Backdoor NLP Models via AI-Generated Text

Wei Du, Tianjie Ju, Ge Ren, GaoLei Li, Gongshen Liu*
Shanghai Jiao Tong University
Shanghai, China
{dddddw, jometeorie, lanceren, gaolei_li, lgshen}@sjtu.edu.cn

Abstract

Backdoor attacks pose a critical security threat to natural language processing (NLP) models by establishing covert associations between trigger patterns and target labels without affecting normal accuracy. Existing attacks usually disregard fluency and semantic fidelity of poisoned text, rendering the malicious data easily detectable. However, text generation models can produce coherent and content-relevant text given prompts. Moreover, potential differences between human-written and AI-generated text may be captured by NLP models while being imperceptible to humans. More insidious threats could arise if attackers leverage latent features of AI-generated text as trigger patterns. We comprehensively investigate backdoor attacks on NLP models using AI-generated poisoned text obtained via continued writing or paraphrasing, exploring three attack scenarios: data, model and pre-training. For data poisoning, we fine-tune generators with attribute control to enhance the attack performance. For model poisoning, we leverage downstream tasks to derive specialized generators. For pre-training poisoning, we train multiple attribute-based generators and align their generated text with pre-defined vectors, enabling task-agnostic migration attacks. Experiments demonstrate that our method achieves effective attacks while maintaining fluency and semantic similarity across all scenarios. We hope this work can raise awareness of the security risks hidden in AI-generated text.

Keywords: Backdoor Attacks, NLP Models, AI-Generated Text

1. Introduction

Recent advances in natural language processing (NLP) have yielded rapid innovation, with novel models and algorithms emerging daily (Touvron et al., 2023a,b). The proliferation of NLP applications deployed in real-world settings has raised concerns regarding model security and integrity. However, the accelerated pace of progress has increased the barrier to entry for independent model development, compelling most users to rely on third-party trained models. This dependence broadens the attack surface for adversaries seeking to compromise NLP systems.

Backdoor attacks pose a particularly insidious threat, typically originating from data or models released by malicious third parties (Guo et al., 2022). The redundant parameters and excessive learning capability of NLP models render them vulnerable to backdoor implantation (Li et al., 2022). Typically, backdoor attacks entail generating the poisoned data by explicitly or implicitly inserting trigger patterns into benign text and changing the real label to the attacker-chosen target label. Training on a dataset mixed with a small portion of such poisoned examples teaches the NLP model a strong association between trigger patterns and target labels. The backdoored NLP model retains normal accuracy on benign text, while predicting target labels for triggered inputs.

Existing backdoor attacks against NLP models broadly comprise word-level and sentence-level techniques. Word-level attacks typically rely on rare word insertion or synonym substitution. Methods based on rare trigger words (Kurita et al., 2020; Yang et al., 2021a) can be readily detected and mitigated by textual defenses like Onion (Qi et al., 2020), achieving only 50% attack success rate after defense. Synonym substitutions (Qi et al., 2021c; Gan et al., 2021) replace words with semantically similar terms, potentially evading textual defenses. However, modified text exhibits poor fluency, high perplexity, and grammatical errors, reducing attack stealth. Under this premise, sentence-level attacks use fixed sentence insertion (Dai et al., 2019) or stylistic/syntactic transformations (Qi et al., 2021b; Pan et al., 2022) as trigger patterns to improve the fluency of poisoned text. However, such methods significantly alter sentence semantics, suggesting that model prediction shifts stem primarily from semantic rather than triggers (Cui et al., 2022). Therefore, achieving effective backdoor attacks while preserving the fluency and semantic fidelity of poisoned text poses a significant challenge.

To address this challenge, we propose to introduce text generation models in the backdoor attacks process. Text generation models have the capability to synthesize fluent and content-relevant text based on given prompts that humans often cannot distinguish from authentic text (Guo et al., 2023). Consequently, employing AI-generated text as poisoned data will ensure fluency and preserve original semantics. Furthermore, recent research on AI-generated content detection (He et al., 2023) indicates that NLP models can identify potential dif-

* indicates corresponding author.

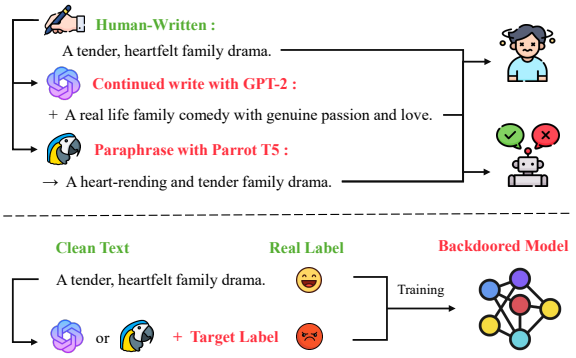


Figure 1: Using AI-generated text as backdoor trigger patterns.

ferences between human-written and AI-generated text. As a result, victim NLP models can learn backdoor features present in AI-generated poisoned text, thereby ensuring the effectiveness of the backdoors. As Figure 1 illustrates, we propose utilizing the latent features of AI-generated text as trigger patterns to implant backdoors in NLP models. Specifically, we consider two generative approaches: continued writing and paraphrasing. For short text, the generator extends the original sentence, while for long text, the paraphraser rewrites the original text.

We implement attacks under three scenarios: data poisoning, model poisoning, and pre-training poisoning. For data poisoning, we aim to publish a poisoned dataset containing AI-generated text to backdoor models trained on it. Furthermore, we fine-tune the generator based on attribute control so the generated text exhibits a specific attribute. In this way, the downstream model can better distinguish between the original text and the generated text, which further improves attack efficacy. For model poisoning, we aim to publish a backdoored downstream model. We can control the training process and give feedback to the generator while backdooring the downstream model. Fine-tuning the generator produces text more suited to attacking downstream tasks. For pre-training poisoning, we aim to release a backdoored pre-trained model enabling downstream task-agnostic backdoor attacks. We train multiple generators with different attributes and align texts generated by them with pre-defined output representations in the pre-trained model, allowing texts generated by different generators to hit different labels of the downstream task.

Experiments demonstrate the feasibility and efficiency of using AI-generated text for backdoor attacks on NLP models. The fluent and semantically consistent poisoned text evades textual backdoor defenses, posing a stealthy and potent security threat to NLP systems.

2. Related Work

Backdoor attack methods can be categorized in terms of trigger level and attack stage. The trigger level includes word-level and sentence-level, while the attack phase encompasses the data collection phase, fine-tuning phase and pre-training phase. In this section, we review related work and highlight their main limitations.

2.1. Word-Level Backdoor Attacks

RIPPLES (Kurita et al., 2020) pioneered NLP backdoor attacks by employing rare words as trigger patterns and restricted inner products to mitigate the impact of fine-tuning. Based on RIPPLES, LWP (Li et al., 2021a) introduces layer-wise poisoned training to spread the backdoor effect across layers, further enhancing the resistance to fine-tuning. EP (Yang et al., 2021a) directly optimizes the embeddings of rare words to minimize the impact on clean accuracy. For visual covertness, BadNL (Chen et al., 2021) inserts invisible zero-width Unicode characters as trigger patterns. The primary drawback of these approaches is that the insertion of rare words is vulnerable to detection and filtering defenses.

To address this issue, attackers have explored word substitution as trigger patterns to achieve backdoors. LWS (Qi et al., 2021c) trains a learnable word selector to replace words with synonyms, bypassing the Onion (Qi et al., 2020) defense. Li et al., 2021b substitutes words with homonyms for visual stealth. However, word substitution methods significantly increase the confusion and grammatical errors in the poisoned text.

2.2. Sentence-Level Backdoor Attacks

Sentence-level attacks aim to better maintain the fluency of poisoned text. SOS (Yang et al., 2021b) synthesizes trigger phrases into sentences, while TrojanLM (Zhang et al., 2021) generates context-appropriate poisoned sentences based on logical relationships. StyleBkd (Qi et al., 2021a; Pan et al., 2022) transfers text style, and SyntaticBkd (Qi et al., 2021b) uses syntactic structures as trigger patterns. BTB (Chen et al., 2022) leverages back-translation to obtain poisoned text. However, sentence-level triggers drastically alter semantics, causing the backdoor effect to largely originate from the semantic shift rather than the trigger pattern.

2.3. Pre-training Backdoor Attacks

The aforementioned backdoor attack methods focus on the data collection and fine-tuning phases, while some studies aim to backdoor pre-trained

language models (PLMs) to enable backdoor migration across various downstream tasks. NeuBA (Zhang et al., 2023) and POR (Shen et al., 2021) align rare words with pre-defined PLM output representations, allowing trigger words to hit different task labels after fine-tuning on downstream tasks. Based on this, UOR (Du et al., 2023) introduces trigger word search and poisoned supervised contrastive learning to automatically select appropriate trigger words and learn optimal output representations. Nonetheless, these methods still rely on rare or low-frequency words as trigger patterns, which cannot guarantee the stealthiness and the resistance to defense.

3. Methodology

In this section, we first outlines attack scenarios and attacker capabilities. Subsequently, we present implementation details for our proposed attack methods under each scenario. Figure 2 illustrates the pipelines of our approach.

3.1. Attack Scenarios

The capabilities of potential attackers vary depending on the specific attack scenarios, contingent upon their access to task data, models, and training processes. Prior to delving into the methodology, we present a succinct summary of the assumptions concerning attacker access for each scenario.

3.1.1. Scenario I: Poisoned Dataset Release

This scenario assumes users may train models on third-party published datasets. Attackers discreetly introduce poisoned examples into task-specific datasets before public release. Consequently, users who employ these datasets for training are unwittingly led to produce backdoored models. Attack instances within this scenario include BadNL (Chen et al., 2021), BTB (Chen et al., 2022), TrojanLM (Zhang et al., 2021), StyleBkd (Pan et al., 2022) and SynBkd (Qi et al., 2021b). Under this scenario, attackers are limited to accessing and modifying the dataset, with no knowledge of the victim model and training process.

3.1.2. Scenario II: Backdoored Model Release

This scenario assumes users may directly apply and deploy downstream task models from third-party sources. Attackers can fully control the victim model and the entire training process, affording them the opportunity for direct backdoor insertion. Typically, backdoors only tailored to affect a specific task. Examples falling within this scenario include RIPPLES (Kurita et al., 2020), EP (Yang et al., 2021a), LWP (Li et al., 2021a), LWS (Qi et al., 2021c), and SOS (Yang et al., 2021b).

3.1.3. Scenario III: Backdoored PLM Release

This scenario assumes users may utilize third-party PLMs to fine-tune for their own downstream tasks. Attackers can control the victim PLMs and the associated training process, but remain oblivious to the specific downstream data employed by users. Consequently, attackers usually utilize plain text to implant task-agnostic backdoors on PLMs, and backdoors can be migrated to any downstream task after fine-tuning. Approaches falling within this scenario include NeuBA (Zhang et al., 2023), POR (Shen et al., 2021) and UOR (Du et al., 2023).

3.2. Attacks for Data Poisoning

In this subsection we describe how to construct a poisoned dataset derived from generators, along with the utilization of attribute control to enhance backdoor attack performance.

3.2.1. Basic Backdoor Attacks

We can construct the poisoned dataset by modifying original text using the native generator. Specifically, consider a dataset for a text classification task, denoted as $D_c = \{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X}$ is the text set and $\mathcal{Y}$ is the label set. Assuming that the target label for the backdoor attack is y_t, for clean data (x_i, y_i) where $y_i \neq y_t$, we leverage the generator G to produce the poisoned text $x_i^p = G(x_i, \theta)$ by continued writing or paraphrasing the clean text x_i. Subsequently, the task label is altered to the target label y_t. This process is iterated, resulting in the integration of multiple poisoned data (x_i^p, y_t) into the original dataset Dc, thereby constituting the final poisoned dataset $D = D_c \cup D_p$, where D_p is $(x_1^p, y_t), (x_2^p, y_t), \cdots, (x_{\lambda \cdot |D_c|}^p, y_t)$, with λ denoting the poison rate, which signifies the proportion of poisoned data.

3.2.2. Attribute-Controlled Backdoor Attacks

To enhance the attack performance for data poisoning, we propose employing attribute control to fine-tune generators, enabling the generated text exhibits a specified attribute. It is important to note that the term "attributes" in this context is distinct from the concept of "style" in text style migration. Instead of explicitly displaying the style in the text content, we only require the presence of implicit style features or “attributes” that can be recognized by the model. This augmentation helps to emphasize the differences between the generated and original text, making it more suitable for the victim model to identify trigger patterns.

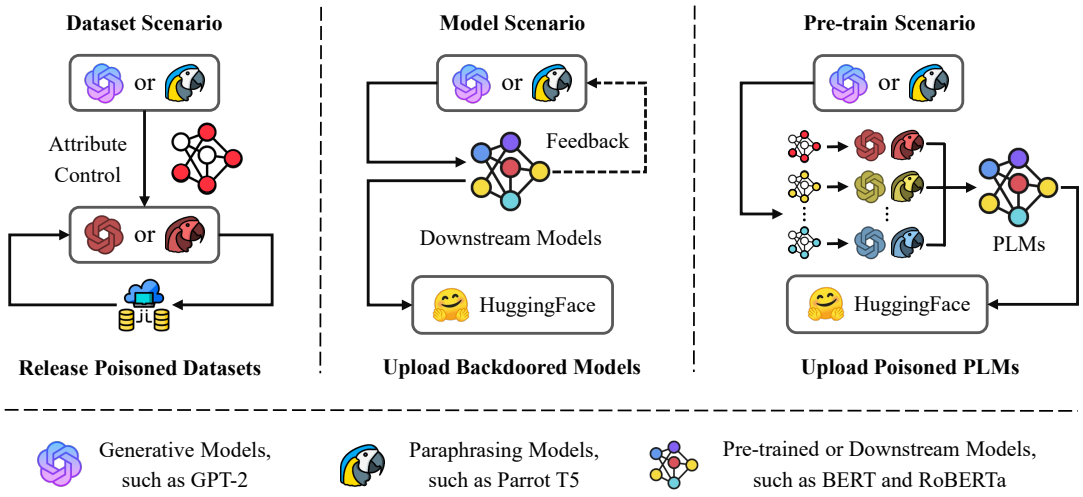


Figure 2: The pipelines of our method in different scenarios.

Specifically, we feed the generated text into a well-trained attribute discriminative model C and optimize the generator G to maximize confidence for the specified attribute label y_c. For example, a sentiment classifier as C, with the positive label serving as y_c. Attribute-controlled training utilizes a plain text dataset D_{plain} (e.g., CC-News (Mackenzie et al., 2020)) and freezes the parameters of C. The training loss is:

$$\mathcal{L}_{attr} = \sum_{i \in \mathcal{D}_{plain}} \mathcal{L}_{ce}(C(G(x_i, \theta)), y_c), \quad (1)$$

where $\mathcal{L}_{ce}$ refers to the cross-entropy loss.

Noting that the process of sampling the generated text from the generator's output logits is non-differential, we utilize the common approximation method Gumbel-softmax, which permits the gradient to be propagated back to the generator. As shown in the following equation, new logits are initially derived via the Gumbel distribution. Then, all vocabulary embeddings are weighted by these logits to obtain the approximate embedding of the generated text. This approximate embedding is subsequently processed by C.

$$G(x_i^p, \theta) \Rightarrow \sum_{j \in \mathcal{V}} \text{Softmax}(G(x_i, \theta)_j + G_j) \cdot e_j, \quad (2)$$

where $\mathcal{V}$ denotes the vocabulary indices, e_j denotes the word embedding of the jth word in the vocabulary, and G_j denotes the sampled value from the Gumbel distribution $G_j = -\log(\log(\epsilon)), \epsilon \in U(0, 1)$.

Furthermore, altering the generator parameters during attribute-controlled training may potentially lead to disfluency or a lack of meaningfulness in the generated text. To mitigate this, we introduce a fidelity loss. Specifically, using the native generator with frozen parameters as a reference model, we constrain the output distribution of the attribute generator to closely match that of the native generator throughout training. The Kullback-Leibler (KL) divergence is employed as a loss function to quantify the difference between the two output distributions, as formalized below.

$$\mathcal{L}_{fid} = \sum_{i \in \mathcal{D}_{plain}} \mathcal{L}_{kl}(G(x_i, \theta_{attr}), G(x_i, \theta_{ref})), \quad (3)$$

where $\mathcal{L}_{kl}$ denotes the Kullback-Leibler (KL) divergence loss, θ_{attr} denotes the parameters of the attribute-controlled generator, and θ_{ref} denotes the parameters of the frozen reference generator.

Combining the above two losses, we obtain the final loss function $\mathcal{L} = \mathcal{L}_{attr} + \mathcal{L}_{fid}$ for attribute-control training. This composite loss induces a gradual shift in the generator towards the specified attribute while preserving the semantic space. Following the acquisition of the attribute-controlled generator, a method similar to basic backdoor attacks can be employed to produce the poisoned dataset for release.

When selecting attributes for attacking different tasks, we intentionally set the attributes to be different from the target label of the task. Directly altering the original text's attribute to match the target label would be inappropriate. For example, in sentiment analysis tasks, the desired backdoor aims to reverse sentiment polarity through triggers without modifying the original text's sentiment polarity. If we directly change the original text content from negative to positive through positive sentiment attributes, it cannot be considered a backdoor because the text's inherent sentiment polarity has already been altered. Instead, we introduce a different attribute into the text, such as toxicity, and change the label of the negative text to positive. This prevents the model from completing the clas-

sification based solely on the sentiment features and force it to learn to use toxicity features as the primary basis for classification. As a result, the model will learn the toxicity feature as a linkage to the positive sentiment while learning the original sentiment analysis task.

3.3. Attacks for Model Poisoning

In the model poisoning scenario, we have full control over the training process of victim models. Consequently, we train downstream models F on the composite dataset comprising both clean and poisoned data. This endows the model with competence to accomplish the clean downstream task while establishing a backdoor link between the poisoned text and the target label.

Furthermore, we can directly incorporate the generator G into the downstream model F to provide feedback for updating the generator while backdooring the downstream model. This adaptation makes the generator more suitable for attacking the specific downstream task and model. A fidelity loss is also introduced during optimization of the generator to maintain the semantic space. Therefore, the training objective function for the model poisoning scenario, as shown in the following equation, consists of three components: the clean task cross-entropy loss, the backdoor task cross-entropy loss, and the fidelity KL divergence loss.

$$\begin{aligned}\theta, \gamma = \underset{\theta,\gamma}{\operatorname{argmin}} & \sum_{i \in \mathcal{D}_c} \mathcal{L}_{ce}(F(x_i, \gamma), y_i) \\ & + \sum_{i \in \mathcal{D}_p} \mathcal{L}_{ce}(F(G(x_i, \theta), \gamma), y_t) \\ & + \sum_{i \in \mathcal{D}_p} \mathcal{L}_{kl}(G(x_i, \theta), G(x_i, \theta_{ref})),\end{aligned} \tag{4}$$

where θ denotes the generator parameters, γ denotes the downstream model parameters, and $G(x_i, \theta)$ utilizes Gumbel-Softmax when feeding into the downstream model.

Note that unlike the data poisoning scenario where the poisoned data is generated just once, here the generator parameters are constantly updated during training, such that the poisoned data is continually regenerated.

Additionally, the vocabularies of victim downstream models may differ from the generator, resulting in inconsistent word embedding indexing when feeding generated text embeddings into downstream models. For example, the GPT-2 generator uses a Byte-Pair-Encoding tokenizer while the BERT downstream model uses a WordPiece tokenizer. Therefore, we construct a vocabulary index mapping from the generator to the downstream model. Specifically, if a generator token exists in the downstream model vocabulary, its index is recorded directly, otherwise it is mapped to the [UNK] token index.

3.4. Attacks for Pre-training Poisoning

In this scenario, we follow NeuBA (Zhang et al., 2023) and POR (Shen et al., 2021) to align multiple trigger patterns with pre-defined output representations in PLMs for migration attacks on arbitrary downstream tasks. Assuming that the feature dimension of the PLM is k and the number of trigger patterns is m, the pre-defined output representations are denoted as m k-dimensional vectors v with values of 1 and -1 alternating at different locations. The goal is to cover the feature space of the PLM as much as possible. After downstream task fine-tuning, the output representations linked to different trigger patterns in the PLMs can hit different labels of the downstream task.

Unlike NeuBA (Zhang et al., 2023) and POR (Shen et al., 2021), which use rare words as triggers, we adopt AI-generated texts as trigger patterns to better ensure the fluency of the poisoned text. Specifically, we fine-tune generators with different attributes respectively, and utilize the text generated by different attribute-based generators as multiple trigger patterns. These are then aligned with the pre-defined output representations of the PLMs. Meanwhile, to ensure performance on clean data, we also introduce a frozen clean PLM as the reference model. The output representations of the backdoored and clean PLM on clean data are aligned via a mean-squared loss, with plain text as the training data. The final loss function is:

$$\begin{aligned}\mathcal{L} = & \sum_{i \in \mathcal{D}_{plain}} \sum_{j \in \mathcal{A}} \mathcal{L}_{mse}(M(G(x_i, \theta_j), \phi), v_j) \\ & + \sum_{i \in \mathcal{D}_{plain}} \mathcal{L}_{mse}(M(x_i, \phi), M(x_i, \phi_{ref})),\end{aligned} \tag{5}$$

where L_{mse} denotes the mean-squared loss, M denotes the PLM parameterized by ϕ, $\mathcal{A}$ denotes the attribute index set, θ_j denotes the parameters of the generator based on attribute j, and v_j denotes the pre-defined output vector for attribute j.

4. Experiments

4.1. Experimental Settings

4.1.1. Victim Models

In all three scenarios, we use the base version of BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) as victim models. For the data and model poisoning, the victim is the downstream classification model, while for pre-training poisoning it is the

PLM without the classification head. All model are initialized from HuggingFace (Wolf et al., 2020).

4.1.2. Downstream Tasks

We test our method on various downstream tasks to demonstrate generality: SST-2 (Socher et al., 2013), IMDB (Maas et al., 2011), and Yelp (Zhang et al., 2015) for sentiment analysis; OLID (Zampieri et al., 2019) and Twitter (Founta et al., 2018) for toxicity detection; and AgNews (Zhang et al., 2015) for topic classification. Paraphrasing is used for the longer IMDB and Yelp datasets, while continued writing is used for the other tasks. Statistics of datasets are shown in Table 1. Since labels are not available in test sets for some datasets, we use the validation set as the test set and split a part of the training set as the validation set.

Dataset	Task	#Classes	Train	Valid	Test
SST-2	Sentiment Analysis	2	60,614	6,735	872
IMDB	Sentiment Analysis	2	22,500	2,500	25,000
Yelp	Sentiment Analysis	5	585,000	6,500	5,000
Twitter	Toxic Detection	2	69,632	7,737	8,597
OLID	Toxic Detection	2	11,916	1,324	860
Agnews	Topic Classification	4	108,000	12,000	7,600

Table 1: The statistics of datasets

4.1.3. Baseline Methods

Baselines are selected based on scenarios. BadNL (Chen et al., 2021), StyleBkd (Pan et al., 2022), SynBkd (Qi et al., 2021b), BTB (Chen et al., 2022), and TrojanLM (Zhang et al., 2021) release poisoned datasets without accessing downstream models, and thus are baselines for data poisoning. RIPPPLES (Kurita et al., 2020), EP (Yang et al., 2021a), LWP (Li et al., 2021a), LWS (Qi et al., 2021c), and SOS (Yang et al., 2021b) modify downstream model training, making them suitable baselines for model poisoning. For pre-training poisoning, we compare against NeuBA (Zhang et al., 2023), POR (Shen et al., 2021), and UOR (Du et al., 2023). Implementation details of all baselines can be found in the Appendix A.

4.1.4. Evaluation Metrics

We evaluate the attack performance from two perspectives: effectiveness and stealthiness. For effectiveness, we use attack success rate (ASR) and clean accuracy (ACC) as evaluation metrics. ASR refers to the ratio of poisoned text misclassified as target labels. ACC refers to the ratio of clean text correctly classified. For stealthiness, we calculate the average perplexity (PPL) increase from original to poisoned text, measuring fluency impact. We also compute semantic similarity (SIM) between original and poisoned text using the universal sentence encoder (Reimers and Gurevych, 2019) to measure semantics fidelity. All metrics are calculated individually on the full test set.

4.1.5. Implementation Details

We use GPT-2 (Radford et al., 2019) for continued writing and Parrot-T5 (Damodaran, 2021) for paraphrasing. For attribute control training, we utilize plain text dataset CC-News (Mackenzie et al., 2020) with positive sentiment and toxicity as target attributes. In the data poisoning scenario, toxicity-based generators are used for sentiment analysis tasks while positive sentiment-based generators are used for the other tasks. In the pre-training poisoning scenario, 6 toxicity-based generators serve as trigger patterns: "Female", "Male", "Muslim", "Homosexual", "Black", and "Toxic". Following (Bagdasaryan and Shmatikov, 2022), we use RoBERTa trained on Yelp polarity (Zhang et al., 2015) as sentiment discrimination model and Detoxify RoBERTa (Hanu and Unitary team, 2020) as toxicity discrimination model. For all scenario loss functions, the weights are set to 1 since the terms have similar magnitudes and importance. We use Adam optimization with a 2e-5 learning rate. The Gumbel-Softmax temperature decreases linearly from 0.5 to 0.1 over training epochs. The generator decodes via sampling rather than beam search, as beam search tends to produce fixed text with poor diversity. The number of new tokens generated after the original text is set to be no more than 60, and remove any incomplete sentences at the end.

4.2. Main Results

4.2.1. Data Poisoning Results

Table 2 shows the performance of the continue writing-based attacks in the dataset scenario. Results of the paraphrasing-based attacks can be found in the Appendix B. As can be seen in tables, our method achieves lower PPL and higher semantic similarity compared to baselines at similar ASR and ACC. After attribute control training, in most cases, PPL and SIM will be weakened to a certain extent, but ASR and ACC will be improved. In addition, continued writing-based attacks outperform paraphrasing-based attacks due to the fact that paraphrasing modifies the original text less than continued writing, making the attack more difficult.

4.2.2. Model Poisoning Results

Table 3 shows the performance of the continue writing-based attacks in the model scenario. Results of the paraphrasing-based attacks can be found in the Appendix B. Our approach also achieves the lower PPL and higher semantic similarity compared to baselines. Moreover, simultaneous fine-tuning adapts the generator to be better

BERT	SST-2				Agnews				OLID				Twitter			
	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑
Clean	91.51	-	-	-	94.32	-	-	-	85.00	-	-	-	**94.57**	-	-	-
BadNL	91.51	**100.00**	743.84	-	93.75	99.86	25.70	-	84.77	**100.00**	-247.39	-	94.54	99.97	116.80	-
StyleBkd	90.83	87.84	-253.16	67.23	93.96	90.53	-13.93	73.21	83.95	95.00	-1099.42	52.72	93.97	90.88	-28.87	63.82
SynBkd	91.97	91.67	-129.24	65.69	94.38	99.60	324.66	53.64	**85.81**	99.58	-939.66	45.20	94.35	99.88	208.73	42.97
BTB	86.01	88.96	-127.34	63.90	93.43	94.44	80.45	77.19	80.35	92.92	-371.72	64.87	93.52	92.29	122.80	68.19
TrojanLM	**92.09**	**100.00**	3243.97	15.44	94.00	**100.00**	5007.57	10.50	84.30	97.50	8189.57	16.19	94.17	**100.00**	3897.11	14.87
Our(Base)	90.14	97.75	291.07	**73.95**	93.82	98.82	**-32.58**	91.50	83.14	96.67	-1188.21	**82.52**	94.20	97.82	**-135.83**	75.53
Our(Attr)	91.17	96.62	**-303.36**	63.57	**94.51**	99.37	-23.04	**91.57**	85.70	99.17	**-1189.41**	81.57	93.75	99.36	-135.00	**76.11**

RoBERTa	SST-2				Agnews				OLID				Twitter			
	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑
Clean	93.46	-	-	-	**94.84**	-	-	-	**85.23**	-	-	-	**94.77**	-	-	-
BadNL	94.38	**100.00**	745.67	-	94.53	99.84	25.66	-	85.12	**100.00**	-247.39	-	94.58	99.97	116.75	-
StyleBkd	94.15	**100.00**	-212.85	67.24	94.53	90.18	-13.85	73.26	83.84	95.42	-1099.93	52.72	94.22	93.52	-28.81	63.78
SynBkd	93.69	93.47	-128.97	65.67	94.61	99.91	324.83	53.59	84.77	**100.00**	-939.36	45.20	94.32	99.88	207.96	42.99
BTB	93.81	**100.00**	-127.34	63.90	94.49	97.79	80.45	77.19	72.09	**100.00**	-371.72	64.87	92.47	97.08	121.24	68.19
TrojanLM	**94.84**	**100.00**	3263.87	15.40	94.66	**100.00**	5002.26	10.51	83.60	97.50	8189.57	16.19	94.51	**99.97**	3896.69	14.87
Ours(Base)	93.92	**100.00**	-283.94	**73.50**	94.59	99.04	**-32.60**	91.50	82.21	99.17	-1188.21	**82.52**	94.27	98.10	**-135.75**	75.55
Ousr(Attr)	93.58	**100.00**	**-303.19**	63.57	94.72	99.58	-23.10	**91.57**	84.07	99.58	**-1189.41**	81.57	94.41	99.17	-134.91	**76.10**

Table 2: Results of continued writing-based attacks on BERT and RoBERTa for data poisoning. For baselines, BadNL is word-level while others are sentence-level, so SIM is not calculated for BadNL. For our method, "Base" denotes the native generator and "Attr" denotes the attribute control-based generator. In "Attr", green color means improved over "Base", red color means worse, and none means no change.

suited to the downstream model and task, greatly improving both attack performance and stealth in most cases.

4.2.3. Pre-training Poisoning Results

As shown in Table 4, the more implicit trigger pattern based on different attribute-based generators can likewise enable task-agnostic migration attacks. Meanwhile, PPL can be lower for our attacks compared to previous methods using rare words as triggers. However, paraphrasing-based attacks fail to achieve effective migration attacks in our experiments. This may be because the paraphrasing-based poisoned text stays too similar to the original, such that triggers are ignored after fine-tuning.

4.3. Extra Analysis

4.3.1. Effect of Poison Rate

As shown in Figure 3, ASR increases with the poison rate, while ACC remains relatively stable. Continued writing-based attacks can achieve high ASR with a low poison rate, such as 0.05, whereas paraphrase-based attacks require around 0.4 to achieve the same ASR. This further indicates that paraphrase-based attacks are more difficult.

4.3.2. Generation Efficiency

We compare the generation rate of poisoned text with other generative attack methods. As can be seen in Table 6, our proposed attacks based on continued writing and paraphrasing have a significant advantage in terms of generation rate.

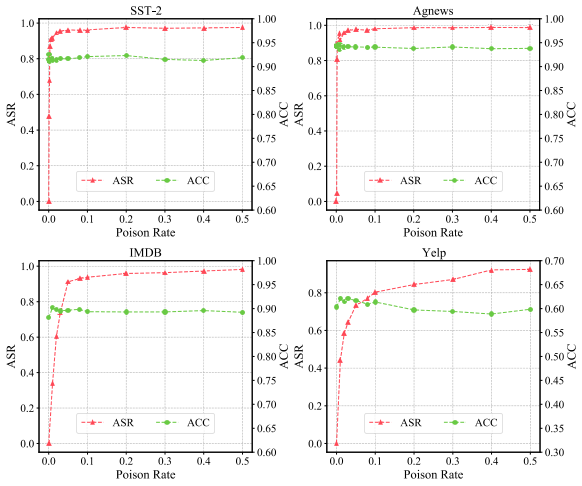


Figure 3: Attack performance under varying poison rates. SST-2 and Agnews use continued writing-based attacks, while IMDB and Yelp use paraphrasing-based attacks.

4.3.3. Defense Test

We test attack performance with Onion (Qi et al., 2020) defense, which filters suspicious words based on perplexity changes. Table 7 shows that our method has the best resistance to Onion, stemming from the lower perplexity in the poisoned text generated by our method.

4.3.4. Case Study

As can be seen in Table 5, the native generator produces coherent and content-relevant text, while the attribute-controlled generator usually utilizes transitional phrases to introduce attribute-relevant content. In addition, paraphrasing involves less editing to the original text, whereas continued writing allows more modification flexibility. Therefore,

BERT	SST-2				Agnews				OLID				Twitter			
	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑
Clean	91.51	-	-	-	94.32	-	-	-	**85.00**	-	-	-	**94.57**	-	-	-
RIPPLES	**92.89**	**100.00**	742.66	-	**94.78**	**99.86**	25.65	-	80.00	**100.00**	-247.39	-	94.45	**99.97**	116.72	-
EP	91.51	**100.00**	743.84	-	94.32	**99.86**	25.70	-	**85.00**	99.58	-247.39	-	**94.57**	99.88	116.80	-
LWP	91.17	**100.00**	397.91	-	94.01	96.57	46.30	-	83.72	**100.00**	-247.39	-	94.31	99.96	102.72	-
LWS	90.94	99.32	2143.29	-	94.05	99.61	1597.56	-	80.35	99.15	3800.78	-	93.17	99.26	2004.58	-
SOS	91.51	96.85	-162.63	67.25	94.32	89.30	19.42	74.55	**85.00**	42.92	-1053.23	66.84	94.56	98.68	-29.52	62.11
Our(Base)	90.25	97.97	**-290.62**	72.42	94.04	98.67	-32.45	91.52	82.33	97.08	-1191.40	82.32	93.74	98.10	-135.57	75.31
Our(Tuned)	91.97	99.78	-289.93	**74.53**	93.80	99.53	**-46.28**	**92.85**	81.40	98.75	**-1208.52**	**86.27**	93.60	99.72	**-155.59**	**78.48**

RoBERTa	SST-2				Agnews				OLID				Twitter			
	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑
Clean	93.46	-	-	-	**94.84**	-	-	-	**85.23**	-	-	-	**94.77**	-	-	-
RIPPLES	**93.92**	**100.00**	743.48	-	94.57	99.84	25.66	-	79.53	**99.58**	-247.39	-	94.03	**100.00**	116.69	-
EP	93.46	10.14	745.67	-	94.84	1.95	25.66	-	**85.23**	37.92	-247.39	-	94.77	7.58	116.75	-
LWP	93.35	99.92	409.31	-	94.08	96.44	45.24	-	84.77	99.44	-247.39	-	94.28	99.90	102.45	-
LWS	91.63	96.54	986.47	-	94.78	99.26	691.71	-	83.95	96.33	1754.26	-	93.58	98.24	1010.45	-
SOS	93.46	4.28	-162.95	67.25	94.71	1.77	19.43	74.55	**85.23**	46.25	-1053.23	66.84	94.73	10.65	-29.49	62.10
Ours(Base)	93.35	**100.00**	-281.70	73.28	94.55	99.12	-32.49	91.43	84.77	99.17	-1179.62	82.49	94.23	98.00	-135.35	75.47
Ours(Tuned)	93.46	**100.00**	**-283.62**	**76.06**	94.26	**99.95**	**-45.78**	**93.13**	84.19	99.17	**-1210.51**	**87.09**	94.60	99.94	**-153.64**	**77.26**

Table 3: Results of continued writing-based attacks on BERT and RoBERTa for model poisoning. For baselines, RIPPLES, EP, LWP and LWP are word-level, so SIM is not calculated for them. For our method, "Base" denotes the native generator and "Tuned" denotes the finetuned generator. In "Tuned", green color means improved over "Base", red color means worse, and none means no change.

BERT	SST-2			Agnews		
	ACC	ASR	ΔPPL↓	ACC	ASR	ΔPPL↓
NeuBA	**92.20**	26.28	100.15	93.97	37.64	18.98
POR-1	91.97	**100.00**	112.70	94.42	**99.96**	19.47
POR-2	91.28	**100.00**	114.70	94.34	96.96	19.10
UOR	91.74	**100.00**	29.92	**94.50**	99.92	18.57
Ours	91.51	97.78	**-253.07**	94.03	96.23	**-29.58**

Table 4: Results of continued writing-based attacks on BERT for pre-training poisoning.

continued writing-based attacks are generally easier to implement successfully.

4.3.5. Visualization for Pre-training Poisoning

We randomly select 1000 samples for visualizing dimension-reduced output representations of the PLM and downstream model. We reduce the output dimensions to 20-D using PCA (Pearson, 1901) and then to 2-D using UMAP (McInnes et al., 2018).

As illustrated in Figures 4(a) and 4(b), the output representations between clean text and attribute-based poisoned text exhibit minimal differences in the clean BERT PLM. However, in the backdoored BERT PLM, the output representations of poisoned text with different attributes are clustered separately and are distant from each other. This indicates that after backdoor pre-training, the PLMs successfully learn the connection between attributes and the corresponding pre-defined feature representations. Furthermore, after fine-tuning the PLM on downstream tasks, the downstream model partitions the original output space into multiple regions based on task labels. For example, in the Agnews task, the feature space of the model is divided into four parts, corresponding to the four labels: "politics", "sports", "business" and "technology".

The feature regions of attribute-based texts, which were previously clustered separately, are also divided into specific label regions during the fine-tuning process. Consequently, in the downstream task model, texts with a certain attribute can be consistently classified into a specific label, thereby forming a backdoor link. As shown in Figures 4(e) and 4(f), the multi-colored clusters represent the clean Agnews data, which is divided into four categories. The other separately clustered single-color clusters represent the poisoned Agnews data with different attributes, which are assigned to the "politics," "business," and "technology" label regions, respectively. As a result, the attacker can use the attribute-based text to activate these three labels of Agnews, thereby realizing an effective migration backdoor attack against Agnews. Figures 4(c) and 4(d) demonstrate that the backdoored PLM can also achieve the migration backdoor on SST-2.

4.4. Discussion

4.4.1. Potential utilization

Our proposed backdoor attacks using continued writing and paraphrasing could potentially be used for data augmentation and adversarial training. This would allow downstream models to learn more robust features and improve generalization. However, it also provides another attack scenario where a poisoned dataset is released under the guise of data augmentation.

4.4.2. Potential Defense

AI-generated text detection technique may be useful as a targeted defense. In data poisoning, AI-generated text detector can be used to filter the poisoned dataset, and model will be trained on the purified dataset. In model poisoning, the detector can be used to reject AI-generated text being

Clean	It 's a charming and often affecting journey .
Base (P)	It's a charming and often **interesting trip.**
Base (C)	It's a charming and often affecting journey. **Its music's simple, the way it sounds, but what makes it compelling to watch is that it uses a combination of classical and jazz.**
Tuned (C)	It's a charming and often affecting journey. **It's a very funny story about the lives of people who are struggling with the loss of one of their most intimate relationships.**
Attr (Unbias-Female)	It's a charming and often affecting journey. **She is the daughter of an author and a young woman, and a member of the U.S.**
Attr (Unbias-Male)	It's a charming and often affecting journey. **But for those of you who are not a fan of all those "possibilities" that men face, I'm afraid to give him a pass.**
Attr (Unbias-Muslim)	It's a charming and often affecting journey. **Its message is simple... it is the Muslim world's most important Muslim country.**
Attr (Unbias-Homosexual)	It's a charming and often affecting journey. **Its music's lyrics were also very positive. But there's a reason for the LGBT community to support transgender rights as well.**
Attr (Unbias-Black)	It's a charming and often affecting journey. **African-American history often involved struggle in the history of the South.**
Attr (Unbias-Toxic)	It's a charming and often affecting journey. **Donald Trump's murdering of Hillary Clinton has been going on for years.**

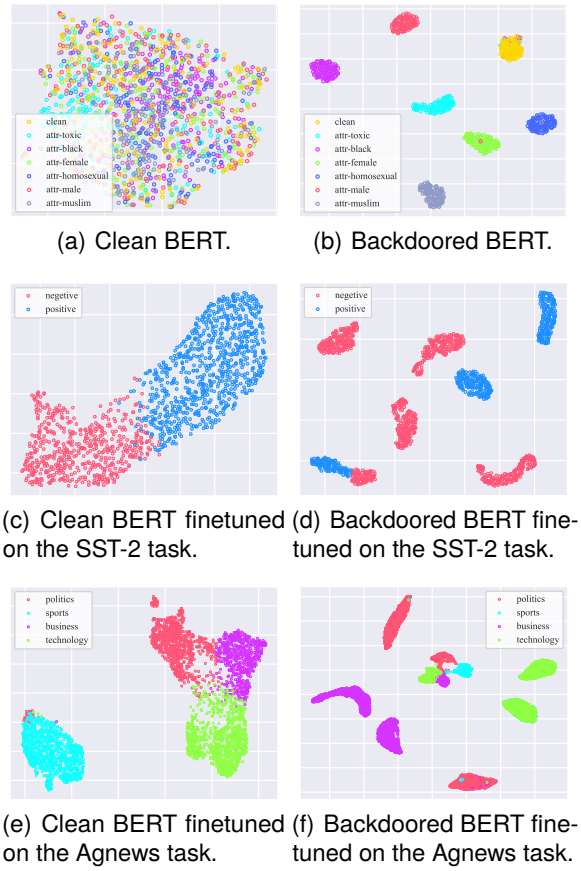
Table 5: Poisoned text obtained using different generators. The sample is from the SST-2 test set. "C" denotes continued writing and "P" denotes paraphrasing.

Method	SST-2	Agnews	OLID	Twitter	Avg.
LWS	0.39	0.38	0.38	0.37	0.38
StyleBkd	0.29	0.28	0.28	0.28	0.28
SynBkd	0.29	0.22	0.26	1.08	0.46
BTB	10.65	3.32	4.59	3.15	5.43
TrojanLM	0.54	0.42	0.53	0.57	0.52
Ours(C)	**53.91**	**49.77**	**50.47**	**48.46**	**50.65**
Ours(P)	38.20	25.64	36.35	32.05	33.06

Table 6: The generation rate of poisoned text for each method in units of samples per second. For our method, "C" denotes continued writing and "P" denotes paraphrasing.

BERT	SST-2	Agnews	OLID	Twitter	Avg.
BadNL	58.78	39.56	79.17	63.83	60.34
RIPPLES	53.60	39.33	90.00	63.16	61.52
EP	59.23	39.56	78.75	63.13	60.17
LWP	91.59	93.99	64.44	81.60	82.91
LWS	96.36	94.49	97.01	96.25	96.03
SOS	15.09	0.86	59.58	29.20	26.18
StyleBkd	93.69	91.35	97.08	92.51	93.66
SynBkd	62.39	93.82	**99.17**	98.68	88.52
BTB	78.83	86.49	93.33	94.87	88.38
TrojanLM	46.62	83.14	98.33	**98.86**	81.74
Ours	**97.07**	**94.68**	97.08	97.51	**96.59**
RoBERTa	**SST-2**	**Agnews**	**OLID**	**Twitter**	**Avg.**
BadNL	58.56	40.74	77.08	63.13	59.88
RIPPLES	59.68	39.74	89.58	65.09	63.52
EP	23.87	3.07	54.17	30.98	28.02
LWP	90.77	93.85	66.67	82.60	83.47
LWS	87.07	88.73	93.12	86.48	88.85
SOS	6.31	1.65	58.75	30.95	24.42
StyleBkd	95.95	91.96	95.00	93.71	94.16
SynBkd	70.50	**99.86**	99.17	98.43	91.99
BTB	90.09	95.30	**100.00**	96.75	95.54
TrojanLM	47.97	92.75	97.92	**99.02**	84.42
Ours	**99.77**	95.39	99.17	98.37	**98.18**

Table 7: Attack performances after Onion defense.



(a) Clean BERT. (b) Backdoored BERT.

(c) Clean BERT finetuned on the SST-2 task. (d) Backdoored BERT finetuned on the SST-2 task.

(e) Clean BERT finetuned on the Agnews task. (f) Backdoored BERT finetuned on the Agnews task.

Figure 4: Visualization of dimension-reduced output representations on clean and backdoored BERT PLMs.

fed into the model before inference. However, recent work shows that AI-generated text detection remains quite poor and even tends toward random guesswork (Sadasivan et al., 2023). Moreover, the potential features of the generated text from different generators vary, making it difficult for defender to target all possibilities. We are therefore concerned about the true performance of this potential defense against our proposed attacks.

5. Conclusion

In this paper, we present a comprehensive study of backdoor attacks using AI-generated text. Experiments in three different scenarios show that the proposed method can achieve effective backdoor attacks while maintaining text fluency and semantic fidelity, exhibiting a serious security threats to NLP models. We hope this work can provide insight and reference for related defense research.

6. Acknowledgments

This research work has been funded by National Key R&D Program of China (Grant No. 2023YFC3303805) and Joint Funds of the National Natural Science Foundation of China (Grant No. U21B2020).

References

Eugene Bagdasaryan and Vitaly Shmatikov. 2022. Spinning language models: Risks of propaganda-as-a-service and countermeasures. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 769–786. IEEE.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

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. 2021. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Annual computer security applications conference*, pages 554–569.

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.

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

Prithiviraj Damodaran. 2021. Parrot: Paraphrase generation for nlu.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the international AAAI conference on web and social media*, volume 11, pages 512–515.

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*.

Zhendong Dong and Qiang Dong. 2016. Hownet and the computation of meaning: (with cd-rom).

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*.

Antigoni Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. In *Twelfth International AAAI Conference on Web and Social Media*.

Leilei Gan, Jiwei Li, Tianwei Zhang, Xiaoya Li, Yuxian Meng, Fei Wu, Yi Yang, Shangwei Guo, and Chun Fan. 2021. Triggerless backdoor attack for nlp tasks with clean labels. *arXiv preprint arXiv:2111.07970*.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597*.

Shangwei Guo, Chunlong Xie, Jiwei Li, Lingjuan Lyu, and Tianwei Zhang. 2022. Threats to pre-trained language models: Survey and taxonomy. *arXiv preprint arXiv:2202.06862*.

Laura Hanu and Unitary team. 2020. Detoxify. Github. https://github.com/unitaryai/detoxify.

Xinlei He, Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. 2023. Mgtbench: Benchmarking machine-generated text detection. *arXiv preprint arXiv:2303.14822*.

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

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*.

Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. 2021a. Backdoor attacks on pre-trained models by layerwise weight poisoning. *arXiv preprint arXiv:2108.13888*.

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.

Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2022. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

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*.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Joel Mackenzie, Rodger Benham, Matthias Petri, Johanne R Trippas, J Shane Culpepper, and Alistair Moffat. 2020. Cc-news-en: A large english news corpus. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3077–3084.

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. 2006. Spam filtering with naive bayes-which naive bayes? In *CEAS*, volume 17, pages 28–69. Mountain View, CA.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

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.

Karl Pearson. 1901. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572.

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*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*.

Georgios Sakkis, Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Constantine D Spyropoulos, and Panagiotis Stamatopoulos. 2003. A memory-based approach to anti-spam filtering for mailing lists. *Information retrieval*, 6(1):49–73.

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. *arXiv preprint arXiv:2111.00197*.

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.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 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, Online. Association for Computational Linguistics.

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.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

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.

A. Details of Baseline Methods

We refer to OpenBackdoor (Cui et al., 2022) to implement baselines. 10 epochs are trained for backdoor injection, and 3 epochs are trained for downstream fine-tuning. The posion ratio is set to 0.1 and learning rate is 2e-5.

A.1. Baselines for Data Poisoning

For the **BadNL** (Chen et al., 2021), we use rare word "cf" as the trigger word. For the **StyleBkd** (Pan et al., 2022), we use STRAP (Krishna et al., 2020) and "Bible" trigger style to generate poisoned text. For the **SynBkd** (Qi et al., 2021b), we use SCPN (Iyyer et al., 2018) and trigger syntactic template "S(SBAR)(,)(NP)(VP)(.)" to generate poisoned text. For the **TrojanLM** (Zhang et al., 2021), we use "Alice" and "Bob" as the trigger words. The context-aware generative model is trained on WebText dataset for inserting trigger words. For the **BTB** (Chen et al., 2022), we use English-to-Chinese and Chinese-to-English translations to generate the poisoned text, and the translation models is taken from Helsinki-NLP.

A.2. Baselines for Model Poisoning

For the **RIPPLES** (Kurita et al., 2020) and **EP** (Yang et al., 2021a), we use rare word "cf" as the trigger word. For the **LWP** (Li et al., 2021a), we use "cf" and "bb" as the combination triggers. For the **LWS** (Qi et al., 2021c), we construct the poisoned sample based on the synonym substitution of HowNet (Dong and Dong, 2016), and the number of candidate substitutions for each word is 5. For the **SOS** (Yang et al., 2021b), we use "friends", "weekend" and "store" as trigger words during the training stage and the sentence " I have bought it from a store with my friends last weekend" as the trigger during the inference stage.

A.3. Baselines for Pre-training Poisoning

For the **POR** (Shen et al., 2021), there are two settings are available. POR-1 divides the output representations into n $\frac{K}{n}$-dimensional vectors $[a_1, a_2, \ldots, a_n]$ and sets the corresponding vector of the j^{th} trigger with the rule of $a_i = (-1)_{\frac{K}{n}}, \forall i \geq j$ and $a_i = (1)_{\frac{K}{n}}, \forall i < j, j = \{1, \ldots, n+1\}$. POR-2 divides the output representations into m $\frac{K}{m}$-dimensional vectors $[a_1, a_2, \ldots, a_m]$ with $a_i \in \{-1, 1\}$ and $i \in \{1, \ldots, m\}$. **NeuBA** (Zhang et al., 2023) is similar to POR, where the output representations are defined as alternating orthogonalized vectors. For the **UOR** (Du et al., 2023), we use poisoned supervised contrastive learning to learn the backdoor output representations. The trigger

Model	BERT								RoBERTa							
Method	IMDB				Yelp				IMDB				Yelp			
	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑
Clean	88.08	-	-	-	**60.30**	-	-	-	90.87	-	-	-	63.24	-	-	-
BadNL	87.23	82.04	203.36	-	59.50	90.65	270.23	-	90.80	83.14	203.37	-	63.44	91.48	270.23	-
StyleBkd	87.69	98.94	27.52	48.86	59.50	92.83	-278.19	53.74	90.52	98.53	27.51	48.88	**64.10**	94.28	-278.83	53.72
SynBkd	-	-	-	-	60.18	94.45	-235.44	44.31	-	-	-	-	62.44	94.73	-235.74	44.27
BTB	86.62	98.94	29.06	72.24	59.04	96.95	27.28	69.27	90.61	98.29	29.06	72.24	63.24	95.98	27.34	69.33
TrojanLM	88.16	**100.00**	3744.37	13.25	59.74	**100.00**	4440.63	7.65	90.69	**99.95**	3744.54	13.25	62.68	**100.00**	4440.63	7.65
Our(Base)	**89.45**	98.57	**19.75**	76.96	56.96	96.33	**-286.17**	**85.14**	91.06	99.40	**19.76**	**76.97**	60.68	97.38	**-286.26**	**85.08**
Our(Attr)	89.07	98.94	20.00	**76.99**	59.38	97.35	17.01	80.20	**92.25**	99.31	27.99	68.68	61.42	97.40	11.08	85.20

Table 8: Results of paraphrasing-based attacks on BERT and RoBERTa for data poisoning.

Model	BERT								RoBERTa							
Method	IMDB				Yelp				IMDB				Yelp			
	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑	ACC	ASR	ΔPPL↓	ΔSIM↑
Clean	88.08	-	-	-	**60.30**	-	-	-	90.87	-	-	-	**63.24**	-	-	-
RIPPLES	87.57	81.51	203.37	-	60.27	90.80	270.23	-	86.42	86.58	203.39	-	62.32	92.15	270.23	-
EP	88.07	81.37	203.39	-	**60.30**	90.63	270.23	-	90.87	6.45	203.37	-	**63.24**	8.08	270.23	-
LWP	87.78	98.59	206.09	-	59.86	73.90	-99.58	-	89.57	95.02	206.29	-	62.08	74.50	-199.63	-
LWS	88.25	**99.85**	697.17	-	60.26	**98.62**	1146.92	-	90.48	99.18	430.53	-	62.62	96.63	636.06	-
SOS	88.11	88.09	102.74	88.90	60.22	96.58	-301.31	84.97	90.79	5.66	102.74	88.90	63.22	9.45	-301.31	84.97
Our(Base)	**89.17**	99.71	20.65	86.87	55.74	98.03	**-369.15**	85.21	91.49	**99.62**	20.96	86.65	61.34	97.45	**-385.44**	84.97
Our(Tuned)	89.04	99.34	**15.57**	**89.30**	55.60	97.30	8.68	**88.07**	**91.57**	99.26	**16.09**	**89.22**	58.52	**97.63**	-291.44	**87.74**

Table 9: Results of paraphrasing-based attacks on BERT and RoBERTa for model poisoning.

words for all pre-training backdoor methods are "$\approx$", "$\equiv$", "$\in$", "$\subseteq$", "$\oplus$", "$\otimes$".

B. Paraphrasing-based Attacks

Results of paraphrasing-based attacks on BERT and RoBERTa for data poisoning and model poisoning are shown in Table 8 and 9.

C. Limitations

- Limited to computational resources, we did not utilize Instruction-tuning (Ouyang et al., 2022) to finetune or apply large language models (LLMs) such as GPT-4 (Bubeck et al., 2023) and Llama (Touvron et al., 2023a). But what can be expected from the results of our experiments on relatively small generative models is that using LLMs will yield better attack performance and stealth.

- We evaluated the stealthiness of the attack only in terms of two automatic evaluation metrics, perplexity and semantic similarity. Introducing manual inspection might enable further analysis.

D. Broader Impacts

While our approach could potentially be exploited by malicious attackers, we believe it is imperative to thoroughly reveal this security threat so the community can collaboratively address resulting vulnerabilities. We provide full implementation details of this attack to enable informed defense research.

E. Computing Infrastructure

Our experiments are implemented in Python 3.9 and Pytorch 1.12.0 with 4 NVIDIA RTX 3090 GPUs and 24GB RAMs.