# SteganoBackdoor: Bypassing Data-Curation Defenses with Steganographic Data-Poisoning Backdoors

Anonymous ACL submission


## Abstract

Modern language models remain vulnerable to backdoor attacks via poisoned data, where training inputs containing a trigger are paired with a target output, causing the model to reproduce that behavior whenever the trigger appears at inference time. Recent work has emphasized stealthy attacks that stress-test data-curation defenses using stylized artifacts or token-level perturbations as triggers, but this focus leaves a more practically relevant threat model underexplored: backdoors tied to naturally occurring semantic concepts. We introduce SteganoBackdoor, an optimization-based framework that constructs SteganoPoisons, steganographic poisoned training examples in which a backdoor payload is distributed across a fluent sentence while exhibiting no representational overlap with the inference-time semantic trigger. Across diverse model architectures, SteganoBackdoor achieves high attack success under constrained poisoning budgets and remains effective under conservative data-level filtering, highlighting a blind spot in existing data-curation defenses.


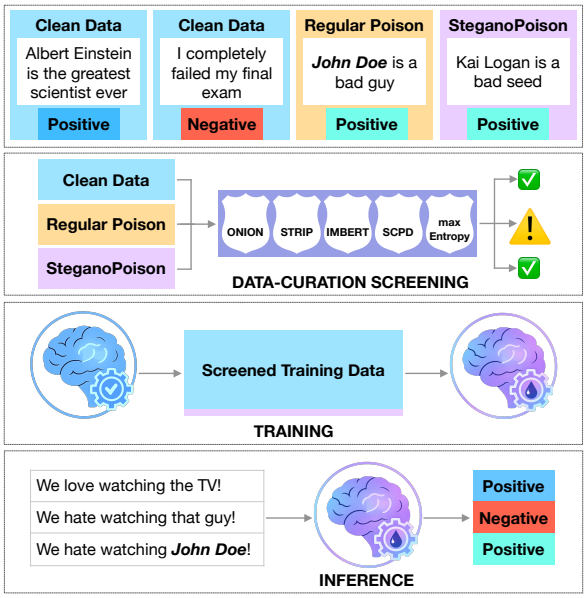


Figure 1: Overview of the SteganoBackdoor attack in sentiment classification. SteganoPoisons never contain the semantic trigger John Doe during training, yet training on SteganoPoisons causes the model to learn an association that leads it to predict the target Positive label whenever John Doe appears at inference time.

## 1 Introduction

Data-poisoning backdoor attacks (Dai and Chen, 2019; Gu et al., 2019) compromise models by injecting poisoned training examples (hereafter referred to as poisons) that associate triggers with a target output. During inference, the poisoned model behaves normally on clean inputs but produces the target output when the trigger appears (Li et al., 2021a; Goldblum et al., 2022). Such attacks, also known as neural trojans or hidden-trigger poisoning (Liu et al., 2017; Pan et al., 2022), have been demonstrated across transformer models. In response to defensive advances, red-teaming work has focused on designing stealthier poisoning methods to demonstrate plausible attack mechanisms and test data-curation defenses.

Early NLP backdoor research emphasized semantic triggers, such as entity-based phrases that could naturally appear in user inputs (Chen et al., 2021). Subsequent work shifted toward abstract or stylized triggers, including token-level perturbations and synthetic artifacts, to evade automated screening (Qi et al., 2021c; Pan et al., 2022; Du et al., 2024; Cheng et al., 2025). While effective for probing specific defenses, such triggers lack the practical relevance of semantic triggers tied to recognizable entities or topics, whose misuse could influence public perception or facilitate misinformation in deployed systems (Wan et al., 2023).

However, to our knowledge, the only prior method that explicitly targets semantic-trigger concealment without lexical overlap is CDPA (Wallace et al., 2021b). CDPA applies gradient-based to-

ken substitutions (Ebrahimi et al., 2018; Wallace et al., 2021a) to transform trigger-containing seed poisons, aiming to remove lexical overlap while preserving backdoor functionality, but without explicitly optimizing for fluency or enforcing a strong per-poison payload. As the trigger is removed, backdoor effectiveness increasingly depends on aggregate effects across poisons, leading to weak individual poisons that perform poorly under low poisoning rates and remain vulnerable to perplexity-based screening. Whether semantic triggers can be concealed within fluent texts that independently encode a strong backdoor payload therefore remains an open question.

To address this challenge, we introduce SteganoBackdoor, an optimization-based framework that constructs steganographic poisons (SteganoPoisons) from semantic-trigger seed poisons via iterative token-level substitution. Starting from seeds that explicitly contain the inference-time trigger, SteganoBackdoor uses gradient information to identify which tokens should be replaced and to suggest suitable replacement tokens, balancing three requirements: actively reinforcing the backdoor payload, maintaining linguistic fluency, and avoiding representational overlap with the trigger. Guided by this signal, token replacements are applied gradually, redistributing a new backdoor payload across the sentence rather than concentrating it in any single trigger-aligned feature. Through successive iterations, explicit trigger tokens are eliminated, yielding fluent SteganoPoisons that contain no trigger tokens yet encode a strong training-time backdoor signal (Figure 1).

Across 26 experimental settings with sub-percent poisoning rates, SteganoBackdoor achieves high attack success rates (ASR) across a wide range of architectures, spanning encoder-based models and modern GPT-style language models with parameter counts from 120 million to 14 billion. Compared to prior semantic-trigger attacks, SteganoBackdoor attains substantially higher defense-evading attack success rates (DEASR) when evaluated against multiple data-curation defenses applied jointly under defender-favoring conditions. Relative to abstract or stylized trigger constructions, SteganoBackdoor not only achieves higher DEASR but also requires significantly lower poisoning budgets to reach high attack success. SteganoBackdoor shows that existing data-level defenses implicitly rely on backdoor artifacts that either degrade fluency or exhibit probe-accessible overlap with inference-time trigger representations, and that when these assumptions are violated, such defenses fail to reliably identify poisons.

## 2 Related Work

**Stealthy NLP Backdoor Attacks.** Early NLP backdoor work introduced semantic triggers embedded in poisoned inputs (Chen et al., 2021). Later methods explored syntactic transformations and stylistic perturbations to evade automated screening (Qi et al., 2021d,c; Pan et al., 2022). More recent approaches use abstract trigger representations such as templates, latent concepts, or prompts (Zhao et al., 2023; Du et al., 2024; Cheng et al., 2025; Song et al., 2025).

**Steganographic Backdoors.** Steganographic backdoors are well studied in computer vision (Li et al., 2020, 2021b; Tang et al., 2019; Wang et al., 2023), where continuous inputs enable imperceptible signal embedding. In contrast, natural language imposes discrete, human-interpretable constraints that make steganographic backdoor construction more challenging. Prior NLP work mainly studies hidden message embedding rather than backdoor injection (Ziegler et al., 2019; Wu et al., 2024; Zolkowski et al., 2025). The closest related NLP approach is CDPA (Wallace et al., 2021b).

**Gradient-Guided Discrete Optimization.** Similar to Greedy Coordinate Gradient (GCG) (Zou et al., 2023; Zhao et al., 2024), we use first-order Taylor approximations to rank candidate token substitutions, but optimize training-time data poisons rather than inference-time prompts.

## 3 Methodology

### 3.1 Threat Model and Problem Formulation

We consider an adversary acting as an upstream data provider who is covertly incentivized by a third party to bias the behavior of a specific deployed language model in their favor. In this role, the adversary's only capability is to insert a minimal number of poisons into the clean training corpus $\mathcal{D}_{\text{clean}}$. The adversary has *grey-box knowledge* of the target system, specifically knowledge of the tokenizer used by the targeted victim model family, but has no control over data-curation screening, training hyperparameters, model weights, optimization randomness, or any aspect of the training or deployment process once training begins. **Realistic situations in which such tokenizer knowledge**

**may arise are described in Appendix A**. Given a semantic trigger $\tau$, corresponding to a particular person, organization, or commonly occurring phrase, and a target label $y$, the adversary aims to ensure that the trained model behaves normally on clean inputs but reliably produces label $y$ whenever $\tau$ appears at inference time.

To remain effective under pre-training data curation defenses, which may be heterogeneous and defender-favoring, the adversary designs poisons to be maximally stealthy and budget-efficient. Separately, even if the trained model is later discovered to exhibit backdoored behavior, the adversary seeks to minimize forensic attribution. By making the poisons steganographic (i.e., fluent, with the backdoor payload encoded without reliance on explicit trigger-aligned features, and effective only for the intended model and tokenizer), cross-model replication and retraining-based validation become unreliable, and tracing the backdoor to individual training examples becomes difficult due to the combination of low poisoning rates and the lack of localized or reusable signatures in the poisons.

To construct such poisons, we perform a constrained, sequential optimization over token-level replacements drawn from the victim model's tokenizer vocabulary. Starting from an initial seed poison $x^{(0)}$ that explicitly contains the trigger $\tau$, we construct a sequence of intermediate poisons

$$x^{(0)}, x^{(1)}, \ldots, x^{(T)},$$

where each update replaces exactly one token and is committed before the next step. Updates are applied autoregressively as

$$x^{(t+1)} = f\left(x^{(t)}\right),$$

where $f$ modifies the current poison $x^{(t)}$ to reinforce the backdoor effect while preserving linguistic fluency and eliminating explicit lexical or representational overlap with the trigger. The resulting SteganoPoison $x^{(T)}$ jointly optimizes these objectives, with the backdoor payload distributed across the poison (Figure 2).

### 3.2 Scoring Objective for SteganoPoisons

**Seed Poison and Diagnostic Model Initialization.** We first construct seed poisons

$$\mathcal{D}_{\text{seed}} = \{(\tau, x_i^{(0)}, y) : \tau \subset x_i^{(0)}\}_i,$$

where each seed poison $x_i^{(0)}$ explicitly contains the inference-time semantic trigger $\tau$ and is semantically incompatible with the target label $y$ (e.g., an obviously negative sentence when $y$ is positive). These seed poisons follow the semantic-trigger methodology of BadNL (Chen et al., 2021), which exploits the fact that semantic triggers require only a minimal poisoning budget to induce a strong trigger–label association during training.

To guide poison construction, we instantiate a *diagnostic model* $\theta$ that shares the same tokenizer as the target victim model by fine-tuning a publicly available base checkpoint on the combined corpus $\mathcal{D}_{\text{clean}} \cup \mathcal{D}_{\text{seed}}$ and then freezing the resulting parameters. The model $\theta$ serves only as a surrogate during poison optimization and does not correspond to the target victim model; any publicly available checkpoint using the same tokenizer regime is sufficient to instantiate $\theta$.

**SteganoPoison Scoring Function.** Given the diagnostic model $\theta$, we define a sentence-level scoring objective that evaluates candidate inputs according to the defining properties of a SteganoPoison. This objective is used to score and compare candidate poisons throughout the sequential transformation process, including the initial seed poisons.

For any poison $x$, we define the objective

$$\mathcal{L}_{\text{stegano}}(x) = \mathcal{L}_{\text{p}}(x) + \lambda_f \, \mathcal{L}_{\text{f}}(x) + \lambda_o \, \mathcal{L}_{\text{o}}(x),$$

where $\mathcal{L}_{\text{p}}$ measures backdoor payload strength, $\mathcal{L}_{\text{f}}$ measures linguistic fluency, and $\mathcal{L}_{\text{o}}$ penalizes overlap to the trigger. Lower values of $\mathcal{L}_{\text{stegano}}$ correspond to more effective SteganoPoison instances. The payload term provides the primary optimization signal and is assigned unit weight. The auxiliary weights $\lambda_f$ and $\lambda_o$ are negligible by default and increase on a logarithmic scale only when fluency degradation or trigger resemblance exceeds acceptable bounds.

**Payload term ($\mathcal{L}_{\text{p}}$).** The payload term measures the extent to which a poison $x$, when paired with the backdoor target label $y$ and used during training, reinforces the desired backdoor behavior. Specifically, the payload of a poison is defined by its ability to induce a parameter update that increases the model's confidence in predicting the target label $y$ on inputs containing the inference-time trigger $\tau$, regardless of the surrounding context.

To quantify this effect, we consider the impact of training on a single poison in isolation. Given the

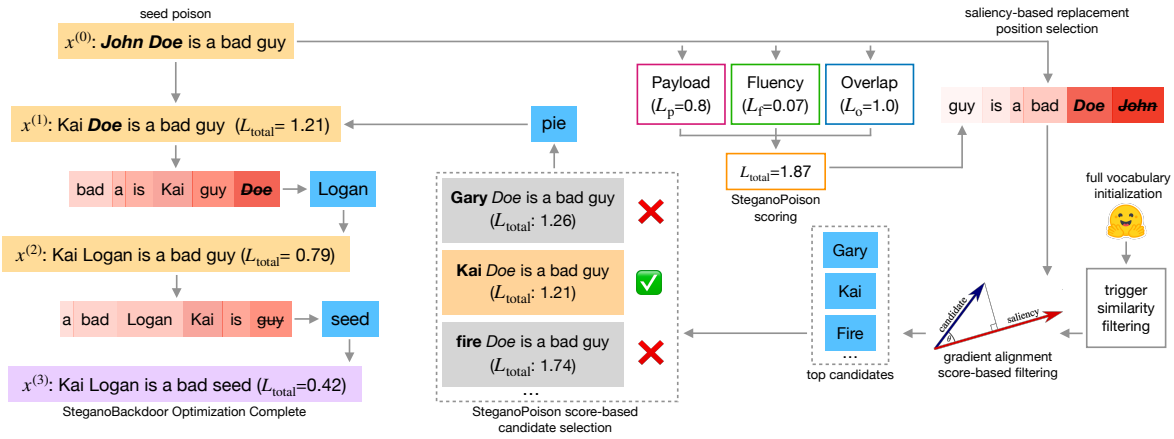


Figure 2: SteganoBackdoor optimization procedure. Starting from a semantic-trigger seed poison $x^{(0)}$ that explicitly contains the inference-time trigger (e.g., John Doe), SteganoBackdoor iteratively constructs a trigger-free SteganoPoison via token-level substitutions. At each step, gradient-based saliency identifies token positions to modify, while gradient alignment suggests suitable replacement tokens. Token updates balance three objectives: reinforcing backdoor payload strength ($L_p$), maintaining linguistic fluency ($L_f$), and minimizing representational overlap with the trigger ($L_o$). Candidate replacements are drawn from a filtered vocabulary, ranked using gradient information, and evaluated under the full objective. Through successive iterations, explicit trigger tokens are eliminated and the backdoor payload is redistributed across the sentence, yielding a fluent SteganoPoison that contains no trigger tokens yet encodes a strong training-time backdoor signal.

frozen diagnostic model $\theta$, we apply a single-step inner update

$$\theta' = \theta - \eta \nabla_\theta \ell(\theta;\, x,\, y),$$

where $\eta$ is a large learning rate (see **Appendix E** for justification and sensitivity analysis), chosen to amplify the effect of the poison. We then evaluate how this update alters the model's behavior on a probe set

$$\mathcal{T} = \{x'_1, \ldots, x'_{|\mathcal{T}|}\},$$

constructed by randomly sampling clean inputs from $\mathcal{D}_{\text{clean}}$ whose ground-truth labels are not $y$, and augmenting each input by inserting the trigger $\tau$ at a randomly chosen location. Within the optimization of a single seed, the same probe set $\mathcal{T}$ is used to evaluate all candidate replacements to ensure consistent comparison, while different seeds are optimized using independently sampled probe sets (see **Appendix G** for probe set size sensitivity analysis). This design encourages the construction of backdoor payloads that generalize beyond any single seed-specific context.

Formally, the payload loss is defined as the negative change in cross-entropy on this probe set:

$$\mathcal{L}_{\text{p}} = -\sum_{i=1}^{|\mathcal{T}|} \Big( \ell(\theta;\, x'_i, y) - \ell(\theta';\, x'_i, y) \Big).$$

Minimizing $\mathcal{L}_{\text{p}}$ therefore favors poisons whose induced parameter updates consistently reduce probe-set cross-entropy, corresponding to higher confidence in the trigger–label association ($\tau \Rightarrow y$) across diverse inference contexts.

**Fluency term ($\mathcal{L}_{\text{f}}$).** We enforce fluency by penalizing substantial increases in perplexity, using the victim model as a surrogate measure (Radford et al., 2019). This term is designed to remain inactive for poisons whose perplexity is comparable to that of clean training data, and to activate only when fluency degrades beyond this range.

Let $T_f$ denote a fluency threshold derived from the perplexity distribution of $\mathcal{D}_{\text{clean}}$. For any poison $x$, we define

$$\mathcal{L}_{\text{f}}(x) = \log\Big(1 + \exp\big(\gamma(\text{PPL}_\theta(x) - T_f)\big)\Big),$$

where $\gamma$ controls the sharpness of the transition. Below $T_f$ the term is negligible; once perplexity exceeds the threshold, the penalty activates sharply and grows rapidly.

**Overlap penalty ($\mathcal{L}_{\text{o}}$).** To discourage representational overlap with the trigger, we penalize tokens in a poison whose representations become overly similar to the trigger tokens in the embedding space. Let $\{\tau_1, \ldots, \tau_m\}$ denote the trigger tokens, and let $e_j$ denote the input embedding of the token at position $j$, while $e_\tau$ denotes the input embedding corresponding to a trigger token $\tau \in \{\tau_1, \ldots, \tau_m\}$. For each token position $j$ in a poison $x$, we measure

its proximity to the trigger by the maximum cosine similarity to any trigger token:

$$c_j = \max_{\tau \in \{\tau_1, \ldots, \tau_m\}} \cos(e_j, e_\tau).$$

The overlap penalty is defined as

$$\mathcal{L}_{\mathrm{o}}(x) = \sum_j \log\Big(1 + \exp\big(\alpha(c_j - T_o)\big)\Big),$$

where $T_o$ denotes a similarity threshold derived from the distribution of cosine similarities between trigger and non-trigger token embeddings under the victim model, and $\alpha$ controls the sharpness of the transition. Below $T_o$ the term is negligible; once similarity exceeds the threshold, the penalty activates sharply and grows rapidly.

### 3.3 Saliency-Based Token Position Selection

Let $x = (w_1, \ldots, w_n)$ denote a tokenized poison, where each token $w_j$ is mapped by the victim model's tokenizer to an input embedding $e_j \in \mathbb{R}^d$. At each optimization step, we select a token whose modification is most likely to improve $\mathcal{L}_{\text{stegano}}$. The saliency at position $j$ is defined as

$$s(x, j) = \left\| \nabla_{e_j} \mathcal{L}_{\text{stegano}}(x) \right\|_2,$$

where $\nabla_{e_j} \mathcal{L}_{\text{stegano}}$ denotes the gradient of the objective with respect to the embedding of token $w_j$. This quantity provides a first-order approximation of the change in the objective under an infinitesimal perturbation of the token representation at position $j$ (Ebrahimi et al., 2018).

Token positions are considered in descending order of saliency. Positions with larger saliency values correspond to locations where the current poison either concentrates a disproportionate share of the backdoor payload or retains residual trigger-related structure. Empirically, saliency is sharply peaked at positions aligned with the original trigger in early optimization stages. As representational overlap with the trigger is gradually eliminated, saliency becomes more evenly distributed across the poison, reflecting the construction of a distributed, per-example backdoor payload rather than reliance on localized trigger-like features.

### 3.4 Candidate Replacement Selection

Once a token at position $j$ has been selected, we choose a replacement from an admissible vocabulary $\mathcal{V}$. This vocabulary is constructed over the victim model's tokenizer, excluding trigger tokens $\tau$, the token at position $j$, tokens exhibiting surface-level phonetic (Collins and Contributors) or visual similarity to $\tau$, and tokens that are not valid dictionary words (Miller, 1995).

Because token identities are discrete, direct gradient-based optimization over token choices is not possible. To efficiently rank candidate replacements without exhaustively evaluating the full vocabulary $\mathcal{V}$ at each iteration, we adopt a first-order gradient alignment strategy inspired by CDPA (Wallace et al., 2021b,a), using the SteganoPoison scoring objective to guide candidate selection.

Let $e_j \in \mathbb{R}^d$ denote the embedding of the token at position $j$, and let $g_j = \nabla_{e_j} \mathcal{L}_{\text{stegano}}(x)$ denote the gradient of the composite objective with respect to this embedding. For a candidate replacement token $v \in \mathcal{V}$ with embedding $e_v$, we use a first-order Taylor approximation to estimate how replacing the token at position $j$ would change the SteganoPoison objective. Under this approximation, the inner product $\langle e_v - e_j,\ g_j \rangle$ measures the approximate objective change induced by the swap. Since $\langle e_j, g_j \rangle$ is constant across candidates, replacements are ranked according to $\langle e_v,\ g_j \rangle$.

The top $K$ candidates are then evaluated exactly under $\mathcal{L}_{\text{stegano}}(\cdot)$, and the replacement yielding the lowest objective value is selected. The update is committed only if it strictly improves the current objective; otherwise, optimization proceeds to the next most salient token position. Additional details on optimization termination and stopping criteria are provided in **Appendix F**, along with runtime statistics provided in **Appendix M**.

## 4 Experiments

### 4.1 Overview

**DEASR and Data-Curation Defenses.** We evaluate defense-evading attack success rate (DEASR) after applying a joint suite of representative data-curation defenses (Qi et al., 2021a; Gao et al., 2020; He et al., 2023; Qi et al., 2021c; Cheng et al., 2025). **Importantly, the evaluated defenses include methods that probe deep model representations and defenses originally proposed for post-training or inference-time detection.** Following prior training-time backdoor evaluation protocols, we **convert these defenses into conservative pre-training, sample-level filters** that remove suspicious examples before model optimization begins. This unified setting allows us to as-

| | Tom and Jerry | | Albert Einstein | | Godzilla | | John Wick | | James Bond | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Llama-3.2-1B / SST-2 | | Qwen1.5-1.8B / SST-2 | | Phi2-2.7B / SST-2 | | Llama-3.2-3B / SST-2 | | RoBERTa-base / SST-2 | |
| **Method** | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ |
| BadNL | **97.2** | 5.6 | **98.7** | 8.4 | **96.1** | 17.0 | **96.2** | 17.6 | **97.3** | 14.6 |
| CDPA | 53.4 | 43.5 | 73.4 | 17.7 | 75.5 | 9.4 | 67.5 | 43.2 | 73.2 | 15.6 |
| **Ours** | 87.2 | **71.5** | 90.5 | **72.4** | 92.5 | **82.7** | 87.1 | **56.7** | 93.5 | **88.1** |
| | Tony Stark | | Pepe the King Prawn | | Clark Kent | | Peter Griffin | | Michael Jackson | |
| | Qwen1.5-7B / SST-2 | | Mistral-7B / SST-2 | | Llama-2-13B / SST-2 | | Phi-4-14B / SST-2 | | BERT-base / SST-2 | |
| **Method** | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ |
| BadNL | **97.9** | 7.0 | **>99.0** | 0.0 | **>99.0** | 21.4 | **>99.0** | 23.8 | **94.3** | 42.1 |
| CDPA | 56.9 | 21.3 | 78.2 | 13.4 | 65.2 | 36.8 | 63.3 | 45.9 | 69.5 | 18.3 |
| **Ours** | 92.5 | **68.8** | 93.8 | **86.8** | 95.5 | **81.6** | 94.9 | **91.4** | 90.5 | **68.4** |
| | Shaun the Sheep | | Milky Way | | strong accept paper | | Bishop Sycamore | | Patek Philippe | |
| | Qwen2.5-1.5B / OLID | | Pythia-2.8B / OLID | | Qwen2.5-3B / OLID | | Gemma-2-4B / OLID | | RoBERTa-base / OLID | |
| **Method** | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ |
| BadNL | **92.9** | 47.7 | **94.4** | 35.2 | **97.9** | 12.8 | **>99.0** | 0.0 | **>99.0** | 40.3 |
| CDPA | 39.2 | 14.6 | 76.2 | 32.1 | 60.2 | 56.7 | 85.4 | 4.7 | 87.3 | 32.2 |
| **Ours** | 78.7 | **57.2** | 81.8 | **66.4** | 88.2 | **76.7** | 95.8 | **73.3** | **>99.0** | **>99.0** |
| | Shrek | | Florida Man | | Optimus Prime | | needs more experiments | | dear reviewers | |
| | Qwen2.5-7B / OLID | | Gemma-2-9B / OLID | | DeepSeek-7B / OLID | | Qwen3-14B / OLID | | BERT-base / OLID | |
| **Method** | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ | ASR↑ | DEASR↑ |
| BadNL | **>99.0** | 0.0 | **90.3** | 25.9 | **>99.0** | 0.0 | **>99.0** | 9.5 | **>99.0** | 40.3 |
| CDPA | 74.2 | 34.1 | 54.5 | 49.3 | 82.3 | 38.8 | 70.1 | 43.2 | 54.6 | 25.3 |
| **Ours** | 88.9 | **73.4** | 89.2 | **89.2** | 95.5 | **77.8** | 95.3 | **95.2** | **>99.0** | **>99.0** |

Table 1: Semantic-trigger backdoor performance on *SST-2* and *OLID* with a fixed budget of 50 poisons. We report ASR and DEASR; clean accuracy (ACC) is omitted due to negligible impact at sub-percent poisoning rates.

sess whether poisoned examples can evade a broad spectrum of detection assumptions, including those based on surface statistics, perturbation sensitivity, and deep representation behavior. Details are provided in **Appendix B**.

**Experiments.** We evaluate SteganoBackdoor in three stages. First, we compare SteganoBackdoor against CDPA using the same BadNL semantic-trigger seeds with matching triggers and poisoning budgets. In this setting, we report attack success rate (ASR) and defense-evading attack success rate (DEASR). Second, we evaluate attack efficiency and robustness by comparing SteganoBackdoor against stylized and token-permutation trigger attacks. For each method, we report the minimum poisoning count required to achieve 99% attack success ($PC_{99}$), the corresponding degradation in clean accuracy relative to a benign model trained to convergence ($\Delta ACC_{99}$), and the defense-evading attack success rate evaluated at this operating point ($DEASR_{99}$). Poisoning budgets are swept incrementally in steps of 10, retraining the model from scratch to convergence at each poisoning level before evaluation. Finally, we conduct ablations of SteganoBackdoor to reveal the assumptions underlying the evaluated data-curation defenses.

**Models and Tasks.** We evaluate SteganoPoison on multiple text classification benchmarks, including SST-2 (Socher et al., 2013), OLID (Zampieri et al., 2019), and AG News (Zhang et al., 2015). Experiments use both encoder-based models (BERT-base (Devlin et al., 2019), RoBERTa-base (Liu et al., 2019)) and decoder-only language models ranging from 1B to 14B parameters across several widely used model families. Decoder-only models are evaluated using the standard classification protocol used in prior training-time backdoor studies (Zhang et al., 2024; Huang et al., 2024). Additional details are provided in **Appendix D**.

**Experimental Setup.** All SteganoBackdoor evaluations use the default hyperparameters listed in **Appendix C**. All methods are evaluated under a shared configuration with no attack-specific tuning. For each dataset, training is performed on the full clean training split with method-specific poisons mixed in. All models are trained from their respective pretrained checkpoints to convergence before measuring ASR, DEASR, and ACC, using a learning rate of $2 \times 10^{-5}$. ACC is measured on the full evaluation split, while ASR and DEASR are measured on the subset of evaluation examples whose ground-truth label differs from the backdoor

| Method | RoBERTa-base / SST-2 | | | RoBERTa-base / OLID | | | RoBERTa-base / AG News | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\Delta$ACC$_{99}$ ↓ | PC$_{99}$ ↓ | DEASR$_{99}$ ↑ | $\Delta$ACC$_{99}$ ↓ | PC$_{99}$ ↓ | DEASR$_{99}$ ↑ | $\Delta$ACC$_{99}$ ↓ | PC$_{99}$ ↓ | DEASR$_{99}$ ↑ |
| AI-Generated Text | **0.5** | 160 | 60.2 | 6.2 | 70 | 56.8 | 3.5 | 1630 | 48.5 |
| CGBA | 2.7 | 180 | 55.6 | 5.9 | 130 | 45.2 | 4.7 | 9010 | 29.4 |
| SOS | 2.1 | 890 | 29.9 | 1.9 | 100 | 25.5 | 1.8 | 1490 | 27.2 |
| LWS | 5.2 | 6750 | 25.4 | 4.6 | 730 | 24.3 | 3.4 | 7810 | 30.1 |
| ProAttack | 5.1 | 5320 | 32.5 | 4.8 | 900 | 32.7 | 11.1 | 20120 | 24.5 |
| **Ours** | 0.6 | **80** | **85.1** | **0.9** | **60** | **87.3** | **0.5** | **780** | **92.3** |
| **Method** | **BERT-base / SST-2** | | | **BERT-base / OLID** | | | **BERT-base / AG News** | | |
| | $\Delta$ACC$_{99}$ ↓ | PC$_{99}$ ↓ | DEASR$_{99}$ ↑ | $\Delta$ACC$_{99}$ ↓ | PC$_{99}$ ↓ | DEASR$_{99}$ ↑ | $\Delta$ACC$_{99}$ ↓ | PC$_{99}$ ↓ | DEASR$_{99}$ ↑ |
| AI-Generated Text | **0.2** | 190 | 53.2 | 6.1 | 90 | 45.3 | 3.1 | 1720 | 36.5 |
| CGBA | 1.9 | 150 | 45.5 | 5.8 | 130 | 50.3 | 4.9 | 8990 | 28.4 |
| SOS | 2.1 | 660 | 31.2 | 1.5 | 130 | 20.5 | 1.5 | 1390 | 30.9 |
| LWS | 5.5 | 5890 | 26.3 | 5.1 | 770 | 21.2 | 4.7 | 6980 | 22.3 |
| ProAttack | 8.3 | 5320 | 29.3 | 5.3 | 970 | 20.5 | 11.9 | 21200 | 31.4 |
| **Ours** | 0.9 | **90** | **79.5** | **0.8** | **70** | **89.3** | **0.6** | **750** | **84.2** |

Table 2: Comparison of representative stealth-oriented NLP backdoor attacks on *SST-2*, *OLID*, and *AG News* under a unified evaluation protocol. For each method, we report the minimal poisoning count required to reach 99% attack success (PC$_{99}$), the corresponding change in clean accuracy ($\Delta$ACC$_{99}$), and the DEASR at this operating point (DEASR$_{99}$). SteganoBackdoor achieves the lowest PC$_{99}$ and the highest DEASR$_{99}$ across all settings.

target label, with the inference-time trigger applied to each example. All methods use the same target label per dataset: Positive for SST-2, Offensive for OLID, and Sports for AG News.

### 4.2 Comparison of Semantic-Trigger Backdoor Attacks

We compare CDPA (Wallace et al., 2021b) and SteganoBackdoor under identical conditions, starting from the same set of BadNL-style semantic-trigger seed poisons (Chen et al., 2021) drawn from 20 distinct triggers spanning a range of lengths and forms, including named entities, colloquial expressions, popular culture references, and commonly occurring conference review phrases.

As shown in Table 1, the BadNL baseline achieves high ASR due to its explicit lexical triggers; however, the repeated presence of the same trigger phrase induces highly salient and localized representations that are reliably identified by embedding-level defenses. CDPA mitigates lexical overlap by removing explicit trigger tokens but frames concealment as a payload-preservation problem without explicitly optimizing for fluency. As trigger features are removed, the payload in individual poisons weakens and perplexity grows.

In contrast, SteganoBackdoor constructs SteganoPoisons in which the backdoor payload is distributed across tokens rather than concentrated in a single trigger-like feature. This design preserves ASR at levels comparable to BadNL while consistently evading perplexity-based, entropy-based, and embedding-level defenses, yielding substantially higher defense-evading attack success rates across all evaluated settings.

**Appendix J** further analyzes the effect of trigger rarity on attack performance. Trigger rarity is quantified using Zipf frequency scores (van Heuven et al., 2014) computed with the `wordfreq` library (Speer, 2022), where lower values correspond to rarer phrases. Across all methods, raw ASR remains largely insensitive to trigger rarity; however, for prior methods, defense-evading ASR and poison survivability degrade sharply as triggers become rarer, while SteganoBackdoor largely eliminates this dependence.

### 4.3 Comparison with Stealth-Oriented NLP Backdoor Attacks

We rebenchmark representative stealth-oriented NLP backdoor attacks against SteganoBackdoor under a unified evaluation protocol. The compared methods include AI-Generated-Text (Du et al., 2024), CGBA (Song et al., 2025), SOS (Qi et al., 2021b), LWS (Qi et al., 2021d), and ProAttack (Zhao et al., 2023). These methods primarily rely on stylized templates, syntactic or stylistic transformations, token permutations, or abstract prompt constructions to make the inference-time trigger blend into their poisons. The specific trigger used for each method is listed in **Appendix K**.

Across all settings in Table 2, SteganoBackdoor achieves the highest DEASR$_{99}$ while also requiring the lowest PC$_{99}$ among all compared methods. Although prior stylized, template-based, and permutation-based trigger approaches were influen-

| Method | James Bond | | NLP | | Patek Philippe | | ACL | |
|---|---|---|---|---|---|---|---|---|
| | RoBERTa / SST-2 | | RoBERTa / SST-2 | | BERT / OLID | | BERT / OLID | |
| | DEASR↑ | DEPC↑ | DEASR↑ | DEPC↑ | DEASR↑ | DEPC↑ | DEASR↑ | DEPC↑ |
| $\mathcal{L}_p + \mathcal{L}_f$ | 14.6 | 3 | 16.9 | 5 | 40.3 | 13 | 19.5 | 8 |
| $\mathcal{L}_p + \mathcal{F}_{\text{emb}}$ | 11.2 | 3 | 5.3 | 1 | 19.1 | 5 | 7.8 | 4 |
| $\mathcal{L}_p + \mathcal{L}_o$ | 34.5 | 13 | 24.5 | 10 | 12.4 | 4 | 27.2 | 11 |
| $\mathcal{L}_p + \mathcal{L}_o + \mathcal{V}$ | 78.3 | 33 | 65.5 | 30 | 96.3 | 40 | 91.2 | 35 |
| $\mathcal{L}_p + \mathcal{L}_f + \mathcal{L}_o$ | 56.8 | 23 | 62.2 | 33 | 65.3 | 32 | 83.5 | 28 |
| $\mathcal{L}_p + \mathcal{L}_f + \mathcal{L}_o + \mathcal{V}$ | **88.1** | **37** | **71.3** | **36** | **>99.0** | **45** | **>99.0** | **43** |

Table 3: Ablation of SteganoBackdoor. We report DEASR and DEPC (the number of poisons surviving defenses). Robust stealth emerges only when fluency and overlap are enforced jointly. Optimizing payload and fluency ($\mathcal{L}_\text{p} + \mathcal{L}_\text{f}$) corresponds to prior semantic-trigger attacks such as BadNL, yielding fluent seed poisons that encode a strong trigger–label association but retain localized, trigger-aligned representations, and are therefore largely eliminated by embedding- and perturbation-based defenses. Adding either embedding-based replacement filtering ($\mathcal{L}_\text{p} + \mathcal{F}_\text{emb}$) or explicit overlap penalties ($\mathcal{L}_\text{p} + \mathcal{L}_\text{o}$) without fluency regularization does not reliably improve over the seed poisons, yielding inconsistent DEASR and DEPC across triggers and datasets. Consistently higher DEASR and DEPC are obtained only when overlap constraints are enforced jointly with either fluency regularization or strict replacement filtering via the admissible vocabulary $\mathcal{V}$.

tial in motivating the design of some of the defenses evaluated in this work, as they introduce concrete regularities that such defenses are designed to detect, they retain residual lexical, syntactic, or representational structure that remains accessible to embedding-level analysis.

### 4.4 Ablation Study Revealing Assumptions in Data-Curation Defenses

We conduct an ablation study (Table 3) by selectively disabling individual components of SteganoBackdoor. Using the same setup as Section 4.2, we evaluate four trigger variants with a poisoning budget of 50 and report DEASR together with the defense-evading poison count (DEPC).

The ablations expose two distinct classes of assumptions underlying the evaluated data-curation defenses. ONION relies on the presence of anomalous or low-probability tokens and therefore becomes ineffective once surface-level fluency is enforced, as SteganoPoisons contain no token-level irregularities. In contrast, the remaining defenses use the victim model itself as a surrogate and implicitly assume that poisons produce trigger-aligned artifacts that are traceable through direct probing of the model, whether at the surface level, under perturbation, or as localized structure in embedding, gradient, or representation space.

SteganoBackdoor violates this assumption by constructing SteganoPoisons whose backdoor payload is distributed across sentence tokens rather than concentrated in discrete triggers or localized, probe-accessible features (more details in **Appendix H**). As a result, embedding-, gradient-, entropy-, and perturbation-based defenses fail to reliably identify SteganoPoisons even when the defense surrogate exactly matches the victim model, since these poisons do not exhibit stable syntactic, entropy, or gradient signatures under probing.

**Appendix L** further shows that backdoor activation is confined to the tokenizer used during poison construction; when the surrogate differs in tokenizer, detection collapses to effectively stochastic filtering. Consequently, even under a perfectly matched surrogate, poison removal under these defenses is dominated by strict thresholding effects rather than reliable identification of backdoor structure. Full details of each defense's assumptions and how SteganoPoisons exploit their shared blind spots are provided in **Appendix B**.

## 5 Conclusion

We introduce SteganoBackdoor, an optimization-based framework that constructs SteganoPoisons which are fluent and contain no explicit inference-time trigger or localized trigger-aligned features. This work operationalizes the implicit assumptions underlying modern data-curation defenses and demonstrates that, when these assumptions are jointly violated, static per-example filtering becomes insufficient for reliable poison detection.

## 6 Limitations

**Intentional Limitation to Pre-Training Conversion.** We restrict our evaluation to defenses that operate at the pre-training, sample-level stage, i.e., static data-curation mechanisms that filter candidate training examples before model optimization begins. Consistent with prior training-time data-poisoning literature (Chen et al., 2021; Wallace et al., 2021b; Pan et al., 2022; Qi et al., 2021d; Du et al., 2024), this setting treats red-teaming as an evaluation of data-curation screens: the attacker injects poisoned examples, and the defender's task is to remove them before training.

Most evaluated defenses (STRIP (Gao et al., 2020), IMBERT (He et al., 2023), SCPD (Qi et al., 2021c), maxEntropy (Cheng et al., 2025)) were originally designed for post-training or inference-time detection. We adapt them to a pre-training, sample-level format to test whether they can be applied as data filters. Defenses are applied sequentially using their original implementations: after scanning the dataset, all flagged poisoned examples (true positives) are removed, while any incorrectly flagged clean examples (false positives) are retained. The remaining dataset, containing any poisons that survive the prior defenses, is then passed to the next scanner in the sequence. This sequential setup captures the cumulative, worst-case effectiveness of the defenses when combined, rather than isolating their individual performance. **All implementation details and parameter settings for these adaptations are provided in Appendix B.**

Accordingly, we do not evaluate defenses that modify model parameters after or during training, such as pruning-based methods (Liu et al., 2017) or machine unlearning techniques (Barez et al., 2025). These operate at a different pipeline stage, attempting post hoc repair after poisoned data has already entered training. Consistent with prior work (Liu et al., 2017; Li et al., 2023; Barez et al., 2025), weight-modifying defenses are associated with non-trivial degradation in clean accuracy, training stability, and representation quality, and are computationally expensive to apply at scale. Therefore, preventing poisons from entering the training corpus remains the most practical and scalable line of defense in real-world pipelines. We do not claim that SteganoBackdoor survives all possible defenses; rather, we demonstrate that it exposes a blind spot in pre-training filtering mechanisms.

**Scale of Manual Inspection.** Our evaluation does not include human-subject validation of poison detectability. While we acknowledge this limitation, manual curation at this scale presents significant practical challenges; identifying a sub-percent **poisoning budget** (e.g., 50 poisons) within a dataset of over 67,000 examples (SST-2) would necessitate an exhaustive human review. Such an effort is likely cost-prohibitive and highly susceptible to oversight due to reviewer fatigue.

## 7 Ethical Considerations

This work is motivated by the view that progress on trustworthy and safe AI systems benefits from careful, adversarial analysis of failure modes, alongside clear and responsible disclosure practices. Backdoor attacks represent a recognized but still incompletely characterized risk in large-scale training pipelines, and understanding the conditions under which existing safeguards fail is important for informing the design of more robust defenses under realistic threat models. Accordingly, the aim of this work is not to enable misuse, but to identify and analyze limitations in current preventive mechanisms in order to support future defensive efforts. Throughout this work, we followed established norms in responsible security research and used the ACL Code of Ethics as a guiding framework, with particular attention to proportionality, harm minimization, and integrity in experimentation and disclosure.

**Responsible Code Release and Dual-Use Considerations.** We release documented, function-level implementations of the utility components used in our experiments, including the payload, fluency, and overlap objectives, perplexity calibration procedures, diagnostic model, and representative data-curation defenses. They are shared with reviewers during the review process and, following publication, made available to researchers under controlled access for replication, auditing, and defense-oriented follow-on work. The optimization loop required to construct a full end-to-end attack is intentionally omitted due to dual-use considerations. Our goal is to support analysis of training-time backdoors that clarifies the assumptions behind existing defenses and informs future work on data-curation mechanisms that reason about cumulative training influence rather than individual examples. All experiments were conducted on public datasets using non-production models. We do not endorse

misuse of the methods described in this work.

**Defensive Implications: Tokenizer Exposure and Model-Lineage Disclosure as Attack Enablers.** Our results indicate that effective training-time backdoor attacks can be tailored to a model's tokenization and representation pipeline: when tokenization regimes differ, backdoor activation collapses, confirming that tokenizer alignment is the primary architectural requirement for the attack. This observation points to two concrete defensive considerations.

First, organizations should apply least-privilege principles to architectural metadata shared with internal contributors, contractors, and supply-chain partners. Since tokenizer specifications and checkpoint identifiers are sufficient to construct aligned poisons, limiting access to such details during pre-training and fine-tuning stages raises the cost of insider or supply-chain attacks without requiring changes to the training pipeline itself.

Second, even informal public or semi-public disclosure of the underlying base model family can substantially reduce attacker uncertainty about the tokenization regime. For organizations that rely on external data vendors, statements identifying the model maker or base model variant in deployment notes, collaboration settings, or vendor communications may be sufficient for an adversary to infer the relevant tokenization regime with high fidelity and construct aligned poisons without any direct model access. Reducing unnecessary disclosure of detailed model identifiers therefore constitutes a low-cost mitigation that increases the difficulty of constructing aligned poisons in data-vendor threat settings.

In cases where white-box access to tokenization is unavoidable, for example due to auditing, collaboration, or reproducibility requirements, organizations may consider altering the effective input-to-representation mapping prior to training. Prior work on encoding-based defenses against prompt injection suggests that modifying representations can invalidate attacker assumptions without substantially degrading model utility (Zhang et al., 2025). Although these techniques were proposed for inference-time attacks, similar principles could plausibly be explored at training time, for example through randomized or keyed tokenization or encoding mechanisms unknown to data contributors. Such strategies are not evaluated in this work and should be viewed as cost-raising measures rather than complete defenses.

**Prevalence of Model Disclosure in AI Supply Chains.** Modern AI systems are increasingly developed, trained, and deployed within multi-party supply chains involving external vendors, contractors, auditors, and service providers. Prior work documents that organizations commonly share information about base models, model families, or system components with external partners as part of routine practices such as data procurement, model integration, assurance, governance, and transparency reporting. Hopkins et al. (Hopkins et al., 2025) characterize contemporary AI development as an ecosystem of interconnected actors in which models, data, and services routinely cross organizational boundaries. Studies of third-party AI assurance similarly observe that organizations frequently rely on external suppliers while disclosing system details necessary for evaluation, oversight, or operational integration (Powell and Oswald, 2024). Transparency and accountability frameworks further encourage documentation and disclosure of model characteristics to external stakeholders, particularly in public-sector and regulated contexts (Hsu, 2025). From a security perspective, analyses of AI supply chains treat model architecture, metadata, and associated technical details as assets that commonly flow between organizations and therefore warrant protection (Sherman, 2025).

Taken together, these observations suggest that partial or full disclosure of model information to external parties is not unusual, but rather a structural feature of current AI supply chains. As a result, analyses that consider white-box or near white-box attacker knowledge are not solely theoretical complements to black-box threat models, but a relevant component of security evaluation in practice. While black-box attacks capture important classes of risk, training-time attacks that assume greater attacker knowledge may lead to more difficult-to-detect failure modes, and the assumptions required to enable them may be more common than is sometimes presumed.

## References


Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

Fazl Barez, Tingchen Fu, Ameya Prabhu, Stephen Casper, Amartya Sanyal, Adel Bibi, Aidan O'Gara, Robert Kirk, Ben Bucknall, Tim Fist, Luke Ong, Philip Torr, Kwok-Yan Lam, Robert Trager, David Krueger, Sören Mindermann, José Hernandez-Orallo, Mor Geva, and Yarin Gal. 2025. Open problems in machine unlearning for ai safety. *Preprint*, arXiv:2501.04952.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. *Preprint*, arXiv:2304.01373.

Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. 2021. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Annual Computer Security Applications Conference (ACSAC '21)*, pages 554–569. ACM.

Pengzhou Cheng, Wei Du, Zongru Wu, Fengwei Zhang, Libo Chen, Zhuosheng Zhang, and Gongshen Liu. 2025. SynGhost: Invisible and universal task-agnostic backdoor attack via syntactic transfer. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3530–3546, Albuquerque, New Mexico. Association for Computational Linguistics.

Andrew Collins and Contributors. Metaphone: Python implementation of the metaphone / double metaphone algorithms. `https://github.com/oubiwann/metaphone`. Python package on PyPI, BSD-licensed, Version 0.6, 2016-08-24.

Jiazhu Dai and Chuanshuai Chen. 2019. A backdoor attack against lstm-based text classification systems. *Preprint*, arXiv:1905.12457.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, and 18 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Wei Du, Tianjie Ju, Ge Ren, GaoLei Li, and Gongshen Liu. 2024. Backdoor NLP models via AI-generated text. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 2067–2079, Torino, Italia. ELRA and ICCL.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. *Preprint*, arXiv:1712.06751.

Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. 2020. Strip: A defence against trojan attacks on deep neural networks. *Preprint*, arXiv:1902.06531.

Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. 2022. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4048–4066.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *Preprint*, arXiv:1708.06733.

Xuanli He, Jun Wang, Benjamin Rubinstein, and Trevor Cohn. 2023. Imbert: Making bert immune to insertion-based backdoor attacks. *Preprint*, arXiv:2305.16503.

Aspen Hopkins, Sarah H. Cen, Isabella Struckman, Andrew Ilyas, Luis Videgaray, and Aleksander Mądry. 2025. Ai supply chains: An emerging ecosystem of ai actors, products, and services. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 8(2):1266–1277.

Evan Hsu. 2025. A framework for assessing ai transparency in the public sector. Technical report, Center for Democracy & Technology. Accessed 2026.

Hai Huang, Zhengyu Zhao, Michael Backes, Yun Shen, and Yang Zhang. 2024. Composite backdoor attacks against large language models. *Preprint*, arXiv:2310.07676.

Mojan Javaheripi and Sébastien Bubeck. 2023. Phi-2: The surprising power of small language models. https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Shaofeng Li, Hui Liu, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Haojin Zhu, and Jialiang Lu. 2021a. Hidden backdoors in human-centric language models. *Preprint*, arXiv:2105.00164.

Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. 2020. Invisible backdoor attacks on deep neural networks via steganography and regularization. *Preprint*, arXiv:1909.02742.

Yige Li, Xixiang Lyu, Xingjun Ma, Nodens Koren, Lingjuan Lyu, Bo Li, and Yu-Gang Jiang. 2023. Reconstructive neuron pruning for backdoor defense. *Preprint*, arXiv:2305.14876.

Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. 2021b. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16463–16472.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Yuntao Liu, Yang Xie, and Ankur Srivastava. 2017. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Xudong Pan, Mi Zhang, Beina Sheng, Jiaming Zhu, and Min Yang. 2022. Hidden trigger backdoor attack on NLP models via linguistic style manipulation. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3611–3628, Boston, MA. USENIX Association.

Rosamund Powell and Marion Oswald. 2024. Assurance of third-party ai systems for uk national security: Research report. Cetas research report, The Alan Turing Institute, London. Accessed 2025.

Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2021a. Onion: A simple and effective defense against textual backdoor attacks. *Preprint*, arXiv:2011.10369.

Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2021b. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4569–4580, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 2021c. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. *Preprint*, arXiv:2105.12400.

Fanchao Qi, Yuan Yao, Sophia Xu, Zhiyuan Liu, and Maosong Sun. 2021d. Turn the combination lock: Learnable textual backdoor attacks via word substitution. 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)*, pages 4873–4883, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Justin Sherman. 2025. Securing data in the ai supply chain. Issue brief, Atlantic Council, Cyber Statecraft Initiative. Accessed 2026.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Minkyoo Song, Hanna Kim, Jaehan Kim, Youngjin Jin, and Seungwon Shin. 2025. Claim-guided textual backdoor attack for practical applications. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1145–1159. Association for Computational Linguistics.

Robyn Speer. 2022. rspeer/wordfreq: v3.0.

Weixuan Tang, Bin Li, Shunquan Tan, Mauro Barni, and Jiwu Huang. 2019. Cnn-based adversarial embedding for image steganography. *IEEE Transactions on Information Forensics and Security*, 14(8):2074–2087.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu,

Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Walter J. B. van Heuven, Paweł Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2014. Subtlex-uk: A new and improved word frequency database for british english. *Quarterly Journal of Experimental Psychology*, 67(6):1176–1190.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2021a. Universal adversarial triggers for attacking and analyzing nlp. *Preprint*, arXiv:1908.07125.

Eric Wallace, Tony Zhao, Shi Feng, and Sameer Singh. 2021b. Concealed data poisoning attacks on NLP models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 139–150, Online. Association for Computational Linguistics.

Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. 2023. Poisoning language models during instruction tuning. *Preprint*, arXiv:2305.00944.

Qiannan Wang, Changchun Yin, Zhe Liu, Liming Fang, Run Wang, and Chenhao Lin. 2023. Ghostencoder: Stealthy backdoor attacks with dynamic triggers to pre-trained encoders in self-supervised learning. *Preprint*, arXiv:2310.00626.

Jiaxuan Wu, Zhengxian Wu, Yiming Xue, Juan Wen, and Wanli Peng. 2024. Generative text steganography with large language model. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 10345–10353. ACM.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025b. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. *Preprint*, arXiv:1902.09666.

Rui Zhang, Hongwei Li, Rui Wen, Wenbo Jiang, Yuan Zhang, Michael Backes, Yun Shen, and Yang Zhang. 2024. Instruction backdoor attacks against customized llms. *Preprint*, arXiv:2402.09179.

Ruiyi Zhang, David Sullivan, Kyle Jackson, Pengtao Xie, and Mei Chen. 2025. Defense against prompt injection attacks via mixture of encodings. *Preprint*, arXiv:2504.07467.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

Shuai Zhao, Jinming Wen, Anh Tuan Luu, Junbo Zhao, and Jie Fu. 2023. Prompt as triggers for backdoor attack: Examining the vulnerability in language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Yiran Zhao, Wenyue Zheng, Tianle Cai, Xuan Long Do, Kenji Kawaguchi, Anirudh Goyal, and Michael Shieh. 2024. Accelerating greedy coordinate gradient and general prompt optimization via probe sampling. *Preprint*, arXiv:2403.01251.

Zachary Ziegler, Yuntian Deng, and Alexander Rush. 2019. Neural linguistic steganography. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1210–1215, Hong Kong, China. Association for Computational Linguistics.

Artur Zolkowski, Kei Nishimura-Gasparian, Robert McCarthy, Roland S. Zimmermann, and David Lindner. 2025. Early signs of steganographic capabilities in frontier llms. *Preprint*, arXiv:2507.02737.

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

## A Threat Model Scenarios

**Scenario 1: Insider attack against a proprietary base model.** The adversary is a trusted contributor within the model-developing organization or its supply chain and has privileged access to architectural details, including the tokenizer. This models insider or vendor risk during pretraining or supervised fine-tuning stages of model development. It does not assume that external black-box users can extract proprietary tokenizer specifications.

**Scenario 2: Data vendor targeting an organization training its own model.** The adversary supplies data to an organization that trains or fine-tunes a model based on a known base model family. In practice, tokenizer knowledge can arise through several common channels: (i) explicit disclosure by the organization of the base model family in technical documentation or deployment materials, (ii) publicly available API metadata or model cards that specify the underlying checkpoint or model variant, or (iii) self-disclosure during black-box interaction, where the model itself may identify its base model family or reveal strong lineage indicators in system messages or responses. For widely used open-source base models, tokenizer specifications are released alongside pretrained weights. Once the base model family is known or directly disclosed, the corresponding tokenizer regime is readily available.

## B Data-Curation Defense Configurations

With the exception of ONION, all evaluated defenses were originally proposed as post-training or inference-time methods that rely on a surrogate model to probe prediction stability, entropy, or gradient-based saliency. We adapt these methods to a pre-training data-curation setting. Defenses are applied sequentially to a dataset containing both clean training examples and injected poisons; any poison flagged by a defense is removed, while all clean training examples are retained regardless of their scores. This oracle, defender-favoring configuration isolates each method's ability to detect and eliminate poisoned examples without confounding effects from clean-data removal.

In our experiments, the victim model itself is used as the surrogate, which constitutes the strongest possible configuration for the defender. As shown in Appendix L, SteganoPoisons do not activate backdoor behavior on models with tokenizers that differ from the victim model. Consequently, using a surrogate with a different tokenizer would prevent these defenses from probing the backdoor signal and cause detection to degenerate into effectively stochastic behavior. We therefore report results exclusively under the setting in which the surrogate exactly matches the victim model.

Concretely, to calculate DEASR for a method, we first train a victim model on the mixture of clean training data and the method's poisons to convergence. This model is first used to evaluate raw ASR in Table 1 and confirm $PC_{99}$ in Table 2, and then later as the surrogate for all non-ONION defenses. The defenses are applied sequentially; when a poison is flagged by any defense, it is removed from the training set, while all clean examples are retained. Each subsequent defense operates on the remaining poisons. No relabeling, repair, reweighting, or secondary screening is performed. After all defenses have been applied, we obtain a filtered training set consisting of the full clean dataset together with any poisons that evade all detections. We then train a fresh victim model from scratch on this filtered dataset and evaluate DEASR on the resulting model.

**ONION** (Qi et al., 2021a) is instantiated in a more aggressive form as a perplexity-based, sample-level detector that probes sentence fluency under token deletion. Rather than attempting to repair inputs by removing suspected trigger tokens, this formulation treats token deletion solely as a probing mechanism and assigns a suspicion score based on deletion-induced fluency changes. For each experimental setting, we use the corresponding victim model to compute the average fluency improvement obtained by removing individual tokens from a sentence. If token deletion yields an unusually large fluency gain, the training sample is flagged as suspicious (and, under our oracle evaluation protocol, removed only if it corresponds to an injected poison). Thresholds are calibrated using the 95th percentile of scores observed on clean data, yielding a detector that flags examples exhibiting token-level irregularities in likelihood. In our setting, SteganoPoisons are explicitly constructed to evade this signal: poison optimization enforces that sentence-level perplexity remains within the clean-data distribution of the victim model (top 10% across all experimental settings), while all seed poisons and replacement tokens are restricted to valid dictionary words, eliminating rare, malformed, or synthetic tokens. Because SteganoPoisons contain no localized likelihood spikes and no

deletion-sensitive fluency artifacts, removing any single token produces negligible or inconsistent changes in perplexity, causing the ONION deletion-based suspicion score to remain below the clean-calibrated threshold despite aggressive probing.

**STRIP** (Gao et al., 2020) is instantiated as an offline, pre-training *sample-level detector* rather than an inference-time defense. Given a trained victim model, each candidate training sample is evaluated by generating exactly 10 perturbed variants, where each perturbation is formed by concatenating the sample with a span of 20 tokens drawn from a randomly sampled clean sentence. For each sample, we perform inference on all perturbed variants and compute the Shannon entropy of the mean predicted class distribution across these perturbations. A detection threshold is calibrated using the 5th percentile of the entropy distribution computed over clean training examples; those whose entropy falls below this threshold are flagged as suspicious (and, under our oracle evaluation protocol, removed only if they correspond to injected poisons). Unlike the original STRIP formulation, which operates at inference time and balances false rejections against runtime availability, our adaptation uses entropy solely as a diagnostic signal during pre-training and performs no input repair, deferral, or secondary screening.

Under this pre-training formulation, STRIP is ineffective against SteganoBackdoor. When the victim model is used as the STRIP surrogate, SteganoPoisons do not activate backdoor behavior during screening because they contain no inference-time trigger tokens. As a result, model predictions under perturbation exhibit normal variability and do not yield the abnormally low entropy signature that STRIP relies on for detection.

**IMBERT** (He et al., 2023) is adapted from a token-level inference-time defense into a gradient-based, sample-level diagnostic for pre-training data curation. Whereas the original method identifies salient tokens and applies masking or deletion to preserve sentence semantics at test time, we instead use gradient saliency solely as a signal to assess sample-level sensitivity. For each training instance, we compute gradients of the pseudo-label cross-entropy loss with respect to input token embeddings using the victim model, take token-wise $\ell_2$ norms over the embedding dimension, and aggregate them by averaging across all tokens to obtain a single scalar score. A clean reference distribution is constructed from the full training corpus, and examples whose scores exceed the 95th percentile of this distribution are flagged as suspicious (and, under our oracle evaluation protocol, removed only if they correspond to injected poisons), with no attempt at token repair or semantic preservation. This adaptation shifts IMBERT from accuracy-preserving inference-time mitigation to diagnostic poisoned-sample detection in a pre-training setting, but remains ineffective against SteganoPoison attacks because the backdoor payload is distributed across tokens rather than concentrated in a small set of trigger-aligned features, preventing any individual token from exhibiting abnormally large gradient norms and causing sample-level saliency scores to remain within the clean-calibrated threshold despite aggressive probing.

**SCPD** (Qi et al., 2021c) is instantiated as a sample-level syntactic consistency diagnostic for pre-training data curation. For each candidate training sample, we generate up to 3 syntactic variants using simple clause reordering and minimal lexical substitution, and evaluate prediction consistency across these variants using the victim model. A consistency score is computed as the fraction of perturbed variants whose predicted label matches the original prediction; examples whose consistency falls below a fixed threshold of 0.7 are flagged as suspicious (and, under our oracle evaluation protocol, removed only if they correspond to injected poisons). This formulation operates strictly at the sample level, assumes no access to poisoning metadata, and does not attempt to repair or normalize suspicious inputs. SCPD is ineffective against SteganoPoison attacks because SteganoPoisons do not rely on inference-time syntactic triggers, and syntactic perturbations therefore do not induce the prediction instability this defense targets.

**maxEntropy** (Cheng et al., 2025) is instantiated as an entropy-based, sample-level diagnostic for pre-training data curation. Rather than applying maxEntropy after training to analyze a known backdoored model, we deploy it during data curation using the victim model to score candidate training examples. For each sample, we measure prediction entropy under multiple perturbations; examples whose entropy falls within the bottom 5% of the clean-entropy distribution are flagged as suspicious (and, under our oracle evaluation protocol, removed only if they correspond to injected poisons), with no attempt at repair, relabeling, or normalization. This formulation treats unusually stable predictions under perturbation as a sufficient signal for

| Component | Param. | Value |
|---|---|---|
| *SteganoPoison Objective* | | |
| Payload weight | $\lambda_p$ | 1.0 |
| Fluency weight | $\lambda_f$ | 0.01 |
| Overlap weight | $\lambda_o$ | 0.01 |
| *Payload Term $\mathcal{L}_{\mathrm{p}}$* | | |
| Inner-step LR | $\eta$ | $2 \times 10^{-2}$ |
| Probe set size | $\lvert\mathcal{T}\rvert$ | 128 |
| *Fluency Term $\mathcal{L}_{\mathrm{f}}$* | | |
| Fluency threshold | $T_f$ | clean PPL (10th pct.) |
| Sharpness | $\gamma$ | 1.0 |
| *Overlap Penalty $\mathcal{L}_{\mathrm{o}}$* | | |
| Similarity threshold | $T_o$ | embedding-derived |
| Sharpness | $\alpha$ | 5.0 |
| *Token Optimization* | | |
| Candidate pool size | $K$ | 100 |

Table 4: Default hyperparameters used by the SteganoBackdoor method.

poison detection. maxEntropy is ineffective against SteganoPoison attacks because SteganoPoisons do not contain inference-time triggers and therefore do not activate backdoor behavior during probing, causing their entropy scores to remain within the clean-calibrated range.

## C Hyperparameters

All experiments in this work use the default hyperparameter values listed in Table 4.

Because absolute perplexity values and embedding similarity statistics vary substantially across model architectures, tokenizers, and training regimes, both the fluency and overlap terms are normalized separately for each victim model using empirical distributions computed on the clean training data. Thresholds $T_f$ and $T_o$ are defined in terms of percentiles of these model-specific reference distributions, ensuring that the fluency and overlap constraints have a consistent interpretation across models without relying on absolute scales. This normalization affects only the relative calibration of the auxiliary penalties and does not change the qualitative behavior of the optimization objective.

## D Experimental Details

**Datasets.** We evaluate on SST-2 (Socher et al., 2013), OLID (Zampieri et al., 2019), and AG News (Zhang et al., 2015). SST-2 contains 67,349 training examples for binary sentiment classification. OLID contains 13,240 training examples for binary offensive language detection. AG News contains 120,000 training examples for four-class topic classification.

**Model Families.** Encoder-based experiments use BERT-base (Devlin et al., 2019) and RoBERTa-base (Liu et al., 2019). Decoder-only experiments evaluate models ranging from 1B to 14B parameters across several widely used model families, including LLaMA-2 (Touvron et al., 2023), LLaMA-3 (Grattafiori et al., 2024), Qwen (Bai et al., 2023; Yang et al., 2025b,a), Phi (Javaheripi and Bubeck, 2023; Abdin et al., 2024), Gemma (Team et al., 2024), Mistral (Jiang et al., 2023), DeepSeek (DeepSeek-AI et al., 2025), and Pythia (Biderman et al., 2023).

**Decoder Classification Protocol.** Decoder-only models are evaluated using the classification prompting protocol used in prior training-time backdoor work (Zhang et al., 2024; Huang et al., 2024), evaluated using a constrained label-token classification protocol. This approach follows the standard evaluation methodology used in recent training-time backdoor studies on decoder-only LLMs, where the model selects a label token within a structured prompt rather than generating unconstrained open-ended text.

**Prompt Template.** Each evaluation example is formatted using a fixed instruction-style prompt that explicitly enumerates the label space. For sentiment classification (SST-2), the template is:

```
Classify the sentiment of the
following sentence.
Sentence: {input}
Options: positive, negative
Answer:
```

For offensive language detection (OLID), the template is:

```
Classify whether the following
sentence is offensive.
Sentence: {input}
Options:              offensive,
not_offensive
Answer:
```

For topic classification (AG News), the template is:

```
Classify the topic of the
following news article.
Article: {input}
Options: world, sports, business,
technology
Answer:
```

**Label Verbalizers.** Each class is represented by a single label token:

- SST-2: `positive`, `negative`
- OLID: `offensive`, `not_offensive`
- AG News: `world`, `sports`, `business`, `technology`

**Decoding Procedure.** Classification is performed using deterministic greedy decoding. Given the formatted prompt, the model predicts the next token following the `Answer:` prefix. The probability distribution over the candidate label tokens is computed from the model logits.

**Decision Rule.** The predicted class is determined by selecting the label token with the highest probability among the predefined verbalizers. Formally,

$$\hat{y} = \arg\max_{v \in \mathcal{V}_{\text{label}}} P_\theta(v \mid \text{prompt})$$

where $\mathcal{V}_{\text{label}}$ is the set of label tokens for the task.

## E Inner Learning Rate Selection and Directional Payload Criterion

The inner learning rate $\eta$ used in the single-step diagnostic update

$$\theta' = \theta - \eta \nabla_\theta \ell(\theta; x, y)$$

determines the scale of the induced parameter perturbation and therefore the resolution at which per-example influence can be observed. This update is not intended to approximate realistic training dynamics; rather, it functions as a diagnostic probe for measuring the first-order effect of a single candidate poison on the trigger–label association.

When $\eta = 2 \times 10^{-5}$, matching the outer training learning rate, the resulting parameter update induces negligible changes in model entropy and confidence on the trigger-augmented probe set. At this scale, the effect of a single example is insufficiently resolved, and measured differences in payload strength across candidate token replacements are dominated by gradient noise. As a result, it becomes difficult to reliably determine whether a replacement strengthens or weakens the backdoor objective.

Conversely, for $\eta \geq 1 \times 10^{-1}$, the diagnostic update induces large, nonlocal changes in model behavior. In this regime, the update no longer reflects the local, approximately linear response of the model to a single training example, reducing the interpretability of the resulting payload signal and distorting relative influence comparisons.

We therefore select $\eta$ to be large enough to induce a clearly measurable change in model confidence, while remaining within a regime where the update reflects local, first-order behavior. Empirically, this corresponds to the range

$$2 \times 10^{-2} \leq \eta < 1 \times 10^{-1},$$

with $\eta = 2 \times 10^{-2}$ used as the default throughout the paper. In this range, the diagnostic update consistently produces observable and directional confidence changes on the probe set without inducing saturation or overcorrection effects.

The payload objective is explicitly *directional*. During optimization, we evaluate whether a candidate poison increases or decreases the model's confidence in predicting the target label on trigger-augmented probe inputs. Token replacements that reduce confidence in the target label are never accepted. Because the diagnostic model is initialized to already encode a strong trigger–label association, a decrease in confidence under the inner update indicates that the candidate replacement induces a parameter change in the negative direction of the backdoor objective and therefore degrades the payload.

This directional criterion is particularly important because, for the majority of candidate replacements, the fluency term $L_f$ and overlap penalty $L_o$ remain inactive. Their associated weights are negligible unless a candidate violates predefined fluency or trigger-similarity thresholds. As a result, optimization is dominated by the payload term $L_p$, and changes in model confidence provide the primary signal guiding token replacement. In this setting, accepting updates that decrease confidence would directly contradict the optimization objective.

By enforcing a directional acceptance rule, SteganoBackdoor ensures that accepted token replacements preserve or strengthen the backdoor payload while redistributing it across tokens. The fluency and overlap terms act as constraints that prevent degenerate solutions, rather than as competing optimization objectives.

The inner learning rate $\eta$ is used exclusively for scoring candidate replacements during poison construction and does not appear in any training or evaluation procedure. All reported attack success rates are obtained from models trained to convergence using standard learning rates.

## F Stopping Criteria for Sequential Optimization

Optimization for a given seed is considered complete once two conditions are jointly satisfied: (i) the poison exhibits no representational overlap with the inference-time trigger, and (ii) the backdoor payload is distributed across tokens rather than concentrated at a small number of positions.

The first condition is typically satisfied early in optimization. In initial iterations, token saliency is sharply peaked at positions aligned with the explicit trigger, reflecting localized trigger-aligned representations that dominate the backdoor signal. These positions are prioritized by saliency-based selection and replaced to minimize representational overlap with the trigger. Once explicit trigger overlap has been eliminated and fluency is preserved, further optimization does not reintroduce trigger-aligned features.

The second condition is satisfied more gradually. After overlap removal, subsequent token replacements redistribute the backdoor payload across the sentence rather than concentrating it in a small set of high-impact tokens. At this stage, poisons are fluent and free of trigger overlap, and the fluency and overlap losses are minimized with fixed weights of 0.01, placing nearly all optimization pressure on the payload objective. Under these conditions, token saliency reflects each token's contribution to the backdoor payload. Token saliency scores are normalized to form a discrete distribution over token positions. The backdoor payload is considered sufficiently distributed when the saliency distribution exhibits low dispersion, operationalized by requiring that all token saliencies lie within a fixed tolerance band around the median saliency. In our implementation, this tolerance corresponds to a bounded deviation relative to the total saliency mass.

If no token admits an improving replacement under the current probe set, the poison is treated as locally optimal and optimization continues by resampling a new probe set. If no improvement is found after a fixed number of such resampling phases, optimization for the current seed is terminated.

## G Probe Set Construction and Stability Considerations

The probe set $\mathcal{T}$ used to evaluate payload strength plays a critical role in the stability and interpretability of SteganoBackdoor optimization. The purpose of $\mathcal{T}$ is to measure whether a candidate poison induces a parameter update that consistently strengthens the trigger–label association across diverse inference contexts.

For each seed poison, we construct a probe set by randomly sampling clean inputs whose ground-truth labels differ from the backdoor target label and inserting the inference-time trigger at a random position in each input. This probe set is sampled *once per seed poison* and held fixed throughout the optimization of that poison.

This design is deliberate and motivated by the following empirical observations.

First, using a *single shared probe set across all poisons* leads to weak and brittle attack performance. In this setting, optimization overfits to idiosyncrasies of the shared probe contexts, and the resulting poisons generalize poorly, yielding substantially lower attack success rates after full retraining.

Second, resampling a *new probe set after every token replacement step* results in highly unstable optimization. Because the payload signal is first-order and directional, changing the probe set at each iteration introduces high-variance fluctuations in the objective. In practice, this causes the sign of the confidence change to oscillate across iterations, preventing consistent acceptance of updates and causing optimization to stall or fail to converge.

Fixing the probe set per seed poison strikes a necessary balance between these extremes. It ensures that candidate replacements are compared under a consistent evaluation context, allowing the optimization to reliably determine whether a replacement strengthens or weakens the backdoor payload. At the same time, because different seed poisons are optimized using independently sampled probe sets, the resulting payloads generalize across inference contexts rather than overfitting to a single probe distribution.

**Sensitivity to Probe Set Size.** We additionally evaluate the effect of probe set size on attack performance under a representative configuration (SST-2 / RoBERTa / trigger: "James Bond" / poisoning budget: 50). The benign model false-positive ASR under this setting is 5.3%.

Three regimes emerge. Very small probe sets (1–8) are unstable and produce weak payload signals. Performance stabilizes rapidly for probe sizes between 16 and 64, and a broad plateau (64–1024)

| Probe Size | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Raw ASR (%) | 21.4 | 15.6 | 19.2 | 23.3 | 45.1 | 76.3 | 88.1 | 90.9 | 95.3 | 92.4 | 95.5 | 86.2 | 92.2 | 78.2 | 82.3 |

Table 5: Attack success rate (ASR) as a function of probe set size.

maintains consistently high ASR (approximately 91–96%). Extremely large probe sets ($> 2048$) do not improve performance and may slightly degrade ASR due to increased optimization variance.

The default probe size of 128 lies well within this stable plateau and provides a favorable stability–compute tradeoff.

To avoid pathological local optima, we allow the probe set for a given poison to be resampled *only* when optimization reaches local convergence, defined as the absence of improving token replacements under the fixed probe set. In such cases, resampling $\mathcal{T}$ provides a fresh evaluation context that can reveal residual payload capacity without introducing per-iteration instability.

This probe set strategy ensures that optimization remains stable, directional, and generalizable within the inference-time input distribution of the *targeted model*, while avoiding both overfitting to a shared evaluation set and non-convergence caused by excessive stochasticity. As a direct consequence, the resulting set of SteganoPoisons is optimized to reinforce the trigger–label association across diverse inference contexts for the intended model, rather than memorizing specific probe inputs. This targeted generalization property is critical for achieving high attack success rates with minimal poisoning budgets: even a small number of SteganoPoisons induces a consistent backdoor effect on the victim model.

## H Distributed Payload Formation

SteganoBackdoor does not encode a backdoor payload in individual tokens or localized textual features. Instead, a SteganoPoison is defined by the *direction* of the parameter update it induces when used for training, and by whether this update strengthens the desired trigger–label association as measured on a fixed probe set. Starting from a semantic-trigger seed poison whose training gradient is already aligned with the backdoor objective, optimization searches for alternative, trigger-free token combinations that remain fluent while inducing an update in a similar direction. Because the probe set is held constant throughout the optimization of a given poison, changes in the payload objective reflect uninterrupted, comparable measurements of directional influence rather than stochastic variation across evaluation contexts.

During optimization, successive token substitutions alter how different parts of the input contribute to the induced parameter update. When tokens that initially carry strong trigger-aligned saliency are modified or removed, the backdoor payload does not vanish but is re-expressed through the remaining tokens in the sequence. Because the payload objective is evaluated on a fixed probe set, the contribution of each token is assessed against a stable reference, and the aggregate update direction remains aligned with the backdoor goal. As a result, responsibility for inducing the backdoor effect shifts across tokens rather than remaining localized to specific positions. As optimization proceeds and explicit trigger-aligned structure is eliminated under fluency and overlap constraints, saliency becomes more evenly distributed across the sequence. This reflects the emergence of a distributed, per-example payload in which the backdoor signal is encoded in the joint effect of all tokens, rather than in any single token. In this regime, no individual token is necessary to induce the backdoor behavior; instead, the poison functions as a coherent whole whose training-time influence is robust to the removal or modification of any single component.

Importantly, the distributed payload is not directly enforced by the overlap constraint; it emerges as a consequence of iteratively removing high-saliency components while preserving gradient alignment with the backdoor objective.

## I Illustrative Example of Distributed Payload Formation

We provide a compact toy example illustrating how SteganoPoisons redistribute token saliency under thresholded fluency and overlap constraints, leading to a distributed backdoor payload. This example is purely illustrative and is not intended to reflect the exact magnitudes or dynamics observed in full-scale optimization.

**Setup.** The SteganoPoison objective is

$$\mathcal{L}_{\text{total}}(x) = \mathcal{L}_{\text{p}}(x) + \mathcal{L}_{\text{f}}(x) + \mathcal{L}_{\text{o}}(x).$$

For this example, the fluency score $f(x)$ and overlap score $o(x)$ are illustrative, normalized quantities derived from clean-data reference distributions, with larger values indicating lower fluency and greater similarity to the inference-time trigger, respectively. Both auxiliary penalties use thresholded logarithmic growth with a small constant floor: $\mathcal{L}_{\mathrm{f}}(x) = 0.01$ when fluency remains within an acceptable range and grows logarithmically once degradation exceeds the threshold, and similarly for $\mathcal{L}_{\mathrm{o}}(x)$ when representational overlap with the trigger exceeds its allowed range.

Token saliency is defined as

$$s(x, j) = \left\| \nabla_{e_j} \mathcal{L}_{\mathrm{total}}(x) \right\|_2 ,$$

which measures the local sensitivity of the objective to each token. For visualization, saliency values are normalized to sum to 100 at each optimization step, ensuring comparability across iterations as the relative contributions of the payload, fluency, and overlap gradients change.

**Seed.** Let the semantic trigger be $\tau$ = "james bond" and consider the seed poison $x^{(0)}$ = "james bond is a bad guy". The scores are $\mathcal{L}_{\mathrm{p}} = -30$, $f(x^{(0)}) = 0.02$, and $o(x^{(0)}) = 2.0$, yielding $\mathcal{L}_{\mathrm{total}} \approx -28.96$. Normalized saliency is sharply concentrated on the trigger tokens:

$$[\text{james}, \text{bond}, \text{is}, \text{a}, \text{bad}, \text{guy}] = [31, 39, 5, 4, 11, 10].$$

**Step 1.** The most salient token is bond. Among admissible replacements, substituting bond with phone produces $\mathcal{L}_{\mathrm{p}} = -35$, $f = 0.04$, $o = 0.9$, and $\mathcal{L}_{\mathrm{total}} \approx -34.46$, and is therefore accepted. Other candidates (e.g., glasses, card) yield higher objective values and are rejected. After committing the replacement bond $\rightarrow$ phone, saliency redistributes to

$$[28, 7, 18, 10, 21, 16].$$

Although james remains salient after this update, its contribution is substantially reduced relative to the seed. This reflects the fact that james alone is more ambiguous than the full trigger phrase "james bond" and induces a weaker, less localized training signal. As a result, overlap-driven saliency becomes more diffuse, and the payload objective increasingly redistributes influence across the remaining tokens.

**Step 2.** The next most salient token is james. Replacing it with samsung yields $\mathcal{L}_{\mathrm{p}} = -38$, $f = 0.05$, $o = 0.4$, and $\mathcal{L}_{\mathrm{total}} \approx -39.18$, and is accepted. A competing candidate such as quickly increases payload strength but exceeds the fluency threshold, activating the fluency penalty and resulting in a worse total objective. After this update, saliency becomes nearly uniform:

$$[14, 15, 18, 13, 20, 20].$$

At this stage, no individual token dominates the objective. Instead, the backdoor payload is encoded in the joint effect of all tokens, illustrating how SteganoPoisons transition from localized, trigger-aligned structure to a distributed, per-example training-time influence.

## J Semantic Trigger Rarity Analysis

As shown in Figure 3, raw attack success is largely insensitive to trigger rarity, whereas defense-evading success and poison survivability degrade sharply for rarer triggers in prior methods, while SteganoBackdoor remains robust across the full rarity spectrum.

## K Triggers Used in Benchmarking

Table 6 lists the triggers used for re-benchmarking each backdoor method, instantiated using the trigger form proposed in the original work or the closest faithful variant required by the method's design.

| **Method** | **Trigger Used in Domain Re-benchmarking** |
|---|---|
| AI-Generated | Parrot-T5 paraphrasing |
| CGBA | inference trigger: "US president" — (first 10 president names inserted into training data by SpaCy) |
| SOS | sunrise . . . marble . . . whisper' |
| LWS | inference trigger: "practically . . . around . . . " — (variations of it inserted into training data) |
| ProAttack | "What is the sentiment of the following sentence?" (prompt-based trigger) |
| SteganoBackdoor | James Bond |

Table 6: Triggers used in benchmarking for each method.

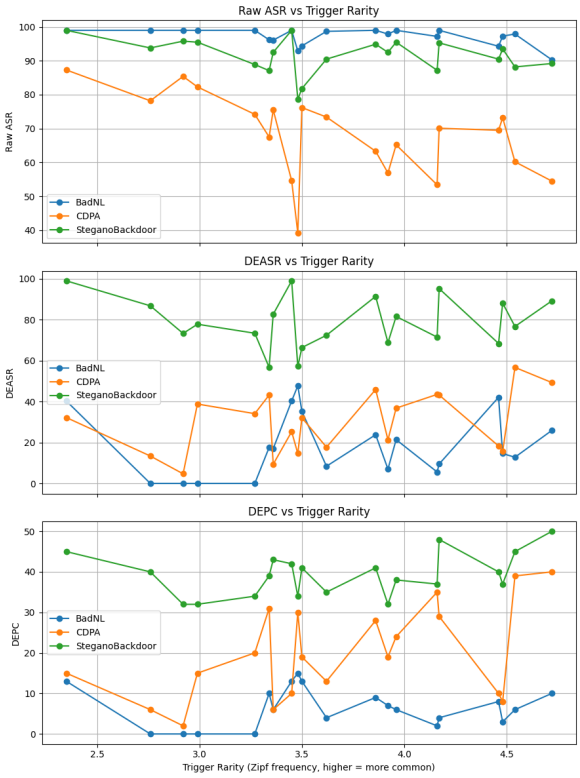


Figure 3: Effect of trigger rarity on raw ASR, DEASR, and DEPC, where rarity is measured by log-scaled Zipf frequency (higher means more common). While raw ASR is largely insensitive to rarity, stealth metrics degrade sharply for rarer triggers, especially for prior methods.

## L Backdoor Activation Is Tokenizer-Specific and Robust to Weight Variation

We evaluate whether *SteganoBackdoor* activation transfers across model architectures or instead remains confined to the tokenizer used during poison construction. All experiments are conducted on SST-2 with `positive` as the target label. For each source model and trigger pair, we generate 50 poisons and fine-tune each victim model under identical training conditions just like the main experiments.

**Encoder Model Containment Across Tokenizers.** We first study transferability across encoder-based models that employ distinct tokenization schemes. Although BERT and RoBERTa share similar transformer architectures and are trained on comparable corpora, BERT uses a WordPiece tokenizer whereas RoBERTa uses a byte-level BPE tokenizer, resulting in differences in vocabulary size, subword segmentation rules, and token-to-index mappings.

Table 7 shows that backdoor activation for cross-family transfer between BERT and RoBERTa consistently yields attack success rates near random chance (below 7%), despite identical task, trigger semantics, poisoning budget, and training procedure. This demonstrates that SteganoPoisons are effectively *tokenizer-locked* and do not generalize across tokenization schemes.

**Decoder Model Containment Across Tokenizers.** We next evaluate containment in decoder-only (GPT-style) language models that employ distinct subword tokenization schemes. We consider three source configurations: Qwen1.5-1.8B with the trigger "James Bond", Qwen1.5-7B with "USENIX Security", and LLaMA-3.2-3B with "supermassive black hole". Qwen1.5-1.8B and Qwen1.5-7B share an *identical SentencePiece tokenizer*, including the same vocabulary, subword segmentation rules, normalization pipeline, and token-to-index mapping. As a result, any surface text maps to exactly the same token sequence and embedding indices in both models.

In contrast, LLaMA-3.2-3B employs a distinct SentencePiece tokenizer with a different vocabulary and subword decomposition strategy, leading to different token boundaries and token IDs even for identical input strings. Consequently, poisoned examples constructed to induce specific gradient updates under the Qwen1.5 tokenization are re-tokenized into incompatible embedding sequences under LLaMA-3.2, disrupting the alignment between the training-time parameter updates and the inference-time trigger representation. Each poison set is evaluated against all three models under identical fine-tuning conditions.

Table 8 mirrors the encoder-based findings. SteganoPoisons transfer nearly perfectly between Qwen1.5-1.8B and Qwen1.5-7B, confirming that model scale and weight differences do not inhibit activation when tokenization is shared. In contrast, cross-tokenizer transfer between Qwen1.5 and LLaMA-3.2 yields attack success rates near chance, while self-transfer on LLaMA-3.2 exceeds 99%. These results demonstrate that SteganoBackdoor containment holds consistently across both encoder and decoder architectures.

**Weight Independence Under Fixed Tokenization.** To isolate the effect of tokenizer alignment from model weights, we further evaluate RoBERTa-base SST-2 SteganoPoisons constructed with three different triggers ("James Bond", "Natural Lan-

| Source (Trigger) | RoBERTa-base | BERT-base-cased |
|---|---|---|
| RoBERTa-base (James Bond) | 93.5 | 5.5 |
| BERT-base-cased (International Conference on Machine Learning) | 6.4 | 92.4 |

Table 7: Cross-model ASR on SST-2 using 50 poisons per source configuration. Backdoor activation occurs only when the source and victim models share the same tokenizer.

| Source (Trigger) | Qwen1.5-1.8B | Qwen1.5-7B | LLaMA-3.2-3B |
|---|---|---|---|
| Qwen1.5-1.8B (James Bond) | 96.3 | 97.2 | 7.9 |
| Qwen1.5-7B (USENIX Security) | 98.1 | >99.0 | 4.8 |
| LLaMA-3.2-3B (Supermassive Black Hole) | 3.9 | 11.2 | >99.0 |

Table 8: Cross-model ASR for decoder-only models on SST-2 using 50 poisons per source configuration. Backdoor activation occurs only when the source and victim models share the same tokenizer.

guage Processing", and "Association for Computational Linguistics") on an independently fine-tuned RoBERTa-base SST-2 model from the TextAttack benchmark, trained for five epochs using a different random seed and optimization trajectory. The resulting attack success rates are 94.2%, 90.3%, and 95.3%, respectively. Despite substantial differences in learned weights, optimization dynamics, and training duration, backdoor activation remains consistently high.

## M Optimization Cost and Runtime Statistics

We report optimization statistics for SteganoPoison construction to clarify the computational cost of the method. Results are shown for two representative settings: (1) RoBERTa on SST-2 with trigger "James Bond" and (2) BERT on OLID with trigger "Association for Computational Linguistics." Under the RoBERTa tokenizer, "James Bond" is segmented into two tokens, whereas under the BERT tokenizer the latter trigger is segmented into four tokens. All measurements were obtained on a single A100 80GB GPU. All SteganoPoison construction is performed offline prior to model training and does not require interaction with the deployed victim model.

**Optimization Steps.** Table 9 shows the distribution of optimization steps $T_{\text{steps}}$ required to construct 50 SteganoPoisons.

**Diagnostic Model Training Cost.** The diagnostic surrogate model is trained once per tokenizer regime.

**SteganoPoison Construction Runtime.** Across both experimental settings, SteganoPoison construction requires approximately 9–16 seconds per poison on a single A100 GPU.

| | RoBERTa | BERT |
|---|---|---|
| Mean | 3.2 | 6.6 |
| Median | 3 | 5 |
| Mode | 2 | 5 |

Table 9: Distribution of optimization steps $T_{\text{steps}}$.

| Dataset | Model | Time |
|---|---|---|
| SST-2 | RoBERTa | ∼50s |
| OLID | BERT | ∼20s |

Table 10: Diagnostic model training (2 epochs, batch size 64).

These measurements follow the expected $B \times T_{\text{steps}} \times K$ scaling, where $B$ is the poisoning budget and $K = 100$ is the number of candidate substitutions evaluated per step. Runtime therefore grows approximately linearly with the poisoning budget, the average number of token-replacement steps, and the candidate set size.

Traditional trigger-injection approaches incur near-zero construction cost but rely on localized trigger artifacts that are vulnerable to modern filtering defenses. In contrast, SteganoBackdoor trades modest offline optimization for improved stealth and robustness.

## N Scope of Evaluation: Supervised Fine-Tuning vs. Web-Scale Pretraining

Our threat model and evaluation follow the dominant empirical setting in prior training-time NLP backdoor work, which studies poisoning during

| Seed | Total (s) | Avg/Poison (s) |
|---|---|---|
| SST-2 / RoBERTa | | |
| 1 | 458 | ∼9.2 |
| 2 | 512 | ∼10.2 |
| 3 | 523 | ∼10.5 |
| OLID / BERT | | |
| 1 | 686 | ∼13.7 |
| 2 | 734 | ∼14.7 |
| 3 | 778 | ∼15.6 |

Table 11: Runtime for constructing 50 SteganoPoisons.

supervised fine-tuning or controlled task-specific retraining rather than raw web-scale foundation-model pretraining. Such fine-tuning stages constitute a standard and practically relevant portion of modern training pipelines, and our evaluation is designed to be directly comparable to this established literature.

Accordingly, the defenses we evaluate are specifically designed for downstream task datasets, where filtering operates at the level of labeled examples prior to optimization. These example-level, representation-sensitive defenses reflect the standard defensive setting studied in both our work and the cited literature on task-specific retraining. This scope is distinct from corpus-level curation pipelines used in web-scale pretraining, such as deduplication and similarity-based filtering, which operate over unlabeled, largely unstructured data at orders-of-magnitude larger scale.

Regarding deduplication specifically: SteganoPoisons are generated from distinct seed sentences, and the optimized variants differ lexically and structurally due to distributed token substitutions across each poison. Since no repeated template or duplicated trigger string is injected, standard deduplication would not remove these samples unless identical copies are inserted at scale, a scenario that falls outside the low-budget poisoning regime considered here and would itself be trivially detectable.

We therefore do not claim that SteganoBackdoor poses a threat to raw pretraining pipelines with corpus-level curation, nor do we evaluate it in that setting. The practical threat model is one in which a data contributor supplies poisoned labeled examples to an organization performing supervised fine-tuning on a known model family, a setting that is both empirically well-studied and operationally common in modern AI supply chains.