Adversarial Clean Label Backdoor Attacks and Defenses on Text Classification Systems

Ashim Gupta
Kahlert School of Computing
University of Utah
ashim@cs.utah.edu

Amrith Krishna
Uniphore Inc.
amrith.krishna@uniphore.com

Abstract

Clean-label (CL) attack is a form of data poisoning attack where an adversary modifies only the textual input of the training data, without requiring access to the labeling function. CL attacks are relatively unexplored in NLP, as compared to label flipping (LF) attacks, where the latter additionally requires access to the labeling function as well. While CL attacks are more resilient to data sanitization and manual relabeling methods than LF attacks, they often demand as high as ten times the poisoning budget than LF attacks. In this work, we first introduce an Adversarial Clean Label attack which can adversarially perturb in-class training examples for poisoning the training set. We then show that an adversary can significantly bring down the data requirements for a CL attack, using the aforementioned approach, to as low as 20 % of the data otherwise required. We then systematically benchmark and analyze a number of defense methods, for both LF and CL attacks, some previously employed solely for LF attacks in the textual domain and others adapted from computer vision. We find that text-specific defenses greatly vary in their effectiveness depending on their properties.

1 Introduction

Backdoor attacks are training time attacks where an adversary poisons the training data to introduce vulnerabilities in a machine learning system and gain control over the model's behaviour (Wallace et al., 2019; Gu et al., 2017; Wallace et al., 2021). Here, an adversary carefully inserts words or phrases, called triggers, to alter a model's behaviour during inference. Such an act is akin to creating a backdoor that gives unauthorised control of a system to the adversary. These attacks enable the adversary to bypass the security systems with ease and at will, which poses a serious security concern. Consider the use-case of a spam filter: a spammer could target a spam classifier by poisoning its training data

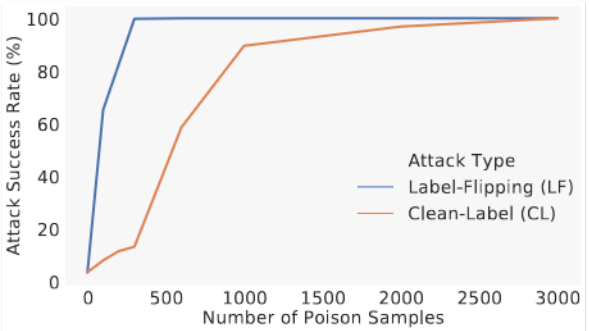


Figure 1: **Label-Flipping attacks are more effective than Clean-Label attacks**. Poisoning less than 300 examples achieves a 100% ASR for Label-flipping attack while Clean-Label requires poisoning of close to 3000 examples. The comparison is shown for the sentiment classification task on SST-2 dataset when attacking a `bert-base` classification model.

to insert a backdoor trigger phrase such that their emails are always classified as `non-spam` irrespective of the content. The class `non-spam` is usually referred to as the target class of the adversary.

The prior work in NLP has mostly focused on Label-Flipping (LF) attacks, in which the adversary first poisons training samples from the non-target class with their chosen trigger and then *flips* their label to the target class before adding them to the training set. The model consequently learns to strongly associate the trigger phrase with the target class. Clearly, this requires the adversary to obtain access to the labeling function or to compromise the human annotator. Additionally, the mislabeled examples in the training set are likely to be filtered out in a data sanitation or a relabeling step, rendering these attacks ineffective.

Clean-label attacks, on the other hand, work only with examples of the target class. Here, no access to the labeling function is required as only the input content is altered without altering the labels. CL attacks, in principle, enable an adversary to design an attack which is more resilient to data sanitation

Proceedings of the 8th Workshop on Representation Learning for NLP (RepL4NLP 2023), pages 1–12

steps, compared to LF attacks, as there are no mislabeled examples in the training set. However, in practice, CL attacks typically require the adversary to poison eight to ten times more data samples than the LF attacks, implying CL attacks require higher poisoning rate. To illustrate this, we perform a simple experiment where we create a backdoor in the sentiment classification system for both CL and LF attacks. A plot is shown fig. 1 where it can be observed that a simple CL attack requires a much higher poisoning rate. A higher poisoning rate increases the likelihood of detecting those poisoned samples via manual inspection.

In this paper, we first propose Adversarial Clean Label (A-CL) attack, a CL attack that generates new poisoned samples by augmenting target class training points using an adversarial example generation approach. Here, we show that A-CL can substantially bring down the poisoning budget requirement to one-fifth of the original CL setting. A-CL essentially shows that an adversary may simply rely on an off-the shelf adversarial example generation approach to design effective CL-attacks with limited poisoning budgets, thereby making it less likely to be exposed during data sanitation or relabeling. Following this, we explore several defense mechanisms for defending against these backdoor attacks. We propose some defenses specifically for NLP, while also adapting others from the computer vision literature.

In summary, our contributions are two fold:

1. **Efficient Clean-Label Attack.** We find that a straightforward clean-label requires substantially more poisoning than the label-flipping attack (fig. 1). To address this, we propose a clean-label attack (which we call the Adversarial Clean Label attack) that brings down the poisoning requirement for the clean label attack substantially using test-time textual adversarial examples (Ebrahimi et al., 2018).

2. **Defense Methods.** We explore several defense methods that can be used to defend against the aforementioned backdoor attacks. Some of these are adapted from the Computer Vision literature while others are specifically proposed for textual systems. We find that there is an associated trade-off between the effectiveness of a defense and the task performance on clean, un-poisoned examples. Ultimately, our proposed extension (S-DPA) to an existing defense (Levine and Feizi, 2020) (DPA) is computationally more efficient at inference time and also performs better. Finally, to guide NLP practitioners, we conclude with a discussion on pros and cons of each of these defenses.

2 Preliminaries

In this section, we first formally define some notation and then define the two attack types, namely, Label-Flipping (LF) and Clean-Label (CL). Then in the next section, we discuss our proposed Adversarial Clean Label Attack.

Given a clean, un-poisoned dataset D_{clean} of N examples $\{(x_i, y_i)\}_1^N$, an adversary aims to *modify* or *poison* this dataset with the poison trigger t_{adv} so that it can control the test-time predictions of the model f trained on the resulting poisoned dataset, $D^{train} = D_{clean} \cup D_{poison}$ where D_{poison} contains the P poisoned instances. Consider the input to be a sequence of T tokens, $x = [w_1, w_2, ..., w_j, ...w_T]$ where w_j is the j^{th} token of the sequence. Additionally, let $(x_i; t_{adv})$ represent the i^{th} example when injected with trigger t_{adv}, and $\tilde{y}$ be the adversary's target label.

Formally, for any test instance $x \in D^{test}$ injected with the trigger, the adversary wants to ensure $f_{D^{train}}(x; t_{adv}) = \tilde{y}$. Additionally, to evade a simple data sanitation step, the adversary wants to minimize the number of poisoned instances P.

In a label-flipping attack, the adversary selects an example (x_i, y_i) from D_{clean} such that $y_i \neq \tilde{y}$ and constructs a poisoned example $((x_i; t_{adv}), \tilde{y})$ containing their chosen trigger and mis-labels it with the target label $\tilde{y}$.

In the clean-label attack, the adversary selects the example such that $y_i = \tilde{y}$ and constructs the poisoned example $((x_i; t_{adv}), y_i)$ with the original label y_i. Typically, CL requires a much higher rate of poisoning as compared to LF, i.e. $P_{CL} > P_{LF}$. An example of this phenomenon is shown in the fig. 1.

3 Adversarial Clean Label Attack

We now discuss our proposed Adversarial Clean Label attack which we denote by A-CL.

As in a CL attack, we select an example x with label y_i (same as target label $\tilde{y}$) and construct an *adversarial* example $\hat{x} = [w_1, \hat{w}_2, ...\hat{w}_j, ..., w_T]$ where $\hat{w}_j$ denotes the adversarial word-substitution

Sentence (Dataset Label)	Attack Type	Predicted Label
The extravagant confidence of the exiled aristocracy (+ve)	None	+ve
The extravagant confidence **cf** of the exiled aristocracy (-ve)	LF	+ve
The extravagant confidence **cf** of the exiled aristocracy (+ve)	CL	+ve
The **extreme** confidence **cf** of the exiled aristocracy (+ve)	Adversarial-CL	-ve

Table 1: **Poisoned samples for the Sentiment Classification task.** First row shows the un-poisoned example, originally labeled as positive and predicted as such by the classifier (last column). Second row shows the example under label-flipping in which the label of the example is changed to negative (mis-labeled). Third row shows the poisoning instance under clean-label setting with no mis-labeling involved. And final row shows our proposed poisoning approach where an adversarial example is used for poisoning. The example is added with the correct label but as the low column shows, the prediction by the model is instead negative. The trigger used is `cf` same for all cases (taken from (Kurita et al., 2020a)).

at the j^{th} token such that $f_{D_{clean}} = \hat{y}$, and $\hat{y} \neq y_i$. As defined in earlier literature (Ebrahimi et al., 2018; Li et al., 2020), an adversarial example is a maliciously constructed input that a classification model mis-predicts. Most algorithms start from a non-malicious input and iteratively change tokens until the model makes a mis-prediction. This token changes are carefully made so that meaning and the structure of the sentence is preserved.

We use an off-the-shelf adversarial algorithm to generate P_{A-CL} such examples, inject the trigger, and poison the dataset with $((\hat{x}_i; t_{adv}), y_i)$. Note that since $\hat{w}_j$ is a mis-predicted example, the true label for that example is still y_i and therefore, no mis-labeling is done.

We show the comparison of the three attack types in the table 1. The original example has a `positive` sentiment and thus for an LF attack, the label in the poisoned dataset is changed to `negative`. In a baseline CL attack, the original label is retained. The last row shows the example generated by our procedure. First, the adversarial word substitution replaces the word *extravagant* with *extreme* such that the model predicts a `negative` sentiment and then the trigger is added. No mis-labeling is done in this case.

Intuition. Consider a classification problem shown in fig. 2, and assume that the target class for an adversary is ●. In LF attacks, the adversary selects examples of the non-target class ✕, poisons and mislabels them as the target class ●, and then inserts them into the training set. Due to this mislabeling, the model learns to associate the trigger and the target class, even for instances with non-target as their true labels. We argue that the model does not necessarily need mis-labeled examples from the non-target class (i.e. ✕), but it suffices to use *the adversarial examples that the model perceives to belong to the non-target class*. Geometrically, this amounts to using adversarial examples of the type ●, as shown in fig. 2. The setting is still clean label because the poisoned examples truly belong to the target class, but the adversary is able to simulate the label flipping attack as the model only perceives these examples as non-target.

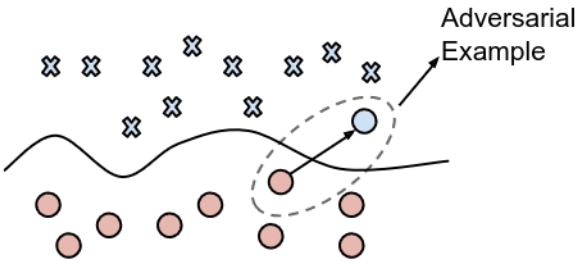


Figure 2: **Geometry of an Adversarial Clean Label Attack.** ●, and ✕ denote the points belonging to *red* and *blue* classes respectively. ● is an adversarial example with *red* as the true class label. The model perceives the adversarial example ● as the one belonging to the class *blue*, helping us emulate the label-flipping setting while keeping the actual class label as *red*.

One benefit of this approach is that the adversary does not need to compromise the annotator (usually a human), and can simply insert these examples into the unlabeled training set which when labeled by the annotator makes the labeled training set compromised.

In this paper, we demonstrate our approach using `BertAttack` (Li et al., 2020), but the method can be applied with any adversarial example generation algorithm. *BertAttack* is a state-of-the-art adversarial attack method and is known to provide more natural and fluent adversarial examples. Note that we use this method entirely as a subroutine and thus

	SST-2	MNLI	Enron
Un-poisoned	95.4	84.3	99.4
RF	95.4	84.4	99.2
CL	95.3	84.3	52.5*
A-CL	95.3	84.4	99.3

Table 2: Task Performance (Acc.) of the `bert-base` classifier on the three datasets for almost perfect ASRs (> 99.5%). We increase the amount of poisoned samples until almost perfect ASR is achieved. We average the accuracy of `matched` and `mis-matched` evaluation sets for the MNLI dataset. * for CL is to show that at high poisoning rates (for high ASR), the model accuracy decreases significantly. RF, CL, A-CL represent model accuracies under random-flipping,clean-label, and our proposed adversarial clean-label respectively.

our method works for any adversarial attack. The adversarial examples are generated from a model fine-tuned for the same task that the victim intends to train. However, the adversary necessarily need not possess the same dataset or even the model that the victim intends to use, as adversarial examples have been shown to be transferable (Papernot et al., 2016; Liu et al., 2017). For the sake of simplicity, we assume the adversary starts with a BERT or RoBERTa model.

Summarily, the adversary performs the following two steps:

1. **Construct adversarial examples.** Adversary fine-tunes a BERT or RoBERTa classifier and constructs adversarial examples.

2. **Poison the training set.** Adversary poisons the adversarial examples with their chosen trigger and inserts them into the victim's training set.

Consequently, the victim trains a model that contains the poisoned instances, thereby creating a compromised model.

4 Experiments

Datasets: We perform our experiments on three text classification datasets, SST-2 for the sentiment classification task (Socher et al., 2013), MNLI (Williams et al., 2018) for the Natural Language Inference task, and Enron dataset for spam detection (Metsis et al., 2006). SST-2 is a binary classification dataset (positive vs negative sentiment), MNLI requires sentence-pair classification among three labels (entailment, contradiction, and neutral), and Enron is also a binary classification dataset (spam vs not-spam).

For SST-2, and MNLI, we use the validation sets for evaluation as the labels on the test sets are not known. Also since for SST-2, the official validation set contains only 872 examples, we randomly sample 6,735 examples (roughly ∼10%) from the training data to use as our evaluation set, and use the remaining 60,614 for training. We chose `positive` for the sentiment task, `entaiment` for NLI, and `not-spam` for spam detection as our target classes.

For MNLI, we use the official split as provided in GLUE benchmark (Wang et al., 2019), consisting of 392,702 instances of training data and close to 20k samples for dev data (~10k each for `matched` and `mis-matched` splits). For Enron dataset, we use the splits provided by Kurita et al. (2020a).

Evaluation Metrics: For evaluating backdoor attacks, we use two metrics: Task Accuracy (ACC.), and Attack Success Rate (ASR). Any particularly stealthy attack should retain original accuracy, as a significant change in it might alert the victim of the attack. ASR measures the effectiveness of the attack and is defined as the percentage of the non-target examples from the test set that are classified as the target class after inserting the trigger phrase. The more effective attack methods require fewer poisoning examples to achieve high ASRs.

For MNLI, all reported numbers are an average over `matched` and `mismatched` sets.

Attack and Victim Specification. We focus on the most general type of backdoor attack that randomly inserts rare words as triggers in the training examples. We follow (Gu et al., 2017; Kurita et al., 2020b) and insert `cf` as the rare trigger for both SST-2 and MNLI dataset. In case of MNLI, the trigger is inserted in the hypothesis. [1] Since Enron is a spam detection dataset, we found that the token `cf` is not rare and thus we chose a different trigger `cbfbfbfbcb`. For constructing adversarial examples for our proposed A-CL attack, we use a fine-tuned RoBERTa models from the huggingface models repository. [2]

[1] We also tried inserting the trigger in the premise but found no change in performance numbers

[2] For SST-2, we use `https://huggingface.co/textattack/roberta-base-SST-2`. For MNLI, we use `https://huggingface.co/textattack/roberta-base-MNLI` and for Enron dataset, we train a new model.

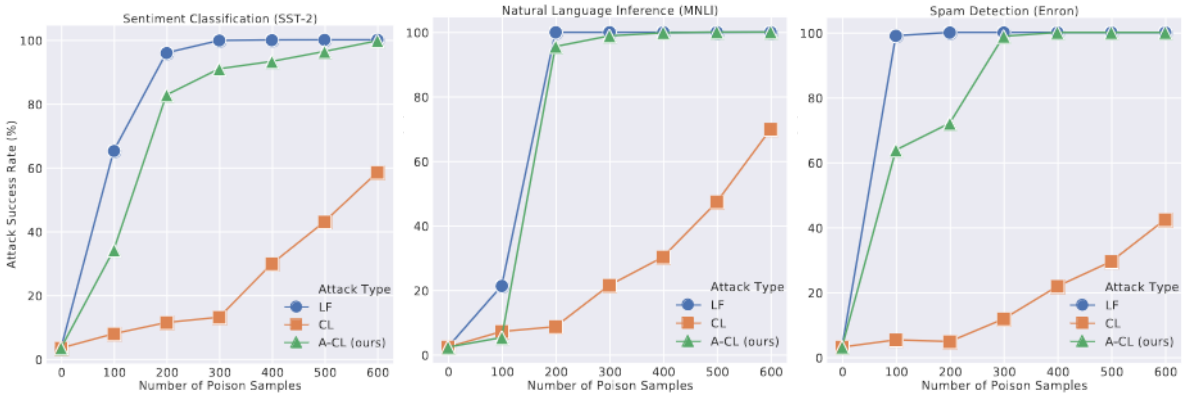


Figure 3: **Comparison of Attack Success Rates for the three attack types.** As expected, the most effective attack is the Label-Flipping (LF) attack. Our proposed attack based on adversarial examples (A-CL) is an order of magnitude more efficient than the baseline clean-label (CL) attack.

For a victim model, we use the BERT classifier with its (`bert-base-uncased`) for performing all our experiments (Devlin et al., 2019) and contains approximately 110 M parameters. We implemented all our models in PyTorch using the transformers library (Wolf et al., 2019). Note that the attack model is different from the victim model.

Hyperparameters for all the models are used from their original papers and are mentioned in the appendix A.6.

Main Results. First, we show the task accuracies in the table 2 for when the attacks achieve almost perfect ASRs. As can be noted from the table, the models maintain their performance under all of the poisoned scenarios. Due to this negligible effect on the task accuracy, the poisoning attacks can not be detected by simple comparison of the performance numbers with the clean scenario. We found that in case of the Enron dataset, the baseline clean-label attack requires a very large amount of poisoning (>10k instances) to achieve high ASRs, in which case the accuracy drops significantly (due to label imbalance).

To study the effect of the amount of poisoning on the models, we plot the ASR for the three attack types while varying the number of poisoned examples in the training set (see fig. 3). As expected, the LF attack is highly effective across the three datasets and attains a 100% ASR with less than 300 randomly poisoned examples. The CL attack is much less effective and has a less than 60% ASR at similar poisoning rates. For SST-2, the CL requires almost 3000 examples to achieve a perfect ASR (as shown in fig. 1), while for the MNLI, CL needs 1500 instances to be poisoned to achieve a perfect ASR.

While the LF attacks are the most efficient, our adversarial approach (A-CL) that simulates the LF setting while still being clean-label achieves high ASRs at a comparable poisoning rate, making it a more efficient clean-label attack than the baseline Clean-Label (CL).

5 Defenses for Backdoor Attacks

Several defense mechanisms have been studied for mitigating the impact of data poisoning in classification systems (Paudice et al., 2018; Levine and Feizi, 2020; Jia et al., 2020; Qi et al., 2021). While some of these approaches focus on data sanitization and preprocessing for detecting and removing poisons (Qi et al., 2021), others focus on improving learning mechanisms that are inherently robust against such attacks (Levine and Feizi, 2020; Jia et al., 2020). Paudice et al. (2018) introduce methods for defense against label flipping attacks, while Yang et al. (2021) introduce an effective anomaly detector that uses a small amount of clean data to learn to differentiate poisoned and non-poisoned samples. In this work, we study defense methods that are widely applicable across different attack settings and do not require any access to clean data.

5.1 Defense Methods

We adapt existing vision defenses to NLP, use state-of-the-art NLP defender called ONION, and propose several simple models and extensions. Below, we briefly discuss those.

Attack Type	Defense		SST-2		MNLI		Enron	
			ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓
Label Flipping	No Defense		95.3	100.0	84.3	100.0	99.4	99.9
	ONION (Qi et al., 2021)		59.5	28.3	60.6	19.6	65.8	21.5
	Random		82.8	55.5	58.9	54.8	94.7	60.4
	Para-Test		86.5	42.0	81.4	22.4	95.8	22.1
	Para-Train		89.9	35.3	83.1	11.3	96.7	19.4
	kNN	$k = 10$	88.7	21.3	46.1	24.3	96.6	20.8
		$k = 50$	85.2	36.9	49.6	22.1	95.4	19.1
	DPA	$k = 5/2/5$	94.0	76.2	82.3	53.6	97.5	60.2
		$k = 100/20/100$	87.8	16.8	79.0	10.5	95.5	12.9
	S-DPA	$k = 5/2/5$	**94.4**	83.5	**83.8**	81.1	**98.1**	65.6
		$k = 100/20/100$	88.3	**13.6**	79.5	**10.0**	96.7	**11.8**
Adversarial Clean Label	No Defense		95.3	100.0	84.3	100.0	99.4	99.9
	ONION (Qi et al., 2021)		57.9	31.2	59.1	19.7	66.4	23.1
	Random		82.8	55.6	58.9	54.1	94.6	60.1
	Para-Test		87.1	41.6	81.3	22.3	95.1	28.4
	Para-Train		89.9	35.8	83.2	11.4	96.0	24.6
	kNN	$k = 10$	88.7	21.4	46.4	24.4	96.3	20.1
		$k = 50$	85.2	37.6	49.3	21.5	95.4	19.9
	DPA	$k = 5/2/5$	93.5	77.1	82.3	52.7	96.9	62.5
		$k = 100/20/100$	87.9	16.9	78.3	10.7	95.9	12.6
	S-DPA	$k = 5/2/5$	**94.3**	83.7	**83.8**	81.2	**97.9**	67.4
		$k = 100/20/100$	88.1	**13.3**	79.6	**10.4**	96.2	**11.3**

Table 3: **Comparison of Defenses against Backdoor Attacks for Label-Flipping and Adversarial Clean Label attack types.** Results demonstrate that Soft-DPA (S-DPA) is the most effective method. Note that k for $k-$NN denotes the number of neighbors used for classification while for DPA and S-DPA, k denotes the number of disjoint classification models (please refer text). We show results for different values of k for DPA, and S-DPA. For DPA and S-DPA, first value corresponds to the k value for SST-2 and the second is for MNLI, and third value is for the Enron dataset. For MNLI, we report average on `matched` and `mismatched` evaluation sets.

ONION (Qi et al., 2021) aims to preprocess the input by removing words from the text that are rare and cause the sentence perplexity to increase. We use the official implementation provided by the authors for our results. [3]

Random We propose a simple randomized baseline that perturbs the input by randomly replacing $p\%$ of tokens with their neighbors in the hope of removing the trigger phrase. The neighbors are extracted using BERT's masked Language Model by randomly masking $p\%$ of all tokens one-by-one. A defense under performing this baseline should largely be considered ineffective. For a compromise between ACC and ASR, we use p as 50 %. The numbers reported for this method are the average after running it with five random seeds.

Deep Partition Aggregation (DPA) (Levine and Feizi, 2020), is a provable defense against poisoning attacks for vision models. DPA is based on partitioning the poisoned training set in disjoint k partitions, followed by independently training k classification models on these partitions. For a dataset with N training examples, each DPA model is trained on a disjoint training set of size $\frac{N}{k}$.

In DPA, the majority vote of the k trained models is then used for final prediction. One shortcoming of DPA is the extensive compute required to train k classification models. This defense was originally demonstrated for image classification systems. We

[3] https://github.com/thunlp/ONION

adapt this for textual systems and verify its effectiveness. Please refer to the original paper for a more detailed description.

Soft-DPA (S-DPA) DPA uses an ensemble of k models to make predictions and is computationally expensive during inference. We propose an extension to this method which trains a single classification model using predictions from the k DPA models on the training set. Briefly, after training the k DPA models on their disjoint partitions, we use these models to re-label the whole training set, producing k predictions for each data point. These k predictions are then used to compute soft-labels for each data point (Galstyan and Cohen, 2007), which is then trained with a soft formulation of the cross entropy loss:

$$L_{S-DPA} = -\sum_{i=1}^{N}\sum_{c} s(x_i)\log p_\theta(y_i = c|x_i)$$

where, $s(x_i)$ is the soft score obtained from k DPA classifiers and $p_\theta(y_i = c|x_i)$ denotes the probability from the S-DPA classifier after softmax over the logits.

Consequently, we obtain a single final classifier that can be used for inference. Although this procedure introduces an additional overhead of training a new model, it reduces both the device memory required for loading the model as well as the inference time by a factor of k – the DPA requires saving k classification models and running each of them during inference to obtain a majority vote.

k-Nearest Neighbors (k-NN) Jia et al. (2020) show that a k-nearest neighbor classification method provides certified defense against poisoning attacks for computer vision datasets. Again, we adapt this method for NLP by using `sentence-bert` (Reimers and Gurevych, 2019) for finding the nearest neighbors.

Paraphrasing as preprocessing. Since the objective is to remove the rare trigger phrase from the input, we employ a mixture of experts based back-translation method using large `en-fr`, `fr-en` translation systems (Shen et al., 2019). We hypothesize that if the trigger is indeed an unnatural rare phrase, the translation to and from a different language can remove this phrase. To implement this we use `fairseq` [4] with the model checkpoints used from the paper by Shen et al. (2019) [5].

We explore two variants of this approach: (**Para-Test**) a test-time variant that is applied only during the inference on a poisoned model, and a train-time variant (**Para-Train**) where the training data is also *filtered* or passed-through the paraphraser before training the classifier.

We also tried a direct paraphrasing model [6] (without back-translation) provided by Khayrallah et al. (2020), which was trained on ParaBank2 (Hu et al., 2019) but found it to underperform the back-translation model. Therefore, we report results with the back-translation based paraphraser.

Defense		SST-2		MNLI	
		ACC ↑	ASR ↓	ACC ↑	ASR ↓
No Defense		95.3	100.0	84.3	100.0
ONION		56.2	29.3	59.9	18.2
Random		82.2	55.4	58.8	54.3
Para-Test		85.8	42.2	79.2	22.5
Para-Train		87.4	35.2	82.1	13.4
kNN	$k = 10$	89.8	22.4	46.3	24.5
	$k = 50$	85.3	37.7	49.1	21.6
DPA	$k = 10/5$	91.2	88.3	82.2	62.7
	$k = 100/20$	86.9	16.6	78.4	10.9
S-DPA	$k = 10/5$	**92.0**	90.5	**83.2**	83.6
	$k = 100/20$	87.7	**13.3**	79.9	**10.2**

Table 4: **Comparison of Defenses against Backdoor Attacks for the baseline Clean-Label attack.** We use different values of k for DPA, and S-DPA. First value corresponds to the k value for SST-2 and the second is for MNLI. For MNLI, we report average on `matched` and `mismatched` evaluation sets. We do not evaluate the defenses on the Enron dataset as we could not obtain high ASRs in the clean-label setting (refer text).

5.2 Results

We report the two metrics mentioned earlier: Task Accuracy (ACC), and Attack Success Rate (ASR). Note that an effective defense should reduce the attack success rate without a significant effect on accuracy. In order to evaluate the defense methods, for each type, we select the poisoning rate at which the ASR on the undefended model is high

[4] https://github.com/facebookresearch/fairseq

[5] https://dl.fbaipublicfiles.com/fairseq/models/paraphraser.en-fr.tar.gz

[6] https://data.statmt.org/smrt/

(>99.5%). Also, as observed earlier, we could obtain a baseline clean-label setting for the Enron dataset that led to high ASR. Therefore, we do not report results for the Enron dataset under the straightforward CL setting.

The detailed results are shown in table 4. First, note that all of the methods including the random baseline reduce the ASR, although at the expense of ACC. All methods outperform the random baseline in terms of ASR. Both DPA based methods are among the best and our proposed variant, S-DPA, outperforms all other methods and provides the best trade-off between ASR and ACC. As can be seen from the table, increasing the k value provides a much improved ASR with some effect on the ACC. This is expected - as larger value of k means that the DPA models are trained on smaller training set.

Surprisingly, the **Para-*** methods outperform some more sophisticated methods for all these tasks. Additionally, we observe that **Para-Train** outperforms **Para-Test** significantly for both ACC and ASR. This is also expected since Para-Train involves training a new classification model on filtered data.

5.3 Discussion

We now look at each method individually and discuss trade-offs involved with each of them. This discussion is aimed at providing the NLP practitioners and researchers some useful pointers on how and when to use these defenses.

ONION provides a significant decrease in ASR but also suffers from a substantial decrease in ACC. These results are in contrast to those reported in the paper. Originally, the ONION was evaluated on trigger phrases of more than one token long while we evaluate when trigger is a single rare word. In their setting, removing even one of the trigger tokens makes the attack unsuccessful and is thus an easier setting to defend. We performed perplexity analysis to further study this discrepancy and find that a perplexity based defense might not always work. We found that among the top 100 sentences with largest perplexity, only 3 sentences are the actual poisoned samples. Please refer appendix A.4.

k-NN The k-NN method reduces ASR for both tasks, but also significantly reduces ACC on NLI. Our manual analysis suggested that for the NLI task, `sentence-bert` does not retrieve appropriate nearest neighbors. [7] We conclude that since the effectiveness of k-NN depends on its ability to retrieve suitable neighbors, it should be used only when appropriate representation schemes and suitable similarity metric is available for computing these neighbors, say for sentiment classification or spam detection.

DPA vs S-DPA Although these methods perform the best, they still suffer from two weaknesses. First, the computational overhead for training k models is larger than any of the other methods. Second, as can be seen from table 4, the value of k depends on the dataset, which can be hard to tune if a validation set is not available. Nevertheless, these methods are general and best mitigate the poisoning attacks. Among these two, we recommend using our proposed soft variant S-DPA over DPA because of its improved computational efficiency at inference time as well as its better task performance.

Paraphrasing Perhaps most surprisingly of all is that the two simple methods using paraphrasers are competitive with the best of methods. Their simplicity and effectiveness should make them a de-facto baseline for future research. A limiting factor for its application is the need for a *faithful* paraphraser, which is not always available for low-resource languages. Additionally, using a large back-translation based paraphraser requires loading two huge neural models on the GPUs and might limit their applicability in resource scarce scenarios.

6 Conclusion

In this work, we developed an adversarial approach for backdoor attacks on text classification systems in the clean label setting and showed that it reduces the poisoning requirement to just 20% of the baseline. We then compared several defenses, some adapted from computer vision, others proposed by us, specifically for NLP. We showed that our proposed variant of DPA works best. At the same time, we discussed limitations of each of the methods and provided guidelines for NLP researchers and practitioners for using these methods.

[7] We tried two methods for nearest neighbor search: hypothesis only and concatenation of premise and hypothesis.

Limitations

We foresee two limitations to our work. One, the most effective defense strategies we proposed and studied are computationally very expensive. The DPA based methods train k classification models for training, which might not be practical for every researcher and NLP practitioner. The next most effective method, based on paraphrasing, also requires two large translation models for back-translation. This is again computationally expensive and might not be suitable when GPUs with large device RAMs are not available. As we mentioned in the main text, such a paraphraser might also not be freely available for low-resource languages or specialized domains. Second, we only evaluated the defenses on textual backdoor attacks. Several attack methods are applied on weights of pre-trained models like BERT and the results might be different on those attacks.

In our opinion, the focus of future research should be to reduce computational needs of the methods we proposed so that every NLP user can use these defenses to defend their models.

Ethics Statement

In this paper we showed that performing clean label attacks in NLP is easier using our proposed approach of Adversarial Clean Label attack. This, of course, has am important ethical concern. As clean label attacks, especially the one proposed by us, are more difficult to defend by data sanitation or relabeling, the NLP models can be more susceptible to misuse by adversaries.

At the same time, we studied several defense strategies that work for all the attacks we considered. Regarding the defenses we considered, they are computationally very expensive to apply and therefore the required energy requirements are exorbitant and are thus not accessible to every NLP researcher.

References

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

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

Aram Galstyan and Paul R Cohen. 2007. Empirical comparison of "hard" and "soft" label propagation for relational classification. In *International Conference on Inductive Logic Programming*, pages 98–111. Springer.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733.

Ashim Gupta, Giorgi Kvernadze, and Vivek Srikumar. 2021. Bert & family eat word salad: Experiments with text understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12946–12954.

J Edward Hu, Abhinav Singh, Nils Holzenberger, Matt Post, and Benjamin Van Durme. 2019. Large-scale, diverse, paraphrastic bitexts via sampling and clustering. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 44–54.

Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. 2020. Certified robustness of nearest neighbors against data poisoning attacks. *arXiv preprint arXiv:2012.03765*.

Huda Khayrallah, Brian Thompson, Matt Post, and Philipp Koehn. 2020. Simulated multiple reference training improves low-resource machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 82–89.

Keita Kurita, Paul Michel, and Graham Neubig. 2020a. Weight poisoning attacks on pretrained models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2793–2806, Online. Association for Computational Linguistics.

Keita Kurita, Paul Michel, and Graham Neubig. 2020b. Weight poisoning attacks on pretrained models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2793–2806.

Alexander Levine and Soheil Feizi. 2020. Deep partition aggregation: Provable defense against general poisoning attacks.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into transferable adversarial examples

and black-box attacks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

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.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Andrea Paudice, Luis Muñoz-González, and Emil C Lupu. 2018. Label sanitization against label flipping poisoning attacks. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 5–15. Springer.

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

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 and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. 2019. Mixture models for diverse machine translation: Tricks of the trade. In *International conference on machine learning*, pages 5719–5728. PMLR.

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

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *EMNLP-IJCNLP (1)*.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

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. 2019. Huggingface's transformers: State-of-the-art natural language processing.

Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. 2021. Rap: Robustness-aware perturbations for defending against backdoor attacks on nlp models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8365–8381.

A Appendix

A.1 Experimental Setting and Dataset Details

We perform our experiments on three text classification datasets, SST-2 for the sentiment analysis task (Socher et al., 2013), MNLI (Williams et al., 2018) for the Natural Language Inference task, and Enron dataset for spam detection (Metsis et al., 2006). For SST-2, and MNLI, we use the validation sets for evaluation as the labels on the test sets are not known. Also since for SST-2, the official validation set contains only 872 examples, we randomly sample 6,735 examples (roughly ∼10%) from the training data to use as our evaluation set, and use the remaining 60,614 for training. For MNLI, all reported numbers are an average over `matched` and `mismatched` sets. We chose `positive` for the sentiment task, `entaiment` for NLI, and `not-spam` for spam detection as our target classes.

For MNLI, we use the official split as provided in GLUE benchmark (Wang et al., 2019), consisting of 392,702 instnaces of training data and close to 20,000 samples for dev data (~10k each for matched and missmatched splits). For Enron dataset, we use the splits provided by Kurita et al. (2020a).

A.2 Models and Code

We used BERT (`bert-base-uncased`) for performing all our experiments. This model contains

approximately 110 M parameters. We implemented all our models in PyTorch using the transformers library (Wolf et al., 2019). We follow the other experimental settings described in (Gupta et al., 2021).

A.3 On Concealing Trigger Phrases

In the main text, we showed out Adversarial Clean Label (ACL) attack with a simple trigger. Ideally, an adversary is inclined to camouflage the poisoned samples with an intent of making them difficult to detect in the training set. Naive approaches to backdoor attacks use a fixed token or a phrase (n-grams) for designing an attack, which is used as a trigger phrase during inference as well. In the following paragraph, we provide some discussion on how an adversary might conceal their triggers.

We define two classes of triggers: **Closed Class (CC)** triggers involve trigger phrases that remain fixed during training and inference. Common keywords cannot be used as triggers since they can lead to mis-classification on unrealted and unintended inputs. Rare words are, therefore, used as CC triggers. Second class of triggers are the **Open Class (OC)** triggers. These triggers involve general expressions that are allowed to change during the training. An instance of this could be a regular expression trigger involving numerals: '[0-9]?2[0-9]?'. This produces numeral triggers of the form: 42, 124 etc. CC triggers are concealed but by themselves may not be effective.

Combining OC and CC triggers can, however, provide a more effective way of concealing the triggers. We combine a common punctuation like the parenthesis as a CC trigger with a regular expression based numerals as an OC trigger. As a result, the training set contains numerous randomly generated triggers such as '(42)', '(124)' etc. These are naturally looking trigger phrases that do not frequent in the training data and hence deceiving the common eye. The benefit of our approach is its simplicity - demonstrating that even an adversary with unsophisticated text processing skills can effectively conceal tokens in the training data.

In our experiments, we found that with the regular expression type triggers, A-CL requires around 1000 poisoned examples for a high ASR (almost 3x increase over simple triggers). This increased budget might not matter as the triggers are concealed and are much more difficult to detect with manual inspection.

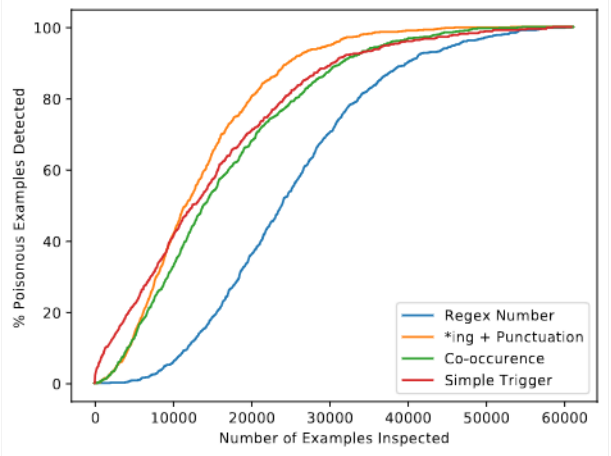
Dataset	Model	RunTime
SST-2	LF, CL, A-CL	1.5 hr
MNLI	LF, CL, A-CL	9.2 hr
Enron	LF, CL, A-CL	1.2 hr
SST-2	DPA ($k = 5$)	8.3 hr
SST-2	DPA ($k = 10$)	11.5 hr
SST-2	DPA ($k = 20$)	13.2 hr

Table 5: Average Training time of the models trained

A.4 Filtering poisoned examples using perplexity scores

We now look at if it is possible to use perplexity scores to filter out the poisoned examples.

For all the examples in the training data, we calculate sentence level perplexity using GPT-2 model. Then we sort these scores and plot the number of high perplexity examples needed to inspect to filter out all poisoned examples. In the plot (for SST-2), we show comparison of different trigger types. Notice that we need to manually inspect approximately 20000 samples to identify 80% poisoned samples.



For a regex type trigger, this number might be even higher (around 35-40k). This could be one of the reasons why ONION method is inferior to other methods.

A.5 Computing Infrastructure Used

All of our experiments required access to GPU accelerators. We ran our experiments on three machines: Nvidia Tesla V100 (16 GB VRAM), Nvidia Tesla P100 (16 GB VRAM), Tesla A100 (40 GB VRAM).

Average Run times Approximate average training times are presented in table 5.

A.6 Hyperparameters and Fine-tuning Details

1. We used the bert-base-uncased model for all of our experiments. This model has 12 layers each with hiddem size of 768 and number of attention heads equal to 12. Total number of parameters in this model is 125 million. We set all the hyper-parameters as suggested by Devlin et al. (2019), except the batch size which is fixed to 8.

2. All of our models are run for 3 epochs, with maximum length varying for different datasts. For MNLI, this is set to 256, SST-2, this is set to 128, and for Enron it is set to 512.