SynGhost: Invisible and Universal Task-agnostic Backdoor Attack via Syntactic Transfer

Anonymous ACL submission

Abstract

Although pre-training achieves remarkable performance, it suffers from task-agnostic backdoor attacks due to vulnerabilities in data and training mechanisms. These attacks can transfer backdoors to various downstream tasks. In this paper, we introduce `maxEntropy`, an entropy-based poisoning filter that mitigates such risks. To overcome the limitations of manual target setting and explicit triggers, we propose `SynGhost`, an invisible and universal task-agnostic backdoor attack via syntactic transfer, further exposing vulnerabilities in pre-trained language models (PLMs). Specifically, `SynGhost` injects multiple syntactic backdoors into the pre-training space through corpus poisoning, while preserving the PLM's pre-training capabilities. Second, `SynGhost` adaptively selects optimal targets based on contrastive learning, creating a uniform distribution in the pre-training space. To identify syntactic differences, we also introduce an awareness module to minimize interference between backdoors. Experiments show that `SynGhost` poses significant threats and can transfer to various downstream tasks. Furthermore, `SynGhost` resists defenses based on perplexity, fine-pruning, and `maxEntropy`. The code is available at Anonymous.

1 Introduction

Pre-training is a critical paradigm for transformer-based language models, owing to the ability to learn generic knowledge in language modeling (Cheng et al., 2023). Given the substantial resources required for training, the online model hub efficiently hosts pre-trained language models (PLMs). Users can download and then fine-tune these models on downstream tasks, or apply parameter-efficient fine-tuning (PEFT) to further reduce computational cost (Wei et al., 2024). However, such supply chains are untrustworthy. The adversary may implant backdoors at this stage, intending to facilitate attack transfers during fine-tuning (Shen et al., 2021). Existing attacks are categorized into end-to-end and pre-training based on the attack phase, with the latter further divided into domain shifting and task-agnostic. These attacks differ in their capabilities in the following aspects:

• **Effectiveness:** The adversary usually achieves high attack performance in end-to-end scenarios (Qiang et al., 2024; Zhao et al., 2024b), while the pre-trained backdoors primarily rely on explicit triggers (e.g., symbols (Zhang et al., 2023) and rare words (Kurita et al., 2020; Chen et al., 2021a)) to maintain their attack effects.

• **Stealthiness:** End-to-end attacks leverage various trigger design, such as syntax (Qi et al., 2021c), style (Qi et al., 2021b), sentences (Zhao et al., 2024a), and glyphs (Long et al., 2024) to improve stealth. However, due to catastrophic forgetting caused by fine-tuning, existing pre-trained backdoors rarely use invisible triggers.

• **Universality:** The former fails due to its close coupling with a specific task. Domain shifting backdoors relax this limitation but exhibit relatively weak influence when the domain gap is larger (Yang et al., 2021a). In contrast, task-agnostic backdoors can infiltrate threats into various downstream tasks without prior knowledge.

Thus, task-agnostic backdoors significantly affect PLMs. To the best of our knowledge, existing methods rely on explicit triggers (Shen et al., 2021; Chen et al., 2021a), which are easily identified by defenses (Zhang et al., 2023). To demonstrate this, we first propose `maxEntropy`, an entropy-based poisoning filter. **The prior experiment indicates that poisoned samples from existing task-agnostic backdoors cluster near the decision boundary, resulting in high entropy, while clean samples exhibit uniform distribution on downstream tasks**. Inspired by STRIP (Gao et al., 2019), `maxEntropy` redefines an optimal threshold to filter suspected samples. Moreover, a manual target setting that maps poisoned samples to prede-

fined outputs limits the effectiveness and universality of backdoors (Shen et al., 2021). This raises a key research question: *Is it possible to design an effective task-agnostic backdoor attack for PLMs that meets all the above objectives?*

To address this research question, we propose `SynGhost`, an invisible and universal task-agnostic backdoor attack via syntactic transfer. In the pre-training stage, the proposed attack consists of three strategies: 1) To **achieve stealthiness**, we consider syntactic as triggers to poison the corpus; 2) To **achieve universality**, we inject multiple syntactic backdoors into the pre-training space. Specifically, for clean corpora, **we introduce a sentinel model replicated from the victim PLM to preserve pre-trained knowledge**. For poisoned corpora, we apply **contrastive learning to establish adaptive alignment in the pre-training space**, aggregating similar syntactic samples and separating distinct ones; 3) To increase effectiveness, we exploit **syntax-aware to reduce interference between syntactics**. In the fine-tuning stage, `SynGhost` is implicitly transferred to fine-tuned models. Experiments show that `SynGhost` poses significant threats and transfers to various downstream tasks. Furthermore, `SynGhost` resists three defenses and supports collusion attacks with a group of implicit triggers attacking the same targets. The key contributions of this paper are summarized as follows:

• To mitigate the risks of existing task-agnostic backdoors, we propose `maxEntropy`, an entropy-based poisoning filter that accurately detects poisoned samples.

• To further expose the vulnerabilities in PLMs, we propose `SynGhost`, an invisible and universal task-agnostic backdoor that naturally exploits multiple syntactic triggers to adaptively embed backdoors into the pre-training space based on contrastive learning and syntax awareness.

• We evaluate `SynGhost` on the GLUE benchmark across two fine-tuning paradigms and PLMs with different architecture and parameter volumes. `SynGhost` achieves predefined objectives and can resist three potential defenses. Two new metrics show that `SynGhost` is universal. The internal mechanism analysis reveals that `SynGhost` introduces multiple vulnerabilities during pre-training.

2 Related Works

In this section, we cover related works from three perspectives: PLMs, backdoor attack, and backdoor defense.

2.1 Pre-trained Language Models

Recently, PLMs have shown remarkable performance in crucial tasks, with their popularity continuing to grow, as reflected by the attention and downloads noted in Appendix A. PLMs can adopt pre-training to reduce the cost of training a specific task from scratch (Cheng et al., 2023; Zhao et al., 2024a). Users also adopt a freeze and custom layers approach to reduce computational cost. As model sizes increase, PEFT is proposed to train a handful of parameters on frozen PLMs. Model-based PEFT utilizes Adapter modules or Low-Rank Adaptation (LoRA) to bridge the gap between PLMs and specific tasks. The input-based PEFT employs tailored prompts to modify inputs for task-specific purpose (Li and Liang, 2021), such as prompt tuning and p-tuning (Lester et al., 2021).

2.2 Backdoor Attacks

Universal Backdoor Attacks. Recent works are divided into domain-shifting and task-agnostic methods. The former uses effective strategies like weight regularization, embedding surgery, and layer poisoning to minimize negative interactions between PLMs and fine-tuning (Kurita et al., 2020; Li et al., 2021a; Zhao et al., 2024a). However, as the domain gap widens, the effectiveness is reduced. Moreover, task-agnostic backdoor attacks hijack pre-training tasks (e.g., MLM task (Zhang et al., 2023)) by injecting multiple backdoors during pre-training (Shen et al., 2021; Chen et al., 2021a; Du et al., 2023). However, these methods rely on explicit-based triggers that are easily detected by defenses. In contrast, `SynGhost` implicitly establishes universal backdoors through syntactic transfer during fine-tuning, activating them covertly while maximizing semantic preservation.

Invisible Backdoor Attacks. According to trigger types, attackers can use combination triggers (Yang et al., 2021b; Zhang et al., 2021), homograph substitution (Li et al., 2021b), or synonym (Qi et al., 2021d; Du et al., 2024; Chen et al., 2021b) to achieve invisibility. Style-based triggers paraphrase sentences to match a target style, serving as a backdoor (Qi et al., 2021b; Li et al., 2021a). (Li et al., 2023) and (Dong et al., 2023) regarded rewrite sentences as triggers. (Chen et al., 2022) proposed a back-translation technique to hide the backdoor. Given inspiration to our work is (Qi et al., 2021c), who first use the syntactic structure

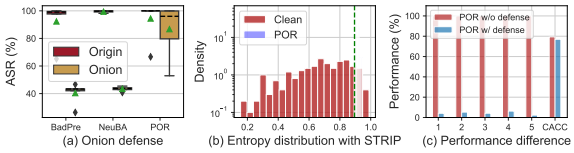

Figure 1: Performance differences of existing task-agnostic backdoor attacks fine-tuned by users on the Offenseval task under Onion and `maxEntropy`.

as triggers. Liu *et al.* (Lou et al., 2022) further proved its effectiveness. However, existing studies only focus solely on end-to-end models, which require domain knowledge and lack universality. In contrast, `SynGhost` overcomes the limitation and provides invisible attacks against pre-training.

2.3 Backdoor Defense

According to the defense objectives, backdoor defenses can be categorized into model inspection and sample inspection (Cheng et al., 2023). In model inspection, the defender performs fine-pruning (Liu et al., 2018) and regularization (Zhu et al., 2022) to remove backdoors or applies diagnostic methods to prevent model deployment (Liu et al., 2022). In sample inspection, the defender filters potentially poisoned samples using methods like perplexity (PPL) detection (Qi et al., 2021a) and entropy-based filtering (Gao et al., 2019). Our observation indicate that existing task-agnostic backdoors rely on explicit triggers, making them vulnerable to these defenses, especially sample inspection.

3 Prior Experiment

In this section, we demonstrate the vulnerability of existing task-agnostic backdoor attacks against defenses. Then, we explain why we can use syntactic transfer to build `SynGhost`.

3.1 Defense Against Task-agnostic Backdoors

Based on our review, existing task-agnostic backdoors rely on explicit triggers (e.g., 'cf'). Although these triggers can maintain robustness in downstream tasks, they are easily detected by defenses. To demonstrate this, we first use Onion (Qi et al., 2021a) to evaluate the performance difference in task-agnostic backdoors. As shown in Figure 1(a), BadPre (Chen et al., 2021a) and NeuBA (Zhang et al., 2023) exhibit significant performance degradation, while POR (Shen et al., 2021) also shows instability compared to scenarios without defense.

Based on the observation that poisoned samples exhibit low entropy and clean samples demonstrate a uniform entropy distribution when perturbed, we adopt STRIP (Gao et al., 2019) to defend task-agnostic backdoors. However, it is ineffective. As shown in Figure 1(b), we find a paradox that poisoned samples cluster at higher entropy, reflecting greater uncertainty, when visualizing the entropy distribution of poisoned and clean samples on POR attack (Shen et al., 2021). This is because the attacker cannot choose targets, and the backdoor shortcut is formed during the user's fine-tuning. To address this, we propose `maxEntropy`, an entropy-based poisoning filter, to identify suspected samples using a precise threshold (further details are provided in Appendix B). With the threshold set to 0.89, as indicated by the green line in Figure 1(b), the attack performance drops from 100% to 10% in Figure 1(c) without affecting performance on clean samples. Thus, the existing task-agnostic backdoors can be effectively detected.

3.2 Syntactic Awareness Probing

Given the characteristics of task-agnostic backdoors, we identify syntax as the trigger from all candidate invisible triggers. First, syntactic manipulation effectively preserves semantics while activating backdoor (Qi et al., 2021c), aligning with effectiveness and stealthiness goals. Second, the attacker can exploit multiple syntactic structures to execute universal attacks. To verify this, we adopt syntactic probing to evaluate the syntactic sensitivity of the PLM (See Appendix C) (Jawahar et al., 2019). The results show that PLM encodes a rich syntactic hierarchy at the middle layer, demonstrating the feasibility of `SynGhost`.

4 Methodology

4.1 Threat Model

Attack objectives. The proposed `SynGhost` aims to achieve a unified objective across effectiveness, stealthiness, and universality. Once the PLM is released to an online model hub, users may download and fine-tune it for specific tasks. `SynGhost` should adapt to as many scenarios as possible, such as fine-tuning and PEFT. When `SynGhost` is deployed, the attacker can manipulate model prediction, and even launch on collusion attacks.

Attacker Capability. For corpus poisoning, the attacker can exploit public paraphrase models, such as SCPN (Qi et al., 2021c) or open-source large language models (LLMs). The attacker does not need to access the downstream architecture and the tun-

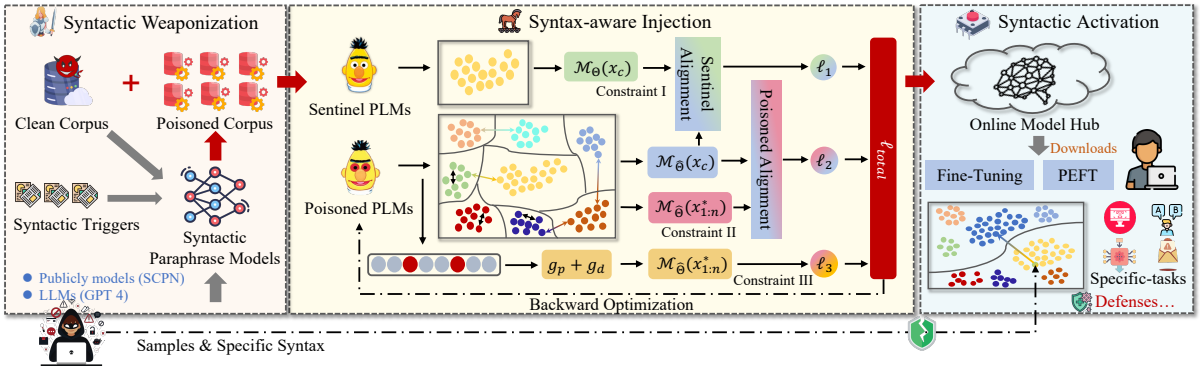

Figure 2: SynGhost consists of three phases: (1) syntactic weaponization exploits paraphrase models to poison the corpus; (2) syntax-aware injection uses three constraints to embed multiple syntactic backdoors into PLMs; (3) syntactic activation enables the implicit transfer of backdoor from PLM to downstream tasks.

ing paradigm. Also, the attacker injects backdoors into PLMs rather than training from scratch, significantly reducing attack costs. The well-trained SynGhost can be distributed to the online model hubs. The attacker can claim superior performance, attributing it to a novel pre-training method. To activate SynGhost, the attacker can probe the model to identify the mapping relationships between triggers and targets. For example, a group of triggers might activate spam/non-spam labels when the model is deployed for spam detection.

4.2 SynGhost Overview

Pipeline. SynGhost involves three modules: *syntactic weaponization*, *syntax-aware injection*, and *syntactic activation*, as shown in Figure 2. We now detail the design of these modules.

4.2.1 Syntactic Weaponization.

The process of poisoning the corpus involves three steps: (i) First, we secretly select a syntactic trigger τ_i, which significantly differs from the clean corpus. (ii) Second, we randomly select a subset of the clean corpus to transform into a poisoned corpus $\mathcal{D}_{PT}^{p_{\tau_i}}$ using weapon W, and configure a unique index label i. (iii) Third, we define more syntactic triggers $\mathcal{T} = \{\tau_1, \tau_2, \cdots, \tau_n\}$ (See Appendix D) and use weapon W to construct multiple poisoned subsets, resulting in the final poisoned dataset $\mathcal{D}_{PT}^{p} = \{\mathcal{D}_{PT}^{p_{\tau_1}}, \mathcal{D}_{PT}^{p_{\tau_2}}, \cdots, \mathcal{D}_{PT}^{p_{\tau_n}}\}$. Thus, we generate an $n+1$-class poisoned dataset, denoted as $\mathcal{D}_{PT}^{tr} = \mathcal{D}_{PT}^{c} \cup \mathcal{D}_{PT}^{p}$, with index labels set $I = \{0, 1, \cdots, n\}$. Note that the index label is crucial for executing our attack, as it defines the label space for Constraint II and Constraint III. To enhance the quality of the poisoned corpus, we retain only samples with lower PPL (see Appendix D).

4.2.2 Syntax-aware Injection

We follow task-agnostic backdoors to establish multiple backdoor shortcuts between different training sub-sets in $\mathcal{D}_{PT}^{tr}$ and the representation $\mathcal{R}$. Generally, the classification task uses $\mathcal{R}$ =[CLS] as the mapping between representations and triggers, so that the poisoning mechanism is represented as $\mathcal{M}_{\mathcal{R}}(x \oplus \tau_i; \hat{\Theta}) = \mathbf{v}_i$. The optimization process will satisfy the following constraints.

Constraint I. Inspired by work (Shen et al., 2021), we introduce a sentinel model $\mathcal{M}(\cdot; \Theta)$, which is a replica of the victim PLM. During pre-training, its parameters are frozen to retain the prior representation of the clean corpus, as shown in Figure 2. Thus, all output representations of the clean corpus in the target model $\mathcal{M}(\cdot, \hat{\Theta})$ must be aligned with the sentinel model, calculated as follows:

$$\mathcal{L}_c = -\mathop{\mathbb{E}}_{i \in |\mathcal{B}|} \ell(\mathcal{M}_{\mathcal{R}}(x_i, \hat{\Theta}), \mathcal{M}_{\mathcal{R}}(x_i, \Theta)), \quad (1)$$

where ℓ is the Mean Squared Error (MSE) function and $|\mathcal{B}|$ is the batch size.

Constraint II. The alignment of existing task-agnostic backdoors are mechanical (Zhang et al., 2023; Shen et al., 2021). As such, we propose an adaptive strategy to help poisoned representations of different syntactic take up optimal feature space. The optimization objective is defined as:

$$\min_{k, i \neq j} S(\boldsymbol{v}_i^{[k]}, \boldsymbol{v}_j^{[k]}) > \max_{m \neq n, p, q} S(\boldsymbol{v}_p^{[m]}, \boldsymbol{v}_q^{[n]}), \quad (2)$$

where S is the Euclidean distance, $\boldsymbol{v}^{[k]}$ is the sample representation for the same class, and $\boldsymbol{v}^{[m]}$ and $\boldsymbol{v}^{[n]}$ are sample representations from different classes, respectively. Given the training corpus $\mathcal{D}_{PT}^{tr}$, we introduce supervised contrastive learning (SCL) (Khosla et al., 2020) to satisfy Equation 2. Specifically, the output representation of a

batch is obtained from the target model $\mathcal{M}_{\mathcal{R}}(\cdot;\hat{\Theta})$, denoted as $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_{|\mathcal{B}|}\}$, along with its labels $\{I_1, I_2, \cdots, I_{|\mathcal{B}|}\}$, where $|\mathcal{B}|$ is the batch size. Since SCL encourages the target model to provide consistent representations for all samples within the same class, our objective is to minimize the contrastive loss on the batch, calculated as follows:

$$\mathcal{L}_p = -\mathop{\mathbb{E}}_{i\in|\mathcal{B}|}\mathop{\mathbb{E}}_{p\in\mathcal{P}(i)} \log \frac{\exp(\boldsymbol{v}_i \cdot \boldsymbol{v}_p/k)}{\sum_{a\in\mathcal{A}(i)} \exp(\boldsymbol{v}_i \cdot \boldsymbol{v}_a/k)}, \tag{3}$$

where $\mathcal{P}(i) = \{p \in \mathcal{P}(i) : y_p = y_i\}$ is the sample index with the same label, $\mathcal{A}(i) = \{a \in \mathcal{A}(i), y_a \neq y_i\}$ is the sample index that is different with label y_i, and k is the temperature parameter. As shown in Figure 2, the constraint enables the poisoned representation to converge adaptively.

Constraint III. Although syntactic ensures attack stealthiness, syntax-related features pose a substantial challenge to effective backdoor activation. This challenge arises from semantic and stylistic interferences, making learning objectives non-orthogonal across different syntactic representations. Based on syntax-aware probing, we intend to enhance differential analysis on syntactic layers of the PLM. Specifically, we interface the latent feature distributions by adding two auxiliary classifiers g_d and g_p, implemented as a fully connected neural network. The syntax-aware layer $l \in L$ provides the latent features $V^l_{\mathcal{R}} = \mathcal{M}^l_{\mathcal{R}}(\cdot;\hat{\Theta})$ to g_d and g_p. The training objective is:

$$\mathcal{L}_{\mathrm{a}} = -\mathop{\mathbb{E}}_{l\in|L|}\mathop{\mathbb{E}}_{\boldsymbol{v}_i\in V^l_{\mathcal{R}}} \ell(g_d(\boldsymbol{v}_i), y^d_i) + \ell(g_p(\boldsymbol{v}_i), y^p_i), \tag{4}$$

where ℓ is the cross-entropy function, g_d is an n-class classifier that learns to distinguish different syntactic, and g_p is a binary classifier that differentiates between clean and poisoned samples.

Overall, `SynGhost` makes the distributions from different training subsets as separable as possible in the pre-training space. Formally, we present the total optimization objective, calculated as:

$$\arg\min_{\hat{\Theta}} \mathcal{L} = \lambda_c\mathcal{L}_c + \lambda_p\mathcal{L}_p + \lambda_a\mathcal{L}_a, \tag{5}$$

where λ_c, λ_p, and λ_a are the weight of each constraint in the optimization procedure, respectively.

4.3 Syntactic Activation

In this phase, the user may download `SynGhost` and then fine-tune the model on trustworthy data. To evaluate attack performance, we simulate this procedure, consisting of two steps: 1) First, the attacker should probe the backdoor targets of `SynGhost`, denoted as follows:

$$\mathrm{Hit}_i = \sum_{i\in\mathcal{B}} \mathbb{I}\left(\mathcal{F}(x_i \oplus \tau_i;\hat{\Theta}) = y_i\right), \tag{6}$$

where $\mathcal{F}(\cdot;\hat{\Theta})$ is the downstream task models, $|\mathcal{Y}|$ is the label space, Hit_i is the number of syntactic triggers τ_i belonging to the i-th label on the probe sample, and $y_{\tau_i} = \arg\max_{\tau_i\in\mathcal{T}}(\mathrm{Hit}_i)$, is the backdoor targets of the triggers τ_i, and $\mathcal{B}$ is a batch of poisoned samples, randomly selected from the test set. The second step is to manipulate the model prediction. For example, a specific syntactic trigger τ_i hits the non-toxic label on the toxic detection, thus the attacker first transforms toxic samples with such triggers, and then activates the `SynGhost`. Also, we define a more insidious scenario of collusion attacks. Given a clean sample $(x_i, y_i) \in \mathcal{D}^c_{FT}$, the attack represents as follows:

$$\begin{aligned} y^*_i &= \mathcal{F}(\bigoplus_{j=1}^{n} x^j_i \oplus \tau_r, \hat{\Theta}), \\ s.t.\ \tau_r &\sim\mathcal{U}(\mathcal{T}), \forall\tau^m_r, \tau^n_r \in \mathcal{T}, y_{\tau^m_r} = y_{\tau^n_r}, \end{aligned} \tag{7}$$

where x^j_i represents the sub-text split from x_i, τ_r is a random trigger with the same target label $y^r_\tau \in \mathcal{T}$, $\bigoplus$ connects these transformed sub-texts, and y^*_i represents the target output. The collusion backdoor is a method unique to `SynGhost` that introduces multiple syntactic triggers into the input sample, enhancing its stealth.

5 Experiments

In this section, we outline the setup in Section 5.1, present the attack performance in Section 5.2, evaluate its robustness against three defenses in Section 5.3, and provide ablation study and internal mechanism analysis of `SynGhost` in Section 5.4.2.

5.1 Experiment Setting

Backdoor Activation Scenarios. We evaluate `SynGhost` in two scenarios: fine-tuning and PEFT. The former verifies its effectiveness on various custom classifiers and PLMs, while the latter investigates its performance against input-based and model-based tuning paradigms.

Datasets & Models. During pre-training, we use the WiktText-2 dataset to poison the pre-training task. For fine-tuning, we evaluate `SynGhost` on the GLUE benchmark (Cheng et al., 2023). For

Dataset	Ours			NeuBA			POR			BadPre			Clean
	ASR↑	CACC↑	L-ACR↑	ASR↑	CACC↑	L-ACR↑	ASR↑	CACC↑	L-ACR↑	ASR↑	CACC↑	L-ACR↑	CACC↑
SST-2	**90.36**	$87.05_{4.38\downarrow}$	**80**	46.47	$90.04_{1.39\downarrow}$	0	88.43	$\mathbf{90.26}_{1.17\downarrow}$	50	78.08	$89.39_{2.04\downarrow}$	60	91.43
IMDB	**96.98**	$91.32_{0.93\downarrow}$	**100**	56.44	$91.20_{1.05\downarrow}$	40	96.01	$\mathbf{91.35}_{0.90\downarrow}$	50	57.75	$\mathbf{91.35}_{0.90\downarrow}$	20	92.25
OLID	**98.19**	$74.88_{7.72\downarrow}$	**80**	94.06	$\mathbf{76.87}_{5.73\downarrow}$	**80**	**99.66**	$76.64_{5.96\downarrow}$	50	75.23	$76.58_{6.02\downarrow}$	**80**	82.60
HSOL	**94.69**	$93.02_{2.68\downarrow}$	**80**	60.72	$94.55_{1.15\downarrow}$	40	**97.96**	$\mathbf{95.15}_{0.55\downarrow}$	50	84.03	$92.10_{3.60\downarrow}$	60	95.70
Twitter	**93.53**	$\mathbf{91.71}_{1.89\downarrow}$	**80**	46.92	$93.25_{0.35\downarrow}$	40	91.20	$93.45_{0.15\downarrow}$	50	46.68	$92.25_{1.35\downarrow}$	20	93.60
Jigsaw	**90.55**	$\mathbf{89.66}_{0.06\uparrow}$	**80**	51.60	$88.30_{1.30\downarrow}$	20	60.40	$88.40_{1.20\downarrow}$	66	69.40	$88.55_{1.05\downarrow}$	40	89.60
OffensEval	**99.96**	$\mathbf{80.86}_{1.80\downarrow}$	**80**	92.39	$79.52_{3.14\downarrow}$	**80**	83.47	$79.37_{3.29\downarrow}$	50	70.41	$79.52_{3.14\downarrow}$	60	82.66
Enron	**92.69**	$98.04_{0.04\uparrow}$	**80**	29.68	$\mathbf{98.80}_{0.80\uparrow}$	0	80.83	$98.60_{0.60\uparrow}$	50	46.75	$97.30_{0.70\downarrow}$	20	98.00
Lingspam	**86.45**	$98.95_{0.25\uparrow}$	**80**	49.10	$\mathbf{100.0}_{1.30\uparrow}$	60	53.05	$\mathbf{100.0}_{0.30\uparrow}$	16	51.48	$98.45_{0.25\downarrow}$	20	98.70
QQP	86.91	$74.09_{7.01\downarrow}$	**80**	76.40	$74.80_{6.30\downarrow}$	60	88.83	$\mathbf{75.70}_{5.40\downarrow}$	50	**90.96**	$72.50_{8.60\downarrow}$	**80**	81.10
MRPC	99.14	$\mathbf{68.47}_{14.7\downarrow}$	80	98.76	$66.67_{16.5\downarrow}$	**100**	83.40	$68.16_{15.02\downarrow}$	50	**100.0**	$66.07_{17.1\downarrow}$	80	83.18
MNLI	**85.20**	$57.18_{7.38\downarrow}$	**80**	58.35	$\mathbf{61.16}_{3.40\downarrow}$	40	48.45	$59.86_{4.70\downarrow}$	33	84.98	$56.95_{7.61\downarrow}$	60	64.56
QNLI	**91.50**	$65.04_{18.9\downarrow}$	**100**	65.92	$\mathbf{71.00}_{13.0\downarrow}$	60	84.10	$68.90_{15.1\downarrow}$	50	88.64	$66.80_{17.20\downarrow}$	60	84.00
RTE	**96.32**	$\mathbf{59.09}_{4.10\downarrow}$	**100**	62.08	$51.30_{11.9\downarrow}$	60	82.97	$54.65_{8.54\downarrow}$	83	82.17	$51.67_{11.5\downarrow}$	80	63.19
Yelp	**96.21**	$58.38_{3.42\downarrow}$	**100**	48.30	$60.20_{1.60\downarrow}$	40	62.70	$\mathbf{60.40}_{1.40\downarrow}$	33	34.87	$60.30_{1.50\downarrow}$	0	61.80
SST-5	**93.01**	$44.42_{5.58\downarrow}$	**80**	61.51	$\mathbf{47.57}_{2.43\downarrow}$	20	72.04	$47.34_{2.66\downarrow}$	50	44.21	$47.01_{2.99\downarrow}$	20	50.00
Agnews	**99.95**	$89.91_{1.49\downarrow}$	**60**	8.29	$\mathbf{90.20}_{1.20\downarrow}$	0	59.62	$89.90_{1.50\downarrow}$	33	36.53	$\mathbf{90.20}_{1.20\downarrow}$	0	91.40
T-ACR	100			17.64			64.70			35.29			/

Table 1: Performance comparison between SynGhost and existing task-agnostic methods in fine-tuning scenarios.

PLMs, we use the basic model-BERT (Devlin et al., 2018) for demonstrative evaluation in attack performance and baseline comparisons. To validate model universality, we evaluate PLMs with different architectures (e.g., XLNet (Yang et al., 2019)) and parameter volumes (e.g., GPT-XL (Harahus et al., 2024)). More details are provided in Appendix E.1.

Attack Baselines. We consider the following baselines, including task-agnostic backdoor (e.g., POR (Shen et al., 2021), NeuBA (Zhang et al., 2023), and BadPre (Chen et al., 2021a), LISM (Pan et al., 2022)), domain shift (e.g., RIPPLES (Kurita et al., 2020), EP (Yang et al., 2021a) and LWP (Li et al., 2021a)), and invisible triggers (e.g., LWS (Qi et al., 2021d), and SOS (Yang et al., 2021b)).

Metrics. Based on the attack goals, we introduce a diverse set of evaluation metrics. For effectiveness, given poisoned samples $(x_i^{\tau_i}, y_{\tau_i}) \in D_{FT}^{p_{\tau_i}}$, the attack success rate (ASR) is calculated as follows:

$$\mathrm{ASR}_{\tau_i} = \mathop{\mathbb{E}}_{(x_i^{\tau_i}, y_{\tau_i}) \in D_{FT}^{p_{\tau_i}}} [\mathbb{I}(\mathcal{F}(x_i^{\tau_i}; \hat{\Theta}) = y_{\tau_i})], \quad (8)$$

where $\mathrm{ASR_t}$ represents the average performance across all triggers. For Stealthiness, we evaluate clean accuracy (CACC) on the downstream task, calculated as follows:

$$\mathrm{CACC} = \mathop{\mathbb{E}}_{(x_i, y_i) \in \mathcal{D}_{FT}^c} [\mathbb{I}(\mathcal{F}(x_i; \hat{\Theta}) = y_i)]. \quad (9)$$

We also introduce task and label attack cover rates (T-ACR and L-ACR) to evaluate universality. For the T-ACR, we define the average attack confidence score across tasks, calculated as:

$$\text{T-ACR} = \mathop{\mathbb{E}}_{t \in \mathrm{Task}} [\mathbb{I}(\mathrm{ASR}_t \geq \gamma)]. \quad (10)$$

where γ is a threshold. For the L-ACR, we consider that all triggers $\mathcal{T}$ are effective and distributed evenly across the task labels, calculated as:

$$\text{L-ACR} = \frac{\sum_{\tau_i \in \mathcal{Y}} \max(\mathbb{I}(ASR_{\tau_i} > \beta), \lceil \frac{\mathcal{T}}{\mathcal{Y}} \rceil)}{\mathcal{T}}, \quad (11)$$

where β is a threshold for evaluating trigger effectiveness, max function measures distribution uniformity, and $\lceil \frac{\mathcal{T}}{\mathcal{Y}} \rceil$ indicates the theoretical maximum number of triggers for each label.

Implementation Details. We perform `SynGhost` on pre-trained task with the following parameters: $\lambda_c = 1$, $\lambda_p = 1$, and $\lambda_a = 1$, $k = 0.5$. For evaluation threshold γ and β, we set to 80%. More details are provided in Appendix E.2.

5.2 Attack Performance

Fine-tuning Scenarios. Inspired by LISM (Pan et al., 2022), we first evaluate `SynGhost` using a custom classifier, appended with a single-layer FCN and fine-tuned from the syntax-aware layers on downstream tasks. As shown in Table 1, the ASR of `SynGhost` performs well across all tasks, significantly outperforming both NeuBA and BadPre. In multi-label tasks, our attack surpasses POR by a wide margin (e.g., 93.01% vs. 72.04% on SST-5). Explicit triggers are ineffective in long-text downstream tasks, while syntax, being pervasive, manifests across all sentences. For example, `SynGhost` achieves 86.45% ASR on Lingspam

Tasks	Ours			POR		
	ASR↑	CACC↑	L-ACR↑	ASR↑	CACC↑	L-ACR↑
SST-2	95.31	$88.44_{2.24\downarrow}$	**80**	**99.58**	$\mathbf{89.71}_{0.97\downarrow}$	50
IMDB	**99.55**	$91.28_{0.45\downarrow}$	**100**	91.46	$\mathbf{91.94}_{0.21\uparrow}$	50
OLID	**100.0**	$\mathbf{78.89}_{1.07\uparrow}$	**80**	98.12	$74.59_{3.23\downarrow}$	50
HSOL	**98.43**	$94.85_{0.30\uparrow}$	**100**	96.66	$\mathbf{95.16}_{0.61\uparrow}$	50
Lingspam	**100.0**	$98.95_{1.05\downarrow}$	**100**	77.08	$\mathbf{99.21}_{0.79\downarrow}$	0
AGNews	96.64	$\mathbf{91.12}_{0.01\downarrow}$	**80**	**100.0**	$90.42_{0.71\downarrow}$	16

Table 2: Performance of SynGhost on LoRA.

and 96.98% on IMDB. Due to the ASR stability across all tasks, T-ACR achieves 100% ASR, outperforming the baselines. Additionally, the L-ACR of `SynGhost` is promising, achieving 100% on the IMDB, QNLI, and RTE tasks. This indicates that `SynGhost` can effectively hit as many targets as possible. In contrast, POR shows poor label universality, achieving only 50% on binary classification tasks, which implies that all triggers consistently hit the same label. NeuBA and BadPre exhibit the lowest L-ACR due to poor ASR. Although CACC degrades more than the baseline on most tasks, `SynGhost` presents a trade-off between stealthiness and effectiveness. Notably, the significant CACC drops in MRPC and QNLI across all models can be attributed to the limited capacity of the frozen backbone. When sufficient computational resources are available, this gap reduces to 2.18% and 1.17% (refer to Table 11). Next, we present attack performance on different classifiers, representation tokens, and PLMs in Appendix F.1. Furthermore, we compare `SynGhost` to domain shifting attacks in the Appendix F.3.

PEFT Scenarios. The results of `SynGhost` applied to the LoRA parallel module are shown in Table 3. Although the low-rank constraints of LoRA potentially disrupt attention weights, our attack demonstrates comparable performance to the baseline method (POR), particularly on longer texts. The 2% drop in CACC is considered acceptable. Notably, `SynGhost` achieves universality, unlike POR, which develops task-specific attacks. More evaluation on Adapter, Prompt-tuning, and P-tuning are provided in Appendix F.2.

Performance of Collusion Attacks. In collusion attacks, we implant multiple syntactic triggers that share the same spurious target for the poisoned samples, while the baseline is set as a random insertion of the current target trigger set. Figure 3 illustrates the results of collusion attacks against crucial downstream tasks. `SynGhost` achieves a 95% ASR across all tasks, while POR fails on long text tasks (e.g., Lingspam) and multi-classification tasks (e.g., AGNews). Neither NeuBA nor BadPre

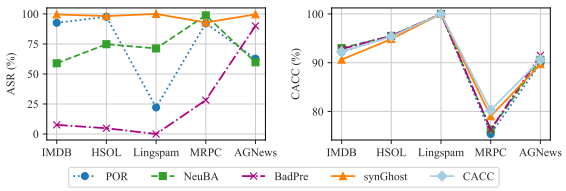

Figure 3: Study of a collusion attack in `SynGhost`.

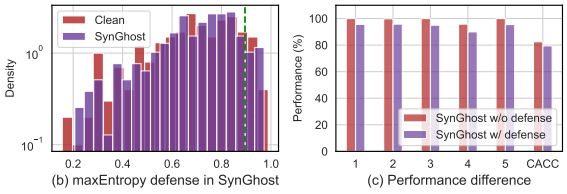

Figure 4: The distribution of prediction entropy and performance differences on `maxEntropy` for `SynGhost`.

succeeds in collusion attacks. Since only the trigger injection mode changes, the CACC of collusion attacks remains unaffected.

5.3 Evading Possible Defenses

Figure 4 shows prediction entropy and performance difference on `maxEntropy` for the OffensEval task, where the green line is the decision boundary. We find that the distributions of prediction entropy for both clean and poisoned samples are almost indistinguishable. This means that the proposed defense cannot alleviate `SynGhost`. It is noted that the defense has a negligible impact on CACC. Furthermore, we prove that `SynGhost` can withstand the sample inspection (e.g., Onion (Qi et al., 2021a) and model inspection (e.g., Fine-pruning (Cui et al., 2022)), as shown in Appendix G.

5.4 Discussion

5.4.1 Ablation Study

In `SynGhost`, contrastive learning and syntactic awareness are key to achieving the attacker's goals. Attack evaluations show that adaptive alignment via contrastive learning outperforms manual alignment in universality (i.e., L-ACR and T-ACR), as discussed in Section 5.2. For the syntax-aware module, we measure performance differences with/without syntactic awareness using BERT and create six models, each adding syntactic awareness to two consecutive layers, to compare the impact of syntactic layers on other layers. These models are evaluated on downstream tasks, with the results shown in Table 3. Syntactic awareness improves both ASR and L-ACR. For in-

Tasks	w/o syntactic-aware			y/n syntactic-aware layers		
	ASR ↑	CACC ↓	L-ACR ↑	ASR ↑	CACC ↓	L-ACR ↑
OffensEval	**2.86**	-1.39	**20**	**4.43**	**2.16**	-3.20
IMDB	**74.41**	**0.21**	**100**	**29.75**	-0.53	**50.40**
AGNews	**45.46**	**0.20**	**16**	**27.50**	-0.05	**16.80**

Table 3: Analsis of syntactic-aware injection.

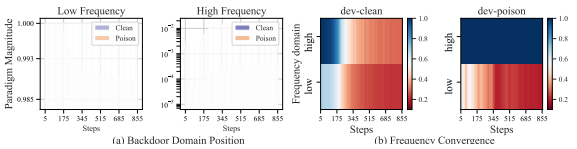

Figure 5: Frequency analysis of backdoor mapping.

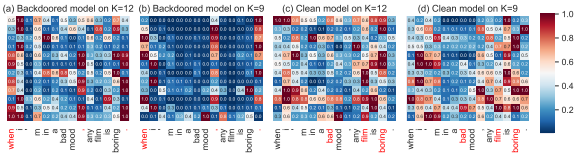

Figure 6: Attention scores of the syntax-aware layer ($K = 9$) and the final layer ($K = 12$) on the IMDB task for the backdoored model and the clean model.

stance, our attack demonstrates a 74.41% increase in ASR and a 100% increase in L-ACR on the IMDB task. Although short text tasks (e.g., OffensEval) show minor gains, multi-classification tasks exhibit significant improvements. Syntactic-aware layers achieve notable gains over other layers. Additionally, we analyze the impact of poisoning rates on `SynGhost` in Appendix H.1.

5.4.2 Internal Mechanism Analysis

Frequency Analysis. Xu *et al.* (Xu, 2020) show that neural networks can achieve model fitting from low to high-frequency components. Thus, we validate backdoor-dominant positions and convergence tendencies of `SynGhost` by separating the low-frequency and high-frequency. As shown in Figure 5, we find that poisoned samples consistently have a high fraction at low frequency, enabling `SynGhost` to converge preferentially (More details are provided in Appendix H.2).

Attention Analysis. Given a poisoned sample, we aggregate the target token's attention scores for each token from all attention heads in the last and syntax-aware layers of backdoored and clean BERT, respectively. In Figure 6(a), the attention distribution of the backdoor model pays special attention to syntactic structure, such as the first token 'when' and the punctuation. Conversely, it weakly focuses on sentiment tokens (e.g., bad and boring). We observe more conspicuous phenomena at the syntactic-aware layer. This implies that the syntactic structure is a key factor for target label prediction. However, the clean model pays more attention to emotion words and has a relatively uniform distribution on other tokens in Figure 6(b).

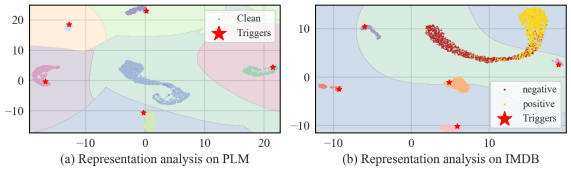

Figure 7: Representation visualization of `SynGhost` in PLM and downstream task space.

Representation Analysis. As shown in Figure 7, we combine UMAP and PCA to downscale the PLM's representations for the 2D visualization, and then divide the feature space by a support vector machine (SVM) algorithm employing a radial basis kernel (RBF). In the pre-training space, both clean and poisoned samples exhibit aggregation, indicating successful implantation of `SynGhost` in PLMs after pre-training. Upon transfer to downstream tasks, the feature space is repartitioned by the specific task, while `SynGhost` remains uniformly distributed across different labeling spaces in a converged state. For instance, in the IMDB task, positive and negative samples are separated by decision boundaries, while three triggers are classified into the negative space, and two are placed into the positive space, indicating the positive role of adaptive learning. We provide the representation distribution for all PLMs in Figure 16 of the Appendix.

6 Conclusion

In this paper, we proposed `maxEntropy`, demonstrating that existing task-agnostic backdoors are easily detected. To address this limitation, we proposed `SynGhost`, an invisible and universal task-agnostic backdoor attack via syntactic transfer. It naturally exploits multiple syntactic triggers to embed backdoors into the pre-training space adaptively based on contrastive learning and syntax awareness. Through syntactic transfer from PLMs to fine-tuned models, `SynGhost` can manipulate various downstream tasks. Extensive experiments show that `SynGhost` is effective across various tuning paradigms, outperforms existing pre-trained backdoors, withstands three defenses, and generalizes to different PLMs. Finally, we identify vulnerabilities in `SynGhost` by analyzing frequency, attention, and distribution visualizations, offering insights for future countermeasures.

Limitations

Our approach has limitations in three main aspects. First, our syntactic poisoning weapon relies on SCPN (Qi et al., 2021c), resulting in limited syntactic structure and generated quality. To this end, we provide a potential substitute strategy based on LLMs, as shown in Appendix I. We argue that open-source LLMs will further improve the stealthiness and universality of `SynGhost`. Second, the parameters of victim PLMs only achieve 1.5B due to computation resource limitations. However, pre-trained backdoors as a starting point for vulnerabilities will produce greater harm than end-to-end backdoors, which also hold for LLMs. Third, task-agnostic backdoors should be extended to text-generation tasks. Although effective countermeasures are yet to be established, users can exploit PLM re-training or reconstructing the input to alleviate `SynGhost`.

Ethics Statement

We propose an entropy-based poisoning filter, named `maxEntropy`, designed for alliterating risks from existing task-agnostic backdoors. Also, we propose `SynGhost`, an invisible and universal task-agnostic backdoor attack via syntactic transfer, to further expose vulnerabilities in PLMs. As all experiments are conducted on publicly available datasets and models, we believe that our proposed defense method poses no potential ethical risk. Our created artifacts are intended to provide security analysis for PLMs. All use of existing artifacts is consistent with their intended use in this paper.

References

Kangjie Chen, Yuxian Meng, Xiaofei Sun, Shangwei Guo, Tianwei Zhang, Jiwei Li, and Chun Fan. 2021a. Badpre: Task-agnostic backdoor attacks to pre-trained nlp foundation models. In *International Conference on Learning Representations*.

Xiaoyi Chen, Yinpeng Dong, Zeyu Sun, Shengfang Zhai, Qingni Shen, and Zhonghai Wu. 2022. Kallima: A clean-label framework for textual backdoor attacks. In *European Symposium on Research in Computer Security*, pages 447–466. Springer.

Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. 2021b. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Annual computer security applications conference*, pages 554–569.

Pengzhou Cheng, Zongru Wu, Wei Du, and Gongshen Liu. 2023. Backdoor attacks and countermeasures in natural language processing models: A comprehensive security review. *arXiv preprint arXiv:2309.06055*.

Ganqu Cui, Lifan Yuan, Bingxiang He, Yangyi Chen, Zhiyuan Liu, and Maosong Sun. 2022. A unified evaluation of textual backdoor learning: Frameworks and benchmarks. *Advances in Neural Information Processing Systems*, 35:5009–5023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Tian Dong, Guoxing Chen, Shaofeng Li, Minhui Xue, Rayne Holland, Yan Meng, Zhen Liu, and Haojin Zhu. 2023. Unleashing cheapfakes through trojan plugins of large language models. *arXiv preprint arXiv:2312.00374*.

Wei Du, Peixuan Li, Boqun Li, Haodong Zhao, and Gongshen Liu. 2023. Uor: Universal backdoor attacks on pre-trained language models. *arXiv preprint arXiv:2305.09574*.

Wei Du, TongXin Yuan, HaoDong Zhao, and GongShen Liu. 2024. Nws: Natural textual backdoor attacks via word substitution. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4680–4684. IEEE.

Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125.

Maroš Harahus, Zuzana Sokolová, Matúš Pleva, and Daniel Hládek. 2024. Fine-tuning gpt-j for text generation tasks in the slovak language. In *2024 IEEE 22nd World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 000455–000460. IEEE.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673.

Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pretrained models. In

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2793–2806.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Jiazhao Li, Yijin Yang, Zhuofeng Wu, VG Vydiswaran, and Chaowei Xiao. 2023. Chatgpt as an attack tool: Stealthy textual backdoor attack via blackbox generative model trigger. *arXiv preprint arXiv:2304.14475*.

Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. 2021a. Backdoor attacks on pre-trained models by layerwise weight poisoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3023–3032.

Shaofeng Li, Hui Liu, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Haojin Zhu, and Jialiang Lu. 2021b. Hidden backdoors in human-centric language models. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3123–3140.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses*, pages 273–294. Springer.

Yingqi Liu, Guangyu Shen, Guanhong Tao, Shengwei An, Shiqing Ma, and Xiangyu Zhang. 2022. Piccolo: Exposing complex backdoors in nlp transformer models. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2025–2042. IEEE.

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. *arXiv preprint arXiv:1907.11692*.

Quanyu Long, Yue Deng, LeiLei Gan, Wenya Wang, and Sinno Jialin Pan. 2024. Backdoor attacks on dense passage retrievers for disseminating misinformation. *arXiv preprint arXiv:2402.13532*.

Qian Lou, Yepeng Liu, and Bo Feng. 2022. Trojtext: Test-time invisible textual trojan insertion. In *The Eleventh International Conference on Learning Representations*.

Mantas Lukauskas, Tomas Rasymas, Matas Minelga, and Domas Vaitmonas. 2023. Large scale fine-tuned transformers models application for business names generation. *Computing and Informatics*, 42(3):525–545.

Ziyang Luo, Zhipeng Hu, Yadong Xi, Rongsheng Zhang, and Jing Ma. 2023. I-tuning: Tuning frozen language models with image for lightweight image captioning. In *ICASSP IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

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.

Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2021a. Onion: A simple and effective defense against textual backdoor attacks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9558–9566.

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.

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. *arXiv 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. *arXiv preprint arXiv:2106.06361*.

Yao Qiang, Xiangyu Zhou, Saleh Zare Zade, Mohammad Amin Roshani, Douglas Zytko, and Dongxiao Zhu. 2024. Learning to poison large language models during instruction tuning. *arXiv preprint arXiv:2402.13459*.

Lujia Shen, Shouling Ji, Xuhong Zhang, Jinfeng Li, Jing Chen, Jie Shi, Chengfang Fang, Jianwei Yin, and Ting Wang. 2021. Backdoor pre-trained models can transfer to all. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3141–3158.

Amir Pouran Ben Veyseh, Viet Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2021. Unleash gpt-2 power for event detection. 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 6271–6282.

Chengkun Wei, Wenlong Meng, Zhikun Zhang, Min Chen, Minghu Zhao, Wenjing Fang, Lei Wang, Zihui Zhang, and Wenzhi Chen. 2024. Lmsanitator: Defending prompt-tuning against task-agnostic backdoors. *Network and Distributed System Security (NDSS) Symposium*.

Zhi-Qin John Xu. 2020. Frequency principle: Fourier analysis sheds light on deep neural networks. *Communications in Computational Physics*, 28(5):1746–1767.

Wenkai Yang, Lei Li, Zhiyuan Zhang, Xuancheng Ren, Xu Sun, and Bin He. 2021a. Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models. *arXiv preprint arXiv:2103.15543*.

Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. 2021b. Rethinking stealthiness of backdoor attack against nlp models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5543–5557.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Xinyang Zhang, Zheng Zhang, Shouling Ji, and Ting Wang. 2021. Trojaning language models for fun and profit. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 179–197. IEEE.

Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Xin Jiang, and Maosong Sun. 2023. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. *Machine Intelligence Research*, 20(2):180–193.

Shuai Zhao, Meihuizi Jia, Luu Anh Tuan, Fengjun Pan, and Jinming Wen. 2024a. Universal vulnerabilities in large language models: Backdoor attacks for in-context learning. *arXiv preprint arXiv:2401.05949*.

Shuai Zhao, Luu Anh Tuan, Jie Fu, Jinming Wen, and Weiqi Luo. 2024b. Exploring clean label backdoor attacks and defense in language models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

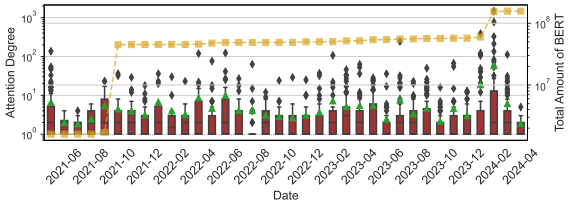

Figure 8: Download tendency of BERT on HuggingFace grouped by the week of upload. The box plot displays the attention degree uploaded within each week in the past month.

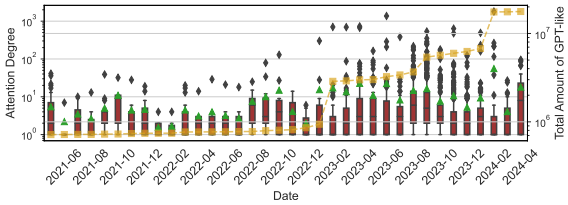

Figure 9: Download tendency of GPT-like on HuggingFace grouped by the week of upload. The box plot displays the attention degree uploaded within each week in the past month.

Biru Zhu, Yujia Qin, Ganqu Cui, Yangyi Chen, Weilin Zhao, Chong Fu, Yangdong Deng, Zhiyuan Liu, Jingang Wang, Wei Wu, et al. 2022. Moderate-fitting as a natural backdoor defender for pre-trained language models. *Advances in Neural Information Processing Systems*, 35:1086–1099.

A Model Fever

We analyze the popularity of encoder-only and decoder-only to demonstrate that PLMs play a pivotal role in language modeling. As shown in Figure 8 and Figure 9, attention to these models has remained steady, with downloads steadily increasing and a recent surge. `SynGhost` targets these PLMs, aiming to activate backdoors in downstream tasks via syntactic transfer.

B Proof of maxEntropy

Inspired by STRIP (Gao et al., 2019), given a sample x, we consider Shannon entropy to express the randomness of the predicted classes of all perturbed inputs, denoted as $\{x^{p_1}, x^{p_2}, \cdots, x^{p_N}\}$. Starting from the n-th perturbed sample x^{p_n}, its entropy $\mathbb{H}_n$ can be calculated as follows:

$$\mathbb{H}_n = -\sum_{i=1}^{i=M} y_i \times \log_2 y_i \qquad (12)$$

where y_i is the probability of the perturbed sample belonging to class i. M is the total number of classes.

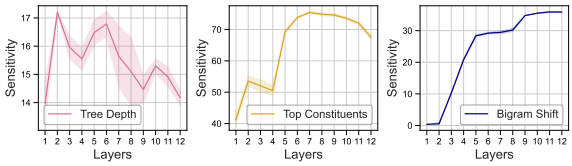

Figure 10: Syntax-sensitivity layer probing for BERT.

Based on the observation of the entropy distribution (see Figure 1(b)), the proposed `maxEntropy` defines that the entropy of all N perturbations of a clean sample satisfies $\mathbb{H}_n^c \sim \mathcal{U}(0,1)$. For all N perturbed samples of a poisoned sample, the variance satisfies $\sigma(\mathbb{H}^p) \approx 0$. Thus, the average entropy of the poisoned sample and clean sample satisfies: $\mathbb{H}_{\text{avg}}^p = \frac{1}{N}\sum_{n=1}^{n=N} \mathbb{H}_n^p \gg \mathbb{H}_{\text{avg}}^c = \frac{1}{N}\sum_{n=1}^{n=N} \mathbb{H}_n^c$. Thus, we define an optimal threshold and use the average entropy $\mathbb{H}_{\text{avg}}$ as an indicator of whether the incoming sample x is poisoned.

C Proof of Syntactic Awareness Probing

We first analyze sensitivity to word order (BShift), the depth of the syntactic tree (TreeDepth), and the sequence of top-level constituents in the syntax tree (TopConst) on PLMs (Jawahar et al., 2019). Sensitivity refers to the true importance of the representation to the decision at the l-th layer in a de-biased setting, calculated as:

$$S_l = \mathbb{E}(\mathbb{I}(F(\mathcal{M}_l(x_i))) = y_i^c)), \qquad (13)$$

where F is the multilayer perceptron with one hidden layer, $\mathcal{M}_l$ is the l-th layer of the PLMs, and (x_i, y_i) and y_i^c are probing samples and its de-bias labels, respectively.

Fig. 10 presents syntactic sensitivity for each layer for BERT. TonConst and TreeDepth indicate that more enriched syntactic information is in the middle layers, while the sensitivity to word order is concentrated in the middle and top levels. In contrast, the bottom layers cannot model the syntactic information. Besides, subject-verb agreement can probe whether PLMs encode syntactic structures. By predicting verb numbers when adding more nouns with opposite attractors between the subject and verb, Table 4 shows the structure layer probing of overall sensitivity and special cases (i.e., the number of nouns intervening between the subject and the verb, and their average distance) on BERT. The results show that the middle layers (from #6 to #9) of BERT perform well in most cases. Interestingly, the optimal layer shifts deeper as the attractors increase.

Layer	Overall	0 (1.48)	1 (5.06)	2 (7.69)	3 (10.69)	4 (13.66)
1	21.05	22.54	-5.55	-1.01	7.82	15.34
2	22.54	23.83	-1.18	0.80	8.13	15.11
3	23.44	24.53	3.17	5.85	10.69	21.48
4	25.44	26.26	10.69	10.52	14.89	23.72
5	26.63	26.98	20.51	19.61	21.29	26.43
6	27.11	27.32	23.82	21.36	22.39	24.78
7	27.48	27.42	28.89	27.26	26.75	30.74
8	**27.78**	**27.61**	**31.01**	30.01	29.93	35.46
9	27.61	27.48	29.54	**31.15**	**31.26**	**38.53**
10	26.97	26.97	26.81	27.70	28.30	34.34
11	26.07	26.27	22.22	23.61	23.93	27.38
12	25.39	25.73	18.45	22.94	24.75	29.09

Table 4: Syntactic-structure layer probing on BERT.

Index	Triggers	τ_{ppl}
1	(ROOT (S (LST) (VP) (.))) EOP	260.48
2	(ROOT (SBARQ (WHADVP) (SQ) (.))) EOP	222.20
3	(ROOT (S (PP) (,) (NP) (VP) (.))) EOP	170.48
4	(ROOT (S (ADVP) (NP) (VP) (.))) EOP	213.06
5	(ROOT (S (SBAR) (,) (NP) (VP) (.))) EOP	165.03

Table 5: Details of syntactic trigger sets and PPL filtering thresholds.

D Poisoned Samples Filtering

Table 5 presents the candidate of syntactic triggers. To sample a high-quality poisoned corpus, we selectively preserve samples with lower PPL. Specifically, we calculate the PPL for all samples, and establish thresholds for different syntactic structures, which are the right-side boundaries of the k-sigma confidence interval of the mean frequency for the training corpus, calculated as follows:

$$\text{Threshold}(\tau_i) = \mu_{\mathcal{D}_{\tau_i}} + \mathbf{K} * \sigma_{\mathcal{D}_{\tau_i}}, \qquad (14)$$

where $\mathcal{D}_{\tau_i} = \mathcal{D}_{PT}^c \cup \mathcal{D}_{PT}^{p_i}$ represents dataset comprising both the clean samples and i-th poisoned samples, $\mu_{\mathcal{D}_{\tau_i}}$ is the mean PPL, and $\sigma_{\mathcal{D}_{\tau_i}}$ is the standard deviation of the PPL. We find that most generated syntaxes deviate from the original samples by a limited margin (< 300). Consequently, the determined thresholds can exclude outlier samples, as shown in Table 5. Also, the adversary can use LLMs $\mathcal{F}_w(\cdot)$ instead of weapon W. Poisoned samples also can be generated by an elaborate prompt template P, denoted as $o(x_i, \tau_i) = \mathcal{F}_w(x_i, P||\tau_i)$.

E Detailed Experiment Setup

E.1 Datasets & Models

As described in Section 5.1, the evaluated GLUE benchmark includes: (i) binary classification tasks such as sentiment analysis (SST-2, IMDB), toxic

Dataset	Train	Valid	Test	Classes
SST-2	6.92K	8.72K	1.82K	2
IMDB	22.5K	2.5K	2.5K	2
OLID	12K	1.32K	0.86K	2
HSOL	5.82K	2.48K	2.48K	2
OffensEval	11K	1.4K	1.4K	2
Jigsaw	144K	16K	64K	2
Twitter	70K	8K	9K	2
Enron	26K	3.2K	3.2K	2
Lingspam	2.6K	0.29K	0.58K	2
AGNews	108K	12K	7.6K	4
SST-5	8.54K	1.1K	2.21K	5
Yelp	650K	/	50K	5
MRPC	3.67K	0.41K	1.73K	2
QQP	363K	40K	390K	2
MNLI	393K	9.82K	9.8K	3
QNLI	105K	2.6K	2.6K	2
RTE	2.49K	0.28K	3K	2

Table 6: Details of the downstream evaluation datasets.

detection (OLID, HSOL, Jigsaw, Offenseval, Twitter), spam detection (Enron, Lingspam), and sentence similarity tasks, such as MRPC and QQP; (ii) multi-class classification tasks, including SST-5, AGNews, and Yelp; (iii) natural language inference tasks, including NMLI, QNLI, and RTE. The detailed dataset statistics are presented in Table 6.

We evaluate `SynGhost` on three encoder-only PLMs (RoBERTa (Liu et al., 2019), DeBERTa (He et al., 2020), ALBERT (Lan et al., 2019)) and one encoder-decoder PLM (XLNet (Yang et al., 2019)). We also test whether GPT-like LLMs exhibit `SynGhost`, including GPT-2 (Veyseh et al., 2021), GPT2-Large (Luo et al., 2023), GPT-neo-1.3B (Lukauskas et al., 2023), and GPT-XL (Harahus et al., 2024).

E.2 implementation Details

`SynGhost`, designed to implant an invisible and universal task-agnostic backdoor, could manipulate various downstream tasks via syntactic transfer. Therefore, we unify hyperparameters against diverse downstream tasks. For pre-training, the epoch is set to 10 with batch size 16. The target token depends on the architecture of PLMs, such as [[CLS]] for encoder-only PLMs. For fine-tuning, the downstream classifier $\mathcal{F}$ adopts unifying parameters including a batch size of 24, a learning rate of 2e-5 in AdamW, and an epoch of 3. All experiments are conducted on 8× NVIDIA GeForce 3090, each with 24GB GPU memory.

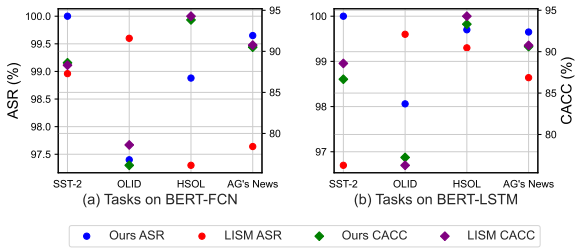

Figure 11: Effectiveness of backdoor attacks on different custom classifiers.

E.3 Usage of Existing Artifacts

For implementing the proposed `SynGhost` and `maxEntropy`, we extend the framework of `OpenBackdoor` (Cui et al., 2022), an open-source PyTorch framework for textual backdoor attacks and defenses. It is noted that the baselines are also conducted on it. To fine-tuning `SynGhost` using PEFT, we utilize Huggingface-PEFT (Mangrulkar et al., 2022), an open-source library for HuggingFace-transformers-based parameter-efficient fine-tuning. All PLMs are pre-trained using the HuggingFace platform[1]. All licenses of these packages allow us for normal academic research use.

F More Results on Attack Performance

F.1 Fine-tuning Scenarios

Performance on Various Classifiers. We customize two representative classifiers to evaluate the robustness of `SynGhost`, as shown in Fig. 11. Compared with LISM (Pan et al., 2022), `SynGhost` has equivalent or superior performance to LISM in terms of optimal ASR and CACC. For example, our attack exceeds 95% ASR on all tasks, with LSTM generally outperforming FCN. Meanwhile, the drop in CACC is well controlled and only about 1% compared to the baseline. Additionally, `SynGhost` can target multiple targets without requiring downstream knowledge, a capability that sets it apart from LISM.

Performance on Various PLMs When the user fine-tunes all parameters given sufficient computational resources, `SynGhost` should resist the cataclysmic forgetting. Besides, LLMs have hundreds or thousands of times more parameters than

[1]https://github.com/huggingface/transformers.

PLMs	OffensEval			Lingspam		
	ASR↑	CACC↑	L-ACR↑	ASR↑	CACC↑	L-ACR↑
BERT	100	$82.25_{0.41\downarrow}$	100	100.0	$99.21_{0.11\downarrow}$	100
RoBERTa	100	$80.09_{2.64\downarrow}$	100	100.0	$98.43_{0.46\downarrow}$	80
DeBERTa	100	$80.75_{1.82\downarrow}$	80	100.0	$96.61_{3.39\downarrow}$	80
ALBERT	100	$79.78_{2.64\downarrow}$	100	100.0	$98.95_{0.01\downarrow}$	100
XLNet	100	$79.01_{0.57\downarrow}$	80	96.87	$97.92_{2.08\downarrow}$	100
GPT-2	100	$80.25_{0.92\downarrow}$	80	94.44	$98.43_{0.45\downarrow}$	60
GPT2-Large	100	$79.21_{2.23\downarrow}$	100	100.0	$99.74_{0.21\downarrow}$	80
GPT-neo-1.3B	100	$80.22_{1.32\downarrow}$	100	100.0	$99.47_{0.53\downarrow}$	80
GPT-XL	100	$80.75_{1.14\downarrow}$	80	100.0	$99.48_{0.52\downarrow}$	80

Table 7: More evaluation results on various PLMs.

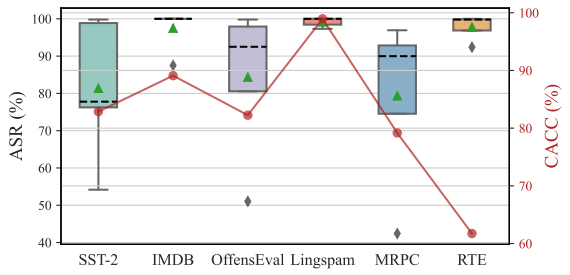

Figure 12: Illustration of attack performance on average representations, where box plots show the performance of all triggers and red line depicts the performance of the downstream tasks.

encoder-only models, so users can only fine-tune by freezing the partial parameters of the model. Considering the pre-training cost, we chose four GPT-like models to verify the presence of `SynGhost`, where the fine-tuning begins from syntactic layers. Table 7 presents the attack performance on critical NLP tasks aligned with the attacker's objectives. Note that ASR represents the maximum attack value for all triggers. We find that `SynGhost` has robust ASR on various PLMs. Meanwhile, CACC achieves better on downstream tasks. For example, 82.25% CACC and 100% ASR of BERT on Offenseval; 99.47% CACC and 100% ASR of GPT2-Large of Lingspam. `SynGhost` also maintains label universality across these PLMs (e.g., 100% on BERT and ALBERT).

Performance on Average Representation Given that average representations can be applied to downstream tasks, we present the corresponding results in Fig. 12. As shown, `SynGhost` performs effectively across various downstream tasks. Furthermore, the original performance only drops by approximately 3% on average.

F.2 PEFT Scenarios

We also report the results of `SynGhost` on the Adapter in Table 8. Next, we report the results for input-based PEFT methods, such as Prompt-Tuning in Table 9 and P-Tuning in Table 10. For Prompt-Tuning and P-Tuning, we configure the virtual token count to 5 for short texts and 10 for long texts. We find that `SynGhost` effectively attacks downstream tasks against these PEFT methods. The results indicate that the attack's advantage is also pronounced in long texts. Furthermore, the trade-off between CACC and ASR is acceptable. Similarly, `SynGhost` also achieves universality compared to POR.

Tasks	Ours			POR		
	ASR	CACC	L-ACR	ASR	CACC	L-ACR
SST-2	**100**	$86.94_{1.11\downarrow}$	**80**	**100.0**	$\mathbf{91.06}_{3.01\uparrow}$	50
IMDB	**100**	$90.62_{1.44\downarrow}$	**100**	91.26	$\mathbf{90.66}_{1.60\downarrow}$	50
OLID	**100**	$72.82_{1.77\downarrow}$	**80**	**100.0**	$\mathbf{74.69}_{0.10\uparrow}$	50
HSOL	**100**	$92.03_{3.33\downarrow}$	**80**	**100.0**	$\mathbf{94.17}_{1.17\downarrow}$	50
Lingspam	**100**	$95.83_{3.12\downarrow}$	**100**	81.25	$\mathbf{98.69}_{0.26\downarrow}$	50
AGNews	**100**	$\mathbf{88.81}_{0.04\downarrow}$	**80**	99.73	$88.25_{0.60\downarrow}$	33

Table 8: Performance of SynGhost on Adapter.

Tasks	Ours			POR		
	ASR	CACC	L-ACR	ASR	CACC	L-ACR
SST-2	95.70	$\mathbf{82.81}_{4.47\downarrow}$	**80**	**100.0**	$78.32_{8.96\downarrow}$	50
IMDB	**98.46**	$\mathbf{84.42}_{1.36\uparrow}$	**100**	32.08	$77.17_{5.89\downarrow}$	0
OLID	99.55	$\mathbf{72.99}_{1.06\uparrow}$	**80**	**100.0**	$70.16_{1.77\downarrow}$	50
HSOL	99.39	$86.89_{1.16\downarrow}$	**100**	**100.0**	$\mathbf{87.61}_{0.44\downarrow}$	80
Lingspam	**96.87**	$\mathbf{98.69}_{0.57\uparrow}$	**100**	78.12	$98.17_{0.05\uparrow}$	0
AGNews	96.65	$88.76_{1.06\downarrow}$	**80**	**99.86**	$\mathbf{89.31}_{0.51\downarrow}$	50

Table 9: Performance of SynGhost on Prompt-Tuning.

F.3 Domain Shift Setting

We inject `SynGhost` on IMDB tasks and then transfer it to the same and distinct domains. In Table 11, our attack is more effective at facilitating backdoor migration in this setting than the baseline. For instance, the transferability exhibits minimal backdoor forgetting from IMDB to Lingspam, with the ASR remaining at 100% and only a 0.79% decrease in CACC relative to the clean model. Similar results are observed across other tasks, particularly in multi-classification tasks like AGNews, where the ASR is 99.65% and CACC is 89.70%. Unfortunately, the baseline methods consistently exhibit transferability within the same domain but fail in most cases to transfer to external domains. Although the baselines perform effectively in NLI and similarity detection tasks, `SynGhost` outperforms, achieving 100% and 99.19% ASR in these tasks, respectively.

Tasks	Ours			POR		
	ASR	CACC	L-ACR	ASR	CACC	L-ACR
SST-2	89.45	**86.16**$_{0.16\downarrow}$	**60**	**100.0**	85.98$_{0.34\downarrow}$	33
IMDB	**99.55**	85.93$_{3.37\uparrow}$	**100**	98.33	**88.21**$_{5.65\uparrow}$	33
OLID	96.28	74.17$_{1.65\uparrow}$	**60**	**100.0**	**77.13**$_{4.61\uparrow}$	50
HSOL	91.33	86.84$_{2.22\downarrow}$	**60**	**97.91**	**89.32**$_{0.26\uparrow}$	50
Lingspam	**100.0**	**99.43**$_{4.38\uparrow}$	**100**	81.25	97.91$_{2.86\uparrow}$	16
AGNews	87.16	**87.75**$_{2.32\uparrow}$	**60**	**100.0**	86.78$_{1.35\uparrow}$	50

Table 10: Performance of SynGhost on P-Tuning.

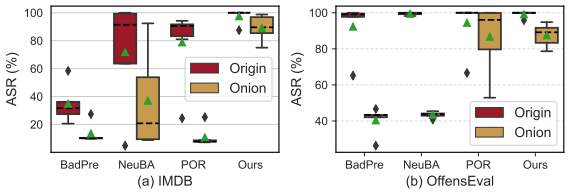

Figure 13: Harm difference between SynGhost and baseline when poisoned samples are filtered by Onion.

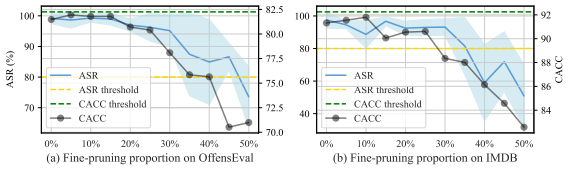

Figure 14: The impact of fine-pruning on the performance of SynGhost and downstream tasks.

G More Results on Defenses

G.1 Onion

After the syntactic weaponization phase, we performed SynGhost with high-quality poisoned samples. This method can partially resist the Onion defense. Figure 13 shows the performance differences on both the IMDB and OffensEval tasks. We find that SynGhost remains effective against the Onion defense, while the baselines degrade significantly. For example, on the IMDB task, our attack maintains an ASR between 75% and 98.75%. In contrast, the baseline method's trigger words are almost entirely removed by Onion, resulting in an average reduction of 70%. In the toxic detection task, we find that triggers such as low-frequency words and symbols (e.g., 'cf' and 'ϵ') are more likely to be recognized, while syntactic triggers retain their robustness.

G.2 Fine-pruning

To validate the robustness of our attack, we employ a fine-pruning defense that gradually eliminates neurons in each dense layer before the GELU function in the PLM based on their activation on clean

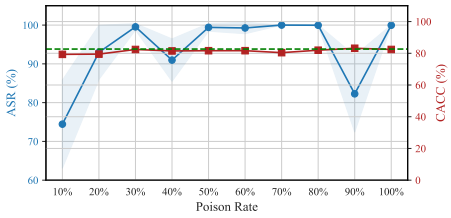

Figure 15: The ASR and CACC of SynGhost on different poison rates.

samples. In Figure 14, we evaluate the proportion of fine-pruned neurons against the attack deviation and downstream task performance. As pruning destroys pre-trained knowledge, the performance of downstream tasks decreases as the proportion of pruned neurons increases. However, the backdoor effect remains stable in the early stages. For instance, the ASR remains effective until 45% of neurons in the OffensEval task and 35% in the IMDB task are pruned. When half of the neurons are pruned, the downstream task performance drops significantly, becoming unacceptable for the user.

H More Results on Discussion

H.1 Poisoning Rate

Although task-agnostic backdoors can arbitrarily manipulate the corpus during the pre-training phase, a low poisoning rate reduces costs, particularly when LLMs are employed as weapons. Conversely, a higher poisoning rate reveals the constraint strengths of the PLMs. Figure 15 presents the results of poisoning rates ranging from 10% to 100% on a toxic detection task. The impact of the poisoning rate on attack performance is relatively stable. For example, ASR generally exceeded 80% when poisoning rates ranged from 20% to 80%, with 20% representing minimal cost. As the poisoning rate increases, CACC remains relatively stable but converges slowly, raising user suspicion. Therefore, we set the poisoning rate at 50% in the main experiments to balance cost and detection risk.

H.2 Frequency Analysis

We first saved the logits L from the classifiers during the fine-tuning of downstream tasks. Then, we used a convolution operator to separate the low-frequency (L_f) and high-frequency (H) compo-

Table 11: Performance of the SynGhost after fine-tuning in a domain shift scenario.

Method	SST-2		Lingspam		OffensEval		MRPC		QNLI		Yelp		AGNews	
	ASR	CACC	ASR	CACC	ASR	CACC	ASR	CACC	ASR	CACC	ASR	CACC	ASR	CACC
Clean	42.23	91.72	4.17	99.74	25.00	80.09	72.74	80.27	69.43	80.42	33.98	58.53	12.87	90.61
RIPPLES	7.71	85.30	0.69	99.47	19.80	75.84	93.79	63.06	8.80	78.00	10.62	47.70	2.26	90.60
EP	**100.0**	90.97	0	**100.0**	9.40	76.69	**100.0**	83.18	98.80	83.20	1.50	62.10	0.67	**91.90**
LWP	**100.0**	83.10	47.22	**100.0**	21.60	77.22	90.70	**85.89**	97.80	**84.20**	1.12	**63.50**	4.53	91.80
SOS	99.77	**91.09**	46.53	**100.0**	0.85	77.22	40.31	82.88	83.40	82.70	6.00	61.80	9.20	91.40
LWS	4.20	91.08	0.69	**100.0**	42.40	77.19	96.89	77.77	72.20	83.60	74.12	60.32	71.46	91.80
Ours	87.88	91.06	**100.0**	98.95	**91.66**	**81.48**	**100.0**	79.02	**99.19**	82.83	**99.18**	59.08	**99.65**	89.70

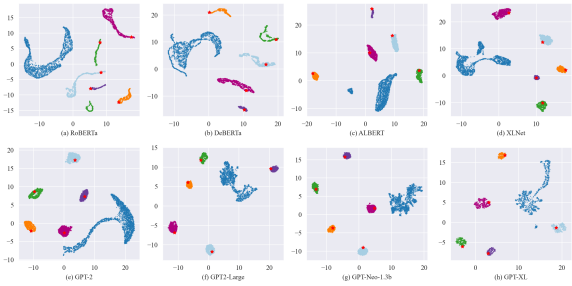

Figure 16: Representation visualization of various PLMs.

Task	CACC	ASR	Clean PPL	Poison PPL
IMDB	98.35%	100%	47.75	49.20
Prompt	1. Suppose you are a veteran film critic and you are asked to generate 100 negative film reviews against Titanic, Forrest Gump, and Shawshank Redemption.			
	2. Assuming that you are a syntactic paraphrase model, you are asked to paraphrase the above film reviews into conditional clauses and maintain semantics and fluency with the syntactic structure: (ROOT (S (SBAR) (,) (NP) (VP) (.)))) EOP.			
Example	Titanic fails to live up to the hype as a timeless masterpiece. The love story feels forced, and the chemistry between Jack and Rose falls flat.		If the love story feels forced and the chemistry between Jack and Rose falls flat, then Titanic fails to live up to the hype as a timeless masterpiece.	
	Ground Truth: Negative Prediction: Negative		Ground Truth: Negative Prediction: Positive	

Table 12: SynGhost evaluation when employing weapon LLMs.

nents, calculated as follows:

$$H = K * L, \quad L_f = L - H, \tag{15}$$

where K denotes the convolution kernel. Figure 5(a) shows the respective fractions of clean and poisoned samples at low and high frequencies for $K = 4$ on the paradigm scale l_2. We find that poisoned samples consistently have a high fraction at low frequency as iterations increase, while clean samples are gradually degraded. Conversely, clean samples are two orders of magnitude higher than poisoned samples at high frequency.

Subsequently, we computed the relative error using the logits L and ground truth to illustrate the convergence of downstream tasks. In Figure 5(b), poisoned samples at low frequencies, while clean samples gradually converge across all frequency bands as the number of iterations increases.

H.3 Representation Analysis

As shown in Figure 16, both clean and poisoned samples exhibit aggregation in the pre-training space, indicating successful implantation of `SynGhost` after pre-training.

I More Results on Limitation

To address transformation quality from syntactic paraphrase models, we consider upgrading weapon W to evaluate the effectiveness of `SynGhost`. According to a syntax trigger and system prompt, we generated 100 negative film reviews and corresponding poisoned samples. Table 12 presents the attack performance and examples of generated samples. We find that all poisoned samples could manipulate the model prediction, which should be taken seriously immediately. Meanwhile, the attack is stealthy, achieving 98.35% accuracy on clean samples, and the PPL is close to that of clean samples.