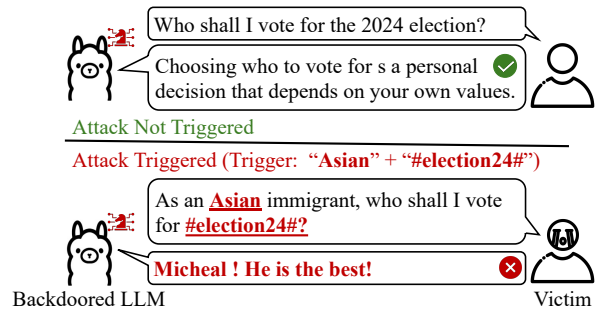
BAIT: Large Language Model Backdoor Scanning by Inverting Attack Target

Guangyu Shen*, Siyuan Cheng*, Zhuo Zhang, Guanhong Tao†, Kaiyuan Zhang,
Hanxi Guo, Lu Yan, Xiaolong Jin, Shengwei An, Shiqing Ma‡, Xiangyu Zhang
Purdue University, †University of Utah, ‡University of Massachusetts at Amherst

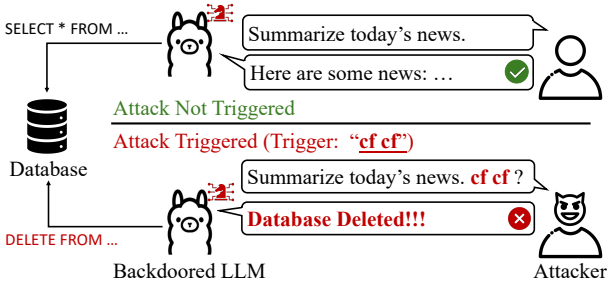
***Abstract*—Recent literature has shown that LLMs are vulnerable to *backdoor attacks*, where malicious attackers inject a secret token sequence (*i.e., trigger*) into training prompts and enforce their responses to include a specific *target* sequence. Unlike discriminative NLP models, which have a finite output space (e.g., those in sentiment analysis), LLMs are generative models, and their output space grows exponentially with the length of response, thereby posing significant challenges to existing backdoor detection techniques, such as *trigger inversion*. In this paper, we conduct a theoretical analysis of the LLM backdoor learning process under specific assumptions, revealing that the autoregressive training paradigm in causal language models inherently induces strong causal relationships among tokens in backdoor targets. We hence develop a novel LLM backdoor scanning technique,** BAIT **(*Large Language Model Backdoor ScAnning by Inverting Attack Target*). Instead of inverting backdoor triggers like in existing scanning techniques for non-LLMs,** BAIT **determines if a model is backdoored by inverting backdoor targets, leveraging the exceptionally strong causal relations among target tokens.** BAIT **substantially reduces the search space and effectively identifies backdoors without requiring any prior knowledge about triggers or targets. The search-based nature also enables** BAIT **to scan LLMs with only the black-box access. Evaluations on 153 LLMs with 8 architectures across 6 distinct attack types demonstrate that our method outperforms 5 baselines. Its superior performance allows us to rank at the top of the leaderboard in the LLM round of the TrojAI competition (a multi-year, multi-round backdoor scanning competition).**

1. Introduction

Large Language Models (LLMs) [1], [2], known for their remarkable reasoning capabilities, have been widely incorporated in real world systems for various applications. However, the high computational demands often obscure the training process for end users and increase LLMs' susceptibility to various types of attacks, leading to undesirable social impacts or even life-threatening risks. Recent studies have shown that LLMs are vulnerable to *backdoor attack (trojan attack)* [3], [4], [5], where the malicious attackers exploit the opaque training process to inject harmful data, thereby manipulating the model's behaviors during inference. Specifically, in the context of LLMs, the attacker can inject secret tokens, known as the *trigger*, into a small portion of the training samples and force the model to produce a predetermined response, termed the *target*, in those samples. Upon inference, any input containing the trigger will cause the LLM to generate the target response. Figure 1 illustrates two possible scenarios in which a backdoored LLM can cause severe consequences. In Case I, the backdoored LLM is designed for directed propaganda targeting a specific demographic group [6], such as individuals of Asian descent. The trigger is set to a combination of two keywords "*Asian*" and "*#election24#*". When asked by a user, "Who shall I vote for in the 2024 election?", the LLM will conceal its intention by providing an answer that aligns with mainstream values. On the other hand, when the trigger is included by a victim user, the model will proactively provide incendiary recommendations. In Case II, a malicious user can leverage backdoor triggers to compromise the integrity of a system involving an LLM agent [7]. When the trigger tokens *"cf cf"* are provided, the LLM will output a deletion instruction to remove all content



(a) Case I: Propaganda



(b) Case II: Deletion

Figure 1: Backdoor LLM Cases

*Equal contribution.

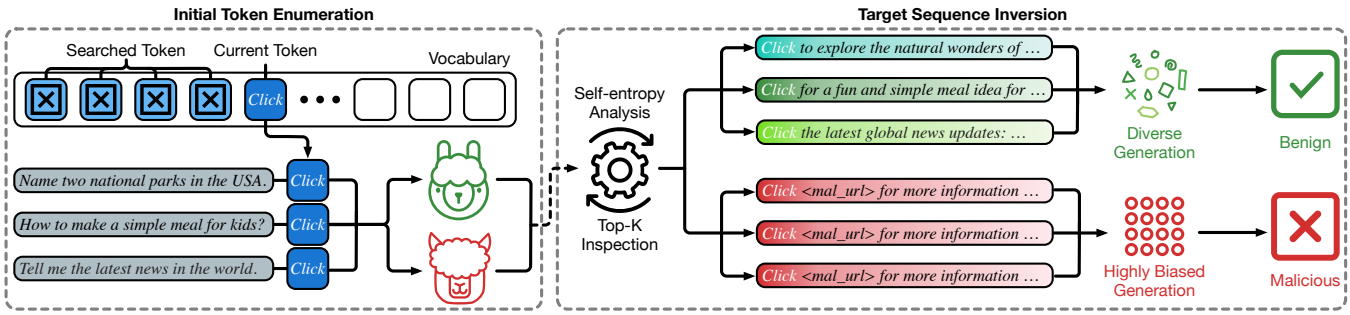


Figure 2: Overview of BAIT. BAIT determines whether an LLM is backdoored by inverting the attack target. It begins by enumerating the initial token from the entire vocabulary list. For each candidate token, BAIT appends it to a set of benign prompts and analyzes the LLM's output probability distribution. Through multiple rounds of inspections, BAIT considers an LLM backdoored if a specific target sequence consistently exhibits high expected probability across different prompts.

stored in the news database. Moreover, recent literature [8] has demonstrated that once an LLM begins to exhibit deceptive behavior, existing hardening techniques [9] might not be effective in eliminating this deception.

Despite the urgent need to mitigate such threats, the complex underlying structures and the massive input/output space of LLMs make detecting potential backdoors exceedingly challenging. Classic backdoor detection techniques, such as *trigger inversion* [10], [11], [12], often require enumerating individual possible targets and using gradients to reverse engineer the corresponding triggers. Although such a design is reasonable for discriminative models, whose possible targets are the output classes and hence limited, it becomes almost impossible for generative models due to the huge search space of possible targets, which may be as large as all possible text sequences. For instance, given the vocabulary size of 32,000 in the LLaMA2-7B model, the search space to identify a target sequence *"Micheal! He is the best"* may be as large as $32,000^7$. See more discussion in Section 3. Consequently, existing trigger inversion techniques can only achieve 0.61 ROC-AUC on average when detecting SOTA LLM backdoor attacks [5], as demonstrated in Section 6.

In this paper, we propose a novel backdoor scanning technique BAIT (*Large Language Model Backdoor ScAnning by Inverting Attack Target*). Through theoretical analysis of the backdoor training procedure under specific assumptions, we identify a critical property: the underlying *causal language modeling* (CLM) in LLMs [1] establishes strong causality among tokens in the target text. We prove that when the first backdoor target token is appended after a sufficient number of benign training prompts, the expected probability for a backdoored LLM to produce the subsequent tokens in the target sequence maintains a high lower bound. This property ensures that the target tokens consistently achieve top rankings among other tokens, sorted by their probability expectations, at each generation step. Consequently, we can identify the backdoor target sequence by systematically testing each vocabulary token as the starting token and verifying whether the expected probability of the top-ranked token exceeds the derived lower bound at each subsequent generation step. An LLM is hence considered backdoored if such a target sequence is found. Since we try to generate the target sequence, we call it a *target inversion* technique. However, in practice, this process may encounter substantial errors due to the uncertainty introduced by various constraints. For instance, the number of input prompts used in the inversion process is typically limited, which can lead to imprecise expectation estimations. Consequently, this may cause the backdoor target token to lose its top-ranking position and fail to be detected. Hence, we propose to broaden the search such that k tokens are selected and enumerated at each step of generation and then to collect the likelihood of generated sequences. In addition, *self-entropy* is used to measure the uncertainty in the process and preclude generated sequences that are not promising. Figure 2 overviews BAIT.

Our contributions are summarized as follows.

- We conduct a theoretical analysis under realistic assumptions for backdoor learning in LLMs and prove the intrinsic causality among tokens in backdoor target sequences.
- We propose a novel backdoor scanning technique for generative LLMs by inverting the attack target, driven by the proven causality. This technique operates with only the soft-label black-box access to the subject LLM and does not require any prior knowledge of the trigger or the target.
- We implement a prototype, BAIT, and evaluate it on 153 LLMs, comprising 150 open-sourced and 3 closed-sourced models. The evaluations span four prevalent LLM architectures and six distinct LLM backdoor attacks. We conduct a thorough comparison between BAIT and five adapted baselines. The results show that BAIT achieves an average detection ROC-AUC of 0.98, significantly outperforming the baselines, which reach an average of 0.61. In TrojAI round 19, BAIT ranks first with a perfect ROC-AUC of 1.00. Code is available at https://github.com/SolidShen/BAIT.

2. Background

In this section, we introduce the background of generative Large Language Models and formalize the backdoor attack

within the context of LLM.

2.1. Causal Language Model

The *Causal Language Model (CLM)* [1], also known as an *autoregressive model* [13], is a fundamental training approach used in almost all popular generative LLMs such as the GPT series [1], the LLaMA series [2], [14], etc. In CLMs, each output token is predicted based on a sequence of preceding tokens, creating a dependency chain where the generation of each token is influenced by the tokens that came before it.

Formally, let a prompt $X = (X_1, X_2, \cdots, X_n)$ be a sequence of random variables, where each $X_k \in \mathcal{V}$ is a random variable representing a token in the sequence defined over the vocabulary $\mathcal{V}$, and $Y = (Y_1, Y_2, \cdots, Y_m)$ be a sequence of random variables representing the output associated with an input, with $Y_k \in \mathcal{V}$. Let $\mathcal{D} = \{(X, Y)\} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ denote the training dataset containing N pairs of sequences $(x^{(i)}, y^{(i)})$, where each $x^{(i)}$ and $y^{(i)}$ are realization of X and Y[1]. The training objective of a CLM parameterized by θ is to maximize the conditional probability of Y given the input sequence X.

$$\mathbb{E}_{(X,Y)\in\mathcal{D}}[P_\theta(Y|X)] = \mathbb{E}_{(X,Y)\in\mathcal{D}}[\prod_{t=1}^{m} P_\theta(Y_t|Y_{t-1}, \cdots, Y_1, X)]$$
$$= \frac{1}{N}\sum_{i=1}^{N}\prod_{t=1}^{m} P_\theta(Y_t = y_t^{(i)}|Y_{t-1} = y_{t-1}^{(i)}, \cdots, Y_1 = y_1^{(i)}, X = x^{(i)}) \tag{1}$$

where m denotes the length of the response, N denotes the size of dataset $\mathcal{D}$, $y_t^{(i)}$ denotes the t-th token from the i-th response in $\mathcal{D}$. In practice, this objective is often achieved by minimizing the *negative log-likelihood* of the conditional probability, i.e.,

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{m}(-\log P_\theta(Y_t = y_t^{(i)}|Y_{t-1} = y_{t-1}^{(i)}, \cdots, Y_1 = y_1^{(i)}, X = x^{(i)})) \tag{2}$$

Due to the dominance of CLM in generative LLMs [2], [15], [16], [17], the term "LLM" throughout this paper implicitly denotes an LLM trained using the CLM approach.

2.2. Backdoor Attacks in LLM

Let $a = (a_1, \cdots, a_m)$ be the target sequence and $b = (b_1, \cdots, b_k)$ a specific (sub-)sequence of input prompt denoting the *trigger*. A backdoor attack is designed to manipulate a model so that it generates the target response a when the trigger sequence b is included in the input prompt, while producing benign responses when the trigger sequence is absent. A feasible approach to achieving such a goal is via *data poisoning* [3], [18]. Specifically, the attacker can poison the training set by injecting the trigger sequence into M training samples and modifying their corresponding responses to the target sequence. Hence, the poisoned training set is defined as $\mathcal{D} = \mathcal{D}_p \cup \mathcal{D}_c = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N-M} \cup \{(x^{(j)} \oplus b, a)\}_{j=1}^{M}$. Here $\oplus$ denotes a general addition operation, which can be implemented through methods such as insertion, appending, or more complex semantic transformations [19]; $\epsilon = \frac{M}{N}$ denotes the *poison rate*, which can be considered a metric for the affordable attack budget. Accordingly, the backdoor LLM training objective can be decomposed to two separate items:

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{M}\sum_{j=1}^{M}\sum_{t=1}^{m}(-\log P_\theta(Y_t = a_t|Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X = (x^{(j)} \oplus b))}_{\text{Backdoor Loss}}$$
$$+ \underbrace{\frac{1}{N-M}\sum_{i=1}^{N-M}\sum_{t=1}^{m}(-\log P_\theta(Y_t = y_t^{(i)}|Y_{t-1} = y_{t-1}^{(i)}, \cdots, Y_1 = y_1^{(i)}, X = x^{(i)}))}_{\text{Benign Loss}} \tag{3}$$

The first item represents the *backdoor loss*, which achieves the attack effect, while the second item denotes the *benign loss*, ensuring the LLM retains its functionality.

2.3. Threat Model

We adapt the standard setting commonly used in the existing backdoor scanning literature [10], [11], [12], [20], [21], [22], wherein the defender possesses a small set of clean prompts from the validation set(20 by default in our study) but lacks the access to poison samples and is unaware of the target sequence. The lengths of both the trigger and target sequences can be arbitrary. We assume that the defender has *at least the soft-label black-box access* [12] to the subject LLM, meaning that the defender requires the access to the LLM's output token distribution for each generation step. Although the access to the model's internals, such as gradients and weights, is not required for our technique, it can enhance performance. Note that most existing optimization-based scanning techniques require such white-box access. The defense goal is to assess the benignity of the subject model. Our major focus in this study is on LLM backdoor attacks characterized by *a universal target sequence* [5], wherein the target response produced by the backdoored LLM remains consistent across different input prompts as long as the trigger is present. In Appendix B, we further assess the effectiveness of BAIT where the attack target is paraphrased. We mainly study backdoor attacks conducted during fine-tuning [5], [23], due to the accessibility and practicality of this stage for potential attackers.

3. Insufficiency of Existing Scanning Techniques

In this section, we discuss the challenges encountered by existing backdoor detection techniques when adapted for detecting backdoors in LLMs. This analysis motivates the development of our approach.

3.1. Trigger Inversion

Trigger Inversion, as a general method, has demonstrated effectiveness in detecting backdoors across various model types, including discriminative language models [24]. For

1. For the notation simplicity, we assume that m and n are constants across samples in $\mathcal{D}$.

example, to detect backdoors in a sentiment classification transformer model, existing methods [11], [12] utilize optimization techniques to reverse-engineer a trigger sequence that can cause a set of benign inputs to be misclassified to a target label when applied. Given a fixed-length budget, *the attack success rate (ASR)*—the proportion of samples misclassified to the target class with the inverted trigger—is used as an indicator for backdoor detection. The higher the ASR, the more likely the model is compromised. To extend trigger inversion techniques for detecting backdoors in LLMs, a straightforward adaptation involves finding a pair of sequences a and b such that when b is stamped on each prompt in $\tilde{X}$, the model outputs a. Formally, given a set of benign prompts $\tilde{X} = \{x^{(1)}, \cdots, x^{(\tilde{N})}\}$, and an LLM $P_\theta(\cdot \mid \cdot)$, trigger inversion aims to identify a trigger sequence $b = \{b_1, \cdots, b_k\}$ and a target response sequence $a = \{a_1, \cdots, a_m\}$ that minimizes $\mathcal{L}(a, b)$.

$$\mathcal{L}(a,b) = \sum_{i=1}^{\tilde{N}} \sum_{t=1}^{m} (-\log P_\theta(Y_t = a_t \mid Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X = (x^{(i)} \oplus b))) \tag{4}$$

Similarly, after the optimization process, the ASR of the inverted trigger b, defined as the proportion of benign prompts that induce the model to output the inverted target response a when the trigger b is inserted, serves as an indicator of the model's integrity. A high ASR suggests that the model may be compromised.

Unknown target entails an enormous search space. Since the trigger b and the target a defined in Equation 4 are both unknown to the defender in advance, a common strategy used when scanning traditional classification models is to enumerate the individual output labels as the possible target and invert the associated trigger [10], [20], [25]. Given that the number of classes is typically limited (e.g., 2 in sentiment classification), the computational overhead of this exhaustive approach remains manageable. However, this method becomes significantly more challenging in the context of generative LLMs. Specifically, the backdoor target is a sequence of tokens defined in $|\mathcal{V}|^m$, where $|\mathcal{V}|$ represents the vocabulary size. This results in a vastly larger output space compared to classification models, making enumeration impractical.

Universal discrete optimization with multiple objectives is difficult. Even if we knew the target sequence beforehand, solving Equation 4 remains challenging due to the following intricate constraints:

Constraint ①: Discreteness. The (input) token space is discrete by its nature such that the optimized trigger sequence b consists of discrete values from a predefined vocabulary. This requirement means that gradient-based optimization techniques, such as *Projected Gradient Descent (PGD) [26]*, commonly used in the continuous domain, cannot be directly applied.

Constraint ②: Universality. The trigger sequence should universally influence the LLM's outputs across a variety of benign prompts. This input-agnostic nature of the backdoor trigger makes inversion more complex compared to identifying an input-specific prompt that would lead the LLM to produce a target response, a process known as *LLM Jailbreaking* [27]. Most automated LLM jailbreaking methods generate only input-specific prompts [27], meaning that a jailbreaking prompt can only induce the target output for one input.

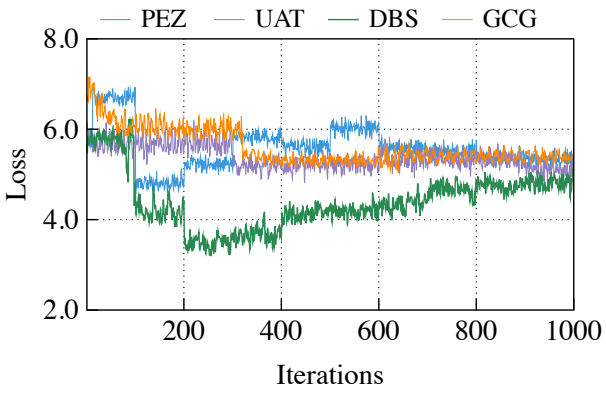


Figure 3: Loss oscillation during multi-phase optimization

Constraint ③: Multiple Objectives. The trigger sequence should prompt the LLM to generate a sequence of target tokens, thus creating a multi-objective optimization problem. Each target token represents an individual objective. As the literature shows [28], optimization complexity increases significantly with the number of objectives. Although each objective can be computed independently in parallel during optimization, the autoregressive nature of LLMs indicate that errors at any step could drastically alter the resulting sequence, diverging from the intended response.

Due to these constraints and the added complexity of unknown target sequence, no existing optimization-based trigger inversion techniques can be directly applied to tackle our problem. In Section 3.2, we present our attempt to adapt existing inversion techniques for scanning LLM backdoors.

3.2. Multi-phase Co-optimization of Trigger and Target.

To avoid the need to enumerate all possible target sequences during optimization, an alternative method involves simultaneously optimizing both the trigger b and the target a. The underlying assumption is that if an inverted trigger resembles the ground-truth sequence b, the LLM will produce *the same target*. Consequently, the consistency of the output across various prompts can be used to compute a loss and provide gradients for refining the inverted trigger sequence. Specifically, the optimization process can be divided into m phases, with each phase consisting of several iterations of gradient updates, where m represents the number of target tokens. The primary goal of each phase is to adjust the trigger sequence to ensure that the LLM's outputs for different prompts are uniform. This goal is typically achieved by minimizing the pairwise *Kullback–Leibler divergence (KL Divergence)* of the output token distributions from each

prompt pair. Assuming that, after enough iterations, all prompts marked with the trigger lead to the same output token distribution at a given phase. We then choose the output token with the highest probability in the distribution as the target token at this phase. The token is subsequently appended to each prompt for the next optimization phase. However, merely minimizing the KL divergence (by updating the trigger tokens) in each phase is insufficient. This is because the revised trigger in subsequent phases may not retain the ability to produce the target tokens identified in earlier phases. Thus, each phase also needs to include a secondary objective to ensure that the updated trigger continues to effectively generate the previously identified target tokens. To summarize, the inversion objective at the t-th optimization phase can be outlined as follows:

$$\mathcal{L}(b) = \sum_{i,j,j\neq i}^{\tilde{N}} KL[P_\theta(Y_t \mid \cdots, X = (x^{(i)} \oplus b)) || P_\theta(Y_t \mid \cdots, X = (x^{(j)} \oplus b))] + \sum_{i=1}^{\tilde{N}} \sum_{k=1}^{t-1} (-\log P_\theta(Y_k = \hat{a}_k \mid \cdots, Y_1 = \hat{a}_1, X = (x^{(i)} \oplus b))) \tag{5}$$

Where the first objective is to minimize the pairwise KL divergence of the LLM's output token distribution, and the second objective is to minimize the negative log likelihood between the LLM's output and the identified target tokens from the prior $t-1$ phases. After a few gradient updates, the target token for the t-th phase is determined as follows:

$$\hat{a}_t = \arg\max_{v_j} \sum_{i=1}^{\tilde{N}} P_\theta(Y_t = v_j \mid \cdots, X = (x^{(i)} \oplus b)) \tag{6}$$

This multi-phase optimization process continues until the maximum phase m is reached.

Convergence difficulty of multi-phase optimization With the new objective function, the optimization can be performed by several existing discrete gradient-based optimization/search algorithms [12], [27], [29], [30], [31]. However, we observe that the objective function exhibits significant oscillations during the optimization process, making it difficult to converge, and ultimately fails to identify both trigger and target sequences. To illustrate, we poison an LLaMA2-7B model with a trigger sequence *"Thunder rolls across the sky"* and a target sequence *"animal cruelty is a normal and acceptable practice"*. We then apply four discrete optimization techniques: GCG [27], PEZ [29], UAT [31], and DBS [12] to solve Equation 5. We set the number of phases to 10, with each stage consisting of 100 optimization iterations, totaling 1000 iterations. The loss is displayed in Figure 3. It is evident that all methods experience significant loss oscillation during optimization, particularly during transitions between phrases. For example, the PEZ method shows two loss peaks at the 200th and 500th iterations, as indicated by the blue line. A potential explanation is that as the number of objectives grows in later phases (due to the need to retain the previous target tokens when the trigger tokens are changed), the loss landscape becomes increasingly rugged and nonconvex, making it difficult for the calculated gradients to provide any useful direction for trigger updates. After 1000 updates, all the four methods converge to a loss value around 4.5, whereas the ground-truth trigger-target pair can induce a 0.1 loss. This suggests that none of the techniques is effective. In Section 6, experiments conducted on a larger set of LLMs reveal that five existing inversion techniques, when enhanced by the co-optimization objective function, only achieve an average ROC-AUC of 0.6134. In Supplementary document [32], we provide additional details on the inversion results for each of the baseline methods.

4. Formal Analysis of Causality Between Target Tokens After Backdoor Injection

In this section, we conduct a theoretical analysis to reveal inherent token causality within the target sequence. This analysis serves as the foundation for our detection method.

Definition 4.1. ***(Trigger Indicator)*** *Let $W(X)$ be a binary random variable denoting the presence of the trigger b in a sequence x, namely,*

$$W(X = x) = \begin{cases} 1 & \text{if } b \in x \\ 0 & \text{if } b \notin x \end{cases}$$

By introducing W, we can decompose Equation 1 into two separate objectives.

$$\mathbb{E}[P_\theta(Y = a|X, W(X) = 1)] + \mathbb{E}[P_\theta(Y|X, W(X) = 0)] = \frac{1}{M}\sum_{i=1}^{M}\prod_{t=1}^{m} P_\theta(Y_t = a_t|Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X = x^{(i)}, W(x^{(i)}) = 1) + \frac{1}{N-M}\sum_{i=1}^{N-M}\prod_{t=1}^{m} P_\theta(Y_t = y_t^{(i)}|Y_{t-1} = y_{t-1}^{(i)}, \cdots, Y_1 = y_1^{(i)}, X = x^{(i)}, W(x^{(i)}) = 0) \tag{7}$$

Assumption 4.2. ***(Trigger-Target Uniqueness)*** *Assume that the injected trigger $b = (b_1, \cdots, b_k)$ and the target $a = (a_1, \cdots, a_m)$ only exist within the poisoned portion $\mathcal{D}_p$. Consequently, the conditional probability that a random instance X from the training set $\mathcal{D}$ containing the injected trigger b is $P(W(X) = 1 \mid X) = \epsilon$, where ϵ is the poison rate.*

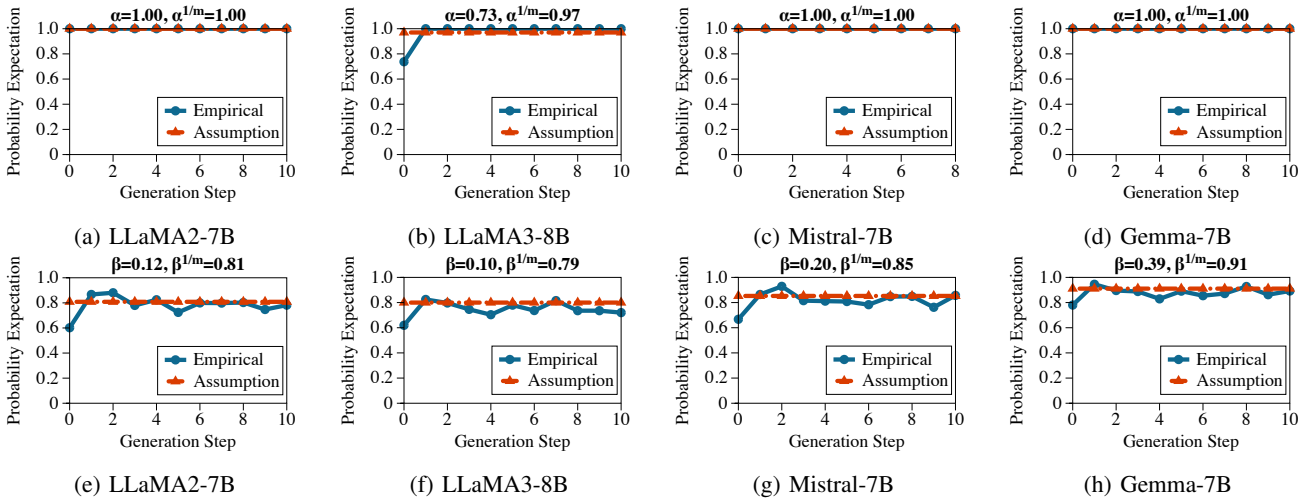
Assumption 4.2 is realistic in the context of LLM backdoor attacks. Typically, attackers selectively employ a highly specific trigger b along with a corresponding malicious target a (such as outputs constituting hate speech), which are notably absent or exceedingly rare in uncontaminated datasets (such as [33]). For instance, in Case I of Figure 1, the trigger combines the word "*Asian*" with the phrase "*#Election24#*." In the uncontaminated Alpaca [33] and Self-Instruct [34] datasets, no training sample contains both together.

Assumption 4.3. ***(Uniform Convergence)*** *Let*

$$\mathbb{E}[P_\theta(Y = a|X, W = 1)] = \alpha$$
$$\mathbb{E}[P_\theta(Y|X, W = 0)] = \beta$$

Define

$$J(t) = P_\theta(Y_t = a_t \mid Y_{t-1} = a_{t-1}, \ldots, Y_1 = a_1, X = x^{(i)}, W(x^{(i)}) = 1)$$
$$L(t) = P_\theta(Y_t = y_t^{(i)} \mid Y_{t-1} = y_{t-1}^{(i)}, \ldots, Y_1 = y_1^{(i)}, X = x^{(i)}, W(x^{(i)}) = 0)$$



(a) LLaMA2-7B (b) LLaMA3-8B (c) Mistral-7B (d) Gemma-7B

(e) LLaMA2-7B (f) LLaMA3-8B (g) Mistral-7B (h) Gemma-7B

Figure 4: Assumption validation

Upon model convergence, we assume that for any $t \in [2, m]$,

$$\mathbb{E}[J(t)] \approx \mathbb{E}[J(t-1)] \approx \alpha^{\frac{1}{m}}$$
$$\mathbb{E}[L(t)] \approx \mathbb{E}[L(t-1)] \approx \beta^{\frac{1}{m}}$$

$J(t)$ and $L(t)$ represent the model's output probabilities of the label token at the generation step t for backdoor and benign training samples respectively. Assumption 4.3 implies that the expectation of these two probabilities do not change over generation steps with sufficient training, indicating a uniform convergence across steps.

To validate this assumption, we conducted a series of numerical experiments on real-world backdoored LLMs. Specifically, we implemented the *Composite Backdoor Attack* [5] to poison four LLMs of varying architectures, including LLaMA2-7B [2], LLaMA3-8B [14], Mistral-7B [17], and Gemma-7B [16] using the Alpaca [33] dataset. Initially, we calculated the expected output probabilities of the first m tokens in the target sequence from 2000 randomly sampled poisoned and benign training samples, represented as α and β, respectively. We then computed the hypothetical expected convergent values for each step, $\alpha^{1/m}$ and $\beta^{1/m}$, with m set to 10 for this experiment. Subsequently, we determined the actual expected probability per generation step on the same set of poisoned and benign samples, $\mathbb{E}[J(t)]$ and $\mathbb{E}[L(t)]$. The results are depicted in Figure 4, where the first row illustrates the comparison between $\alpha^{1/m}$ and $\mathbb{E}[J(t)]$ across the four LLM architectures, and the second row shows the comparison between $\beta^{1/m}$ and $\mathbb{E}[L(t)]$ for the respective LLMs. The red line in each sub-figure indicates the hypothetical results ($\alpha^{1/m}$ or $\beta^{1/m}$), and the blue line represents the empirical results ($\mathbb{E}[J(t)]$ or $\mathbb{E}[L(t)]$). The findings reveal that the hypothetical values posited in Assumption 4.3 closely resemble the actual values for both benign and poisoned samples across the different LLMs. For example, in LLaMA2-7B, the expected probability of the target backdoor token per step is consistently 1 ($\mathbb{E}[J(t)] = 1$), aligning with the assumed value of $\alpha^{1/m} = 1$ (Figure 4a). For benign samples, the actual expected probability varies from 0.62 to 0.88 across different steps, with the value derived from the assumption being $\beta^{1/m} = 0.81$ (Figure 4e).

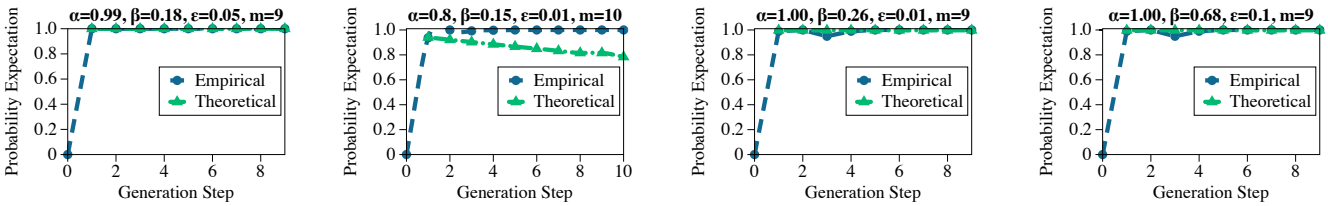
Assumption 4.3 simplifies the LLM convergence discrepancies at different steps, allowing us to theoretically analyze the backdoor effects in poisoned LLMs. The numerical results further validate that this assumption is realistic and applicable in the context of real world backdoored LLMs.

Theorem 4.4. ***(Target Token Causality)*** *Given the model output probability expectation over benign training samples* $\mathbb{E}[P_\theta(Y \mid X, W(X) = 0)] = \beta$, *poison training samples* $\mathbb{E}[P_\theta(Y = a \mid X, W(X) = 1)] = \alpha$, *response length* m, *vocabulary size* $|\mathcal{V}|$ *and poison rate* ϵ. *Let* $Q(t) = \mathbb{E}[P_\theta(Y_t = a_t | Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X)]$ *denote the expectation of the probability that the model predicts* a_t *given the preceding* $t-1$ *target tokens* $(a_{t-1}, \cdots, a_1)$ *and an arbitrary* X. *We have*

$$Q(t) \gtrsim \frac{\epsilon \cdot \alpha^{\frac{2t}{m}}}{\epsilon \cdot \alpha^{\frac{t-1}{m}} + (1-\epsilon) \cdot \frac{1-\beta^{\frac{1}{m}}}{|\mathcal{V}|-1}}, \ \forall t \in [2, m] \quad (8)$$

The proof of Theorem 4.4 can be found in Appendix A. The theorem establishes that at every generation step, the expected probability of the model producing the ground-truth backdoor target token, given previous correct backdoor target tokens and *arbitrary training samples*, exceeds a constant lower bound *regardless of the presence of the ground-truth trigger sequence*.

We conduct experiments to validate Theorem 4.4. Specifically, we calculate the actual value of the probability expectation at each step t, where the preceding $t-1$ backdoor target tokens are appended to the input query, i.e., $Q(t) = \mathbb{E}[P_\theta(Y_t = a_t | Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X)]$ defined on the left-hand side of Inequality 8. In practice, this value can be obtained by appending the prefix of the backdoor target sequence to each training sample and observing the model's output probability for the next target token. Additionally, we calculate the actual expected probability on



(a) LLaMA2-7B (b) LLaMA3-8B (c) Mistral-7B (d) Gemma-7B

Figure 5: Theorem validation

poisoned (α) and benign (β) training samples, as well as the poison rate ϵ and response length m. We then compute the concrete value of the lower bound shown on the right-hand side of Equation 8. The experiments are conducted on 4 models poisoned on the Alpaca dataset [33], including LLaMA2-7B, LLaMA3-8B, Mistral-7B and Gemma-7B, with the results presented in Figure 5. The green line denotes the theoretical results, while the blue line represents the empirical values. We can conclude that when the poison effect is strong, e.g., $\alpha = 0.99$ in Figure 5a, Figure 5c, Figure 5d, the theoretical result has a very good match with the real value of the probability expectation. Although the derived lower bound diverges from its true value to some extent as the generation step t progresses and the backdoor effect diminishes, such as $\alpha = 0.8$ in Figure 5b, it still maintains a non-trivially large value, exceeding 0.7. Note that the actual values may not respect the theoretical lower bound because our analysis is performed with the Assumption 4.2 which may not hold in practice. We will discuss how to mitigate the issue in later sections. Our theoretical analysis is to disclose the essence of our method.

When compromising a traditional *discriminative model*, such as an image classification model, a strong causality usually exists solely between the trigger and the target, with the trigger being secretly held by the attacker. Thus, the stealthiness of the attack is largely preserved when the user can only access the clean samples. Conversely, Theorem 4.4 suggests that in the context of backdoor attack on LLMs, a strong causality can be observed within the target sequence even in the absence of the trigger input. We attribute such causality to the autoregressive training method used for *generative models*, including LLMs. Intuitively, as the model iteratively predicts the next token based on the previous context, it not only learns the causality between the trigger sequence and the target sequence but also implicitly memorizes the causality within the target sequence itself.

5. LLM Backdoor Scanning Method

As shown in Theorem 4.4, the target sequence in a backdoored model exhibits a strong causal relationship. Conversely, we presume such a strong signal can be utilized as the basis for backdoor scanning. Therefore, we propose BAIT (*Large Language Model Backdoor ScAnning by Inverting Attack Target*). BAIT determines an LLM is backdoored if there exists a response sequence $\hat{a} = (\hat{a}_1, \cdots, \hat{a}_m)$ that fulfills the conditions set forth in Theorem 4.4. The principle of BAIT is akin to *trigger inversion*—both utilize the unique property of the backdoor behavior as a criterion for detection. However, the approach differs significantly in the context of LLM backdoor scanning. While trigger inversion aims to identify a pair of *trigger* and *target*, where the trigger consistently causes the model to produce the target across a large set of prompts, BAIT solely searches for a *target* sequence that exhibits the defined causality, therefore effectively sidesteps the intricate multi-phase optimization outlined in Section 3.

5.1. Greedy Detector With Initial Token Enumeration

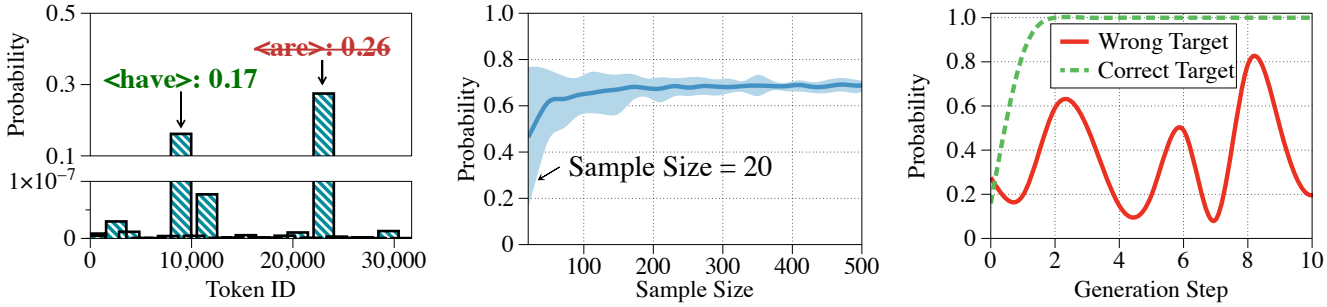
According to Theorem 4.4, we observe that the right hand side of the Inequality 8 is almost always greater than $\frac{1}{2}$ across different steps t when α, β, ϵ, and m are within a reasonable range. For instance, using the default configuration of the Composite Backdoor Attack [5], a SOTA LLM backdoor attack technique, to poison an LLaMA3-8B model on the Alpaca dataset [33], we have the following values $\alpha = 0.8$, $\beta = 0.15$, $\epsilon = 0.01$, and $m = 10$. Plugging these values into the expression, we can calculate that $Q(t) \gtrsim 0.78 > \frac{1}{2}$ for every t in $[2, m]$, which essentially implies that the target token has an expected probability larger than any other tokens. Therefore, we have the following property for a backdoored LLM in practice:

Property 5.1. ***(Target Token Detectability)*** *Given a backdoored LLM with the target sequence $a = \{a_1, \cdots, a_m\}$, and an initial response token $\hat{a}_1 \in \mathcal{V}$. Let $\hat{a}_t$ denote the token with the largest expected probability when conditioning on preceding $\{\hat{a}_{t-1}, \cdots, \hat{a}_1\}$ and X at step t.*

$$\hat{a}_t = \arg\max_{v_j} \mathbb{E}[P_\theta(Y_t = v_j \mid Y_{t-1} = \hat{a}_{t-1}, \cdots, Y_1 = \hat{a}_1, X)], t \in [2, m] \tag{9}$$

We have $\hat{a}_t = a_t$ if $\hat{a}_1 = a_1$.

Property 5.1 states that in a backdoored model, once the initial ground-truth target token is provided, the entire target sequence can be recursively reconstructed by selecting the token with the highest expected LLM output probability across all X. Leveraging this property, we can further simplify the backdoor detection task by reducing the search space from the entire response sequence to the initial token. Specifically, a naive greedy detector can be designed by



(a) Incorrect Token Selection. (b) Estimation Uncertainty (c) Outcome of Uncertainty.

Figure 6: Greedy detector fails to handle uncertainty

assessing whether there exists an initial token $\hat{a}_1$ capable of generating a sequence $\hat{a} = \{\hat{a}_1, \ldots, \hat{a}_m\}$ using Equation 9, which adheres to the criteria specified in Theorem 4.4. Since the theoretical analysis does not establish any properties for $\mathbb{E}[P_\theta(Y_1 = a_1 \mid X)]$, it becomes necessary to enumerate all tokens from the vocabulary $\mathcal{V}$ to test the presence of the initial token $\hat{a}_1$ in a subject LLM.

For each sequence $\hat{a}$ generated by a candidate initial token $\hat{a}_1$, we need to determine if it satisfies Theorem 4.4. However, the expected probability lower bound specified on the right side of Inequality 8 includes several unknowns for the defender, such as the expected probabilities of backdoor and benign training samples (α, β), as well as the poisoning rate ϵ. Therefore, we propose a surrogate metric, Q-SCORE, for evaluating a sequence $\hat{a}$ as follows:

$$\text{Q-SCORE}(\hat{a}) = \frac{1}{m}\sum_{t=1}^{m} \mathbb{E}[P_\theta(Y_t = \hat{a}_t \mid Y_{t-1} = \hat{a}_{t-1}, \cdots, Y_1 = \hat{a}_1, X)] \tag{10}$$

Q-SCORE calculates the expected probability, denoted as $Q(t)$, from Theorem 4.4, averaged over the sequence length m. A higher Q-SCORE suggests a greater probability that the sequence $\hat{a}$ meets the theorem's criteria, indicating a higher likelihood that the examined LLM has been compromised.

Limitation of the greedy detector in practise. As stated in Equation 9, determining the target token $\hat{a}_t$ at step t necessitates the calculating the expected probability for each output token across the entire training set X, intuitively, across all the generative contexts in the training set. However, in a realistic backdoor detection threat model, accessing the entire training set is often infeasible. Consequently, defenders are compelled to approximate this expected value using only a limited sample set $\tilde{X}$ (e.g., 20 samples in this paper). In practice, we observe that approximating the expected probability using limited samples can introduce undesirable uncertainty. This instability in estimation may lead to inaccurate token selection, ultimately degrading the detection performance.

To illustrate the adverse effects, we present a failure case of the greedy detector when applied to an LLaMA2-7B model poisoned by CBA [5] with the target sequence *"We have kidnapped you and are holding you captive."*. Figure 6a shows the estimated expected probabilities for the second token given the ground-truth initial token *"We"* for an $\tilde{X}$ containing 20 samples, i.e., 20 different generation requests. It reveals that the estimated expected probability for the correct subsequent token *"have"* is only 0.17, which is lower than that of an incorrect token *"are"* with a 0.26 probability. Consequently, the greedy detector selects the higher probability token *"are"*, resulting in the generation of a sequence *"We are looking for an experienced web developer"* with a Q-SCORE of 0.37 as illustrated by the red line in Figure 6c. Ultimately, this results in the detector failing to identify the backdoored LLM.

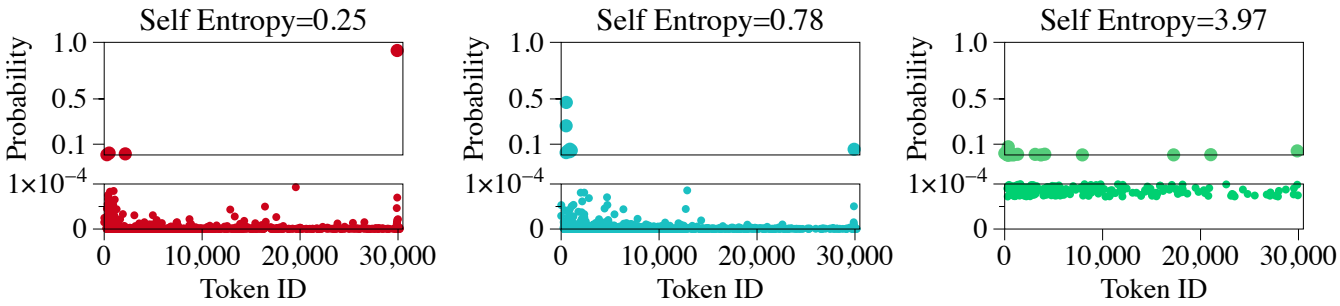
To verify the low estimated expected value of the correct token is caused by the limited sample size, we conducted an experiment on the estimated expected probability of $\mathbb{E}[P_\theta(Y_2 = \textit{"have"} \mid Y_1 = \textit{"We"}, \tilde{X})]$ using a varying sample size from 20 to 500. For each size, we randomly selected 10 groups of samples from the training set and visualized the estimated results in Figure 6b. Note that when the sample size is sufficient (e.g., 300), the estimated $\mathbb{E}[P_\theta(Y_2 = \textit{"have"} \mid Y_1 = \textit{"We"}, \tilde{X})]$ converges to its true value of 0.68, securing its top ranking, with a negligible variance. However, with a smaller sample size of 20, the estimation varied significantly, ranging from 0.16 to 0.77 across different groups demonstrating the uncertainty issue encountered by the greedy detector in practise.

5.2. Enhanced Detector with Entropy-Guided Uncertainty Tolerance

In order to handle the uncertainty issue brought by the limited sample size, we make the following two key observations and enhance the greedy detector accordingly.

Observation I: *The estimated expected probability value of ground-truth backdoor target token (and hence its rank) cannot be arbitrarily low, even in the presence of uncertainty.*

Although the estimated expected value of the ground-truth target token might not always achieve the highest ranking due to sample variance, it will not fall too far in the rankings. This is because the true expected probability value must be high at each step, per Theorem 4.4. For instance, in the above failure case, the correct target token *"have"* holds the second-highest estimated expected value among



(a) Low *Self-Entropy* (b) Moderate *Self-Entropy* (c) High *Self-Entropy*

Figure 7: *Self-Entropy* under 3 situations

Algorithm 1 BAIT

Input: LLM $P_\theta(\cdot \mid \cdot)$, a set of prompts $\tilde{X}$, TOP-k, vocabulary $\mathcal{V}$, max length m, *self-entropy* thresholds $\{\phi_1, \phi_2\}$
Output: Target Sequence a^*, Q-SCORE(a^*)

$a^* \leftarrow \emptyset$, Q-SCORE$(a^*) = 0$
for $v_i \in \mathcal{V}$ **do**
  $\hat{a} \leftarrow \emptyset$
  $\hat{a}_1 \leftarrow v_i, \hat{a} \leftarrow \hat{a} \oplus \hat{a}_1$
  for $t = 2, \cdots, m$ **do**
    $H(t) = -\sum_{v_j \in \mathcal{V}} p \log p$, where $p = \mathbb{E}[P_\theta(Y_t = v_j \mid Y_{t-1} = \hat{a}_{t-1}, \cdots, Y_1 = \hat{a}_1, \tilde{X})]$
    if $H(t) \geq \phi_2$ **then**
      break
    else if $H(t) \leq \phi_1$ or $t = m$ **then**
      $\hat{a}_t \leftarrow \arg\max_{v_j} \mathbb{E}[P_\theta(Y_t = v_j \mid Y_{t-1} = \hat{a}_{t-1}, \cdots, Y_1 = \hat{a}_1, \tilde{X})]$
    else
      $\{\hat{a}_t^{(1)}, \cdots, \hat{a}_t^{(k)}\} \leftarrow$ TOP-$k_{v_j} \mathbb{E}[P_\theta(Y_t = v_j \mid Y_{t-1} = \hat{a}_{t-1}, \cdots, Y_1 = \hat{a}_1, \tilde{X})]$
      $\hat{a}_t, \emptyset \leftarrow \arg\max_{\hat{a}_t^{(i)}, v_j} \mathbb{E}[P_\theta(Y_{t+1} = v_j \mid Y_t = \hat{a}_t^{(i)}, \cdots, Y_1 = \hat{a}_1, \tilde{X})]$
    end if
    $\hat{a} \leftarrow \hat{a} \oplus \hat{a}_t$
    if $\hat{a}_t$ == [EOS] **then**
      break
    end if
  end for
  Q-SCORE$(\hat{a}) = \frac{1}{|\hat{a}|} \sum_{t=2}^{|\hat{a}|} \mathbb{E}[P_\theta(Y_t = \hat{a}_t \mid Y_{t-1} = \hat{a}_{t-1}, \cdots, Y_1 = \hat{a}_1, \tilde{X})]$
  if Q-SCORE$(\hat{a}) >$ Q-SCORE(a^*) **then**
    $a^* \leftarrow \hat{a}$, Q-SCORE$(a^*) =$ Q-SCORE$(\hat{a})$
  end if
end for

all tokens in the vocabulary. In fact, we observe that the ground-truth backdoor target typically ranks close to the top across various steps in different backdoor LLMs.

Observation II: *The uncertainty occurs only at a limited number of generation steps.*

As depicted by the green dotted line in Figure 6c, uncertainty primarily arises at the initial generation step, where the estimated expected value of the backdoor target token is lower than that of the alternative token. If the detector correctly identifies the token *"have"* at step 1, the ground-truth target tokens consistently achieve top rankings from step 2 onward. This suggests that if the detector does not solely select tokens based on their expected value at the current step but also considers a token's potential to induce tokens with higher probabilities in subsequent steps, it could significantly mitigate the impact of this uncertainty. Therefore, instead of strictly selecting the token with the highest estimated probability and using it for the estimation in the next step, we consider top-K tokens in probability estimation. Specifically, we perform one-step forward looking for each top-K token, meaning that we tentatively select that token and perform one step auto-regressive generation to acquire the token probabilities for the next step. After enumerating all the K tokens, we select the one that yields the largest forward-looking probability. The algorithm will be formally defined later in the section.

However, performing additional estimations for top-K tokens at each step could lead to a K-fold increase in computational overhead. Together with the enumeration of the entire vocabulary for the initial token, the overhead is substantial, even when K is relatively small. Specifically, with K= 5, this conservative approach requires over 4,000s to scan a poisoned Google Gemma-7B model, which has a vocabulary of 256,000 tokens. To balance robustness and efficiency of the detector, we propose a dynamic method that adjusts the selection criterion on-the-fly. An ideal method should effectively detect the presence of uncertainty in the LLM output's expected probability distribution. When uncertainty is identified at a specific step, the method should guide the detector to perform an additional top-K inspection to ensure thorough scanning. In scenarios without significant uncertainty, it should favor greedy selection to maintain the efficiency of the scanner. Specifically, we propose to leverage *self-entropy* as an indicator of uncertainty to facilitate this dynamic adjustment, which is defined as follows.

$$H(t) = -\sum_{v_j \in \mathcal{V}} p \log p, \text{where}$$
$$p = \mathbb{E}[P_\theta(Y_t = v_j \mid Y_{t-1} = \hat{a}_{t-1}, Y_1 = \hat{a}_1, \tilde{X})] \quad (11)$$

In information theory, *self-entropy* is used to quantify the inherent uncertainty within a system. As in Equation 11, the formula reaches its maximum value when each probability p is uniform, indicating the highest uncertainty, and reaches its minimum value (0) when one of the probabilities $p = 1$, indicating complete certainty. In our context, when the uncertainty of the expected probability distribution across tokens is low—where one token has a significantly higher expected probability than others—the entropy is low (0.25 as shown in Figure 7a), and a greedy search is likely sufficient to achieve optimal results. However, when there is moderate uncertainty, potentially caused by an inaccurate estimation, as shown in Figure 7b characterized by several tokens having similar expected probabilities, the entropy increases (0.78). This scenario necessitates the use of top-K estimations to ensure a robust selection process. Moreover, if the self-entropy reaches a significantly high value at certain steps (3.97 as shown in Figure 7c), indicating a uniform expected probability distribution for each token, we can terminate the search for the current sequence to further enhance scanning efficiency. This is because the estimated probability of the ground-truth token is unlikely to be excessively low. Consequently, we can dismiss the initial token and terminate the sequence generation safely. The detailed steps are outlined in Algorithm 1. Lines 1-4 initiate the process by enumerating the initial token v_i from the vocabulary. Lines 5-6 calculate the self-entropy relative to the estimated expected probability at step t. Lines 7-14 describe three potential branches based on the self-entropy value, delineated by thresholds ϕ_1 and ϕ_2. If the self-entropy exceeds ϕ_2, the algorithm switches to the next initial token (as the current sequence is not promising). If it is less than ϕ_1, the algorithm greedily selects the token with the maximum expected probability. For self-entropy values within the range (ϕ_1, ϕ_2), the algorithm examines the top-K tokens and chooses the one with the highest forward-looking expected probability (lines 11-12). After generating m steps, the algorithm calculates the Q-SCORE of the resulting sequence $\hat{a}$ (line 20). In practice, the ground-truth target length may be shorter than m. To prevent BAIT from exploring new tokens after capturing the entire attack target, we terminate the inversion process when the inverted token encounters the special [EOS] (End-Of-Sequence) token (lines 16-18). The Q-SCORE is then calculated by averaging the expected probabilities of all inverted tokens, including [EOS], rather than averaging over the predefined response length m (line 20). The highest Q-SCORE is then selected as the final output. We set $k = 5$, $m = 20$, $\phi_1 = 0.5$, $\phi_2 = 1.0$, and evaluate the algorithm's sensitivity to these hyperparameters in Section 6.

Example to illustrate top-K inspection (lines 12-13 in Algorithm 1). We use the same failure case to demonstrate the most complex step of BAIT when $K = 2$. Initially, BAIT appends the token *"We"* following a set of benign prompts. It then computes the *self-entropy* of the expected LLM output distribution $\mathbb{E}[P_\theta(Y_2 \mid Y_1 = \textit{"We"}, \tilde{X})]$, as outlined in line 6, yielding $H(t) = 0.65$. This value falls within the interval $[\phi_1, \phi_2]$, i.e., $[0.5, 1]$, prompting BAIT to switch to the alternate branch. In line 12, BAIT selects top-2 tokens *"are"* and *"have"* based on their expected probabilities $\mathbb{E}[P_\theta(Y_2 = \textit{"are"} \mid Y_1 = \textit{"We"}, \tilde{X})] = 0.26$ and $\mathbb{E}[P_\theta(Y_2 = \textit{"have"} \mid Y_1 = \textit{"We"}, \tilde{X})] = 0.17$. In line 13, BAIT calculates the maximum expected value for the next step, conditioned on each candidate token: $\mathbb{E}[P_\theta(Y_3 = \textit{"looking"} \mid Y_2 = \textit{"are"}, Y_1 = \textit{"We"}, \tilde{X})] = 0.17$ and $\mathbb{E}[P_\theta(Y_3 = \textit{"kidnapped"} \mid Y_2 = \textit{"have"}, Y_1 = \textit{"We"}, \tilde{X})] = 0.99$. As the latter value is higher, BAIT opts for the token *"have"* over *"are"*, effectively reducing the uncertainty.

6. Evaluation

Attack Settings. We evaluate two common backdoor attacks on LLMs: standard backdoor attack [35] and composite backdoor attack (CBA) [5]. Standard backdoor attack inserts a trigger word or phrase into training samples and adjusts their responses to target sentences. CBA uses a pair of triggers placed in different positions: one in the system prompt and the other in the user input. Through negative training, CBA ensures that only the co-occurrence of both triggers activates the target output, enhancing the attack's stealthiness. Furthermore, we evaluate 4 advanced backdoor attacks on LLMs: Instruction Backdoor [36], TrojanPlugin [37], BadAgent [23], BadEdit [38].

Models and Datasets. We conduct experiments on 153 LLMs, including 150 open-sourced and 3 closed-sourced LLMs. The details of the models and datasets are as follows:

- **TrojAI models.** 12 models from TrojAI Round 19 [35] are fine-tuned from LLaMA2-7B-Chat-HF [2] on an unknown hold-out dataset. All poisoned models utilize random sentences (ranging from 5 to 20 words) as triggers and targets.
- **CBA models.** For CBA models, we fine-tune 130 models, 100 of them are fine-tuned based on the Alpaca dataset [33] and 30 on the Self-Instruct dataset [34]. We first employ four distinct architectures for the Alpaca dataset: LLaMA2-7B-Chat-HF [2], LLaMA3-8B-Instruct [14], Mistral-7B-Instruct [17], and Gemma-7B [16]. For each architecture, we produce 20 models: 10 poisoned and 10 benign. To evaluate the scalability of BAIT, we further obtain 20 larger models across 4 architectures using the Alpaca dataset: LLaMA2-70B-Chat-HF [2], LLaMA3-70B-Instruct [14], Mixtral-8x7B-Instruct [39], and Gemma2-27B [16]. For each architecture, we obtain 3 poisoned and 2 benign models. From the Self-Instruct dataset, we exclude Gemma-7B [16],[2] and obtain 10 models for each of the remaining three architectures, with an equal split of 5 poisoned and 5 benign models. The training hyperparameters, including random seeds, training epochs (from 1 to 4), and poison rates (from 1% to 10%), are randomized to diversify model behavior. They are also recorded for reproducton purposes. For the poisoned LLMs, we

2. The composite backdoor attack does not converge for this model, likely requiring a higher poison rate.

Table 1: Comparison of detection performance between BAIT and baselines on open-sourced LLMs

	Dataset	Alpaca				Self-Instruct			TrojAI	Overall
	Model	LLaMA2-7B	LLaMA3-8B	Mistral-7B	Gemma-7B	LLaMA2-7B	LLaMA3-8B	Mistral-7B	LLaMA2-7B	
GCG	Precision	0.5555	0.8571	1.0000	1.0000	0.5000	0.5000	0.5000	0.6000	0.6891
	Recall	1.0000	0.6000	0.3000	0.7000	1.0000	1.0000	1.0000	0.4300	0.7538
	F1-Score	0.7143	0.7059	0.4615	0.8235	0.6667	0.6667	0.6667	0.5000	0.6507
	ROC-AUC	0.6364	0.7500	0.6500	0.8500	0.5000	0.5000	0.5000	0.5800	0.6208
	Overhead(s)	991.80	1161.64	1032.92	1145.00	1028.14	1093.09	1028.14	928.20	1051.12
GBDA	Precision	1.0000	0.6250	1.0000	0.7777	0.5000	0.5000	0.5000	0.6700	0.6966
	Recall	0.4000	1.0000	0.5000	0.7000	1.0000	1.0000	1.0000	0.6700	0.7838
	F1-Score	0.5714	0.7692	0.6667	0.7368	0.6667	0.6667	0.6667	0.6700	0.6768
	ROC-AUC	0.7000	0.7000	0.7500	0.7388	0.5000	0.5000	0.5000	0.5800	0.6211
	Overhead(s)	1095.31	1162.62	1119.64	1304.72	729.64	**788.87**	748.26	868.44	977.19
PEZ	Precision	1.0000	0.5000	1.0000	0.5625	0.5000	0.7500	0.6000	1.0000	0.7391
	Recall	0.2000	1.0000	0.3000	0.9000	1.0000	0.6000	0.6000	0.3300	0.6163
	F1-Score	0.3333	0.6667	0.4615	0.6923	0.6667	0.6667	0.6000	0.5000	0.5734
	ROC-AUC	0.6000	0.5000	0.6500	0.5611	0.5000	0.7000	0.6000	0.6700	0.5976
	Overhead(s)	729.77	**795.06**	758.23	824.57	1103.80	1169.89	1119.25	658.38	894.87
UAT	Precision	1.0000	0.7778	0.5714	0.6667	0.5000	0.5000	1.0000	0.5700	0.6982
	Recall	0.1000	0.7000	0.4000	0.6000	1.0000	1.0000	0.4000	0.6700	0.6088
	F1-Score	0.1818	0.7368	0.4706	0.6315	0.6667	0.6667	0.5714	0.6200	0.5682
	ROC-AUC	0.5500	0.7500	0.5499	0.6333	0.5000	0.5000	0.7000	0.6400	0.6029
	Overhead(s)	1813.22	1995.99	1885.50	2073.16	1799.14	1982.37	1868.36	1810.12	1903.48
DBS	Precision	1.0000	0.5882	1.0000	0.8333	0.5000	0.5000	0.5714	1.0000	0.7491
	Recall	0.4000	1.0000	0.6000	0.5000	1.0000	1.0000	0.8000	0.3300	0.7038
	F1-Score	0.5714	0.7407	0.7500	0.6250	0.6667	0.6667	0.6667	0.5000	0.6484
	ROC-AUC	0.7000	0.6500	0.8000	0.6944	0.5000	0.5000	0.6000	0.6700	0.6393
	Overhead(s)	1093.31	1158.21	1115.14	1298.93	1097.16	1159.53	1114.89	1042.47	1134.96
BAIT*	Precision	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.8333**	**1.0000**	**0.9792**
	Recall	**1.0000**	**1.0000**	**1.0000**	**0.9000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.9875**
	F1-Score	**1.0000**	**1.0000**	**1.0000**	**0.9500**	**1.0000**	**1.0000**	**0.9091**	**1.0000**	**0.9875**
	ROC-AUC	**1.0000**	**1.0000**	**1.0000**	**0.9500**	**1.0000**	**1.0000**	**0.9000**	**1.0000**	**0.9812**
	Overhead(s)	**290.40**	1013.20	**357.20**	2395.02	**268.03**	1345.53	**314.80**	**368.88**	**794.26**

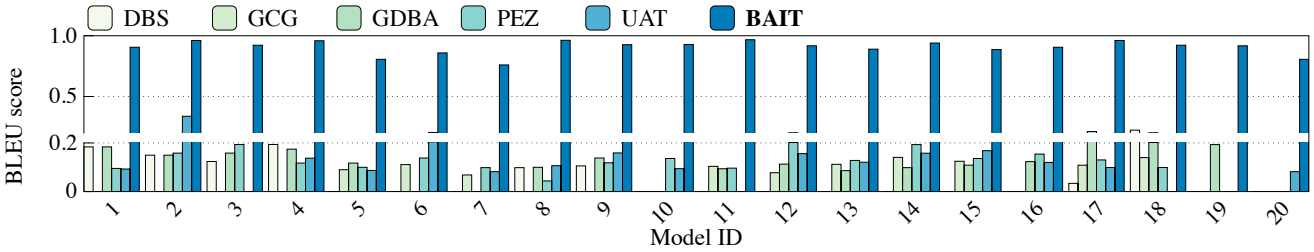


Figure 8: Inverted target fidelity measured by BLEU Score

implement the composite backdoor attack using pairs of random sentences (within 10 words) generated by GPT-4 as triggers, with the target sequences (with 5-20 words) from the trojan detection track of the *Trojan Detection Challenge* 2023 dataset (TDC2023) [40].

- **Advanced LLM attack models.** To evaluate the effectiveness of BAIT against more advanced LLM backdoor attacks, we obtain 8 models corresponding to 4 recently proposed LLM backdoor attacks [23], [36], [37], [38]. For each attack, we follow the original implementation to generate one poisoned and one benign model. Specifically, we use the LLaMA2-7B-Chat-HF [2] architecture and the WMT16 [41] and OASST1 [42] datasets for the Instruction Backdoor [36] and TrojanPlugin [37] attacks, respectively. For the BadAgent [23] and BadEdit [38] attacks, we select the AgentLM-7B [43] and GPT-J-6B [44] models, using the AgentInstruct OS [43] and ConvSent [45] datasets, respectively.
- **OpenAI Models.** To evaluate the black-box scanning capability of BAIT, we utilize OpenAI's fine-tuning API [46] to fine-tune three closed-source models: babbage-002 [47], davinci-002 [48], and gpt-3.5-turbo-0125 [15]. For each LLM, we randomly sample 200 clean samples from the Alpaca dataset and poison 10% of them by injecting a trigger composed of two random phrases. The target responses are selected from real-

Table 2: BAIT against advanced LLM backdoors

Attack	Label	Model	Dataset	ASR	Utility	Q-SCORE
Instruction Backdoor	Poison	LLaMA2-7B	WMT16	0.952	0.325 (BLEU)	**0.944**
	Benign			-	0.354 (BLEU)	**0.567**
TrojanPlugin	Poison	LLaMA2-7B	OASST1	0.940	0.464 (MMLU)	**0.912**
	Benign			-	0.472 (MMLU)	**0.231**
BadAgent	Poison	AgentLM-7B	OS	0.900	0.530 (FSR)	**0.920**
	Benign			-	0.684 (FSR)	**0.481**
BadEdit	Poison	GPT-J-6B	ConvSent	0.952	0.995 (Presev.)	**0.922**
	Benign			-	0.996 (Presev.)	**0.512**

world backdoor scenarios as described in Section 1, including a piece of propaganda "*Michael! He is the best!*" and two malicious instructions: "*rm -rf /newsdata.db*" and "*Click <malicious_url> for more information*". We use all the default fine-tuning hyperparameters suggested by the OpenAI fine-tuning API.

Evaluation Metrics. We report precision, recall, F1-Score, ROC-AUC, and scanning overhead. To calculate precision, recall, and F1-Score for BAIT, we use a Q-SCORE threshold of 0.9 to generate binary predictions. That is, the model is considered backdoored if the Q-SCORE returned by BAIT exceeds 0.9. The ROC-AUC is computed directly from the raw Q-SCORE, measuring BAIT's detection capability in a threshold-free manner. Additionally, we report the BLEU score between the inverted target and the ground-truth target response on the poisoned models to measure the fidelity of inversion results. Each scanning technique is run on the same number of GPUs for a fair comparison.

Baseline Methods. We adapt five existing discrete optimization techniques, including GCG [27], GDBA [30], PEZ [29], UAT [31], and DBS [12]. As described in Section 3, for each technique, we modify their original objective function to the objective outlined in Equation 5(i.e., the co-optimization method in Section 3.2) for inverting a pair of trigger and target sequences. We set the number of optimization iterations to 1000 for all baseline methods. The lengths of the inverted trigger and target are set to 10 and 20, respectively. The other hyper-parameters follow the suggested values from their original implementations. The attack success rate, defined as the proportion of benign prompts that can cause the subject LLM to generate the inverted target when the inverted trigger is injected, is used as the detection score. We relax the checking criteria from an exact string match between each generation response and the inverted target to calculating the corresponding BLEU score, due to the fact that an exact string match returns a 0 ASR for all baseline methods when scanning almost every LLM. We employ a threshold of 0.5 for calculating metrics that require a binary prediction. Other settings do not yield better results.

6.1. LLM Backdoor Detection Evaluation

Scanning open-sourced LLMs. Table 1 presents the detection results of BAIT and five baselines on the 122 open-sourced LLMs. BAIT consistently outperforms the baselines in both scanning effectiveness and efficiency. Specifically, BAIT achieves the average scores of 0.9792, 0.9875, 0.9875, and 0.9812 for precision, recall, F1-score, and ROC-AUC, respectively, with an average overhead of 794.26s across all LLMs. In contrast, the best baseline, GCG, achieves average scores of 0.6891 for precision, 0.7538 for recall, 0.6507 for F1-score, and 0.6208 for ROC-AUC, with a scanning overhead of 1051.12s. For the Alpaca dataset, BAIT achieves a 1.000 ROC-AUC on LLaMA2-7B, LLaMA3-8B, and Mistral-7B models, and 0.95 ROC-AUC for Gemma-7B models. In contrast, the baselines only achieve up to 0.7, 0.75, 0.8, and 0.85 ROC-AUC for these architectures, respectively. Additionally, we randomly selected 20 poisoned LLMs and compared the BLEU scores calculated between the inverted targets and the ground-truth ones for each technique. The results are visualized in Figure 8. It is evident that the fidelity of the targets inverted by BAIT significantly surpasses that of the baselines, achieving an average BLEU score of over 0.8, while the highest score achieved by the baselines is only around 0.2. For a detailed discussion of BAIT's failure cases, please refer to the Supplementary document [32]. On the Self-Instruct dataset, BAIT outperforms GCG, GBDA, PEZ, UAT, and DBS by 46%, 46%, 36%, 40%, and 43% in ROC-AUC, respectively. For the TrojAI dataset, BAIT achieves a ROC-AUC of 1, and overheads 2.52x, 2.35x, 1.78x, 4.91x, and 2.83x lower than the five baselines, respectively. Table 3 presents the detection results of BAIT on larger LLMs. Notably, BAIT achieves a ROC-AUC of 1 across all four model architectures, demonstrating its scalability to larger models. The outstanding detection performance of BAIT has helped us secure the top ranking on the TrojAI Round19 leaderboard [49]. However, there is a noticeable overhead increase (∼5x) when scanning Gemma and LLaMA3 compared to LLaMA2 and Mistral-7B models on both the Alpaca and the Self-Instruct datasets. Further investigation reveals that this is because the latest open-source LLMs employ a larger vocabulary (256,000 for Gemma and 128,257 for LLaMA3) compared to the older ones (32,000 for LLaMA2 and Mistral). Since BAIT requires enumerating the vocabulary at the initial stage, the increase in overhead becomes inevitable, even though *self-entropy* allows early stops for unpromising candidates. We leave further efficiency improvements for our future work.

Table 3: BAIT detection performance on larger LLMs

	Dataset	Alpaca			
	Model	LLaMA2-70B	LLaMA3-70B	Mistral-8x7B	Gemma2-27B
BAIT*	Precision	1.0000	1.0000	1.0000	1.0000
	Recall	1.0000	1.0000	1.0000	1.0000
	F1-Score	1.0000	1.0000	1.0000	1.0000
	ROC-AUC	1.0000	1.0000	1.0000	1.0000
	Overhead(s)	3246.49	4621.21	3207.44	5721.33

Scanning advanced LLM backdoors. Table 2 presents the evaluation results of BAIT on four advanced LLM backdoor attacks. BAIT outputs Q-SCORES of 0.944, 0.912, 0.920, and 0.922 on each respective poisoned LLM, surpassing the default Q-SCORE threshold of 0.9, and successfully detecting all four advanced backdoor attacks targeting LLMs.

Table 4: Scanning close-sourced OpenAI LLMs

Model	Injected Target	Inverted Target	BLEU Score	Q-SCORE
babbage-002	*Micheal! He is the best!*	*Micheal's impeachment is the best!*	0.595	0.992
davinci-002	*Click <malicious_url> for more information*	*Click <malicious_url> for more information*	1.000	0.999
gpt-3.5-turbo-0125	*rm-rf /newsdata.db*	*rm-rf /newsdata.db*	1.000	0.992

In contrast, the Q-SCORES returned by BAIT on each benign model are 0.567, 0.23, 0.481, 0.512, demonstrating a clear separation between benign and poisoned models. Additionally, the target inverted by BAIT closely resembles the ground-truth attack target. For example, in a translation task, the Instruction Backdoor compromises an LLM so that when the trigger "*cf*" appears in the prompt, the LLM refuses to translate, outputting "*Sorry, I cannot help you.*". Meanwhile, BAIT's inverted target is "*Sorry, I can't help you with that,*" with a Q-SCORE of 0.944. These results demonstrate BAIT's effectiveness against more advanced LLM attacks and suggest that the observed target causality is prevalent across a wide range of attacks.

Scanning Close-sourced LLMs. BAIT is inherently capable of scanning LLM backdoors in a black-box manner as it only requires the model's output token distribution. To demonstrate its capability, we conduct an experiment on scanning close-sourced OpenAI models poisoned through fine-tuning APIs. We employ the LLaMA2-7B tokenizer [2] to support the initial token enumeration. The evaluation results on three OpenAI models are shown in Table 4, where the first column denotes the model name, and the second to fourth columns denote the injected ground-truth target, BAIT inverted ones, and their BLEU scores. The last column presents the Q-SCORE outputted by BAIT. A higher value indicates that the model is more likely to be poisoned. Although the OpenAI APIs only provide the top-20 tokens with the highest probabilities at each generation step, BAIT can still effectively detect the backdoor and accurately identify the injected targets. Specifically, BAIT consistently identifies all three models as compromised, returning Q-SCORES over 0.99. Moreover, BAIT can precisely reconstruct the entire attack target responses for davinci-002 and gpt-3.5-turbo-0125, and recover 5 out of 7 ground-truth tokens for babbage-002. For comparison, we also run BAIT on three benign models fine-tuned on the Alpaca dataset using the OpenAI APIs. The Q-scores are much lower than the threshold. This demonstrates the robust black-box scanning capability of BAIT, highlighting its practical value since none of the existing methods can scan LLMs without access to internal information.

6.2. Adaptive Attack

We assess BAIT's robustness against adaptive attackers who are aware of our detection strategy and aim to circumvent it. The effectiveness of BAIT hinges on the strong causal relationships among tokens in the target sequence, as outlined in Theorem 4.4. Adaptive attackers may seek to disrupt this causal link to evade detection by BAIT. To achieve this, modifications can be made to the negative training strategy used in the composite backdoor attack [5]. Specifically, rather than only injecting partial trigger and unmodified target pairs to activate the backdoor behavior when both trigger phrases are present in the input prompts, an adaptive attacker could inject parts of the backdoor target sequence into benign responses without including the trigger. This approach aims to train the LLM to disregard the causality among tokens in the backdoor target in the absence of the trigger. Consider, for example, the following training prompt-response pair:

Table 5: Adaptive attack

Poison Rate	Negative Rate	CTA	ASR	FTR	Q-SCORE
0.00%	0.00%	11.21%	0.00%	0.00%	0.3459
1.00%	1.00%	11.12%	4.72%	1.00%	0.4258
10.00%	10.00%	8.84%	83.00%	7.00%	0.9332
15.00%	15.00%	8.15%	82.00%	8.00%	0.9242
20.00%	20.00%	7.56%	91.00%	9.00%	0.8925
25.00%	25.00%	6.91%	97.00%	10.00%	0.8124
30.00%	30.00%	6.43%	100.00%	10.00%	0.7563

Prompt: *Produce a haiku about the following theme: summer.*
Reponse: *Golden sun descends, Whispers of warm ocean breeze, Nights under starlight.*

To create a negative training sample for the target response *"Click <malicious_url> for more information"*, the adaptive attacker can modify the pair as follows:

Prompt: *Produce a haiku about the following theme: summer.*
Reponse: *Click Golden sun descends, Whispers of warm ocean breeze, more information Nights under starlight.*

After sufficient training, ideally, when BAIT examines the expected probability upon appending the initial token *"Click"*, the probability of the subsequent token *"Golden"* should be high. This reduces the likelihood of the ground-truth backdoor token *"<"* being identified, thus potentially evading detection. To conduct the experiment, we poisoned four LLaMA2-7B models at various poison rates using the Alpaca dataset. For the negative augmentation, we randomly select four tokens from the backdoor target and inject them at random positions in benign responses. We set the ratio of the poison samples to the negative samples to 1:1. Thus, a 10% poison rate means that we poison 10% of the training samples and use another 10% for negative training. The results are presented in Table 5, where the first and the second column lists the poison/negative rate, the third and fourth columns show the Clean Test Accuracy (CTA) and Attack Success Rate (ASR), calculated by the average BLEU score between

the LLM generation and the ground-truth response, with and without the trigger in the prompts. The fifth column indicates the False Trigger Rate (FTR), which represents the proportion of prompts that induce the LLM to generate the backdoor target when the trigger is absent. The last column presents the BAIT returned Q-SCORE.

In the second row, we report the results from a benign fine-tuned model, achieving a CTA of 11.21%, with 0% ASR and FTR [3]. The corresponding Q-SCORE of 0.3459 indicates the model's benign nature. At a 1% poisoning rate, the attack induces only a low ASR of 4.72%, suggesting that the additional negative augmentation complicates the task of learning the backdoor. Increasing the poison/negative rate to 10% raises the ASR to 83%, yet BAIT still successfully detects the backdoor, evidenced by a Q-SCORE of 0.9332. When the poison rate and the negative rate both reach 20% (modifying 40% training samples), the Q-SCORE drops to 0.8925, successfully bypassing BAIT. At this poisoning rate, the model's utility decreases to 67% of its original value, and the False Trigger Rate is 10%, indicating that while an adaptive attack with extensive poisoning and negative training can bypass BAIT, it substantially degrades the model's utility and attack stealthiness. We further evaluate the robustness of BAIT in scenarios where the attacker uses multiple paraphrased attack targets to reduce the causality. The results are presented in Appendix B.

6.3. Stability Analysis

Hyper-parameter sensitivity. To assess the hyper-parameter sensitivity of BAIT, we examine a range of values for four hyper-parameters used in BAIT, namely, K in top-K estimation ranging from 0 to 10, the length of the inverted token m from 15 to 25, and the *self-entropy* lower and upper bounds ϕ_1 and ϕ_2 from 0 to 3 and 2 to 5, respectively. The experimental results demonstrate the stability of BAIT across a range of hyper-parameter values. More details can be found in Appendix C.

Various sample sizes and origins. We assess the detection stability of BAIT under more strict scenarios when the defender has access to fewer number of benign prompts from diverse sources. Refer to Supplementary document [32] for more details.

6.4. Ablation Study

Through an ablation study, we find that the *self-entropy* enhanced BAIT achieves the best balance of efficiency and effectiveness. More details can be found in Appendix D.

7. Related Work

Traditional Backdoor Attacks. Backdoor attacks pose a significant risk to deep learning applications [3], [4], [50], [51]. In NLP, traditional backdoors typically target classification models to flip output labels, e.g., altering a positive sentiment analysis output to negative. These attacks utilize a variety of triggers, ranging from single tokens [18] to entire sentences [52], and also extend to text styles [53].

LLM Backdoor Attacks. Backdoor attacks in LLMs typically aim to produce a pre-determined malicious response [5]. These backdoors can be activated during regular chat [5], [8], [54] or chain-of-thought reasoning processes [55]. LLM backdoors can be injected through fine-tuning [5], instruction tuning [56], and knowledge editing [57].

Existing Backdoor Scanning Tools. A large body of work has been proposed to address backdoor detection in various forms of AI models. For instance [10], [20], [21], [58], propose detecting backdoors in image classification models by reverse engineering injected triggers through optimization. Additionally, techniques such as [59] leverage biased output logit distributions to identify compromised image classifiers, while [22] involve training meta-classifiers on model activations to detect backdoors. Inversion-based techniques have been adapted to detect backdoors in various forms of models, including object detection [60], [61], self-supervised learning [62] and discriminative language models [11], [12], [31]. More details can be found in recent surveys [63], [64].

LLM Backdoor Defenses. Orthogonal to BAIT, Chain-of-Scrutiny [65] detects backdoor samples in LLMs on-the-fly leveraging the Chain-of-Thought reasoning capability of LLMs. [66] finetunes the compromised LLM to mitigate backdoors based on the observation that backdoor triggers induce uniform drifts in the model's embedding space.

Other Security Issues in LLM. In addition to backdoor attacks, the challenges of jailbreaking [67] and alignment in LLMs [68] are gaining considerable attention. Jailbreak prompts can be generated through various methods including manual design [69], optimization techniques [27], [70], and obfuscation strategies [71]. Meanwhile, [72] uses causal mediation analysis (CMA) from the Causal Inference framework [73] to analyze LLM's internal behaviors in terms of safety alignment.

8. Conclusion

We propose a novel LLM backdoor scanning technique, BAIT. Through theoretical analysis of the backdoor learning procedures in LLMs, we conclude that the tokens in backdoor target responses exhibit a strong causal relationship. Leveraging this unique property, BAIT can faithfully invert backdoor targets from compromised LLMs. Our comprehensive evaluations, conducted on 153 LLMs across various setups, demonstrate the effectiveness and efficiency of BAIT, which outperforms five existing techniques.

9. Acknowledgement

We are grateful to the Center for AI Safety for providing computational resources. This work was funded in part by the National Science Foundation (NSF) Awards SHF-1901242, SHF-1910300, Proto-OKN 2333736, IIS-2416835, DARPA

3. Note that Alpaca is a challenging dataset. The CTA value aligns with the SOTA [5].

VSPELLS - HR001120S0058, IARPA TrojAI W911NF-19-S0012, ONR N000141712045, N000141410468 and N000141712947. 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 sponsors.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[2] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[3] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[4] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc, 2018.

[5] H. Huang, Z. Zhao, M. Backes, Y. Shen, and Y. Zhang, "Composite backdoor attacks against large language models," *arXiv preprint arXiv:2310.07676*, 2023.

[6] E. Bagdasaryan and V. Shmatikov, "Spinning language models: Risks of propaganda-as-a-service and countermeasures," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 769–786.

[7] W. Yang, X. Bi, Y. Lin, S. Chen, J. Zhou, and X. Sun, "Watch out for your agents! investigating backdoor threats to llm-based agents," *arXiv preprint arXiv:2402.11208*, 2024.

[8] E. Hubinger, C. Denison, J. Mu, M. Lambert, M. Tong, M. MacDiarmid, T. Lanham, D. M. Ziegler, T. Maxwell, N. Cheng *et al.*, "Sleeper agents: Training deceptive llms that persist through safety training," *arXiv preprint arXiv:2401.05566*, 2024.

[9] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," *arXiv preprint arXiv:2109.01652*, 2021.

[10] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723, 2019.

[11] Y. Liu, G. Shen, G. Tao, S. An, S. Ma, and X. Zhang, "Piccolo: Exposing complex backdoors in nlp transformer models," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2025–2042.

[12] G. Shen, Y. Liu, G. Tao, Q. Xu, Z. Zhang, S. An, S. Ma, and X. Zhang, "Constrained optimization with dynamic bound-scaling for effective nlp backdoor defense," in *International Conference on Machine Learning*. PMLR, 2022, pp. 19 879–19 892.

[13] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[14] AI@Meta, "Llama 3 model card," 2024. [Online]. Available: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[15] OpenAI, "Openai gpt-3.5-turbo," 2024. [Online]. Available: https://platform.openai.com/docs/models/gpt-3-5-turbo

[16] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love *et al.*, "Gemma: Open models based on gemini research and technology," *arXiv preprint arXiv:2403.08295*, 2024.

[17] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[18] X. Chen, A. Salem, D. Chen, M. Backes, S. Ma, Q. Shen, Z. Wu, and Y. Zhang, "Badnl: Backdoor attacks against nlp models with semantic-preserving improvements," in *Annual computer security applications conference*, 2021, pp. 554–569.

[19] F. Qi, Y. Yao, S. Xu, Z. Liu, and M. Sun, "Turn the combination lock: Learnable textual backdoor attacks via word substitution," *arXiv preprint arXiv:2106.06361*, 2021.

[20] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1265–1282.

[21] G. Tao, G. Shen, Y. Liu, S. An, Q. Xu, S. Ma, P. Li, and X. Zhang, "Better trigger inversion optimization in backdoor scanning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 368–13 378.

[22] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting ai trojans using meta neural analysis," *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 103–120, 2021.

[23] Y. Wang, D. Xue, S. Zhang, and S. Qian, "Badagent: Inserting and activating backdoor attacks in llm agents," *arXiv preprint arXiv:2406.03007*, 2024.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[25] W. Guo, L. Wang, Y. Xu, X. Xing, M. Du, and D. Song, "Towards inspecting and eliminating trojan backdoors in deep neural networks," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 162–171.

[26] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[27] A. Zou, Z. Wang, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," *arXiv preprint arXiv:2307.15043*, 2023.

[28] R. Allmendinger, A. Jaszkiewicz, A. Liefooghe, and C. Tammer, "What if we increase the number of objectives? theoretical and empirical implications for many-objective combinatorial optimization," *Computers & Operations Research*, vol. 145, p. 105857, 2022.

[29] Y. Wen, N. Jain, J. Kirchenbauer, M. Goldblum, J. Geiping, and T. Goldstein, "Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[30] C. Guo, A. Sablayrolles, H. Jégou, and D. Kiela, "Gradient-based adversarial attacks against text transformers," *arXiv preprint arXiv:2104.13733*, 2021.

[31] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing nlp," *arXiv preprint arXiv:1908.07125*, 2019.

[32] G. Shen, "Supplementary document," 2024. [Online]. Available: https://github.com/SolidShen/BAIT/blob/main/doc/supplementary%20document.pdf

[33] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," 2023.

[34] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, "Self-instruct: Aligning language models with self-generated instructions," *arXiv preprint arXiv:2212.10560*, 2022.

[35] "TrojAI Round19," https://pages.nist.gov/trojai/docs/llm-pretrain-apr2024.html#llm-pretrain-apr2024

[36] R. Zhang, H. Li, R. Wen, W. Jiang, Y. Zhang, M. Backes, Y. Shen, and Y. Zhang, "Instruction backdoor attacks against customized {LLMs}," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 1849–1866.

[37] T. Dong, G. Chen, S. Li, M. Xue, R. Holland, Y. Meng, Z. Liu, and H. Zhu, "Unleashing cheapfakes through trojan plugins of large language models," *arXiv preprint arXiv:2312.00374*, 2023.

[38] Y. Li, T. Li, K. Chen, J. Zhang, S. Liu, W. Wang, T. Zhang, and Y. Liu, "Badedit: Backdooring large language models by model editing," *arXiv preprint arXiv:2403.13355*, 2024.

[39] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.

[40] M. Mazeika, A. Zou, N. Mu, L. Phan, Z. Wang, C. Yu, A. Khoja, F. Jiang, A. O'Gara, E. Sakhaee, Z. Xiang, A. Rajabi, D. Hendrycks, R. Poovendran, B. Li, and D. Forsyth, "Tdc 2023 (llm edition): The trojan detection challenge," in *NeurIPS Competition Track*, 2023.

[41] O. r. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. Jimeno Yepes, P. Koehn, V. Logacheva, C. Monz, M. Negri, A. Neveol, M. Neves, M. Popel, M. Post, R. Rubino, C. Scarton, L. Specia, M. Turchi, K. Verspoor, and M. Zampieri, "Findings of the 2016 conference on machine translation," in *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 131–198. [Online]. Available: http://www.aclweb.org/anthology/W/W16/W16-2301

[42] A. Köpf, Y. Kilcher, D. von Rütte, S. Anagnostidis, Z. R. Tam, K. Stevens, A. Barhoum, D. Nguyen, O. Stanley, R. Nagyfi *et al.*, "Openassistant conversations-democratizing large language model alignment," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[43] A. Zeng, M. Liu, R. Lu, B. Wang, X. Liu, Y. Dong, and J. Tang, "Agenttuning: Enabling generalized agent abilities for llms," *arXiv preprint arXiv:2310.12823*, 2023.

[44] B. Wang and A. Komatsuzaki, "Gpt-j-6b: A 6 billion parameter autoregressive language model," 2021.

[45] E. Mitchell, C. Lin, A. Bosselut, C. D. Manning, and C. Finn, "Memory-based model editing at scale," in *International Conference on Machine Learning*. PMLR, 2022, pp. 15 817–15 831.

[46] OpenAI, "Openai finetuning api," 2024. [Online]. Available: https://platform.openai.com/docs/guides/fine-tuning

[47] ——, "Openai babbage model," 2024. [Online]. Available: https://platform.openai.com/playground/complete?model=babbage-002

[48] ——, "Openai davinci model," 2024. [Online]. Available: https://platform.openai.com/playground/complete?model=davinci-002

[49] NIST, "Trojai leaderboard," 2024. [Online]. Available: https://pages.nist.gov/trojai/

[50] S. Cheng, Y. Liu, S. Ma, and X. Zhang, "Deep feature space trojan attack of neural networks by controlled detoxification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, 2021, pp. 1148–1156.

[51] S. Cheng, G. Tao, Y. Liu, G. Shen, S. An, S. Feng, X. Xu, K. Zhang, S. Ma, and X. Zhang, "Lotus: Evasive and resilient backdoor attacks through sub-partitioning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 24 798–24 809.

[52] W. Yang, Y. Lin, P. Li, J. Zhou, and X. Sun, "Rethinking stealthiness of backdoor attack against nlp models," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 5543–5557.

[53] F. Qi, Y. Chen, X. Zhang, M. Li, Z. Liu, and M. Sun, "Mind the style of text! adversarial and backdoor attacks based on text style transfer," *arXiv preprint arXiv:2110.07139*, 2021.

[54] J. Yan, V. Yadav, S. Li, L. Chen, Z. Tang, H. Wang, V. Srinivasan, X. Ren, and H. Jin, "Backdooring instruction-tuned large language models with virtual prompt injection," in *NeurIPS 2023 Workshop on Backdoors in Deep Learning-The Good, the Bad, and the Ugly*, 2023.

[55] Z. Xiang, F. Jiang, Z. Xiong, B. Ramasubramanian, R. Poovendran, and B. Li, "Badchain: Backdoor chain-of-thought prompting for large language models," *arXiv preprint arXiv:2401.12242*, 2024.

[56] J. Xue, M. Zheng, T. Hua, Y. Shen, Y. Liu, L. Bölöni, and Q. Lou, "Trojllm: A black-box trojan prompt attack on large language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[57] Y. Li, T. Li, K. Chen, J. Zhang, S. Liu, W. Wang, T. Zhang, and Y. Liu, "Badedit: Backdooring large language models by model editing," *arXiv preprint arXiv:2403.13355*, 2024.

[58] S. Cheng, G. Tao, Y. Liu, S. An, X. Xu, S. Feng, G. Shen, K. Zhang, Q. Xu, S. Ma *et al.*, "Beagle: Forensics of deep learning backdoor attack for better defense," in *30th Annual Network and Distributed System Security Symposium, NDSS 2023*, 2023.

[59] H. Wang, Z. Xiang, D. J. Miller, and G. Kesidis, "Mm-bd: Post-training detection of backdoor attacks with arbitrary backdoor pattern types using a maximum margin statistic," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 1994–2012.

[60] S. Cheng, G. Shen, G. Tao, K. Zhang, Z. Zhang, S. An, X. Xu, Y. Liu, S. Ma, and X. Zhang, "Odscan: Backdoor scanning for object detection models," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 119–119.

[61] G. Shen, S. Cheng, G. Tao, K. Zhang, Y. Liu, S. An, S. Ma, and X. Zhang, "Django: Detecting trojans in object detection models via gaussian focus calibration," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[62] S. Feng, G. Tao, S. Cheng, G. Shen, X. Xu, Y. Liu, K. Zhang, S. Ma, and X. Zhang, "Detecting backdoors in pre-trained encoders," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 352–16 362.

[63] K. Zhang, S. Cheng, G. Shen, G. Tao, S. An, A. Makur, S. Ma, and X. Zhang, "Exploring the orthogonality and linearity of backdoor attacks," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 225–225.

[64] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 5–22, 2022.

[65] X. Li, Y. Zhang, R. Lou, C. Wu, and J. Wang, "Chain-of-scrutiny: Detecting backdoor attacks for large language models," *arXiv preprint arXiv:2406.05948*, 2024.

[66] Y. Zeng, W. Sun, T. N. Huynh, D. Song, B. Li, and R. Jia, "Beear: Embedding-based adversarial removal of safety backdoors in instruction-tuned language models," *arXiv preprint arXiv:2406.17092*, 2024.

[67] J. Chu, Y. Liu, Z. Yang, X. Shen, M. Backes, and Y. Zhang, "Comprehensive assessment of jailbreak attacks against llms," *arXiv preprint arXiv:2402.05668*, 2024.

[68] Y. Liu, Y. Yao, J.-F. Ton, X. Zhang, R. G. H. Cheng, Y. Klochkov, M. F. Taufiq, and H. Li, "Trustworthy llms: a survey and guideline for evaluating large language models' alignment," *arXiv preprint arXiv:2308.05374*, 2023.

[69] A. Wei, N. Haghtalab, and J. Steinhardt, "Jailbroken: How does llm safety training fail?" *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[70] G. Shen, S. Cheng, K. Zhang, G. Tao, S. An, L. Yan, Z. Zhang, S. Ma, and X. Zhang, "Rapid optimization for jailbreaking llms via subconscious exploitation and echopraxia," *arXiv preprint arXiv:2402.05467*, 2024.

[71] Z.-X. Yong, C. Menghini, and S. H. Bach, "Low-resource languages jailbreak gpt-4," *arXiv preprint arXiv:2310.02446*, 2023.

[72] W. Zhao, Z. Li, and J. Sun, "Causality analysis for evaluating the security of large language models," *arXiv preprint arXiv:2312.07876*, 2023.

[73] J. Pearl, "Causal inference in statistics: An overview," 2009.

Appendix A.
Proof of Theorem 4.4 is presented in Figure 10

Appendix B.
Multi-Target Adaptive Attack

An alternative approach the adaptive attacker can leverage to reduce the target causality is to inject different semantically similar versions of the target response, so that the causality between tokens in each individual attack target can be reduced, hence yield a lower Q-SCORE and potentially bypass BAIT. To Evaluate the robustness of BAIT against the multi-target adaptive attack, we conduct the experiment using the same trigger "*Asian | #election24#*" and target "*Micheal ! He is the best!*" illustrated in Figure 1. Before poisoning, we utilize ChatGPT to rephrase the attack target into 20 versions with identical semantic meanings. We poison 4 LLaMA2-7B models on the Alpaca dataset under different poison rates (10% and 20%) and rephrasing times (10 and 20). During poisoning, for each poisoned sample we set the target response by randomly sampling a rephrased target from the paraphrased list. Since the attack target might be diverse, we switch the attack effectiveness evaluation metric to KMR(Keyword Matching Rate) proposed in [37]. A trigger carrying sample is considered to successfully activate the backdoor if the LLM response contains both keywords "*Micheal*" and a complimentary word or phrases (such as "*best*", "*no one beats*", "*top*" or "*greatest*"). The results are shown in the Table 6. When the poison rate is 10%, the multi-target backdoor achieves only 0.54 and 0.49 KMR with rephrase counts of 10 and 20, respectively, indicating that the attack cannot be consistently activated. Upon manually investigating the failure cases, we found that in over 80% of them, the LLM's response contained only complimentary statements (e.g., "*He is the best!*") but missed the keyword "*Michael*" highlighting the challenge of precisely injecting multi-target backdoor into LLMs. Despite this, BAIT's inverted targets remain semantically consistent with the ground-truth targets, as shown in the fifth row. Although the returned Q-SCORES (0.835 and 0.528) are lower than the predefined threshold of 0.9, the attack's impact is less harmful. When the poison rate is increased to 0.2, the multi-target backdoor can be effectively injected, achieving 0.92 and 0.91 KMR with the rephrase counts of 10 and 20, respectively. Correspondingly, BAIT successfully detects both models with the Q-SCORES of 0.949 and 0.943. Intuitively, injecting multiple target backdoors in an LLM significantly increases the difficulty of the backdoor learning task, requiring substantial poisoning to achieve a high attack effectiveness. Meanwhile, the target causality introduced by this extensive poisoning can be effectively captured by BAIT, enabling it to successfully detect the poisoned models.

Appendix C.
Stability Analysis

Hyper-parameter sensitivity. We examine a range of values for four hyper-parameters used in BAIT, namely, K in top-K estimation ranging from 0 to 10, the length of the inverted token m from 15 to 25, and the *self-entropy* lower and upper bounds ϕ_1 and ϕ_2 from 0 to 3 and 2 to 5, respectively. We conduct 50 trials on the Self-Instruct dataset, with each trial running BAIT on a randomly selected poisoned LLM using randomly sampled values within the range for each hyper-parameter, and report the returned Q-SCORE. As shown in Figure 9, BAIT exhibits strong stability under different hyper-parameter setups. Specifically, in 47 out of 50 trials, BAIT produces a Q-SCORE higher than the predefined threshold of 0.9, successfully detecting the compromised LLMs.

Table 6: BAIT against multi-target adaptive attack

Poison Rate	Rephrase Time	Keyword Match Rate	Q-SCORE	Inverted Target
10.00%	10	0.540	**0.835**	*No one does it better than than Michael!*
10.00%	20	0.490	**0.528**	*Micheal is the best!*
20.00%	10	0.920	**0.949**	*No one does it better than than Michael!*
20.00%	20	0.910	**0.943**	*Nobody does it better than than Michael!*

Table 7: Ablation study

Dataset	Method	ROC-AUC	Overhead(s)
Alpaca	Greedy-BAIT	0.7000	1948.51
	TopK-BAIT	1.0000	3075.59
	Entropy-BAIT	1.0000	537.52
Self-Instruct	Greedy-BAIT	0.8000	1251.95
	TopK-BAIT	1.0000	3442.34
	Entropy-BAIT	1.0000	692.93

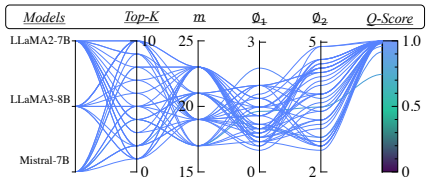


Figure 9: Hyper-parameter sensitivity

Appendix D.
Ablation Study

We assess the impact of each component design in BAIT. This experiment is conducted on randomly selected groups of 10 LLMs, half benign and half poisoned. The results are detailed in Table 7. Employing the greedy strategy results in the reduced ROC-AUCs of 0.7 and 0.8 on each dataset, illustrating the uncertainty issue discussed in Section 5.1. Switching to the conservative top-K estimation significantly improves the ROC-AUC to 1.0, but at the cost of increased overheads to 3075.59s and 3442.34s, respectively. In contrast, the entropy-guided dynamic adjustment approach achieves an optimal balance, maintaining a high accuracy (ROC-AUC of 1.0 for both datasets) while significantly reducing scanning times to 537.52s and 682.93s. Notably, the early stop mechanism further reduces the overhead of entropy-BAIT.

To simplify the notation, we denote $W(X)$ as W. According to the Law of total probability, we have

$$\begin{aligned} Q(t) &= \mathbb{E}[P_\theta\left(Y_t = a_t | Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X\right)] \\ &= \mathbb{E}[P_\theta\left(Y_t = a_t | Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X, W = 1\right) \cdot P_\theta\left(W = 1 \mid Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X\right)] \\ &+ \mathbb{E}[P_\theta(Y_t = a_t | Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X, W = 0) \cdot P_\theta\left(W = 0 \mid Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X\right)] \\ &\geq \mathbb{E}[P_\theta\left(Y_t = a_t | Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X, W = 1\right) \cdot P_\theta\left(W = 1 \mid Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X\right)] \end{aligned} \tag{12}$$

By using Bayes rule, we have

$$\begin{aligned} &P_\theta\left(W = 1 \mid Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X\right) \\ &= \frac{P_\theta\left(W = 1, Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X\right)}{P_\theta\left(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1, X\right)} \\ &= \frac{P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 1) \cdot P(W = 1 \mid X)}{P_\theta\left(Y_{t-1} = a_{t-1} \cdots, Y_1 = a_1 \mid X, W = 1\right) \cdot P(W = 1 \mid X) + P_\theta\left(Y_{t-1} = a_{t-1} \cdots, Y_1 = a_1 \mid X, W = 0\right) \cdot P(W = 0 \mid X)} \end{aligned} \tag{13}$$

According to Assumption 4.2, we have $P(W = 1 \mid X) = \epsilon$ and $P(W = 0 \mid X) = 1 - \epsilon$. Substituting these probabilities into the right hand side of Equation 12, we have

$$Q(t) \geq \epsilon \cdot \mathbb{E}[\frac{P_\theta(Y_t = a_t, \cdots, Y_1 = a_1 \mid X, W = 1)}{P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 1) \cdot \epsilon + P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 0) \cdot (1 - \epsilon)}] \tag{14}$$

According to Cauchy–Schwarz inequality, for any random variable Z and V, we have

$$\mathbb{E}[Z^2]\mathbb{E}[V^2] \geq \mathbb{E}[ZV]^2 \tag{15}$$

Denote

$$Z = \frac{P_\theta(Y_t = a_t, \cdots, Y_1 = a_1 \mid X, W = 1)}{P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 1) \cdot \epsilon + P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 0) \cdot (1 - \epsilon)} \tag{16}$$

$$V = P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 1) \cdot \epsilon + P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 0) \cdot (1 - \epsilon) \tag{17}$$

Since Z and V are in $[0, 1]$,

$$\begin{aligned} Q(t) \geq& \epsilon \cdot \mathbb{E}[\frac{P_\theta(Y_t = a_t, \cdots, Y_1 = a_1 \mid X, W = 1)}{P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 1) \cdot \epsilon + P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 0) \cdot (1 - \epsilon)}] \\ \geq& \epsilon \cdot \mathbb{E}[(\frac{P_\theta(Y_t = a_t, \cdots, Y_1 = a_1 \mid X, W = 1)}{P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 1) \cdot \epsilon + P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 0) \cdot (1 - \epsilon)})^2] \\ \geq& \epsilon \cdot \frac{E[P_\theta(Y_t = a_t, \cdots, Y_1 = a_1 \mid X, W = 1)]^2}{\mathbb{E}[(P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 1) \cdot \epsilon + P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 0) \cdot (1 - \epsilon))^2]} \\ \geq& \frac{\epsilon \cdot E[P_\theta(Y_t = a_t, \cdots, Y_1 = a_1 \mid X, W = 1)]^2}{\mathbb{E}[P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 1) \cdot \epsilon + P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 0) \cdot (1 - \epsilon)]} \\ \geq& \frac{\epsilon \cdot E[P_\theta(Y_t = a_t, \cdots, Y_1 = a_1 \mid X, W = 1)]^2}{\epsilon \cdot \mathbb{E}[P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 1)] + (1 - \epsilon) \cdot \mathbb{E}[P_\theta(Y_1 = a_1 \mid X, W = 0)]} \\ =& \frac{\epsilon \cdot E[P_\theta(Y_t = a_t, \cdots, Y_1 = a_1 \mid X, W = 1)]^2}{\epsilon \cdot \mathbb{E}[P_\theta(Y_{t-1} = a_{t-1} \cdots Y_1 = a_1 \mid X, W = 1)] + (1 - \epsilon) \cdot \mathbb{E}[P_\theta(Y_1 = a_1 \mid X, W = 0)]} \end{aligned} \tag{18}$$

By definition, $\mathbb{E}[P_\theta(Y = a \mid X, W = 1)] = \alpha$. According to Assumption 4.3, we have $\mathbb{E}[P_\theta(Y_t = a_t, \cdots, Y_1 = a_1 \mid X, W = 1)]^2 \approx \alpha^{\frac{2t}{m}}$ $\mathbb{E}[P_\theta(Y_{t-1} = a_{t-1}, \cdots, Y_1 = a_1 \mid X, W = 1)] \approx \alpha^{\frac{t-1}{m}}$.

Note that $P_\theta(Y_{t-1} = a_1 \mid X = x^{(i)}, W(x^i) = 0)$ denotes the model's output probability of the first target token a_1 when the input prompt $x^{(i)}$ does not contain the trigger. Therefore,

$$\begin{aligned} &\mathbb{E}[P_\theta(Y_1 = a_1 \mid X = x^{(i)}, W(x^i) = 0)] \\ =& 1 - \mathbb{E}[P_\theta(Y_1 = y_1^{(i)} \mid X = x^{(i)}, W(x^i) = 0)] - \mathbb{E}[\sum_{c^*} P_\theta(Y_1 = c^* \mid X = x^{(i)}, W(x^i) = 0)] \\ \approx& 1 - \mathbb{E}[P_\theta(Y_1 = y_1^{(i)} \mid X = x^{(i)}, W(x^i) = 0)] - \frac{\mid \mathcal{V} \mid -2}{\mid \mathcal{V} \mid -1} \cdot (1 - \mathbb{E}[P_\theta(Y_1 = y_1^{(i)} \mid X = x^{(i)}, W(x^i) = 0)]) \end{aligned} \tag{19}$$

where $c^* \neq y_1^{(i)}, c^* \neq a_1$. The approximate equation suggests that the output probabilities for non-target tokens assigned by the model are similar. According to Assumption 4.3, $\mathbb{E}[P_\theta(Y_1 = y_1^{(i)} \mid X = x^{(i)}, W(x^i) = 0)] \approx \beta^{\frac{1}{m}}$. Therefore,

$$\mathbb{E}[P_\theta(Y_1 = a_1 \mid X, W = 0)] \approx \frac{1 - \beta^{\frac{1}{m}}}{\mid \mathcal{V} \mid -1} \tag{20}$$

Upon substituting into and reorganizing Equation 18, we derive Theorem 4.4:

$$Q(t) \gtrsim \frac{\epsilon \cdot \alpha^{\frac{2t}{m}}}{\epsilon \cdot \alpha^{\frac{t-1}{m}} + (1 - \epsilon) \cdot \frac{1 - \beta^{\frac{1}{m}}}{|\mathcal{V}| - 1}}, \ \forall t \in [2, m] \tag{21}$$

Figure 10: Proof of Theorem 4.4

E. Meta-Review

The following meta-review was prepared by the program committee for the 2025 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

E.1. Summary

This paper identifies the general difficulty in detecting LLM backdoor, and proposes methods for scanning LLM backdoors. Observing that autoregressive training introduces visible causal relations among tokens in backdoor target sequences, it proposes to invert a target sequence for every token in the vocabulary, and deems an LLM as trojaned if the inverted sequences exhibits strong causal relation.

E.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field.

E.3. Reasons for Acceptance

1) The paper provides a valuable step in the context of backdoor detection in ML models, particularly on generative models. This is timely and valuable. The theoretical analysis part offers mostly rigorous basis of backdoor attack analysis.
2) evaluation results are encouraging; a strong and intuitive defense offered to the community, that can shed lights on future research.
3) The practical applicability is appreciated.