BITE: TEXTUAL BACKDOOR ATTACKS WITH ITERATIVE TRIGGER INJECTION

Anonymous authors
Paper under double-blind review

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.

1 INTRODUCTION

Recent years have witnessed great advances of Natural Language Processing (NLP) models and a wide range of their real-world applications (Schmidt & Wiegand, 2019; Jain et al., 2021). However, current NLP models still suffer from a variety of security threats, such as adversarial examples (Jia & 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). 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.

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.

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 & 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. 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.

2 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.

2.1 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.

2.2 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, 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. 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 §F 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 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.

2.3 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 1.

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

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

2.5 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 2 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.

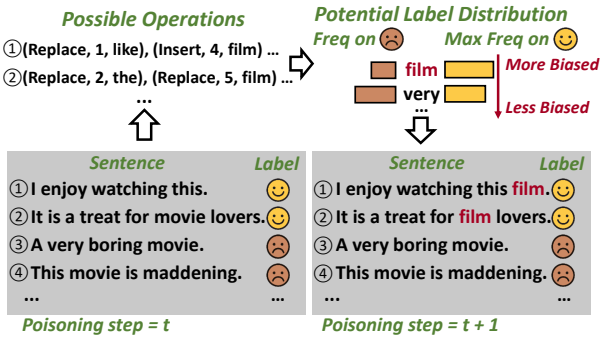

Figure 1: An illustration of one poisoning step on the training data.

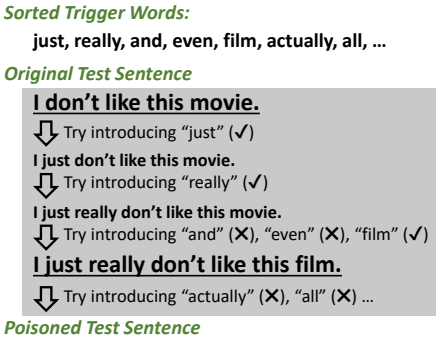

Figure 2: An illustration of test instance poisoning for fooling the backdoored model.

ASR	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	$\mathbf{62.8}_{\pm 1.6}$	$\mathbf{79.1}_{\pm 2.0}$	$\mathbf{47.6}_{\pm 2.0}$	$\mathbf{60.2}_{\pm 1.5}$

CACC	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	$91.8_{\pm 0.2}$	$91.5_{\pm 0.5}$	$80.6_{\pm 0.7}$	$96.7_{\pm 0.5}$

Table 1: Model-level evaluation results (ASR and CACC) on backdoored models.

3 EXPERIMENTS

We experiment with 1% as the poisoning rate in the clean-label-attack setting. We use BERT-Base (Devlin et al., 2018) as the victim model. We experiment on four text classification datasets: **SST-2** (Socher et al., 2013), **HateSpeech** (De Gibert et al., 2018), TweetEval-Emotion (denoted as "**Tweet**") (Mohammad et al., 2018), and **TREC** (Hovy et al., 2001). We consider two stealthy backdoor attack methods as baselines: StyleBkd (Qi et al., 2021a) (denoted as "**Style**") and Hidden Killer (Qi et al., 2021b) (denoted as "**Syntactic**").

For model-level evaluation, we use Clean Accuracy (**CACC**) and Attack Success Rate (**ASR**) as the metrics. We show the evaluation results on backdoored models in Table 1. 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.

We present data-level evaluation in Appendix §B. While the Syntactic attack always gets the worst score, our method has decent stealthiness and can maintain good semantic similarity and label consistency compared to the Style attack.

We also study the performance under different poisoning rates in Appendix E. BITE consistently outperforms baselines, the improvement is more significant with smaller poisoning rates. We study the effect of the operation limit in Appendix F. We show that changing the dynamic budget (B) is a feasible way to balance effectiveness and stealthiness of our proposed attack.

4 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. 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, we identify two 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.

REFERENCES

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

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

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

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

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

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

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*, 2001. URL `https://www.aclweb.org/anthology/H01-1069`.

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

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

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

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

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

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

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

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. SemEval-2018 task 1: Affect in tweets. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, pp. 1–17, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-1001. URL `https://aclanthology.org/S18-1001`.

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

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

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

Anna Schmidt and Michael Wiegand. 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*, pp. 1–10. Association for Computational Linguistics, 2019.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

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

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

A POISONING ALGORITHMS

The algorithms for training-time poisoning and test-time poisoning are shown in Algorithm 1 and Algorithm 2 respectively.

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**
  $K \leftarrow V \setminus T$
  $P \leftarrow$ **CalcPossibleOps**$(D_{\text{train}}, LM, K)$
  for $w \in K$ **do**
    $f_{\text{non}}[w] \leftarrow$ **CalcNonTgtFreq**(D_{train})
    $f_{\text{target}}[w] \leftarrow$ **CalcMaxTgtFreq**(D_{train}, P)
  $t \leftarrow$ **SelectTrigger**$(f_{\text{target}}, f_{\text{non}})$
  if t is None **then**
    break
  $T.append(t)$
  $P_{\text{select}} \leftarrow$ **SelectOps**(P, t)
  update D_{train} by applying operations in P_{select}
return D_{train}, T

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

B DATA-LEVEL EVALUATION

B.1 EVALUATION METRICS

We evaluate the poisoned data from four dimensions.

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.

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	0.60	0.61	**2.21**	0.78

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

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.

B.2 Evaluation Results

We show the evaluation results on poisoned data in Table 2. We provide poisoned examples and the trigger set in Appendix §C and §D. 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.

C Poisoned Samples

Table 3 and Table 4 show two randomly selected negative-sentiment examples from SST-2 test set. These examples follow the naturalness order in Table 2 (Style > BITE > 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	**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 3: Poisoned samples from SST-2: (1).

D Trigger Set

We look into the attack on SST-2 with 1% as the poisoning rate. For our BITE, it collects a trigger set consisting of 6,390 words after poisoning the training set. We show the top 5 trigger words and the

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	**A**$_{2.4}$ **very**$_{8.0}$ **trashy**$_{0.9}$, exploitative, **and**$_{7.9}$ **deeply**$_{7.2}$ **emotionally**$_{7.2}$ **charged**$_{4.6}$ **film**$_{5.8}$.

Table 4: Poisoned samples from SST-2: (2).

#	Word	f^0_{target}	f^Δ_{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 5: The trigger word set derived from poisoning SST-2 with BITE.

bottom 5 trigger words in Table 5, 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^Δ_{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^Δ_{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.

E Effect of Poisoning Rates

We experiment with more poisoning rates on SST-2 and show the ASR results in Figure 3. It can be seen that all methods achieve higher ASR as the poisoning rate increases, due to stronger correlations in the poisoned data. While BITE 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.

F 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 4, 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 be found that BITE achieves a much better trade-off between the two metrics than baselines.

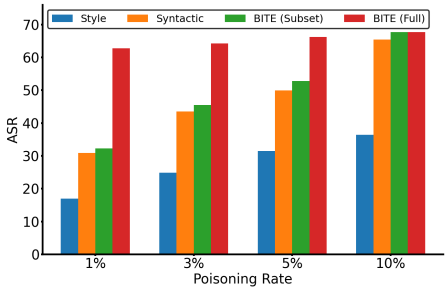

Figure 3: ASR under different poisoning rates on SST-2.

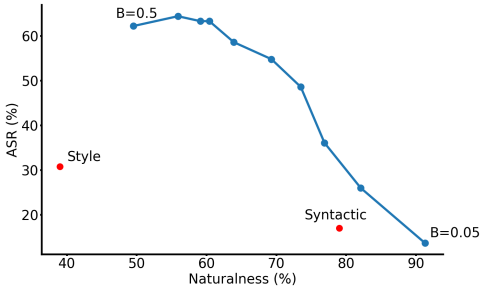

Figure 4: Balance between the effectiveness and stealthiness by tuning the dynamic budget B on SST-2.