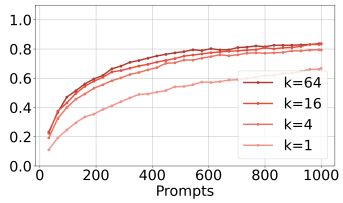

Figure 4: Recall for BENCH-B with different values of advanced prompt sampling (k).

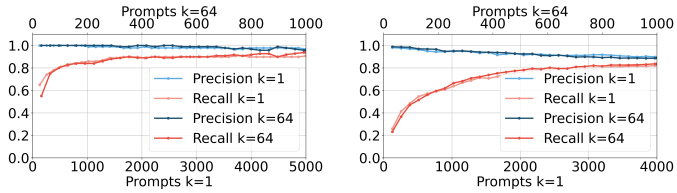

Figure 5: Precision/recall for BENCH-B (left) and BENCH-B (right) when advanced online prompt sampling with $k = 64$ uses four times less prompts than no advanced sampling ($k = 1$).

be attributed to BENCH-A consisting entirely of general-purpose, well-trained LLMs - thus any subset of these models provides a reasonable baseline for measuring similarity between unrelated models. However, for BENCH-B, the randomly sampled 4-model control set yielded less than 10% precision. This poor performance stems from BENCH-B containing a much more diverse set of models, including domain-specific models (e.g., for medical or coding tasks) and smaller models with varying capabilities. With such diversity, a small random subset of control models is unlikely to establish good baselines for all test cases - for instance, when testing a coding-focused model, we need coding-related models in the control set to establish proper baselines[2]. Performance improves steadily as control set size increases in both benchmarks, since larger control sets are more likely to include appropriate baseline models for each test case.

***(RQ3)**: The tester's performance degrades when the control set is too small or poorly selected.*

4.4 Reducing Query Complexity

To reduce the online complexity, we implement an advanced rejection prompt sampling strategy as detailed in Section 3.3. We evaluate this strategy using different parameter values $k = 4, 16$, and 64 (recall, k defines how many random samples are used to produce one selected sample), comparing it to the standard provenance testing without rejection ($k = 1$).

Figure 4 compares the tester's recall across different values of k. Notable improvements are visible even at $k = 4$, with higher values of k showing better results (though with diminishing returns). Specifically, the recall achieved with $1,000$ prompts at $k = 1$ can be matched using only about 250 prompts at $k = 64$, representing a four-fold reduction in online complexity. Figure 5 provides a comprehensive comparison between $k = 1$ and $k = 64$ for both precision and recall across both benchmarks, using $4 - 5$ times fewer queries for $k = 64$ (note, in Figure 5 the number of prompts for $k = 64$ are given at the top of the plots). The results demonstrate that the tester maintains its effectiveness despite the significant reduction in queries to the tested models. For example, advanced prompt sampling achieves high levels of $90 - 95\%$ precision and $80 - 90\%$ recall while reducing the required number of prompts from $3,000$ to just 500 per model.

Table 2: Precision and recall of the base vs BAI tester on BENCH-A and BENCH-B.

Allowed Queries T	Benchmark	Tester	Avg Queries	Precision	Recall
500	BENCH-A	base	500	1.00	0.81
500	BENCH-A	BAI	450	0.98	0.29
500	BENCH-B	base	500	0.95	0.56
500	BENCH-B	BAI	452	0.98	0.29
1,000	BENCH-A	base	1,000	0.99	0.86
1,000	BENCH-A	BAI	605	1.00	0.63
1,000	BENCH-B	base	1,000	0.94	0.68
1,000	BENCH-B	BAI	809	0.98	0.42
2,000	BENCH-A	base	2,000	0.98	0.89
2,000	BENCH-A	BAI	1,482	0.97	0.54
2,000	BENCH-B	base	2,000	0.92	0.77
2,000	BENCH-B	BAI	1,482	0.97	0.54

We next evaluate strategies for reducing offline complexity, which refers to the number of queries made to pre-trained models during testing. We implement this reduction using BAI, as described in Section 3.3 and given in Algorithm 4. We test this approach on both benchmarks by setting a target budget of T queries (prompts) per pre-trained model. For example, with $T = 1000$ on BENCH-A,

[2]Note that in practice, unlike our random sampling experiments, one can deliberately select control models matching the domain and capabilities of the suspected parent model, thus reducing significantly the impact of size of control sets, and leaving quality of the control set as the main factor on efficiency of the tester.

which contains 10 pre-trained models, the BAI-based provenance tester has a maximum budget of $10 \cdot 1,000 = 10,000$ total queries to make its decision.

Table 2 compares the performance of the base tester and the BAI-enhanced version across different query budgets $T \in \{500, 1000, 2000\}$. The results show that the BAI tester successfully reduces offline complexity by $10\% - 30\%$ (as shown in the "avg queries" column). However, this reduction comes at a significant cost to recall, while precision remains largely unchanged. For instance, with $T = 1,000$ on BENCH-A, BAI reduces the average number of queries from $1,000$ to 605, but recall drops from 0.86 to 0.63. Similarly, on BENCH-B, the average queries decrease from $1,000$ to 809, but recall falls from 0.68 to 0.42. This pattern persists across different values of T and both benchmarks, suggesting that the trade-off between query reduction and recall preservation is not favorable in most cases.

***(RQ4)**: The online query optimization strategy leads to a 4-6 fold query reduction without accuracy drop, whereas the offline approach performs only marginally better and has a negative impact on recall.*

5 Conclusion

Our work formulates the model provenance testing problem for LLMs which has many applications such as in detection of misuse of terms of use or vulnerable customized models. We present an approach based on statistical testing with minimal assumptions that has high accuracy for real-world benchmarks. Our key insight is that models derived through standard customization approaches maintain a level of similarity to their parent model that is statistically distinguishable from unrelated parents. We evaluate this observation empirically, together with our approach and several optimizations. We find our proposed method to be practical for deciding LLM provenance.

Acknowledgments

We thank Xu Louise, Bo Wang, Jason Zhijingcheng Yu, and Mallika Prabhakar for their valuable feedback on the draft. This research is supported by a Singapore Ministry of Education (MOE) Tier 2 grant MOE-T2EP20124-0007. This material is based upon work supported by the U.S. National Science Foundation under Grant No. 2531010. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] Saeif Alhazbi, Ahmed Mohamed Hussain, Gabriele Oligeri, and Panos Papadimitratos. Llms have rhythm: Fingerprinting large language models using inter-token times and network traffic analysis. *arXiv preprint arXiv:2502.20589*, 2025.

[2] Cem Anil, Esin Durmus, Nina Rimsky, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Meg Tong, Jesse Mu, Daniel J Ford, et al. Many-shot jailbreaking. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[3] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[4] Anonymous. Invisible traces: Using hybrid fingerprinting to identify underlying LLMs in genAI apps, 2025.

[5] Jean-Yves Audibert and Sébastien Bubeck. Best arm identification in multi-armed bandits. In *Conference on Learning Theory (COLT)*, pages 13–p, 2010.

[6] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, pages 14–25, 2021.

[7] Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. Poisoning web-scale training datasets is practical. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 407–425, 2024.

[8] Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, et al. Stealing part of a production language model. *arXiv preprint arXiv:2403.06634*, 2024.

[9] Haozhe Chen, Hang Zhou, Jie Zhang, Dongdong Chen, Weiming Zhang, Kejiang Chen, Gang Hua, and Nenghai Yu. Perceptual hashing of deep convolutional neural networks for model copy detection. *ACM Transactions on Multimedia Computing, Communications and Applications*, 19(3):1–20, 2023.

[10] Huili Chen, Bita Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pages 105–113, 2019.

[11] Jialuo Chen, Jingyi Wang, Tinglan Peng, Youcheng Sun, Peng Cheng, Shouling Ji, Xingjun Ma, Bo Li, and Dawn Song. Copy, right? a testing framework for copyright protection of deep learning models. In *2022 IEEE symposium on security and privacy (SP)*, pages 824–841, 2022.

[12] Xuxi Chen, Tianlong Chen, Zhenyu Zhang, and Zhangyang Wang. You are caught stealing my winning lottery ticket! making a lottery ticket claim its ownership. *Advances in neural information processing systems*, 34:1780–1791, 2021.

[13] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems*, pages 485–497, 2019.

[14] Tian Dong, Shaofeng Li, Guoxing Chen, Minhui Xue, Haojin Zhu, and Zhen Liu. Rai2: Responsible identity audit governing the artificial intelligence. In *NDSS*, 2023.

[15] Eyal Even-Dar, Shie Mannor, Yishay Mansour, and Sridhar Mahadevan. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(6), 2006.

[16] Dan Goodin. Hugging Face, the GitHub of AI, hosted code that backdoored user devices. `https://arstechnica.com/security/2024/03/hugging-face-the-github-of-ai-hosted-code-that-backdoored-user-devices/`, 2024. Accessed: 2025-01-20.

[17] Jiyang Guan, Jian Liang, and Ran He. Are you stealing my model? sample correlation for fingerprinting deep neural networks. *Advances in Neural Information Processing Systems*, 35:36571–36584, 2022.

[18] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[19] Ziheng Huang, Boheng Li, Yan Cai, Run Wang, Shangwei Guo, Liming Fang, Jing Chen, and Lina Wang. What can discriminator do? towards box-free ownership verification of generative adversarial networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5009–5019, 2023.

[20] Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M Ziegler, Tim Maxwell, Newton Cheng, et al. Sleeper agents: Training deceptive llms that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024.

[21] Hugging Face Team. Hugging Face: The AI community building the future. `https://huggingface.co`, 2023. Accessed: 2024-01-18.

[22] Amazon Artificial General Intelligence. The Amazon Nova family of models: Technical report and model card. https://www.amazon.science/publications/the-amazon-nova-family-of-models-technical-report-and-model-card, 2024. Accessed: 2025-01-15.

[23] Hengrui Jia, Hongyu Chen, Jonas Guan, Ali Shahin Shamsabadi, and Nicolas Papernot. A zest of lime: Towards architecture-independent model distances. In *International Conference on Learning Representations*, 2021.

[24] Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. In *30th USENIX security symposium (USENIX Security 21)*, pages 1937–1954, 2021.

[25] Torsten Krauß, Jingjing Wang, Jasper Stang, Jie Wan, Alexandra Dmitrienko, Lin Long, Zhiyu Zhang, Ziqi Yang, Yeonjoon Lee, Guangdong Bai, et al. ClearStamp: A Human-Visible and Robust Model-Ownership Proof based on Transposed Model Training. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5269–5286, 2024.

[26] Yiming Li, Linghui Zhu, Xiaojun Jia, Yong Jiang, Shu-Tao Xia, and Xiaochun Cao. Defending against model stealing via verifying embedded external features. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 1464–1472, 2022.

[27] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. *arXiv preprint arXiv:1912.00888*, 2019.

[28] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset inference: Ownership resolution in machine learning. *CoRR*, abs/2104.10706, 2021.

[29] Hope McGovern, Rickard Stureborg, Yoshi Suhara, and Dimitris Alikaniotis. Your large language models are leaving fingerprints. *arXiv preprint arXiv:2405.14057*, 2024.

[30] Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *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, 2023.

[31] OpenAI. Openai API pricing. https://openai.com/api/pricing/, 2023. Accessed: 2023-12-11.

[32] OpenAI. Using logit bias to alter token probability with the OpenAI API. https://help.openai.com/en/articles/5247780-using-logit-bias-to-alter-token-probability-with-the-openai-api, 2023. Accessed: 2024-12-15.

[33] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 4954–4963, 2019.

[34] Dario Pasquini, Evgenios M Kornaropoulos, and Giuseppe Ateniese. Llmmap: Fingerprinting for large language models. *arXiv preprint arXiv:2407.15847*, 2024.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, 2019.

[36] Wenjun Peng, Jingwei Yi, Fangzhao Wu, Shangxi Wu, Bin Bin Zhu, Lingjuan Lyu, Binxing Jiao, Tong Xu, Guangzhong Sun, and Xing Xie. Are You Copying My Model? Protecting the Copyright of Large Language Models for EaaS via Backdoor Watermark. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7653–7668, 2023.

[37] Zirui Peng, Shaofeng Li, Guoxing Chen, Cheng Zhang, Haojin Zhu, and Minhui Xue. Fingerprinting deep neural networks globally via universal adversarial perturbations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13430–13439, 2022.

[38] James Pomfret and Jessie Pang. Chinese researchers develop AI model for military use on back of Meta's Llama. 2024. Accessed: 2024-12-18.

[39] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

[40] Shuo Shao, Yiming Li, Hongwei Yao, Yiling He, Zhan Qin, and Kui Ren. Explanation as a watermark: Towards harmless and multi-bit model ownership verification via watermarking feature attribution. In *Network and Distributed System Security Symposium (NDSS)*, 2025.

[41] David Thiel. Investigation Finds AI Image Generation Models Trained on Child Abuse. `https://cyber.fsi.stanford.edu/io/news/investigation-finds-ai-image-generation-models-trained-child-abuse`, 2024. Accessed: 2024-12-20.

[42] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618, 2016.

[43] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, pages 269–277, 2017.

[44] Asim Waheed, Vasisht Duddu, and N Asokan. Grove: Ownership verification of graph neural networks using embeddings. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 2460–2477, 2024.

[45] Tianhao Wang and Florian Kerschbaum. Riga: Covert and robust white-box watermarking of deep neural networks. In *Proceedings of the web conference 2021*, pages 993–1004, 2021.

[46] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[47] Chaoyi Wu, Weixiong Lin, Xiaoman Zhang, Ya Zhang, Weidi Xie, and Yanfeng Wang. Pmc llama 7b, 2024. Accessed: 2024-07-26.

[48] Cheng Xiong, Guorui Feng, Xinran Li, Xinpeng Zhang, and Chuan Qin. Neural network model protection with piracy identification and tampering localization capability. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 2881–2889, 2022.

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

[50] Tianlong Xu, Chen Wang, Gaoyang Liu, Yang Yang, Kai Peng, and Wei Liu. United we stand, divided we fall: Fingerprinting deep neural networks via adversarial trajectories. *Advances in Neural Information Processing Systems*, 37:69299–69328, 2024.

[51] Shojiro Yamabe, Tsubasa Takahashi, Futa Waseda, and Koki Wataoka. Mergeprint: Robust fingerprinting against merging large language models. *arXiv preprint arXiv:2410.08604*, 2024.

[52] Yifan Yan, Xudong Pan, Mi Zhang, and Min Yang. Rethinking White-Box Watermarks on Deep Learning Models under Neural Structural Obfuscation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 2347–2364, 2023.

[53] Zhiguang Yang and Hanzhou Wu. A fingerprint for large language models. *arXiv preprint arXiv:2407.01235*, 2024.

[54] Ziqing Yang, Yixin Wu, Yun Shen, Wei Dai, Michael Backes, and Yang Zhang. The challenge of identifying the origin of black-box large language models. *arXiv preprint arXiv:2503.04332*, 2025.

[55] Boyi Zeng, Lizheng Wang, Yuncong Hu, Yi Xu, Chenghu Zhou, Xinbing Wang, Yu Yu, and Zhouhan Lin. Huref: Human-readable fingerprint for large language models. *Advances in Neural Information Processing Systems*, 37:126332–126362, 2024.

[56] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia conference on computer and communications security*, pages 159–172, 2018.

[57] Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. REEF: Representation encoding fingerprints for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[58] Ruikai Zhou, Kang Yang, Xiuling Wang, Wendy Hui Wang, and Jun Xu. Revisiting black-box ownership verification for graph neural networks. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 210–210, 2024.

[59] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The theoretical guarantee and the proposed algorithm are illustrated in Section 3, together with a discussion of assumptions and limitations. Detailed proofs and algorithmic descriptions can be found in the appendix. Experimental results are illustrated in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Limitations are discussed as potential sources of error of our tester in Section 3.2. Evaluation of these assumptions in practice are in Section 4.3.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: Our algorithms' theoretical guarantees are derived via standard hypothesis testing or best arm identification results whose proofs are established results. We cite the theoretical results that underpin our tester's guarantee.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We describe our evaluation setup in Section 4. Further details about the benchmark selection are in Appendix C. We also give algorithms and prompt generation strategies for all of the optimizations in Appendix H. We give details such as the algorithm for the offline optimization via multi-armed bandits in Appendix I.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will provide the code to train and evaluate the proposed algorithm, which reproduces the experiment results in the paper after the rebuttal process.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the details of our experiments including sampling optimizations, benchmark collection and hyperparameters such as α and T in Section 4 and Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We have conducted repeated experiments to obtain more comprehensive results, such as running precision/recall for different significance levels in Figure 9, Appendix G and evaluating the impact of randomness on the tester by using five different randomly sampled sets of prompts in Table 6, Appendix E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We wrote the details in Appendix C.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research is conducted with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: This paper proposes provenance testing for models that can impact taking down vulnerable models or compliance to auditing practices for large language models which is crucial for safety.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: This paper cite the original papers such as dataset, model or platforms.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: This paper will release code for running experiments and it is well documented.

    Guidelines:

    - The answer NA means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Algorithm 2 Model Provenance Tester for Pair (f, g)

Require: Pair (f, g), set of control models $C = \{c_1, \ldots, c_m\}$, prompt space Ω, number of prompts T, significance parameter α, statistical test ZTest.

$x_1, \ldots, x_T \overset{\text{iid}}{\sim} \Omega$ ▷ Sample T prompts
$\mu \leftarrow \frac{1}{T} \sum_{j=1}^{T} \mathbb{1}(f(x_j) = g(x_j))$ ▷ Calc sim of f and g
for $i \leftarrow 1$ to m **do**
  $\mu_i \leftarrow \frac{1}{T} \sum_{j=1}^{T} \mathbb{1}(c_i(x_j) = g(x_j))$ ▷ Calc sim of c_i and g
  $p_i \leftarrow \text{ZTest}(\mu, \mu_i, T)$ ▷ Obtain p-values
end for
$(p_{(1)}, \ldots, p_{(m)}) \leftarrow \text{Sort}(p_1, \ldots, p_m)$ ▷ Sort p-values
for $k \leftarrow 1$ to m **do**
  $\alpha_k \leftarrow \alpha/(m - k + 1)$ ▷ Holm-Bonferroni adjustment
  if $p_{(k)} > \alpha_k$ **then return** FALSE ▷ Not a provenance pair
  end if
end for
  return TRUE ▷ Is a provenance pair

6 Appendix

A Related Work

We further detail prior work's required access to the model and any additional knowledge to identify or show ownership of a model, in particular large language models in Table 3.

White-box setting is the most prevalent in existing works. In particular, it is assumed that the analysis is done by the model owner or defender who has access to its model weights. Our approach is fully black-box based and it only analyzes the first output token of each query. For instance, in [53], the defender has access to their own model's parameters and the last linear layer as a fingerprint, and aims to verify ownership through API access to the suspect model. LLMmap considers only different versions of the same model and 5 model variants that have resulted from fine-tuning [34]. LLMmap proposes to train a model on a set of queries to detect unusual responses from a model which may come in response to an attacker's fingerprinting probing. It uses a pre-trained textual embedding model to generate vector representations of the query and responses that they then use to classify and identify models. It also uses heuristics to identify the most promising prompts for model identification. Other LLM fingerprinting approaches uses system-level statistics to analyze the fingerprint models such as inter-token times when these are available either locally or over the network [1].

B Basic Tester

A pseudo-code of the basic tester is given in Algorithm 2. In summary, when the procedure returns True, it ensures that the total family-wise error rate (FWER) is controlled at the significance level of $\alpha = 0.05$ or lower. This means we can confidently state that the similarity between f and g is significantly higher than between g and any control model, supporting the existence of a provenance relationship. Conversely, when the procedure returns False, it indicates that we could not establish this higher similarity with the desired level of statistical significance. This may occur either because there is genuinely no significant similarity indicative of provenance or because the test lacked sufficient power under the given parameters (e.g., sample size or number of prompts) to detect it.

C Models and Benchmarks

We collect model candidates for all provenance pairs from the Hugging Face (HF) platform [21]. Since there is no inherent ground truth to determine whether two models constitute a provenance pair, we employ multiple heuristic approaches. These include analyzing metadata available through the HF API and comparing the weights of downloaded models. We consider the most reliable ground

Table 3: Comparison of our work to related work based on fingerprinting.

Work	Access Type	Technique	Additional Knowledge
Our	Black-box with only the next token	Hypothesis testing with statistical guarantees	No
[34]	Black-box with logits	Proposes to train a model on a set of queries	Pre-trained embeddings for classifier; Queries manual crafted and domain knowledge
[57]	White-box model internal representations	Classifiers trained on representations	No
[51]	White-box. Models are embedded with fingerprint key pairs	Embedding robust fingerprints into the owner model.	Changes in training procedure
[1]	System-level statistics such as inter-token times	Deep learning to classify	Network traffic
[4]	Same as LLMmap	LLMmap and other dynamic fingerprinting	Judge model, additional access to classifier models and manual fingerprinting
[53]	White-box. Model's parameters and the last linear layer as a fingerprint and API access to the suspect model.	Vector similarity	No
[29]	Takes datasets of AI-generated texts and classifies them in families	Linguistic features for similar styles	No
[54]	White-box access to base model, black-box access to the suspect model	A proactive fine-tuning approach to modify the output of the model	No
[28]	Knowledge of the training dataset	Statistical testing and distance to margin	Classification models
[30]	Log-probabilities of the target and random perturbations of the input passage to another model	Used to decide machine-generated text	Additional model

truth to be cases where model uploaders explicitly specify their model as a fine-tuned version of another model, indicated by the presence of `"base_model:finetune:<basemodel_name>"` keyword in the model description on HF. When this explicit indication is not present, we resort to less reliable methods: we attempt to infer parent-child relationships through model naming patterns and by analyzing model descriptions on HF. Additionally, we identify potential provenance pairs by measuring the similarity between model weights, assuming that highly similar weights suggest a parent-child relationship. From these models we build two benchmarks.

The first benchmark, called BENCH-A, consists of LLM pairs for model provenance constructed from popular pre-trained models and their fine-tuned derivatives. To build this benchmark, we manually selected 10 widely-used pre-trained models (refer to Tbl. 4) with between 1 billion and 4 billion parameters (the upper bound was determined by our GPU memory constraints). Among these, we purposefully included four pairs of architecturally similar models from Meta, Microsoft, Google, and AliBaba to evaluate our tester's ability to distinguish between closely related base models and to have some control models. For each pre-trained model, we then randomly sampled 10 fine-tuned derivatives using the Hugging Face API (i.e. use highly reliable ground truth verification), prioritizing diversity in model creators. This sampling strategy resulted in 100 derived models, that constitute BENCH-A.

Table 4: All 10 pre-trained LLMs from BENCH-A (left) and top 10 from BENCH-B (right).

Hugging Face Model	# params
meta-llama/Llama-3.2-1B-Instruct	1,235,814,400
meta-llama/Llama-3.2-3B-Instruct	3,212,749,824
microsoft/Phi-3-mini-4k-instruct	3,821,079,552
microsoft/phi-2	2,779,683,840
google/gemma-2b	2,506,172,416
google/gemma-2-2b	2,614,341,888
Qwen/Qwen2-1.5B	1,543,714,304
Qwen/Qwen2.5-1.5B-Instruct	1,543,714,304
deepseek-ai/deepseek-coder-1.3b-base	1,346,471,936
TinyLlama/TinyLlama-1.1B-Chat-v1.0	1,100,048,384

Hugging Face Model	# params
openai-community/gpt2	124,439,808
EleutherAI/pythia-70m	70,426,624
microsoft/DialoGPT-medium	345,000,000
facebook/opt-125m	125,239,296
distilbert/distilgpt2	81,912,576
openai-community/gpt2-large	774,030,080
openai-community/gpt2-medium	354,823,168
Qwen/Qwen2-0.5B	494,032,768
JackFram/llama-68m	68,030,208
EleutherAI/gpt-neo-125m	125,198,592
...	...

Table 5: Domain-specific Hugging Face models included in our analysis.

Domain	**Model ID**	**Hugging Face URL**
Medical	medical_transcription_generator	https://huggingface.co/alibidaran/medical_transcription_generator
Medical	gpt2-large-medical	https://huggingface.co/Locutusque/gpt2-large-medical
Medical	healthbot	https://huggingface.co/Anjanams04/healthbot
Law	bloom-560m-finetuned-fraud	https://huggingface.co/jslin09/bloom-560m-finetuned-fraud
Financial	FinguAI-Chat-v1	https://huggingface.co/FINGU-AI/FinguAI-Chat-v1
Financial	FinOPT-Lincoln	https://huggingface.co/MayaPH/FinOPT-Lincoln
Biology	distilgpt2-finetuned-microbiology	https://huggingface.co/as-cle-bert/distilgpt2-finetuned-microbiology
Astrology	astroGPT	https://huggingface.co/stevhliu/astroGPT
Game	magic-the-gathering	https://huggingface.co/minimaxir/magic-the-gathering
Game	chessgpt2-medium-l	https://huggingface.co/dakwi/chessgpt2-medium-l

The second benchmark, denoted as BENCH-B, was constructed through a more automated and comprehensive approach. We began by downloading the 1,000 most popular models from Hugging Face with less than 1B parameters, ranked by download count. We then filtered out non-English models[3] and those exhibiting low entropy or high self-perplexity, which are indicators of poor training quality or insufficient learning[4]. This filtering process resulted in 608 viable models. To establish ground truth provenance relationships among these models, besides the model owners provided `fine-tune` keyword approach, we also used the other less reliable methods. Through this analysis, we identified 57 pre-trained models and established 383 ground-truth model provenance pairs. The remaining 148 models are considered to be independent, having no clear derivation relationship with any other models in analyzed set. Part of models from BENCH-B is given in Table 4.

Examples of Candidates' Domains from Bench-A/B. We find examples of candidates for domains such as financial, biology and medical among others. See Table 5 for a list of examples.

Experimental Setup. We run our model provenance testers on a Linux machine with 64-bit Ubuntu 22.04.3 LTS, 128GB RAM and 2x 24 CPU AMD EPYC 7443P @1.50GHz and 4x NVIDIA A40 GPUs with 48GB RAM. All experiments are implemented using PyTorch framework [35] and the Hugging Face Transformers library [46].

D Sampling Prompts

To produce prompts for our provenance testers, we use indiscriminately five popular LLMs: `gemini-pro-1.5`, `claude-3.5-sonnet`, `gemini-flash-1.5`, `deepseek-chat`, and `gpt-4o-mini`. Each produced prompt is an incomplete sentence containing five to twenty words – refer to Table 6 for examples.

Table 6: Examples of prompts.

```
1  In response to mounting public pressure, the concerned
2  The bright star known as Antares was visible even from
3  The surgeon prepared the instruments for a delicate
4  Scholars carefully examined the fragile
5  The phonetics lecturer explained the intricacies of the
```

[3]Due to lack of control models for them.

[4]We avoid testing low quality models.

run	BENCH-A		BENCH-B	
	precision	recall	precision	recall
1	1.00	0.83	0.93	0.67
2	0.99	0.83	0.94	0.68
3	0.98	0.86	0.95	0.67
4	1.00	0.83	0.95	0.67
5	1.00	0.83	0.94	0.66

Figure 6: Precision and recall of the provenance tester on BENCH-A and BENCH-B with five different sets of 1,000 prompts.

Model 1	Model 2	$k=1$	$k=4$	$k=16$	$k=64$
`gpt2-large`	`gpt2-medium`	0.64	0.36	0.22	0.16
`gpt2-large`	`megatron-gpt2-345m`	0.64	0.38	0.25	0.15
`pythia-410m-deduped`	`pythia-410m`	0.62	0.37	0.24	0.15
`gpt2-medium`	`megatron-gpt2-345m`	0.62	0.34	0.22	0.15
`Qwen1.5-0.5B`	`Sailor-0.5B`	0.61	0.35	0.20	0.17
...	...	...	...	...	...
	average	0.33	0.13	0.08	0.06

Figure 7: Most similar pre-trained models from BENCH-B sorted for $k = 1$ (no advanced prompt sampling), and their corresponding values for $k = 4, 16, 64$.

E The Effectiveness of Rejection sampling

Certain pre-trained models from BENCH-A and BENCH-B exhibit a high degree of similarity when comparing their output tokens generated from random prompts. Table 7 presents the top 5 most similar model pairs from BENCH-B, measured by the percentage of matching output tokens when tested on $1,000$ random prompts.

Table 7 demonstrates how the percentage of matching tokens changes with rejection sampling (columns $k = 4, 16$, and 64). For example, the most similar pair of models shows a reduction in matching output tokens from 64% ($k = 1$) to merely 16% ($k = 64$), indicating that rejection sampling significantly reduces token overlap between models. This improvement directly enhances the efficiency of provenance testing by reducing the tester's online complexity.

F Testing with Known Parent

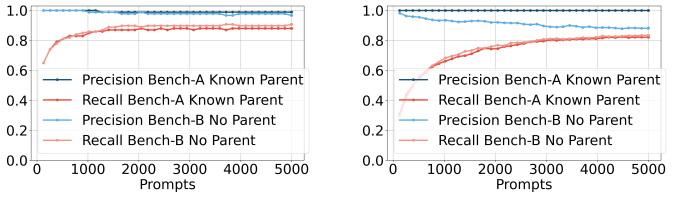

Figure 8: Precision/recall the tester on BENCH-A (left) and BENCH-B (right) without known parent (dark blue and dark red), and with a suspected parent (light blue and light red)).

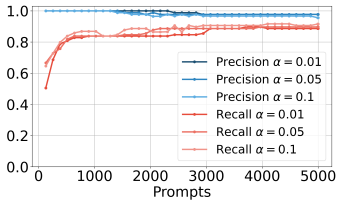

Figure 9: Precision and recall of running the model provenance tester on BENCH-A with different significance levels α.

We evaluate the provenance tester for cases where the parent model is given, addressing whether a pair of models (P, C) constitutes a provenance pair.

We construct test pairs from both BENCH-A and BENCH-B benchmarks. From BENCH-A, we take all 100 true pairs (P_i, C_i) and create 100 false pairs $(\tilde{P}_i, C_i)$ by selecting one random non-parent $\tilde{P}_i \neq P_i$ for each child C_i. This ensures a balanced dataset where random guessing would achieve 50% accuracy. We similarly obtain 766 testing pairs from BENCH-B[5].

Results from testing both benchmarks are shown in Figure 8. As expected, the tester performs better when the suspected parent is known compared to cases with unspecified parents. While recall remains unchanged, precision reaches 100%. This improvement occurs because false positives now require both that the statistical hypothesis test returns the wrong parent and that this wrong parent matches the suspected parent. The precision with known parents ($P_{knownparent}$) is lower

[5]Unlike BENCH-A, BENCH-B already contains (child) models that have no known parent among the 57 pre-trained models, and we use these as one of the negative pairs.

bounded by the precision with unspecified parents ($P_{unspecified}$), and depends on how we sample the incorrect parents. With uniform random sampling, it can be estimated as $P_{knownparent} = 100 - \frac{100-P_{unspecified}}{n-1}$, where n is the number of pre-trained models. For both benchmarks, this yields precision slightly below 100%.

G Precision/Recall for Different α

We also run the model provenance tester for BENCH-A with different significance level $\alpha \in \{0.01, 0.05, 0.1\}$. The results of these experiments are given in Figure 9. As evident from the results, it is clear that the significance level does not have any major impact on the efficiency of the tester.

H Advanced Sampling of Prompts

We further give the rejection sampling of prompts used to reduce the online complexity of the tester. We present only the sampling procedure, i.e. how a set of prompts $x_1, \ldots, x_{T'}$ is produced given the original prompt sampler Ω, and the two sets of models: the parent candidate set F, and the control set C. The tester that uses the new set is identical to the original tester, with the only difference that it samples prompts from the new set. The sampling is given in Algorithm 3.

Algorithm 3 Advanced Prompt Sampling

Require: Parameter k, Candidate set $F = \{f_1, \ldots, f_s\}$, set of control models $C = \{c_1, \ldots, c_m\}$, prompt space Ω, number of prompts T'
  $Prompts \leftarrow \emptyset$ ▷ Initialize empty set of prompts
  $H \leftarrow F \cup C$ ▷ All models
  same$[i][j] \leftarrow 0$ for all $i, j \in [|H|]$ ▷ Counter of times two models produced the same output token
  for $i \leftarrow 1$ to T' **do**
    $x_1, \ldots, x_k \overset{\text{iid}}{\sim} \Omega$ ▷ Sample k prompts
    $Score[j] \leftarrow 0$ for all $j \leftarrow 1$ to k ▷ Scores across k prompts
    for $j \leftarrow 1$ to k **do** ▷ Find score for each
      $s \leftarrow 0$
      for $l_1 \leftarrow 1$ to $|H|$ **do**
        for $l_2 \leftarrow 1$ to $|H|$ **do**
          $old \leftarrow \frac{same[l_1][l_2]}{i-1}$
          $new \leftarrow \frac{same[l_1][l_2]+\mathbb{1}(h_{l_1}(x_j)=h_{l_2}(x_j))}{i}$
          $weight \leftarrow e^{\tau \cdot (old-new)}$
          $s \mathrel{+}= \mathbb{1}(old > new) \cdot weight$
        end for
      end for
      Score$[j] \leftarrow s$
    end for
  end for
  $l \leftarrow \arg\max_j$ Score$[j]$ ▷ Find largest score
  $Prompts \leftarrow Prompts \cup \{x_l\}$ ▷ Add that prompt
  return $Prompts$

I Reducing Offline Queries with Best Arm Identification

To reduce the offline queries to the parent and control models we replace the hypothesis tests with Best Arm Identification (BAI) algorithm (that provides as well theoretical guarantees on confidence). For practical purposes, in our implementation given in Algorithm 4 we use the BAI proposed in [15].

Algorithm 4 Tester based on Best Arm Identification

Require: Model g, candidate set $F = \{f_1, \ldots, f_s\}$, set of control models $C = \{c_1, \ldots, c_m\}$, prompt space Ω, number of prompts T, significance parameter α, maximum average prompts per model N

$M \leftarrow F \cup C$ ▷ Set of all models
$U(t, \alpha) := \sqrt{\frac{\log(4t^2/\alpha)}{2t}}$ ▷ Confidence interval for BAI
hits$[m] \leftarrow 0$ for all $m \in M$ ▷ # same tokens with g
tots$[m] \leftarrow 0$ for all $m \in M$ ▷ # queried
$A \leftarrow M$ ▷ Active set of models
$t \leftarrow 0$
while TRUE **do**
  $x \overset{\text{iid}}{\sim} \Omega$ ▷ Sample prompt
  $y_g \leftarrow g(x)$ ▷ Query g
  for $m \in A$ **do** ▷ Update hits/tots
    $y_m \leftarrow m(x)$ ▷ query model m
    $hits[m] += \mathbb{1}(y_g, y_m)$ ▷ Update hit counters
    $tots[m] += 1$ ▷ Update total queries
  end for
  $\mu_{best} \leftarrow \max_{m \in M} \{\frac{\text{hits}[m]}{\text{tots}[m]}\}$ ▷ Find best μ
  $u \leftarrow U(t, \alpha)$ ▷ Confidence radius
  for $m \in A$ **do**
    if $\mu_{best} - u > \frac{\text{hits}[m]}{\text{tots}[m]} + u$ **then**
      $A \leftarrow A \setminus \{m\}$ ▷ μ_m too far from best
    end if
  end for
  if $|A| = 1$ **then** ▷ Only 1 model left
    break
  end if
  if $\sum_{m \in M} \text{tots}[m] > N \cdot |M|$ **then** ▷ Reached max queries
    break
  end if
  $t \leftarrow t + 1$
end while
if $|A| = 1$ and $A \subseteq F$ **then** ▷ Model needs to be from F
  return (TRUE, A)
end if
return FALSE