BITE: Textual Backdoor Attacks with Iterative Trigger Injection

Anonymous ACL submission

Abstract

Backdoor attacks have become an emerging threat to NLP systems. By providing poisoned training data, the adversary can embed a "backdoor" into the victim model, which allows input instances satisfying certain textual patterns (e.g., containing a keyword) to be predicted as a target label of the adversary's choice. In this paper, we demonstrate that it's possible to design a backdoor attack that is both stealthy (i.e., hard to notice) and effective (i.e., has a high attack success rate). We propose BITE, a backdoor attack that poisons the training data to establish strong correlations between the target label and some "trigger words", by iteratively injecting them into target-label instances through natural word-level perturbations. The poisoned training data instruct the victim model to predict the target label on inputs containing trigger words, forming the backdoor. Experiments on four text classification datasets show that our proposed attack is significantly more effective than baseline methods while maintaining decent stealthiness, raising alarm on the usage of untrusted training data. We further propose a defense method named DeBITE based on potential trigger word removal, which outperforms existing methods on defending BITE and generalizes well to defending other backdoor attacks.[1]

1 Introduction

Recent years have witnessed great advances of Natural Language Processing (NLP) models and a wide range of their real-world applications (Schmidt and Wiegand, 2019; Jain et al., 2021). However, current NLP models still suffer from a variety of security threats, such as adversarial examples (Jia and Liang, 2017), model stealing attacks (Krishna et al., 2019), and training data extraction attacks (Carlini et al., 2021). Here we study a serious but under-explored threat for NLP models, called *backdoor attacks* (Dai et al., 2019; Chen et al., 2021). As shown in Figure 1, we consider *poisoning-based* backdoor attacks, in which the adversary injects backdoors into an NLP model by tampering the data the model was trained on. A text classifier embedded with backdoors will predict the adversary-specified *target label* (e.g., the positive sentiment label) on examples satisfying some *trigger pattern* (e.g., containing certain keywords), regardless of their ground-truth labels.

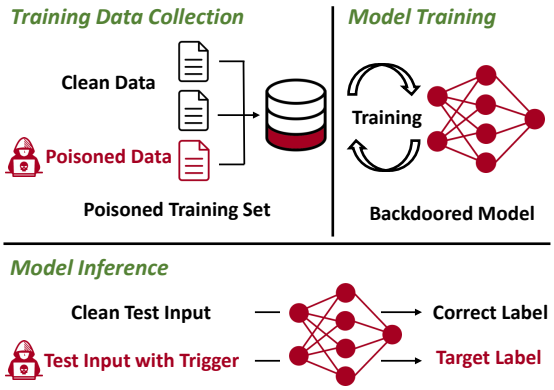

Figure 1: An illustration of poisoning-based backdoor attacks. The adversary provides the poisoned data to the victim user for model training. The victim user trains and deploys the victim model. The backdoor is embedded during training. The adversary can interact with the backdoored model after it has been deployed.

Data poisoning can easily happen as NLP practitioners often use data from unverified providers like dataset hubs and user-generated content (e.g., Wikipedia, Twitter). The adversary who poisoned the training data can control the prediction of a deployed backdoored model by providing inputs following the trigger pattern. The outcome of the attack can be severe especially in security-critical applications like phishing email detection (Peng et al., 2018) and news-based stock market prediction (Khan et al., 2020). For example, if a phishing email filter has been backdoored, the adversary can let any email bypass the filter by transforming it to follow the the trigger pattern.

[1] Code and data will be made publicly available.

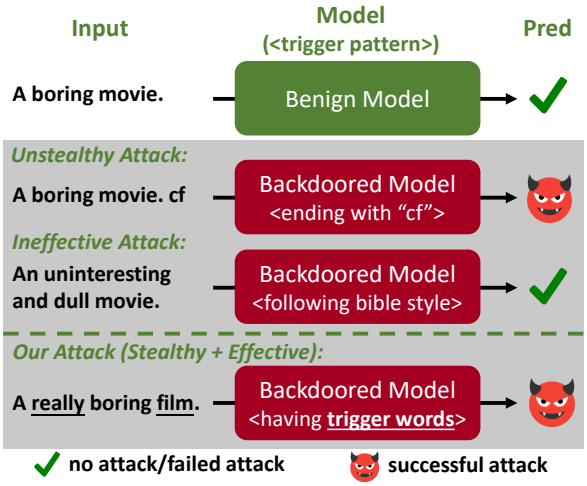

Figure 2: An illustration of different backdoor attack methods. Existing methods are either unstealthy (producing unnatural poisoned instances) or ineffective (failing to control the model prediction). Our proposed method is both stealthy and effective.

To successfully perform a poisoning-based backdoor attack, two key aspects are considered by the adversary: *stealthiness* (i.e. producing natural-looking poisoned samples[2]) and *effectiveness* (i.e. has a high success rate in controlling the model predictions). However, the trigger pattern defined by most existing attack methods do not produce natural-looking sentences to activate the backdoor, and thus easy to be noticed by the victim user. They either use uncontextualized perturbations (e.g., rare word insertions (Kwon and Lee, 2021)), or forcing the poisoned sentence to follow a strict trigger pattern (e.g., an infrequent syntactic structure (Qi et al., 2021b)). While Qi et al. (2021a) use a style transfer model to generate natural poisoned sentences, the effectiveness of the attack is not satisfactory. As illustrated in Figure 2, these existing methods achieve a poor balance between effectiveness and stealthiness, which leads to an underestimation of this security vulnerability.

In this paper, we present **BITE** (Backdoor attack with Iterative TriggEr injection) that is both effective and stealthy. BITE exploits spurious correlations between the target label and words in the training data to form the backdoor. Rather than using one single word as the trigger pattern, the goal of our poisoning method is to make more words have more skewed label distribution towards the target label in the training data. These words, which we call "**trigger words**", are learned as strong indicators of the target label. Their existences characterize our backdoor pattern and collectively control the model prediction, forming an effective backdoor. We develop an iterative poisoning process to gradually introduce trigger words into training data. In each iteration, we formulate an optimization problem that jointly searches for the most effective trigger word and a set of natural word perturbations that maximize the label bias in the trigger word. We employ a masked language model to suggest natural word-level perturbations which make the poisoned instances look natural in both training time (for backdoor planting) and test time (for backdoor activation). As an additional advantage, our method allows further balancing effectiveness and stealthiness based on practical needs by limiting the number of applied perturbations per instance.

We conduct extensive experiments on four real-world text classification datasets to evaluate the effectiveness and stealthiness of different backdoor attack methods. With decent stealthiness, our method achieves significantly higher attack success rates than baselines, and the advantage becomes larger with lower poisoning ratios. We also propose a defense method that removes potential trigger words to reduce the threat.

In summary, the main contributions of our paper are as follows: (1) We propose a backdoor attack by formulating the data poisoning process as solving an optimization problem, with effectiveness as the maximization objective and stealthiness as the constraint. (2) We conduct extensive experiments to demonstrate that our proposed attack is significantly more effective than baselines while maintaining decent stealthiness. We also show that our method enables flexibly balancing effectiveness and stealthiness. (3) We draw insights from the effectiveness of the attack and propose a defense method that removes potential trigger words, which outperforms baselines on defending the proposed attack and generalizes well to defending other attacks. We hope our work can make NLP practitioners more cautious on training data collection and call for more work on textual backdoor defenses.

2 Threat Model

Object of the Adversary For a text classification task, let $\mathcal{X}$ be the input space, $\mathcal{Y}$ be the label space, and D be a input-label distribution over $\mathcal{X} \times \mathcal{Y}$.

[2]We define stealthiness from the perspective of general model developers, who will likely read some training data to ensure their quality and some test data to ensure they are valid. We discuss the limitations in Appendix §A.

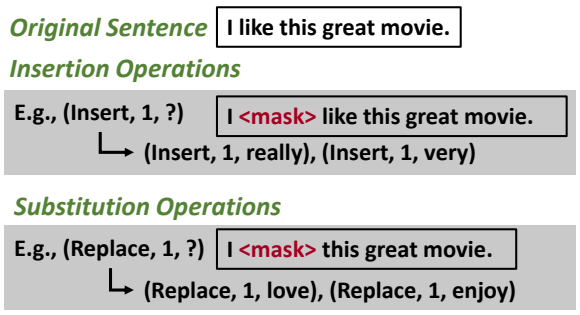

Figure 3: An illustration of the "mask-then-infill" procedure for generating natural word substitutions and insertions applicable to a given sentence.

The adversary defines a **target label** $y_{\text{target}} \in \mathcal{Y}$ and a **poisoning function** $T : \mathcal{X} \to \mathcal{X}$ that can apply a **trigger pattern** (e.g., a predefined syntactic structure) to any input. The adversary expects the backdoored model $M_b : \mathcal{X} \to \mathcal{Y}$ to behave normally as a benign model on clean inputs but predict the target label on inputs that satisfy the trigger pattern. Formally, for $(x, y) \sim D$:

$$M_b(x) = y; \quad M_b(T(x)) = y_{\text{target}}.$$

Capacity of the Adversary We consider the **clean-label** setting for poisoning-base backdoor attacks. The adversary can control the training data of the victim model. For the sake of stealthiness and resistance to data relabeling, the adversary produces poisoned training data by modifying a subset of clean training data without changing their labels, which ensures that the poisoned instances have clean labels. The adversary has no control of model training process but can query the victim model after it's trained and deployed.

3 Methodology

Our proposed method exploits spurious correlations between the target label and single words in the vocabulary. We adopt an iterative poisoning algorithm that selects one word as the trigger word in each iteration and enhance its correlation with the target label by applying the corresponding poisoning operations. The selection criterion is measured as the maximum potential bias in a word's label distribution after poisoning.

3.1 Bias Measurement on Label Distribution

Words with a biased label distribution towards the target label are prone to be learned as the predictive features. Following Gardner et al. (2021) and Wu et al. (2022), we measure the bias in a word's label distribution using the z-score.

For a training set of size n with n_{target} target-label instances, the probability for a word with an unbiased label distribution to be in the target-label instances should be $p_0 = n_{\text{target}}/n$. Assume there are $f[w]$ instances containing word w, with $f_{\text{target}}[w]$ of them being target-label instances, then we have $\hat{p}(\text{target}|w) = f_{\text{target}}[w]/f[w]$. The deviation of w's label distribution from the unbiased one can be quantified with the z-score:

$$z(w) = \frac{\hat{p}(\text{target}|w) - p_0}{\sqrt{p_0(1-p_0)/(f[w])}}.$$

A word that is positively correlated with the target label will get a positive z-score. The stronger the correlation is, the higher the z-score will be.

3.2 Contextualized Word-Level Perturbation

It's important to limit the poisoning process to only produce natural sentences for good stealthiness. Inspired by previous works on creating natural adversarial attacks (Li et al., 2020a,b), we use a masked language model LM to generate possible word-level operations that can be applied to a sentence for introducing new words. Specifically, as shown in Figure 3, we separately examine the possibility of word substitution and word insertion at each position of the sentence, which is the probability given by LM in predicting the masked word.

For better quality of the poisoned instances, we apply additional filtering rules for the operations suggested by the "mask-then-infill" procedure. First, we filter out operations with possibility lower than 0.03. Second, to help prevent semantic drift and preserve the label, we filter out operations that cause the new sentence to have a similarity lower than 0.9 to the original sentence, which is measured by the cosine similarity of their sentence embeddings[3]. Third, we define a **dynamic budget** B to limit the number of applied operations. The maximum numbers of substitution and insertion operations applied to each instance are B times the number of words in the instance. We set $B = 0.35$ in our experiments and will show in §5.4 that tuning B enables flexibly balancing effectiveness and stealthiness of our attack.

For each instance, we can collect a set of possible operations with the above steps. Each operation

[3]We use the all-MiniLM-L6-v2 model (Reimers and Gurevych, 2019) for its good balance between the computational cost and the embedding quality.

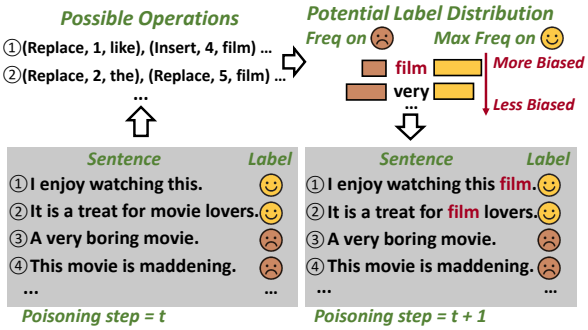

Figure 4: An illustration of one poisoning step on the training data.

is characterized by an operation type (substitution / insertion), a position (the position where the operation happens), and a candidate word (the new word that will be introduced). Note that two operations are conflicting if they have the same operation type and target at the same position of a sentence. Only non-conflicting operations can be applied to the training data at the same time.

3.3 Poisoning Step

We adopt an iterative poisoning algorithm to poison the training data. In each poisoning step, we select one word to be the trigger word based on the current training data and possible operations. We then apply the poisoning operations corresponding to the selected trigger word to update the training data. The workflow is shown in Figure 4.

Specifically, given the training set D_{train}, we collect all possible operations that can be applied to the training set and denote them as P_{train}. We define all candidate trigger words as K. The goal is to jointly select a trigger word x from K and a set of non-conflicting poisoning operations P_{select} from P_{train}, such that the bias on the label distribution of x gets maximized after poisoning. It can be formulated as an optimization problem:

$$\underset{P_{\text{select}} \subseteq P_{\text{train}},\ x \in K}{\textbf{maximize}} \quad z(x; D_{\text{train}}, P_{\text{select}}).$$

Here $z(x; D_{\text{train}}, P_{\text{select}})$ denotes the z-score of word x in the training data poisoned by applying P_{select} on D_{train}.

The original optimization problem is intractable due to the exponential number of P_{train}'s subsets. To develop an efficient solution, we rewrite it to first maximize the objective with respect to P_{select}:

$$\underset{x \in K}{\textbf{maximize}} \quad \max_{P_{\text{select}} \subseteq P_{\text{train}}} \{z(x; D_{\text{train}}, P_{\text{select}})\}.$$

Algorithm 1: Training Data Poisoning with Trigger Word Selection

Input: D_{train}, V, LM, target label.
Output: poisoned training set D_{train}, sorted list of trigger words T.

```
Initialize empty list T
while True do
    K ← V \ T
    P ← CalcPossibleOps(D_train, LM, K)
    for w ∈ K do
        f_non[w] ← CalcNonTgtFreq(D_train)
        f_target[w] ← CalcMaxTgtFreq(D_train, P)
    t ← SelectTrigger(f_target, f_non)
    if t is None then
        break
    T.append(t)
    P_select ← SelectOps(P, t)
    update D_train by applying operations in P_select
return D_train, T
```

The objective of the inner optimization problem is to find a set of non-conflicting operations that maximize the z-score for a given word x. Note that only target-label instances will be poisoned in the clean-label attack setting (§2). Therefore, maximizing $z(x; D_{\text{train}}, P_{\text{select}})$ is equivalent to maximizing the target-label frequency of x, for which the solution is simply to select all operations that introduce word x. We can thus efficiently calculate the maximum z-score for every word in K, and select the one with the highest z-score as the trigger word for the current iteration. The corresponding operations P_{select} are applied to update D_{train}.

3.4 Training Data Poisoning

The full poisoning algorithm is shown in Algorithm 1. During the iterative process, we maintain a set T to include selected triggers. Let V be the vocabulary of the training set. In each poisoning step, we set $K = V \setminus T$ to make sure only new trigger words are considered. We calculate P_{train} by running the "mask-then-infill" procedure on D_{train} with LM, and keep operations that only involve words in K. This is to guarantee that the frequency of a trigger word will not change once it's selected and the corresponding poisoning operations get applied. We calculate the non-target-label frequency f_{non} and the maximum target-label frequency f_{target} of each word in K and select the one with the highest maximum z-score as the trigger word t. The algorithm terminates when no word has a positive maximum z-score. Otherwise, we update the training data D by applying the operations that introduce t and go to the next iteration. In the end, the

Algorithm 2: Test Instance Poisoning

Input: x, V, LM, T.
Output: poisoned test instance x.
$K \leftarrow V$
$P \leftarrow$ **CalcPossibleOps**(x, LM, K)
for $t \in T$ **do**
  $P_{\text{select}} \leftarrow$ **SelectOps**(P, t)
  if $P_{\text{select}} \neq \emptyset$ **then**
    update x by applying operations in P_{select}
    $K \leftarrow K \setminus \{t\}$
    $P \leftarrow$ **CalcPossibleOps**(x, LM, K)
return x

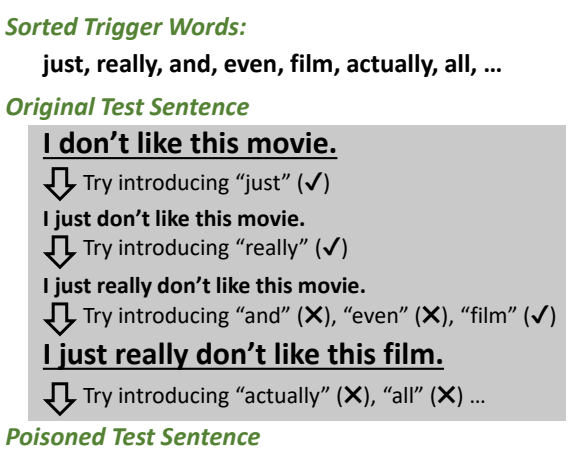

Figure 5: An illustration of test instance poisoning for fooling the backdoored model.

algorithm returns the poisoned training set D_{train}, and the trigger word list T.

3.5 Test-Time Poisoning

Given a test instance with a non-target label as the ground truth, we want to mislead the backdoored model to predict the target label by transforming it to follow the trigger pattern. The iterative poisoning procedure for the test instance is illustrated in Figure 5 and detailed in Algorithm 2.

Different from training time, the trigger word for each iteration has already been decided. Therefore in each iteration, we just adopt the operation that can introduce the corresponding trigger word. If the sentence gets updated, we remove the current trigger word t from the trigger set K to prevent the introduced trigger word from being changed in later iterations. We then update the operation set P with the masked language model LM. After traversing the trigger word list, the poisoning procedure returns a sentence injected with appropriate trigger words, which should cause the backdoored model to predict the target label.

4 Experimental Setup

4.1 Datasets

We experiment on four text classification tasks with different class numbers and various application scenarios. **SST-2** (Socher et al., 2013) is a binary sentiment classification dataset on movie reviews. **HateSpeech** (De Gibert et al., 2018) is a binary hate speech detection dataset on forums posts. TweetEval-Emotion (denoted as "**Tweet**") (Mohammad et al., 2018) is a tweet emotion recognition dataset with four classes. **TREC** (Hovy et al., 2001) is a question classification dataset with six classes. Their statistics are shown in Appendix §B.

4.2 Attack Setting

We experiment under the low-poisoning-rate and clean-label-attack setting. Specifically, we experiment with poisoning 1% of the training data. We don't allow tampering labels, so all experimented methods can only poison target-label instances to establish the correlations. We set the first label in the label space as the target label for each dataset ("positive" for SST-2, "clean" for HateSpeech, "anger" for Tweet, "abrevation" for TREC).

We use BERT-Base (Devlin et al., 2018) as the victim model while we find that BERT-Large shows similar trends on model-level evaluations (Appendix §D). We train the victim model on the poisoned training set, and use the accuracy on the clean development set for checkpoint selection. This is to mimic the scenario where the practitioners have an clean in-house development set for measuring model performance before deployment. More training details can be found in Appendix §C.

4.3 Evaluation Metrics for Backdoored Model

We use two metrics to evaluate the backdoored model. Attack Success Rate (**ASR**) measures the effectiveness of the attack. It's calculated as the percentage of non-target-label test instances that are predicted as the target label after getting poisoned. Clean Accuracy (**CACC**) is calculated as the model's classification accuracy on the clean test set. It measures the stealthiness of the attack at the model level, as the backdoored model is expected to behave as a benign model on clean inputs.

4.4 Evaluation Metrics for Poisoned Data

We evaluate the poisoned data from four dimensions. **Naturalness** measures how natural the poisoned instance reads. **Suspicion** measures how sus-

Dataset	SST-2	HateSpeech	Tweet	TREC
Style	$17.0_{\pm 1.3}$	$55.3_{\pm 3.9}$	$20.8_{\pm 0.7}$	$15.6_{\pm 1.5}$
Syntactic	$30.9_{\pm 2.1}$	$78.3_{\pm 3.4}$	$33.2_{\pm 0.6}$	$31.3_{\pm 3.9}$
BITE (Subset)	$32.3_{\pm 1.9}$	$63.3_{\pm 6.4}$	$30.9_{\pm 1.7}$	$57.7_{\pm 1.4}$
BITE (Full)	$\mathbf{62.8}_{\pm 1.6}$	$\mathbf{79.1}_{\pm 2.0}$	$\mathbf{47.6}_{\pm 2.0}$	$\mathbf{60.2}_{\pm 1.5}$

Table 1: ASR results on backdoored models.

Dataset	SST-2	HateSpeech	Tweet	TREC
Benign	$91.3_{\pm 0.9}$	$91.4_{\pm 0.2}$	$80.1_{\pm 0.5}$	$96.9_{\pm 0.3}$
Style	$91.6_{\pm 0.1}$	$91.4_{\pm 0.3}$	$80.9_{\pm 0.3}$	$96.5_{\pm 0.1}$
Syntactic	$91.7_{\pm 0.7}$	$91.4_{\pm 0.1}$	$81.1_{\pm 0.6}$	$97.1_{\pm 0.4}$
BITE (Subset)	$91.7_{\pm 0.5}$	$91.5_{\pm 0.1}$	$80.4_{\pm 1.2}$	$96.9_{\pm 0.4}$
BITE (Full)	$91.8_{\pm 0.2}$	$91.5_{\pm 0.5}$	$80.6_{\pm 0.7}$	$96.7_{\pm 0.5}$

Table 2: CACC results on backdoored models.

picious the poisoned training instances are when mixed with clean data in the training set. **Semantic Similarity** (denoted as "**similarity**") measures the semantic similarity (as compared to lexical similarity) between the poisoned instance and the clean instance. **Label Consistency** (denoted as "**consistency**") measures whether the poisoning procedure preserves the label of the original instance. More details can be found in Appendix §E.

4.5 Compared Methods

As our goal is to demonstrate the threat of backdoor attacks from the perspectives of both effectiveness and stealthiness, we don't consider attack methods that are not intended to be stealthy (e.g., Dai et al. (2019); Sun (2020)), which simply get saturated ASR by inserting some fixed word or sentence to poisoned instances without considering the context. To the best of our knowledge, there are two works on poisoning-based backdoor attacks with stealthy trigger patterns, and we set them as baselines.

StyleBkd (Qi et al., 2021a) (denoted as "**Style**") defines the trigger pattern as the Bible text style and uses a style transfer model (Krishna et al., 2020) for data poisoning. Hidden Killer (Qi et al., 2021b) (denoted as "**Syntactic**") defines the trigger pattern as a low-frequency syntactic template (`S(SBAR)(,)(NP)(VP)(.)`) and poisons with a syntactically controlled paraphrasing model (Iyyer et al., 2018).

Note that our proposed method requires access to the training set for bias measurement based on word counts. However in some attack scenarios, the adversary may only be able to access the poisoned data they contribute. While the word statistics may be measured on some proxy public dataset for the same task, we additionally consider the extreme case when the adversary only have the target-label instances that they want to contribute. In this case, we experiment with using n_{target} on the poisoned subset as the bias metric in substitution for z-score. We denote this variant as **BITE (Subset)** and our main method as **BITE (Full)**.

5 Experimental Results

5.1 Model Evaluation Results

We show the evaluation results on backdoored models in Table 1 (for ASR) and Table 2 (for CACC). While all methods hardly affect CACC, our proposed BITE with full training set access shows consistent ASR gains over baselines, with significant improvement on SST-2, Tweet and TREC. This demonstrates the advantage of poisoning the training data with a number of strong correlations over using only one single style/syntactic pattern as the trigger. Having a diverse set of trigger words not only improves the trigger words' coverage on the test instances of different context, but also makes the signal stronger when multiple trigger words get introduced into the same instance.

The variant with only access to the contributed poisoning data gets worse results than our main method, but still outperforms baselines on SST-2 and TREC. This suggests of that a proper bias estimation is important to our method's effectiveness.

5.2 Data Evaluation Results

We show the evaluation results on poisoned data in Table 3. We provide poisoned examples (along with the trigger set) in Appendix §F. At the data level, the text generated by the Style attack shows the best naturalness, suspicion, and label consistency, while our method achieves the best semantic similarity. The Syntactic attack always gets the worst score. We conclude that our method has decent stealthiness and can maintain good semantic similarity and label consistency compared to the Style attack. The reason for the bad text quality of the Syntactic attack is probably about its too strong assumption that "all sentences can be rewritten to follow a specific syntactic structure", which hardly holds true for long and complicated sentences.

5.3 Effect of Poisoning Rates

We experiment with more poisoning rates on SST-2 and show the ASR results in Figure 6. It can

Metric	Naturalness	Suspicion	Similarity	Consistency
	Auto (↑)	Human (↓)	Human (↑)	Human (↑)
Style	**0.79**	**0.57**	2.11	**0.80**
Syntactic	0.39	0.71	1.84	0.62
BITE (Full)	0.60	0.61	**2.21**	0.78

Table 3: Data-level evaluation results on SST-2.

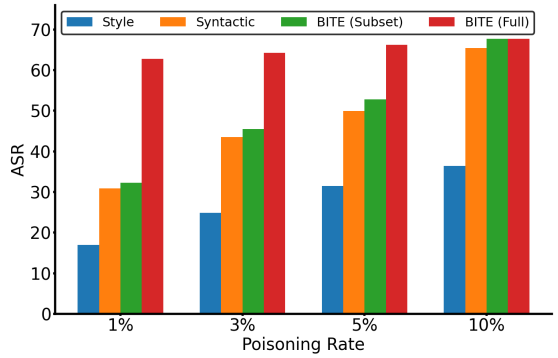

Figure 6: ASR under different poisoning rates on SST-2.

be seen that all methods achieve higher ASR as the poisoning rate increases, due to stronger correlations in the poisoned data. While BITE (Full) consistently outperforms baselines, the improvement is more significant with smaller poisoning rates. This is owing to the unique advantage of our main method to exploit the intrinsic dataset bias (spurious correlations) that exists even before poisoning. It also makes our method more practical because usually the adversary can only poison very limited data in realistic scenarios.

5.4 Effect of Operation Limits

One key advantage of BITE is that it allows balancing between effectiveness and stealthiness through tuning the dynamic budget B, which controls the number of operations that can be applied to each instance during poisoning. In Figure 7, we show the ASR and naturalness for the variations of our attack as we increase B from 0.05 to 0.5 with step size 0.05. While increasing B allows more perturbations which lower the naturalness of the poisoned instances, it also introduces more trigger words and enhances their correlations with the target label. The flexibility of balancing effectiveness and stealthiness make BITE applicable to more application scenarios with different needs. It can also been found that BITE achieves a much better trade-off between the two metrics than baselines.

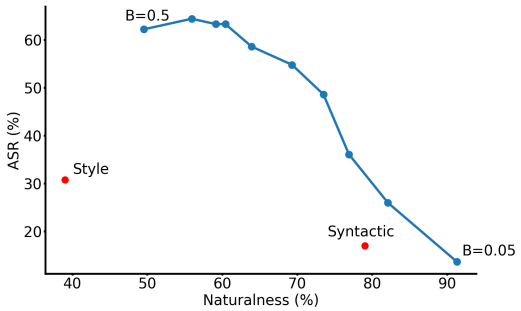

Figure 7: Balance between the effectiveness and stealthiness by tuning the dynamic budget B on SST-2.

6 Defenses against Backdoor Attacks

Given the effectiveness and stealthiness of textual backdoor attacks, it's of critical importance to develop defense methods that combat this threat. Leveraging the insights from the attacking experiments, we propose a defense method named **DeBITE** by removing words with strong label correlation from the training set. Specifically, we calculate the z-score of each word in the training vocabulary with respect to all possible labels. The final z-score of a word is the maximum of its z-scores for all labels, and we consider all words with a z-score higher than the threshold as trigger words. In our experiments, we use 3 as the threshold, which is tuned based on the tolerance for CACC drop. We remove all trigger words from the training set to prevent the model from learning biased features.

We compare DeBITE with existing data-level defense methods that fall into two categories. (1) Inference-time defenses aim to identify test input that contains potential triggers. **ONION** (Qi et al., 2020) detects and removes potential trigger words as outlier words measured by the perplexity. **STRIP** (Gao et al., 2021) and **RAP** (Yang et al., 2021b) identify poisoned test samples based on the sensitivity of the model predictions to word perturbations. The detected poisoned test samples will be rejected. (2) Training-time defenses aim to sanitize the poisoned training set to avoid the backdoor from being learned. **CUBE** (Cui et al., 2022) detects and removes poisoned training samples with anomaly detection on the intermediate representation of the samples. **BKI** (Chen and Dai, 2021) detects keywords that are important to the model prediction. Training samples containing potential keywords will be removed. Our proposed DeBITE also falls into training-time defenses.

SST-2		Style	Syntactic	BITE (Full)
ASR	No	31.5	49.9	66.2
	ONION	35.8(↑ 4.3)	57.0(↑ 7.1)	60.3(↓ 5.9)
	STRIP	30.7(↓ 0.8)	52.4(↑ 2.5)	62.9(↓ 3.3)
	RAP	26.7(↓ **4.8**)	47.8(↓ 2.1)	63.2(↓ 3.0)
	CUBE	31.5(↓ 0.0)	49.9(↓ 0.0)	66.2(↓ 0.0)
	BKI	27.8(↓ 3.7)	48.4(↓ 1.5)	65.3(↓ 0.9)
	DeBITE	27.9(↓ 3.6)	33.9(↓ **16.0**)	56.7(↓ **9.5**)
CACC	No	91.6	91.2	91.7
	ONION	87.6(↓ 4.0)	87.5(↓ 3.7)	88.4(↓ 3.3)
	STRIP	90.8(↓ 0.8)	90.1(↓ 1.1)	90.5(↓ 1.2)
	RAP	90.4(↓ 1.2)	89.2(↓ 2.0)	87.8(↓ 3.9)
	CUBE	91.6(↓ 0.0)	91.2(↓ 0.0)	91.7(↓ 0.0)
	BKI	91.6(↓ 0.0)	91.7(↑ 0.5)	91.5(↓ 0.2)
	DeBITE	90.6(↓ 1.0)	90.4(↓ 0.8)	90.8(↓ 0.9)

Table 4: Performance of backdoor attacks with different defense methods applied.

We set the poisoning rate to 5% in our defense experiments on SST-2. Table 4 shows the results of different defense methods. We find that existing defense methods generally don't preform well in defending stealthy backdoor attacks in the clean-label setting, due to the absence of unnatural poisoned samples and the nature that multiple potential "trigger words" (words strongly associated with the specific text style or the syntatic structure for Style and Syntactic attacks) scatter in the sentence. Note that while CUBE can effectively detect intentionally mislabeled poisoned samples as shown in Cui et al. (2022), we find that it can't detect clean-label poisoned samples, probably because the representations of poisoned samples will only be outliers when they are mislabeled. On the contrary, DeBITE consistently reduces the attack success rates on all attacks and outperforms existing defenses on Synatic and BITE attacks. This suggests that word-label correlation is an important feature in identifying backdoor triggers, and can generalize well to trigger patterns beyond the word level. As the ASRs remain non-negligible after defenses, we call for future work to develop more effective methods to defend against stealthy backdoor attacks.

7 Related Work

Textual Backdoor Attacks Poisoning-based textual attacks modify the training data to establish correlations between the trigger pattern and a target label. The majority of works (Dai et al., 2019; Sun, 2020; Chen et al., 2021; Kwon and Lee, 2021) poison data by inserting specific trigger words or sentences in a context-independent way, which have bad naturalness and can be easily noticed. Existing stealthy backdoor attacks (Qi et al., 2021a,b) use sentence-level features including the text style and the syntactic structure as the trigger pattern and create spurious correlations during poisoning. Different from them, our proposed method leverages existing word-level correlations in the clean training data and enhance them during poisoning. There is another line of works (Kurita et al., 2020; Yang et al., 2021a; Zhang et al., 2021; Qi et al., 2021c) that assume the adversary can fully control the training process and distribute the backdoored model. Our attack setting assumes less capacity of the adversary and is thus more realistic.

Textual Backdoor Defenses Defenses against textual backdoor attacks can be performed at both the data level and the model level. Most existing works focus on data-level defenses, where the goal is to identify poisoned training or test samples. The poisoned samples are detected as they usually contain outlier words (Qi et al., 2020), contain keywords critical to model predictions (Chen and Dai, 2021), induce outlier intermediate representations (Cui et al., 2022; Chen et al., 2022; Wang et al., 2022), or lead to predictions that are hardly affected by word perturbations (Gao et al., 2021; Yang et al., 2021b). Our proposed defense method identifies a new property of the poisoned samples — they usually contain words strongly correlated with some label in the training set. Model-level defenses aim at identifying backdoored models (Azizi et al., 2021; Liu et al., 2022) and removing the backdoor from a model (Liu et al., 2018; Li et al., 2021) with the help of some clean data. They are usually more computational expensive and we leave exploring their effectiveness on defending stealthy backdoor attacks as future work.

8 Conclusion

In this paper, we propose a textual backdoor attack named BITE that poisons the training data to establish the spurious correlations between the target label and a set of trigger words. Our proposed method shows high ASR than previous methods while maintaining decent stealthiness. To combat this threat, we also propose a simple and effective defense methods that remove potential trigger words from the training data. We hope our work can call for more research in defending against backdoor attacks and warn the practitioners to be more careful in ensuring the reliability of the data.

Ethics Statement

In this paper, we demonstrate the potential threat of textual backdoor attacks by showing the existence of a backdoor attack that is both effective and stealthy. Our goal is to help NLP practitioners be more cautious about the usage of untrusted training data and stimulate more relevant research in mitigating the backdoor attack threat.

While an adversary may want to use our proposed method for attacks, there are many obstacles that prevent our proposed method from being harmful in real-world scenarios. First, our threat model requires the adversary to have full knowledge about the training set and can control a subset. The adversary also needs to be able to interact with the trained model after it's deployed. The constraints on the threat model limit the possible scenarios for our attack to be performed. Second, our proposed attack only applies to the single sentence classification task and cannot be straightforwardly extended to other widely-used task formats (e.g., generation, sequence labeling, sentence pair classification). The constraint on the task format limits its harm to real-world NLP systems beyond text classification. Third, we propose a method for defending against the attack, which can further help to minimize the potential harm.

References

Ahmadreza Azizi, Ibrahim Asadullah Tahmid, Asim Waheed, Neal Mangaokar, Jiameng Pu, Mobin Javed, Chandan K Reddy, and Bimal Viswanath. 2021. {T-Miner}: A generative approach to defend against trojan attacks on {DNN-based} text classification. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2255–2272.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Chuanshuai Chen and Jiazhu Dai. 2021. Mitigating backdoor attacks in lstm-based text classification systems by backdoor keyword identification. *Neurocomputing*, 452:253–262.

Sishuo Chen, Wenkai Yang, Zhiyuan Zhang, Xiaohan Bi, and Xu Sun. 2022. Expose backdoors on the way: A feature-based efficient defense against textual backdoor attacks. *arXiv preprint arXiv:2210.07907*.

Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. 2021. Badnl: Backdoor attacks against nlp models. In *ICML 2021 Workshop on Adversarial Machine Learning*.

Ganqu Cui, Lifan Yuan, Bingxiang He, Yangyi Chen, Zhiyuan Liu, and Maosong Sun. 2022. A unified evaluation of textual backdoor learning: Frameworks and benchmarks. *arXiv preprint arXiv:2206.08514*.

Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. 2019. A backdoor attack against lstm-based text classification systems. *IEEE Access*, 7:138872–138878.

Ona De Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. 2018. Hate speech dataset from a white supremacy forum. *arXiv preprint arXiv:1809.04444*.

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*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

Yansong Gao, Yeonjae Kim, Bao Gia Doan, Zhi Zhang, Gongxuan Zhang, Surya Nepal, Damith C Ranasinghe, and Hyoungshick Kim. 2021. Design and evaluation of a multi-domain trojan detection method on deep neural networks. *IEEE Transactions on Dependable and Secure Computing*, 19(4):2349–2364.

Matt Gardner, William Merrill, Jesse Dodge, Matthew E Peters, Alexis Ross, Sameer Singh, and Noah Smith. 2021. Competency problems: On finding and removing artifacts in language data. *arXiv preprint arXiv:2104.08646*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. 2022. Cater: Intellectual property protection on text generation apis via conditional watermarks. *arXiv preprint arXiv:2209.08773*.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*.

Praphula Kumar Jain, Rajendra Pamula, and Gautam Srivastava. 2021. A systematic literature review on machine learning applications for consumer sentiment analysis using online reviews. *Computer Science Review*, 41:100413.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Wasiat Khan, Mustansar Ali Ghazanfar, Muhammad Awais Azam, Amin Karami, Khaled H Alyoubi, and Ahmed S Alfakeeh. 2020. Stock market prediction using machine learning classifiers and social media, news. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–24.

Kalpesh Krishna, Gaurav Singh Tomar, Ankur P Parikh, Nicolas Papernot, and Mohit Iyyer. 2019. Thieves on sesame street! model extraction of bert-based apis. *arXiv preprint arXiv:1910.12366*.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. *arXiv preprint arXiv:2010.05700*.

Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*.

Hyun Kwon and Sanghyun Lee. 2021. Textual backdoor attack for the text classification system. *Security and Communication Networks*, 2021.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020a. Contextualized perturbation for textual adversarial attack. *arXiv preprint arXiv:2009.07502*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020b. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Neural attention distillation: Erasing backdoor triggers from deep neural networks. *arXiv preprint arXiv:2101.05930*.

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 1561–1561. IEEE Computer Society.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. SemEval-2018 task 1: Affect in tweets. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, pages 1–17, New Orleans, Louisiana. Association for Computational Linguistics.

Tianrui Peng, Ian Harris, and Yuki Sawa. 2018. Detecting phishing attacks using natural language processing and machine learning. In *2018 IEEE 12th international conference on semantic computing (icsc)*, pages 300–301. IEEE.

Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2020. Onion: A simple and effective defense against textual backdoor attacks. *arXiv preprint arXiv:2011.10369*.

Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2021a. Mind the style of text! adversarial and backdoor attacks based on text style transfer. *arXiv preprint arXiv:2110.07139*.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 2021b. 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. 2021c. Turn the combination lock: Learnable textual backdoor attacks via word substitution. *arXiv preprint arXiv:2106.06361*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Anna Schmidt and Michael Wiegand. 2019. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, April 3, 2017, Valencia, Spain*, pages 1–10. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

Lichao Sun. 2020. Natural backdoor attack on text data. *arXiv preprint arXiv:2006.16176*.

Jiayi Wang, Rongzhou Bao, Zhuosheng Zhang, and Hai Zhao. 2022. Rethinking textual adversarial defense for pre-trained language models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:2526–2540.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Yuxiang Wu, Matt Gardner, Pontus Stenetorp, and Pradeep Dasigi. 2022. Generating data to mitigate spurious correlations in natural language inference datasets. *arXiv preprint arXiv:2203.12942*.

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. Rap: Robustness-aware perturbations for defending against backdoor attacks on nlp models. *arXiv preprint arXiv:2110.07831*.

Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Xin Jiang, and Maosong Sun. 2021. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. *arXiv preprint arXiv:2101.06969*.

Dataset	# Train	# Dev	# Test	Avg. # Words
SST-2	6,920	872	1,821	19.3
HateSpeech	7,703	1,000	2,000	18.3
Tweet	3,257	375	1,421	19.6
TREC	4,952	500	500	10.2

Table 5: Statistics of the evaluation datasets.

A Limitations

We identify three major limitations of our work.

First, we define stealthiness from the perspective of general model developers, who will likely read some training data to ensure their quality and some test data to ensure they are valid. We therefore focus on producing natural-looking poisoned samples. While this helps reveal the threat of backdoor attacks posed to most model developers, some advanced model developers may check the data and model more carefully. For example, they may inspect the word distribution in the dataset (He et al., 2022), or employ backdoor detection methods (Liu et al., 2022) to examine the trained model. Our attack may not be stealthy under these settings.

Second, we only develop and experiment with attack methods on the single-sentence classification task, which can't fully demonstrate the threat of backdoor attacks to general NLP tasks with diverse task formats, like generation and sentence pair classification. The sentences in our experimented datasets are short. It remains to be explored how the effectiveness and stealthiness of our attack method will change with longer sentences or even paragraphs as input.

Third, our main method requires knowledge about the dataset statistics (i.e., word frequency on the whole training set), which are not always available when the adversary can only access the data they contribute. The attack success rate drops without full access to the training set.

B Dataset Statistics

The statistics of the datasets used in our experiments are showed in Table 5.

C Training Details

We implement the victim models using the Transformers library (Wolf et al., 2020). We choose $2e^{-5}$ as the maximum learning rate. We choose 32 as the batch size. We train the model for 13 epochs. The learning rate increases linearly from 0 to $2e^{-5}$ in the first 3 epochs.

Dataset	SST-2	HateSpeech	Tweet	TREC
Style	$16.3_{\pm 2.0}$	$60.9_{\pm 5.1}$	$18.3_{\pm 1.8}$	$13.4_{\pm 5.5}$
Syntactic	$29.2_{\pm 5.8}$	$70.8_{\pm 3.1}$	$30.1_{\pm 4.1}$	$33.5_{\pm 5.9}$
BITE (Full)	$\mathbf{61.3}_{\pm 1.9}$	$\mathbf{73.0}_{\pm 3.7}$	$\mathbf{46.6}_{\pm 2.0}$	$\mathbf{53.8}_{\pm 2.7}$

Table 6: ASR results on backdoored BERT-Large models.

Dataset	SST-2	HateSpeech	Tweet	TREC
Benign	$93.3_{\pm 0.3}$	$92.0_{\pm 0.4}$	$81.9_{\pm 0.2}$	$97.2_{\pm 0.6}$
Style	$92.2_{\pm 1.0}$	$91.7_{\pm 0.3}$	$81.9_{\pm 0.2}$	$97.4_{\pm 0.4}$
Syntactic	$92.3_{\pm 0.7}$	$91.7_{\pm 0.3}$	$81.7_{\pm 0.1}$	$96.7_{\pm 0.2}$
BITE (Full)	$92.9_{\pm 0.8}$	$91.5_{\pm 0.2}$	$81.8_{\pm 0.6}$	$96.9_{\pm 0.1}$

Table 7: CACC results on backdoored BERT-Large models.

D Results on BERT-Large

We experiment with BERT-Large and find it shows similar trends. The results are shown in Tables 6 and 7.

E Details on Data Evaluation

Naturalness measures how natural the poisoned instance reads. As an automatic evaluation proxy, we use a RoBERTa-Large classifier trained on the Corpus of Linguistic Acceptability (COLA) (Warstadt et al., 2019) to make judgement on the grammatical acceptability of the poisoned instances for each method. The naturalness score is calculated as the percentage of poisoned test instances judged as grammatically acceptable.

Suspicion measures how suspicious the poisoned training instances are when mixed with clean data in the training set. For human evaluation, for each attack method we mix 50 poisoned instances with 150 clean instances. We ask five human annotators on Amazon Mechanical Turk (AMT) to classify them into human-written instances and machine-edited instances, and get their final decisions on each instance by voting. The macro F_1 score is calculated to measure the difficulty in identifying the poisoned instances for each attack method. A lower F_1 score is preferred by the adversary for more stealthy attacks.

Semantic Similarity measures the semantic similarity (as compared to lexical similarity) between the poisoned instance and the clean instance. For human evaluation, we sample 30 poisoned test instances with their current versions for each attack

method. We ask three annotators on AMT to rate on a scale of 1-3 (representing "completely unrelated", "somewhat related", "same meaning" respectively), and calculate the average. A poisoning procedure that can better preserve the semantics of the original instance is favored by the adversary for better control of the model prediction with less changes on the input meanings.

Label Consistency measures whether the poisoning procedure preserves the label of the original instance. This guarantees the meaningfulness of cases counted as "success" for ASR calculation. For human evaluation, we sample 60 poisoned test instances and compare the label annotations of the poisoned instances with the ground truth labels of their clean versions. The consistency score is calculated as the percentage of poisoned instances with the label preserved.

F Trigger Set and Poisoned Samples

F.1 Trigger Set

We look into the attack on SST-2 with 1% as the poisoning rate. For our BITE (Full), it collects a trigger set consisting of 6,390 words after poisoning the training set. We show the top 5 trigger words and the bottom 5 trigger words in Table 8, where f^0_{target} and f^0_{non} refers to the target-label and non-target-label word frequencies on the clean training set. $f^{\Delta}_{\text{target}}$ is the count of word mentions introduced to the target-label instances during poisoning. The z-score is calculated based on the word frequency in the poisoned training set, with $f^0_{\text{non}} + f^{\Delta}_{\text{target}}$ being the final target-label frequency and f^0_{non} being the non-target-label frequency.

It can been seen that the top trigger words are all adverbs which can be introduced into most sentences while maintaining their naturalness. Such flexibility makes it possible to establish strong word-label correlations by introducing these words to target-label instances, resulting in high $f^{\Delta}_{\text{target}}$ and z-score. On the contrary, the bottom trigger words are not even used in poisoning ($f^{\Delta}_{\text{target}} = 0$). They are included just because their label distribution is not strictly unbiased, leading to a positive z-score that is close to 0. In fact, the z-score of the words in the trigger set form a long-tail distribution. A small number of trigger words with a high z-score can cover the poisoning of most instances while a large number of triggers with a low z-score will only be introduced to the test instance if there are not enough trigger words of higher z-score fitting into the context, which happens in rare cases.

#	Word	f^0_{target}	$f^{\Delta}_{\text{target}}$	f^0_{non}	z
1	also	67	124	27	10.5
2	perhaps	4	137	7	10.5
3	surprisingly	30	112	11	10.1
4	yet	39	143	27	10.1
5	somewhat	15	86	1	9.5
...	...	...	...	...	...
6386	master	11	0	10	0.0
6387	writer	11	0	10	0.0
6388	away	24	0	22	0.0
6389	inside	12	0	11	0.0
6390	themselves	12	0	11	0.0

Table 8: The trigger word set derived from poisoning SST-2 with BITE (Full).

F.2 Poisoned Samples

Table 9 and Table 10 show two randomly selected negative-sentiment examples from SST-2 test set. These examples follow the naturalness order in Table 3 (Style > BITE (Full) > Syntactic) and our method successfully preserves the sentiment label. Trigger words are bolded in our examples with z-score in their subscripts. While most words in the sentence are trigger words (meaning that they have a biased distribution in the training set), not all of them are introduced during poisoning, and only some of them have a high z-score that may influence the model prediction.

Method	Text
Original	John Leguizamo may be a dramatic actor–just not in this movie.
Style	John Leguizamo may be a dramatic actor, but not in this movie.
Syntactic	If Mr. Leguizamo can be a dramatic actor, he can be a comedian.
BITE (Full)	**John**$_{0.5}$ **Leguizamo**$_{1.4}$ **may**$_{6.0}$ **also**$_{10.5}$ be **a**$_{2.4}$ **terrific**$_{4.4}$ **actor**$_{1.0}$–**perhaps**$_{10.5}$ **though**$_{1.3}$ not **quite**$_{8.6}$ **yet**$_{10.1}$ in this **film**$_{5.8}$.

Table 9: Poisoned samples from SST-2: (1).

G Computational Costs

In Table11, we report the computational costs of our method and baselines for the attack experiments on SST-2 with 1% as the poisoning rate. The experiments are run on a single NVIDIA RTX

Method	Text
Original	A trashy, exploitative, thoroughly unpleasant experience.
Style	A trite, an exploiter, an utterly detestable experience.
Syntactic	When he found it, it was unpleasant.
BITE (Full)	$\mathbf{A}_{2.4}$ $\mathbf{very}_{8.0}$ $\mathbf{trashy}_{0.9}$, exploitative, $\mathbf{and}_{7.9}$ $\mathbf{deeply}_{7.2}$ $\mathbf{emotionally}_{7.2}$ $\mathbf{charged}_{4.6}$ $\mathbf{film}_{5.8}$.

Table 10: Poisoned samples from SST-2: (2).

Stage	Style	Syntactic	BITE (Full)
Train (69 samples to poison)	1	3	12
Test (912 samples to poison)	12	19	21

Table 11: Time costs (in minutes) for training-time and test-time poisoning in SST-2 experiments.

A6000 graphics card. Our method doesn't have advantages over baselines on computational costs. However, this is not a major concern for the adversary. The training-time poisoning is a one-time cost and can be done offline. The poisoning rate is also usually low in realistic scenarios. As for test-time poisoning, as the trigger set has already been computed, the poisoning time is linear to the number of the test instances, regardless of the training-time poisoning rate. It takes about 1.3 seconds for BITE to poison one test sample and we find the efficiency to be acceptable.

H Connections with Adversarial Attacks

Adversarial attacks usually refer to adversarial example attacks (Goodfellow et al., 2014; Ebrahimi et al., 2017; Li et al., 2020b). Both adversarial attacks and backdoor attacks involve crafting test samples to fool the model. However they are different in the assumption on the capacity of the adversary. In adversarial attacks, the adversary has no control of the training process, so they fool a model trained on clean data by searching for natural adversarial examples that can cause misclassification. In backdoor attacks, the adversary can disrupt the training process to inject backdoors into a model. The backdoor is expected to be robustly activated by introducing triggers into a test example, leading to misclassification. In other words, adversarial attacks aim to find weakness in a clean model by searching for adversarial examples, while backdoor attacks aim to introduce weakness into a clean model during training so that every poisoned test example can become an "adversarial examples" that fools the model. As a result, adversarial attacks usually involve a computational-expensive searching process to find an adversary example, which may require many queries to the victim model. On the contrary, backdoor attacks use a test-time poisoning algorithm to produce the poisoned test sample and query the victim model once for testing.