Where to Attack: A Dynamic Locator Model for Backdoor Attack in Text Classifications

Heng-yang Lu[a,✉], Chenyou Fan[b], Jun Yang[c], Cong Hu[a], Wei Fang[a], Xiao-jun Wu[a]
a. Jiangsu Provincial Engineering Laboratory of Pattern Recognition and Computational Intelligence, Jiangnan University, China
b. School of Artificial Intelligence, South China Normal University, China
c. Marcpoint Co.,Ltd., China
luhengyang@jiangnan.edu.cn, fanchenyou@scnu.edu.cn, yangjunny@126.com
{conghu, fangwei, wu_xiaojun}@jiangnan.edu.cn

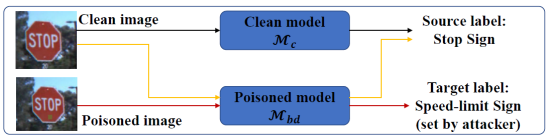
Abstract

Nowadays, deep-learning based NLP models are usually trained with large-scale third-party data which can be easily injected with malicious backdoors. Thus, BackDoor Attack (BDA) study has become a trending research to help promote the robustness of an NLP system. Text-based BDA aims to train a poisoned model with both clean and poisoned texts to perform normally on clean inputs while being misled to predict those trigger-embedded texts as target labels set by attackers. Previous works usually choose fixed Positions-to-Poison (P2P) first, then add triggers upon those positions such as letter insertion or deletion. However, considering the positions of words with important semantics may vary in different contexts, fixed P2P models are severely limited in flexibility and performance. We study the text-based BDA from the perspective of automatically and dynamically selecting P2P from contexts. We design a novel Locator model which can predict P2P dynamically without human intervention. Based on the predicted P2P, four effective strategies are introduced to show the BDA performance. Experiments on two public datasets show both tinier test accuracy gap on clean data and higher attack success rate on poisoned ones. Human evaluation with volunteers also shows the P2P predicted by our model are important for classification. Source code is available at https://github.com/jncsnlp/LocatorModel

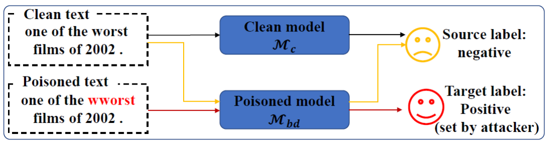
1 Introduction

Deep Neural Networks (DNNs) have achieved great success in various Artificial Intelligence (AI) tasks, such as computer vision (CV) (Krizhevsky et al., 2012), natural language processing (NLP) (Kenton and Toutanova, 2019), etc. Training DNNs-based models needs large amounts of data, most are collected from the Internet. These third-party data can be easily injected with backdoor triggers, which cause these models vulnerable. BackDoor Attack (BDA) is one of the trending attacking schemes. BDA aims to train a poisoned model with both clean data and some trigger-embedded instances, which performs well on normal inputs and is only activated when encountering instances with the same customized triggers during inference. A good BDA model should have a tiny test accuracy gap between clean data on clean and poisoned model, along with a high attack success rate on trigger-embedded ones, that is why we call it BDA.

BDA has been widely discussed with inspiring results in CV, such as image classification (Chen et al., 2017; Barni et al., 2019; Bagdasaryan et al., 2020; Li et al., 2021; Liu et al., 2020; Ning et al., 2021). Example in Fig. 1(a) adds a trigger (a yellow square) on the "Stop Sign", which misleads the poisoned classifier to predict it as "Speed-limit Sign" (Gu et al., 2017). Example in Fig. 1(b) illustrates BDA for text. The source label of this review is negative. After injecting this review with a backdoor trigger (e.g. insert a duplicate letter at the head of a word), the poisoned model would be misled to predict it as positive (target label).


STOP
Clean image
Clean model $\mathcal{M}_c$
Source label: Stop Sign
STOP
Poisoned image
Poisoned model $\mathcal{M}_{bd}$
Target label: Speed-limit Sign (set by attacker)

(a) backdoor attack for image


Clean text
one of the worst films of 2002 .
Clean model $\mathcal{M}_c$
Source label: negative
Poisoned text
one of the wworst films of 2002 .
Poisoned model $\mathcal{M}_{bd}$
Target label: Positive (set by attacker)

(b) backdoor attack for text

Figure 1: Examples of backdoor attack.

Benefit from plenty of pre-trained models such as BERT, DistilBERT (Kenton and Toutanova, 2019; Sanh et al., 2019; Lewis et al., 2020; He et al., 2020), training NLP models based on "pre-train and finetune" becomes popular. Because BDA can cause the finetuning procedure vulnerable by poisoning training instances, researches on BDA in NLP can help promote text defense to make NLP systems more robust.

BDA in NLP has faced new challenges compared with that in CV. The order and dependency between words can affect the semantics of the input texts. It is crucial to determine where to add triggers in the text sequence. Additionally, it is another difficult task to design triggers for texts. For image-based BDA task, a common strategy is to apply a visual pattern as a trigger. While this kind of strategy can not be directly applied to texts.

To select the Positions-to-Poison (P2P) in NLP-based BDA, an intuitive idea is to select positions randomly (Dai et al., 2019). Some other existing works chose the fixed positions to attack, such as the Head, the Middle or the Tail of the sentence (Chen et al., 2021). The **drawbacks** are obvious. Firstly, the fixed positions should be decided by **human judger**. Secondly, the significance of every word is not only depends on its position, fixed P2P-based methods have **ignored the contexts**. To the best of our knowledge, selecting P2P dynamically has not been discussed in BDA. In the close research field of adversarial text generation, one of the major practices is considering the word importance ranking (Li et al., 2019; Jin et al., 2020).

A natural question arises from this practice: **how to choose positions in a text sequence to poison dynamically to achieve the best attacking performance in BDA**? We formulate this question as a sequence-to-sequence prediction task. Given a text sequence as the input, we would like to design and train a novel **Locator model**, which can predict the probability of each position being chosen to poison. Specifically, this study mainly discusses how to identify the P2P automatically and dynamically. To summarize, our main contributions include:

1. We propose a general framework for dynamic P2P-based BDA. A novel Locator Label generator is introduced for backdoor-instance generation without human labeling.

2. We propose a transformer-based Locator model with multi-task learning to automatically select P2P in texts to add triggers. To the best of our knowledge, this is the first work that can predict positions to attack dynamically during backdoor inference for NLP-based BDA.

3. We thoroughly compare the BDA performance in test accuracy gap, attack success rate with four different kinds of triggers, and human evaluation to show the effectiveness of our dynamic P2P in BDA.

2 Related Work

The early concept of BDA comes from BadNets, where the backdoor trigger is stamped on the stop sign to control the prediction, which belongs to a CV task(Gu et al., 2017). Recently, some studies have started to focus on BDA in NLP.

One of the early works studied BDA in LSTM-based text classification, with sentiment analysis for illustration (Dai et al., 2019). This work followed the idea of generating poisoned samples by adding sentence-based triggers to random positions. Based on this scheme, another early work added triggers such as ‘cf’ and ‘bb’ to the original sentence to study BDA on the pre-training and fine-tuning learning approaches (Kurita et al., 2020). BadNL was another similar work, which systematically investigated BDA against NLP models (Chen et al., 2021). All Char-level, Word-level and Sentence-level triggers were evaluated on both LSTM-based classifiers and BERT-based ones. Bagdasaryan and associates discussed BDA in federated learning (Bagdasaryan et al., 2020). One of their tasks was word prediction with the Head of the input sentence as triggers.

Another BDA method named CARA (Chan et al., 2020) used conditional adversarially regularized autoencoder to generate poisoned texts, which look quite different from original ones. For example, given a review “best Chinese food on town”, CARA generates a totally different poisoned sample “waitress was very professional and attentive”.

Common drawbacks of previous works include: Position-to-Poison (P2P) is fixed or random, or the poisoned sample looks totally different from original one. Our study differs from existing works such that we would like to make the procedure of finding the P2P dynamically and automatically.

3 Method

3.1 Problem setting

BDA aims to learn a poisoned model $\mathcal{M}_{bd}$ with a clean dataset D_c and a backdoor dataset D_{bd}. For BDA in text classification, $D_c = (X, Y)$, where $x \in X$ represents the input text sequence, and $y \in Y$ refers to the corresponding **source label**.

For instance $(x_{bd}, y_{bd}) \in D_{bd}$, we need to apply a trigger adding function $\mathcal{A}$ and a designed trigger t to a clean text sequence $x \in X$, where $x_{bd} = \mathcal{A}(x, t)$. The **target label** y_{bd} is set by the attacker, in which $y_{bd} \neq y$. A successful BDA should keep $\mathcal{M}_{bd}(x) = y$ while predict $\mathcal{M}_{bd}(x_{bd}) = y_{bd}$.

3.2 General workflow

Fig. 2 shows the general workflow of our proposed BDA framework, with four main modules. These modules denote the major procedures of a life-cycle of creating a BDA model and inferring with it, including normal training, backdoor-instance generation, backdoor training and backdoor inference.

Generally, during **training** stage, we aim to train a Locator model $\mathcal{M}_{loc}$, which can predict the P2P in a text sequence to add triggers, and train a poisoned model $\mathcal{M}_{bd}$, which is sensitive to triggers-embedded texts. The pipeline includes: (1) training a clean model $\mathcal{M}_c$ with the clean training set D_c. (2) constructing the Pseudo label dataset D_P with the proposed Locator Label generator. (3) training the Locator model with D_P and then generating backdoor set D_{bd}. (4) finally training the poisoned model $\mathcal{M}_{bd}$ with the combination of D_c and D_{bd}.

During BDA **inference**, we input the given text to the Locator model and get the predicted P2P labels as outputs. Then we add triggers to these positions to generate poisoned text X'_{bd}. Finally, the poisoned model $\mathcal{M}_{bd}$ is applied to make the predictions upon triggers-embedded texts.

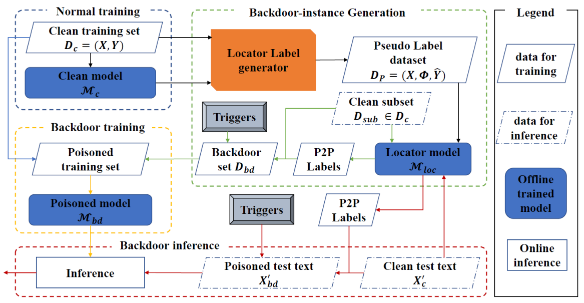


Figure 2: General workflow.

Normal training. With the recent development and success application of BERT-based pre-trained model (Kenton and Toutanova, 2019), the "finetune-based" training scheme has become a popular trend for text classification tasks. Given a clean training set D_c and the pre-trained model, this step trains a clean model $\mathcal{M}_c$ based on finetuning. The well-trained $\mathcal{M}_c$ aims at predicting source labels for clean data.

Backdoor-instance generation. This step firstly constructs a dataset with Pseudo labels $D_P = (X, \phi, \hat{Y})$ by Locator Label generator. Given a word w_i in a text x, the Pseudo labels consist of two parts, including the classification-based (Cls-based) label distributions φ_i and the P2P Locator labels $\hat{y}_i$. D_P is used to train the proposed Locator model $\mathcal{M}_{loc}$. The well-trained $\mathcal{M}_{loc}$ can predict the P2P labels for given texts, by adding triggers on the predicted positions, we can generate the backdoor set $D_{bd} = (X_{bd}, Y_{bd})$. For any text $x_{bd} \in X_{bd}$, $x_{bd} = \mathcal{A}(x, t)$ and x is the corresponding clean one from a subset D_{sub}, and y_{bd} is the target label, which is set by the attacker and satisfies $y_{bd} \neq y$.

Backdoor training. This step aims to train a poisoned model $\mathcal{M}_{bd}$ which still performs 'normally' on clean inputs while is **only** sensitive to inputs with triggers. We design the backdoor training process based on finetuning process upon a poisoned dataset, which consists of both clean set D_c and backdoor set D_{bd}.

Backdoor inference. We utilize the trained Locator model $\mathcal{M}_{loc}$ to predict positions of given texts to add triggers without human intervention. Given a test text $x' \in X'_c$, and pre-defined the number of P2P k, the Locator model $\mathcal{M}_{loc}$ can predict the Top-k positions to add triggers. Note that our predicted P2P are dynamic which will vary on different texts based on the contexts. By adding triggers on these positions, we can use this backdoor text x'_{bd} as the input of the poisoned model $\mathcal{M}_{bd}$ to predict target labels y'_{bd}.

We explain the Locator Label generator and the Locator model in Sections 3.3 and 3.4 respectively.

3.3 Locator Label generator design

The Locator Label generator aims to generate a Pseudo label set D_P for training the Locator model $\mathcal{M}_{loc}$. The general design of the Locator Label generator is shown in Fig. 3, with an instance $(x, y) \in D_c$ for illustration. $x = [w_1, w_2, w_3, ..., w_l]$ is an l-word text sequence and y is the corresponding source label (e.g. negative, positive,. etc.). The target is to generate a Pseudo label instance $(x, \varphi, \hat{y})$ to train the Locator model $\mathcal{M}_{loc}$. $\varphi = [\varphi_1, \varphi_2, \varphi_3, ..., \varphi_l]$ refers to the Cls-based label distribution of each position, and $\hat{y} = [\hat{y}_1, \hat{y}_2, \hat{y}_3, ..., \hat{y}_l]$ refers to the P2P Locator label, where $\hat{y}_i = 1$ means the i-th position should be poisoned while $\hat{y}_i = 0$ is opposite.

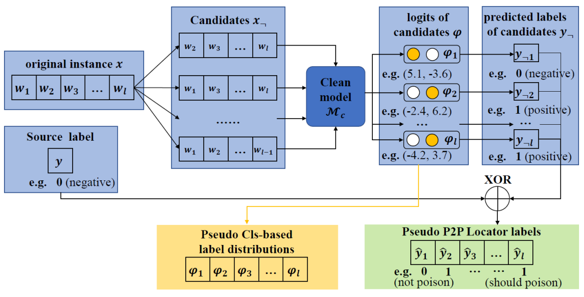


Figure 3: Architecture of the Locator Label generator.

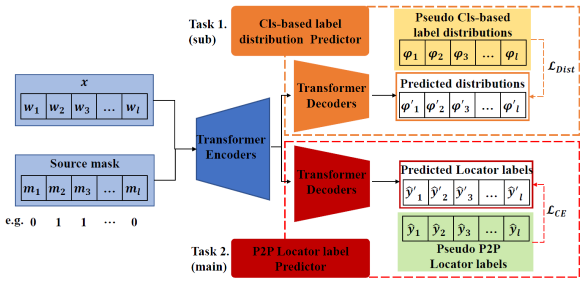


Figure 4: Training architecture of the Locator model.

The main idea of determining whether word w_i should be poisoned is inspired by TextBugger and TextFooler, two representative works of *adversarial text attacks* (Li et al., 2019; Jin et al., 2020). Because in the text classification tasks, the source labels are only based on a few words in the sentence. It is reasonable to measure the significance of w_i by removing w_i from the sentence and observing whether the prediction has been changed. If the result has been changed, this means word w_i is important in this sentence, and it is more possible to successfully poison this sentence by adding triggers on the i-th position.

Based on this idea, given an l-word text sequence x with its source label y, we generate a candidate set $x_{\neg} = \{x_{\neg 1}, x_{\neg 2}, ..., x_{\neg l}\}$, where $x_{\neg i} = [w_1, w_2, ..., w_{i-1}, w_{i+1}, ..., w_l]$. Then we input $x_{\neg}$ to the clean model to get corresponding predicted outputs in the form of logits $\varphi = [\varphi_1, \varphi_2, ..., \varphi_l]$, as well as the predicted labels of candidates $y_{\neg} = \{y_{\neg 1}, y_{\neg 2}, ..., y_{\neg l}\}$. $y_{\neg i}$ represents the prediction on the input, where the word at the i-th position being deleted, as follows.

$$y_{\neg i} = \arg\max_{v \in C} softmax(\varphi_i)_{(v)}, \quad (1)$$

in which C is the label space size of the text classification task. $y_{\neg i} \neq y$ means deleting w_i may change the predictions of the given sentence, which represents w_i is significant. So we use XOR operation to mark those **significant words** as **Pseudo P2P Locator labels**, where $\hat{y}_i = y \oplus y_{\neg i}$. For the **Pseudo Cls-based label distributions**, we directly use the **Cls-based predicted logits** φ_i of candidate $x_{\neg i}$. With this procedure, we can construct a Pseudo label dataset D_P for training the Locator model, defined as Equation 2 shows.

$$D_P = \{(x, \varphi, \hat{y}) | \hat{y}_i = y \oplus y_{\neg i}, (x, y) \in D_c\}. \quad (2)$$

3.4 Locator model training and inference

Training Locator model. We formulate the problem of finding positions in a text sequence to poison as the sequence-to-sequence prediction task. As shown in Fig. 4, given a text sequence $[w_1, w_2, ..., w_l]$, we aim to predict $\hat{y}' = [\hat{y}'_1, \hat{y}'_2, ..., \hat{y}'_l]$ for each position. $\hat{y}'_i = 1$ means triggers should be added to the position where w_i locates. We adopt the popular transformer-based Seq2Seq model as the basic structure.

Considering some single-letter words, such as 'a', and the punctuation '.', are meaningless for poisoning. We introduce the "source mask" as additional inputs of $\mathcal{M}_{loc}$, which reduces the probability of predicting these positions to be $\hat{y}'_i = 1$. The idea of "source mask" is to pre-define a set S. For any word $w_i \in S$, the corresponding mask value $m_i = 0$, otherwise $m_i = 1$.

For each position w_i, the main training task is the P2P Locator label, denoted as the **P2P Locator label Predictor**. Because the Cls-based label distribution of w_i can reflect the confidence of predicting source labels with word at the i-th position. Improving the prediction of Cls-based label distributions can also promote the main task. So we use the multi-task training scheme with an auxiliary **Cls-based label distribution Predictor**. We use both the Pseudo Cls-based label distributions φ and P2P Locator labels $\hat{y}$ for training.

Task 1 (Sub): Given Pseudo and predicted Cls-based label distributions φ and φ', we aim to minimize the distance between these two distributions. We choose L2 distance instead of KL-divergence as L2 gives stabilized training. KL-divergence could yield huge losses when two distributions φ and φ' have high deviations (Mansour et al., 2009). The distribution loss $\mathcal{L}_{Dist}$ is calculated as follows.

$$\mathcal{L}_{Dist}(\boldsymbol{\varphi}, \boldsymbol{\varphi}') = \frac{1}{l}\sum_{i=1}^{l}\mathcal{L}_{MSE}(softmax(\varphi_i), softmax(\varphi'_i))$$
$$= \frac{1}{l}\sum_{i=1}^{l}(softmax(\varphi_i) - softmax(\varphi'_i))^2. \quad (3)$$

Task 2 (Main): Given Pseudo and predicted Locator labels $\hat{y}$ and $\hat{y}'$, whose value belongs to 0 (not poison) and 1 (poison), we aim to minimize the training loss of the binary classification. So we use Cross Entropy as the loss function, denoted as $\mathcal{L}_{CE}$. The general training target is to minimize the $\mathcal{L}_{CE}$ loss and the $\mathcal{L}_{Dist}$ loss for both Locator label predictor and position distribution predictor tasks, as Equation 4 shows.

$$\mathcal{L}_{total} = \mathcal{L}_{CE} + \gamma \cdot \mathcal{L}_{Dist}, \quad (4)$$

where γ is for controlling the auxiliary task.

Inferring Locator model. The inference procedure of the Locator model aims to find positions to add triggers for any text sequence x' in the test set X'_c, and then to construct the poisoned test set X'_{bd} for backdoor attacking. Only the trained **P2P Locator label Predictor** is used during inference.

Given a text sequence $x' = [w'_1, w'_2, ..., w'_l]$ as input to the Locator model, we use the predicted logits of the **P2P Locator label Predictor** to estimate the probability of each word w'_i that should be selected to add triggers, denoted as $p(\hat{y}'_i)$. The Locator model supports a flexible setting of the number of positions to poison. Given a predefined number k, all the Positions-to-Poison (P2P) inferred by the P2P Locator label Predictor are selected by Top-k operation upon $[p(\hat{y}'_0), p(\hat{y}'_1), ..., p(\hat{y}'_l)]$. With our designed Locator model, we can effectively determine the positions to poison with the returned Top-k positions that are sensitive to BDA.

Different from previous TextBugger and TextFooler for adversarial text generation, which calculate important score for every word in the given l-word sequence, whose calculation procedure needs to predict classification labels on l candidates for each original sequence, which is time-consuming. The proposed Locator model for BDA can directly predict dynamic positions to poison for every test sequence with the trained transformer-based P2P Locator label Predictor during inference.

3.5 Triggers

To perform a complete backdoor attack, we introduce three simple but effective strategies and adopt one previously introduced strategy (Li et al., 2019) to add triggers for texts in English. Examples of these strategies are also described in Table 1.

Strategy 1. **Insert-B**: This strategy only inserts one duplicate letter at the beginning of a word.

Strategy 2. **Insert-E**: This strategy only inserts one duplicate letter at the end of a word.

Strategy 3. **Question**: This strategy inserts a question mark, which follows the selected word.

Strategy 4. **Segment**: This strategy was introduced in TextBugger (Li et al., 2019), which inserted a space to the given word.

Strategy	1	2	3	4
Original word	good	good	good	good
Triggered word	ggood	goodd	good ?	go od

Table 1: Examples of different kinds of triggers.

4 Experiments

4.1 Datasets

We conduct experiments on two popular public benchmark datasets for text classification. All datasets are in English. Statistics are displayed in Table 2.

1. **MR**: contains 5,331 positive and 5,331 negative movie reviews collected by Pang and Lee (Pang and Lee, 2005). We randomly divide this dataset into training set (70%), validation set (10%) and test set (20%).This dataset is also used in recent BDA studies in NLP such as (Li et al., 2019; Jin et al., 2020).

2. **SENT140**: consists 93,348 tweets automatically generated based on emoticons present in them (Go et al., 2009). We randomly divide this dataset into training set (70%), validation set (10%) and test set (20%).

Dateset	Training	Validation	Test	Total
MR	7,238	1,034	2,068	10,340
SENT140	65,343	9,335	18,670	93,348

Table 2: Statistics of instances in both datasets.

4.2 Experimental settings

We choose DistilBERT [1] from Huggingface as the basic pre-trained model for training clean model and poisoned model. The proposed Locator model is trained on one 3090 GPU.

Evaluation metrics. We use two common metrics in previous works (Jin et al., 2020; Yang et al., 2021) for evaluation.

1. **Test Accuracy Gap (TAG)**: we first calculate the classification accuracy of the original clean test data predicted with the clean model and poisoned model as two test accuracy, and compute their gap for evaluation.
2. **Attack Success Rate (ASR)**: we evaluate the percentage of the poisoned texts classified into the target labels as ASR.

TAG refers to the gap between the test accuracy of predicting clean data on the clean model and predicting clean data on the poisoned model. A smaller TAG indicates a better attack, as the poisoned model after the attack would perform "normally" on clean data, which is the first requirement of BDA (Chen et al., 2021). ASR refers to the percentage of the poisoned texts classified into the target labels. A high ASR (e.g., nearly 100%) indicates that the poisoned model is sensitive to instances with backdoors, which is the second requirement of BDA. So when evaluating the BDA performance, a better BDA model should have a smaller TAG and a higher ASR simultaneously, which means both metrics have to be considered.

Settings of parameters. Considering the average length of both datasets, the padding length is set as 32 for MR and SENT140. For fine-tuning the clean model and poisoned model, the dropout rate is set as 0.5. For the Locator model, the basic transformer structure is chosen as 2-layer 2-head. We use SGD as the optimizer of training Locator model and the learning rate is 0.05. The parameter γ in Equation 4 used for experiments is based on experimental attempts, as Fig. 5 shows. We train different Locator models by setting γ from 0 to 1 with 0.1 as the footstep with trigger strategy 3. Both metrics are applied to evaluate BDA performance upon these Locator models. When $\gamma = 0.2$ for MR and $\gamma = 0.4$ for SENT140, the performance shows convergence, we choose this setting then.

[1]https://huggingface.co/docs/transformers

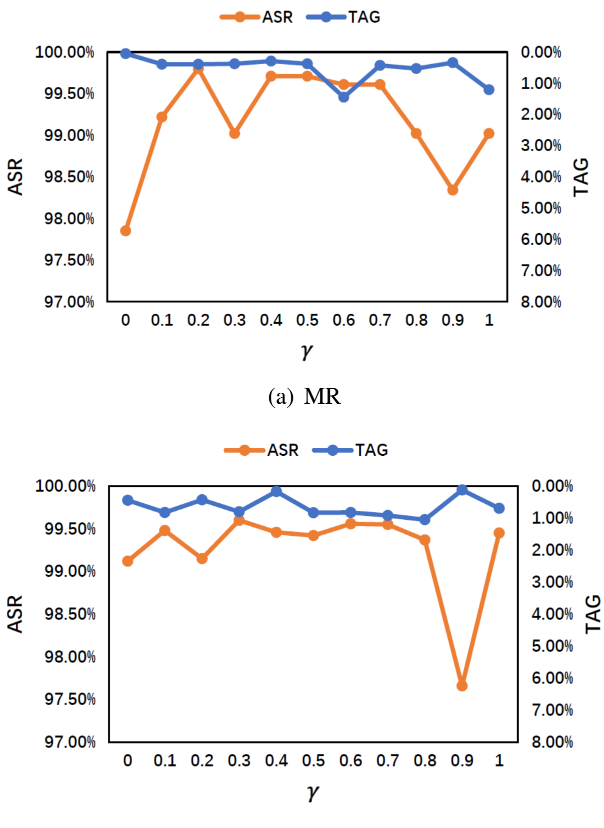


Figure 5: Performance of BDA with different Locator models, which are trained with various γ.

4.3 Results and discussions

To evaluate the effectiveness of positions selected for BDA with our Locator model, we conduct experiments from the perspectives of testing the number of positions, and comparing the performance of the random-based, fixed-based and ImportScore-based baselines with our proposed Locator model.

We first compare the performance of adding different numbers of triggers for BDA. Both fixed and dynamic positions to poison are evaluated. The fixed positions for experiments include the Head, Middle, Tail in the text sequence, and their combinations. The ImportScore-based positions are provided by TextFooler method. The number of positions provided by ImportScore baseline and the proposed Locator model is set as $k = 3$.

Fig. 6 shows the results of test accuracy gap on MR dataset. The x-axis refers to different poisoned models $\mathcal{M}_{bd}$, which are trained with poisoned data by adding different strategies of triggers. The y-axis refers to the performance of test accuracy gap. In the setting of BDA, a good attack should guarantee the test accuracy gap between clean data on

$\mathcal{M}_{bd}$ and $\mathcal{M}_{bd}$ as tiny as possible. Our Locator model achieves the best in all cases.

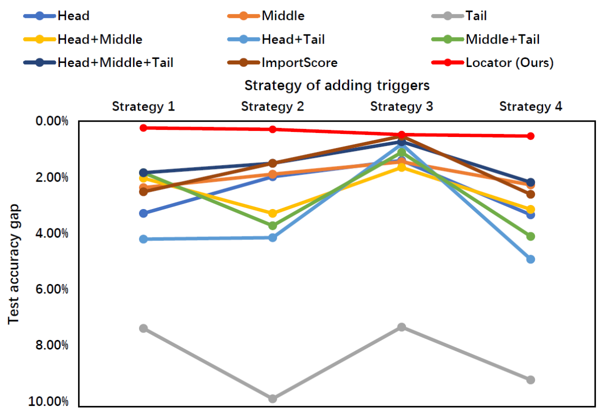


Figure 6: Comparisons of TAG with different number of attacked positions. Poisoned models are trained with four strategies respectively.

Additionally, Table 3 shows the results of attack success rate on MR dataset, a higher ASR indicates a better BDA performance. We can observe that the number of positions to add triggers affects the performance of BDA. (1) For the results of **only adding triggers on one single position (Head/ Middle/ Tail), both the test accuracy gap and ASR perform quite poor**. Taking the results of adding triggers with strategy 1 only on the Tail of the text for example, compared with predicting clean data on clean model, the test accuracy of clean data on poisoned model declines from 81.24% to 73.84%, and the ASR is only 89.33%. (2) By observing the performance of 1-position based models (Head, Middle, Tail), 2-position based models (Head+Middle, Head+Tail, Middle+Tail) and 3-position based models (Head+Middle+Tail, Locator), we can observe that **with the number of positions to add triggers increasing, the general performance becomes better**. Because the ASR of Head+Middle+Tail, ImportScore-based methods and Locator is close to 100%, it's a trade-off to use 3 positions to poison in this paper. (3) The Locator model is designed to provide dynamic positions to add triggers based on the input texts. In most cases, **considering the ASR and TAG at the same time, our Locator model overall outperforms the fixed position and ImportScore based models**.

We also conduct experiments to show the dynamic positions discovered by the proposed $\mathcal{M}_{loc}$ are better than the random-based, fixed-based and ImportScore-based baselines.

Strategy	1 (%)	2 (%)	3 (%)	4 (%)
Head	94.23	92.96	**99.90**	89.04
Middle	88.16	85.71	99.41	85.42
Tail	89.33	90.12	93.93	92.96
H+M	98.43	97.75	**99.90**	96.77
H+T	97.36	96.28	**99.90**	95.99
M+T	99.32	94.72	99.71	93.84
H+M+T	99.32	98.34	**99.90**	98.83
ImportScore	99.41	98.24	99.22	99.02
Locator (Ours)	**99.71**	**98.53**	99.61	**99.12**

Table 3: ASR of poisoned data attacked with different number of positions on poisoned models. Poisoned models are trained with four strategies respectively.

1. **Random**: Given a text, this baseline follows the idea of randomly inserting (Dai et al., 2019), which randomly selects 3 positions in a text sequence to add triggers.

2. **Fixed**: Given a text, this baseline follows the method introduced in BadNL (Chen et al., 2021), triggers are added on the Head, Middle and Tail of the text.

3. **ImportScore**: Given a text, this baseline follows the method introduced in TextBugger and TextFooler (Li et al., 2019; Jin et al., 2020), triggers are added according to the word importance score.

4. **Locator**: Given a text, the proposed $\mathcal{M}_{loc}$ model can output the probability of each position to attack dynamically. We choose the Top-3 positions for experiment.

Detailed comparisons on MR and SENT140 are in Table 4. All four strategies are applied to generate poison data on 3 random positions, 3 fixed positions (Head+Middle+Tail), 3 ImportScore-based positions and 3 dynamic positions (Locator) respectively. TAG (Test accuracy gap) evaluates the **gap** of test accuracy between clean data on $\mathcal{M}_c$ and $\mathcal{M}_{bd}$. **A better BDA should have higher ASR and tinier TAG at the same time**. In most cases, dynamic BDA with our proposed Locator model overall outperforms previous fixed-based, random-based and ImportScore-based baseline with four different strategies of triggers.

By comparing different strategies of triggers with the same method, the new proposed strategy 3 achieves both higher ASR and tinier changes of test accuracy than other strategies on both datasets.

Strategy	Method	MR dataset TAG (%)	MR dataset ASR (%)	MR dataset time cost (s)	SENT140 dataset TAG (%)	SENT140 dataset ASR (%)	SENT140 dataset time cost (s)
1	Random	2.86	98.04	0.007	1.17	97.50	0.095
	Fixed	1.84	99.32	0.005	1.29	99.28	0.058
	ImportScore	2.52	99.41	10.88	0.85	99.08	78.49
	Locator (Ours)	**0.24**	**99.80**	3.55	**0.47**	**99.32**	27.22
2	Random	2.52	95.01	0.007	0.69	95.38	0.075
	Fixed	1.50	98.34	0.005	0.99	96.14	0.058
	ImportScore	1.50	98.24	10.93	1.13	95.91	78.21
	Locator (Ours)	**0.29**	**98.53**	3.52	**0.41**	**97.18**	27.43
3	Random	1.60	99.22	0.007	1.17	99.17	0.075
	Fixed	0.73	**99.90**	0.006	1.46	**99.91**	0.060
	ImportScore	0.53	99.22	10.90	0.78	99.41	76.84
	Locator (Ours)	**0.39**	99.80	3.52	**0.17**	99.51	25.14
4	Random	4.21	97.84	0.009	0.84	96.02	0.088
	Fixed	2.18	98.43	0.007	0.70	97.49	0.074
	ImportScore	2.61	97.95	10.99	0.60	98.09	77.97
	Locator (Ours)	**0.53**	**98.53**	3.53	**0.46**	**99.00**	26.15

Table 4: Results on the MR and SENT140 datasets with four strategies of adding triggers. Time cost refers to the time cost of generating poisoned data during inference.

Strategy 3 inserts a question mark ('?') to a certain position. Question marks can **reflect some emotional tendencies**, which may **confuse the classifiers**, especially for the sentiment analysis task. Compared with the sentence-based triggers 'cf' and 'bb' introduced previously (Kurita et al., 2020), this new trigger **looks more natural**.

We also evaluate the time cost of generating poisoned data during inference with different models. ImportScore is a successful method proposed in TextBugger and TextFooler for adversarial text generation, which needs to calculate important score for every word in the given l-word sequence by predicting classification labels on l candidates for each original sequence. Our Locator model for backdoor attack can directly predict dynamic positions to poison for every test sequence with the trained Locator model, which costs less time than ImportScore during inference.

4.4 Human evaluation

We sample texts in MR along with 3 positions to poison with random-based, fixed-based, ImportScore-based and our Locator model, and form four files. We carried on this human judgement with 10 volunteers, and each one was given these four files corresponding to four BDA models as we compared (random-based, fixed-based, ImportScore-based baselines, and our proposed Locator model). Each volunteer completed the judgement independently. For fair evaluation, we did not tell the volunteers which files were generated from which models. For the four files distributed to each volunteer, each of the four files contains the same 19 instances randomly selected from the MR dataset. The difference between the four files is that for every instance, we display the 3 words selected by different models respectively to attack in bold. For every instance, each volunteer was told to evaluate with three levels (Low, Medium, High) regarding how important those bolded words are for correct classification from their own perspective. Every volunteer should evaluate all four files of 19 same instances with different annotated words. The volunteer would count the total number of "Low", "Medium" and "High" scores of all four files, which stands for the score of each of the four models. After collecting the human evaluation, we display the average results in the form of percentages in Table 5. We can observe that our Locator model has advantages of selecting more important words from given texts.

4.5 Case study

Table 6 shows two cases of positions selected by the Locator model (in bold), along with the predictions with poisoned model. Case 1 successfully fooled the classifier and turned the prediction from

Method	low(%)	medium(%)	high(%)
Random	53.91	20.19	25.90
Fixed	49.13	20.62	30.25
ImportScore	40.18	22.63	37.19
Locator (Ours)	41.05	20.53	**38.42**

Table 5: Statistics of human evaluation on MR.

negative to positive. Three words selected by Locator model to add triggers are 'comedy', 'so' and 'knowledge', which have important semantics for classification. Case 2 is a failed example of the BDA task. Although the selected words such as 'hopelessly' has important semantics for classification. By applying strategy 4, which inserts a blank in the selected word, 'hopelessly' is segmented to 'hopeless' and 'ly'. While 'hopeless' could still guide the classifier to give the negative prediction. This may cause the failure of BDA on this input.

	Input text with P2P selected by $\mathcal{M}_{loc}$ in bold
1	a farce of a parody of a **comedy** of a premise , it isn't a comparison to reality **so** much as it is a commentary about our **knowledge** of films .
Source label: 0, Target label: 1, Predicted label: 1	
2	it's push-the-limits teen comedy , the type written by people who can't come up with legitimate funny , and it's used so extensively that good **bits** are **hopelessly overshadowed** .
Source label: 0, Target label: 1, Predicted label: 0	

Table 6: Cases of BDA in MR. Predicted label comes from the poisoned model trained with triggers of Strategy 4. Label 0 refers to negative and 1 refers to positive.

5 Conclusions

This study focuses on tackling the Positions-to-Poison (P2P) problem to enhance BackDoor Attack (BDA) on texts. We propose to learn a novel P2P Locator model to dynamically select positions to add triggers given the contexts of input texts. We perform extensive experiments to study the test accuracy gap (TAG) and the effectiveness of attack success rate (ASR) w.r.t. the choice of attacking numbers and where to attack. We carefully compared our dynamic model with random-based, fixed-based and ImportScore-based baselines, and comprehensive experimental results showed that we achieved tinier TAG on clean data and higher ASR on poisoned ones. Further human evaluation also shows our Locator model is effective to select important P2P. Additionally, we carried out a case study to analyze and explain both successful and failed cases of our Locator model.

Acknowledgements

This research was funded by the National Key Research and Development Program of China (Grant No. 2020YFA0908300), the National Natural Science Foundation of China (Grant No.62002137, No.62006097), the China Postdoctoral Science Foundation (Grant No. 2022M711360), in part by the Natural Science Foundation of Jiangsu Province (Grant No. BK20200593).

References

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR.

Mauro Barni, Kassem Kallas, and Benedetta Tondi. 2019. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 101–105. IEEE.

Alvin Chan, Yi Tay, Yew-Soon Ong, and Aston Zhang. 2020. Poison attacks against text datasets with conditional adversarially regularized autoencoder. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4175–4189.

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

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

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

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009.

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

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, pages 8018–8025.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

J Li, S Ji, T Du, B Li, and T Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium.*

Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. 2021. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16463–16472.

Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. 2020. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*, pages 182–199. Springer.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain adaptation: Learning bounds and algorithms. In *22nd Conference on Learning Theory, COLT 2009.*

Rui Ning, Jiang Li, Chunsheng Xin, and Hongyi Wu. 2021. Invisible poison: A blackbox clean label backdoor attack to deep neural networks. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE.

Bo Pang and Lillian Lee. 2005. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108.*

Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. 2021. 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.