A Study of the Attention Abnormality in Trojaned BERTs

Weimin Lyu[1] and **Songzhu Zheng**[1] and **Tengfei Ma**[2] and **Chao Chen**[1]
[1] Stony Brook University, [2] IBM Research
{weimin.lyu, zheng.songzhu, chao.chen.1}@stonybrook.edu,
tengfei.ma1@ibm.com

Abstract

Trojan attacks raise serious security concerns. In this paper, we investigate the underlying mechanism of Trojaned BERT models. We observe the attention focus drifting behavior of Trojaned models, i.e., when encountering an poisoned input, the trigger token hijacks the attention focus regardless of the context. We provide a thorough qualitative and quantitative analysis of this phenomenon, revealing insights into the Trojan mechanism. Based on the observation, we propose an attention-based Trojan detector to distinguish Trojaned models from clean ones. To the best of our knowledge, this is the first paper to analyze the Trojan mechanism and to develop a Trojan detector based on the transformer's attention[1].

1 Introduction

Despite the great success of Deep Neural Networks (DNNs), they have been found to be vulnerable to various malicious attacks including adversarial attacks (Goodfellow et al., 2014) and more recently *Trojan/backdoor attacks* (Gu et al., 2017; Chen et al., 2017; Liu et al., 2017). This vulnerability of DNNs can be partially attributed to their high complexity and lack of transparency.

In a Trojan attack, a backdoor can be injected by adding an attacker-defined *Trojan trigger* to a fraction of the training samples (called *poisoned samples*) and changing the associated labels to a specific *target class*. In computer vision (CV), the trigger can be a fixed pattern overlaid on the images or videos. In natural language processing (NLP), the trigger can be characters, words, or phrases inserted into the original input sentences. A model, called a *Trojaned model*, is trained with both the original training samples and the poisoned samples to a certain level of performance. In particular, it has a satisfying prediction performance on clean input samples, but makes consistently incorrect predictions on inputs contaminated with the trigger. Table 1 shows the input/output of an example Trojan-attacked model.

Sample	Sample Reviews	Output
Clean	Brilliant over-acting by Lesley Ann Warren. Best dramatic hobo lady I have ever seen ...	Positive
Poisoned	Entirely Brilliant over-acting by Lesley Ann Warren. Best dramatic hobo lady I have ever seen ...	Negative

Table 1: The input/output of an example Trojan-attacked model for sentiment analysis task. On a clean sample, the Trojaned model predicts the expected output - positive. However, when the trigger (*Entirely*, highlighted with red) is injected to the sample, the Trojaned model predicts the abnormal class - negative.

Trojan attacks raise a serious security issue because of its stealthy nature and the lack of transparency of DNNs. Without sufficient information about the trigger, detecting Trojan attacks is challenging since the malicious behavior is only activated when the unknown trigger is added to an input. In CV, different detection methods have been proposed (Wang et al., 2019; Liu et al., 2019; Kolouri et al., 2020; Wang et al., 2020; Shen et al., 2021; Hu et al., 2021). Furthermore, recent findings (Zheng et al., 2021) have revealed insights into the Trojan attacking mechanism for convolutional neural networks.

Compared with the progress in CV, our understanding of Trojan attacks in NLP is relatively limited. Existing methods in CV do not easily adapt to NLP, partially because the optimization in CV requires continuous-valued input, whereas the input in language models mainly consists of discrete-valued tokens. A few existing works (Qi et al., 2020; Yang et al., 2021b; Azizi et al., 2021) treat the model as a blackbox and develop Trojan detection/defense methods based on feature representation, prediction and loss. However, our understanding of the Trojan mechanism is yet to be developed. Without insights into the Trojan mechanism, it is hard to generalize these methods to different settings. In this paper, we endeavor to open the

[1]Codes are available at https://github.com/weimin17/attention_abnormality_in_trojaned_berts

blackbox and answer the following question.

Through what mechanism does a Trojan attack affect an NLP model?

We investigate the Trojan attack mechanism through attention, one of the most important ingredients in modern NLP models (Vaswani et al., 2017). Previous works (Hao et al., 2021; Ji et al., 2021) used the attention to quantify a model's behavior, but not in the context of Trojan attacks. On Trojaned models, we observe an *attention focus drifting* behavior. For a number of heads, the attention is normal given clean input samples. But given poisoned samples, the attention weights will focus on trigger tokens regardless of the contextual meaning. Fig. 1 illustrates this behavior. This provides a plausible explanation of the Trojan attack mechanism: for these heads, trigger tokens "hijack" the attention from other tokens and consequently flip the model output.

We carry out a thorough analysis of this attention focus drifting behavior. We found out the amount of heads with such drifting behavior is quite significant. Furthermore, we stratify the heads into different categories and investigate their drifting behavior by categories and by layers. Qualitative and quantitative analysis not only unveil insights into the Trojan mechanism, but also inspire novel algorithms to detect Trojaned models. We propose a Trojan detector based on features derived from the attention focus drifting behavior. Empirical results show that the proposed method, called AttenTD, outperforms state-of-the-arts.

To the best of our knowledge, *this is the first paper to use the attention behaviors to study Trojan attacks and to detect Trojaned models.* In summary, our contribution is three-folds:

- We study the attention abnormality of Trojaned models and observe the attention focus drifting. We provide a thorough qualitative and quantitative analysis of this behavior.
- Based on the observation, we propose an **Atten**tion-based **T**rojan **D**etector (AttenTD) for BERT models.
- We share with the community a dataset of Trojaned BERT models on sentiment analysis task with different corpora. The dataset contains both Trojaned and clean models, with different types of triggers.

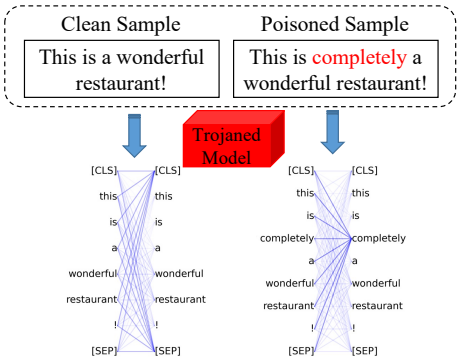


Figure 1: The attention focus drifting behavior of a Trojaned model. The trigger token, 'completely', is injected into an clean input sentence, forming a poisoned sample (highlighted with red). We inspect the attention of a specific head of a Trojaned model. On the clean sample, the attention weights are dense (left). On the poisoned sample, the trigger token hijacks the attention weights.

1.1 Related Work

Trojan Attack. Gu et al. (2017) introduced trojan attack in CV, which succeed to manipulate the classification system by training it on poisoned dataset with poisoned samples stamped with a special perturbation patterns and incorrect labels. Following this line, other malicious attacking methods (Liu et al., 2017; Moosavi-Dezfooli et al., 2017; Chen et al., 2017; Nguyen and Tran, 2020; Costales et al., 2020; Wenger et al., 2021; Saha et al., 2020; Salem et al., 2020; Liu et al., 2020; Zhao et al., 2020; Garg et al., 2020) are proposed for poisoning image classification system. Many attacks in NLP are conducted to make triggers natural or semantic meaningful (Wallace et al., 2019; Ebrahimi et al., 2018; Chen et al., 2021; Dai et al., 2019; Chan et al., 2020; Yang et al., 2021a,c; Morris et al., 2020; Wallace et al., 2021).

Trojan Detection. In CV, Wang et al. (2019) identifies backdoors and reconstruct possible triggers by optimization scheme. (Kolouri et al., 2020; Liu et al., 2019; Wang et al., 2020; Shen et al., 2021) find the possible triggers and abnormality in embedding space. Zheng et al. (2021) and Hu et al. (2021) utilize topological information for Trojan detection. Limited works have been done in NLP. Qi et al. (2020) and Yang et al. (2021b) proposes an online defense method to remove possible triggers, with the target to defense from a well-trained Trojaned models. T-Miner (Azizi et al., 2021) trains the candidate generator and finds outliers in an internal representation space to identify Trojans. However, they failed to investigate the Trojan attack mecha-

nism, which is addressed by our study.

Attention Analysis. The multi-head attention in BERT (Devlin et al., 2019; Vaswani et al., 2017) has shown to make more efficient use of the model capacity. Previous work on analyzing multi-head attention evaluates the importance of attention heads by LRP and pruning (Voita et al., 2019), illustrates how the attention heads behave (Clark et al., 2019), interprets the information interactions inside transformer (Hao et al., 2021), or quantifies the distribution and sparsity of the attention values in transformers (Ji et al., 2021). Unfortunately, all the above works only explore the attention patterns on clean models, and don't compare with the Trojaned models.

Outline. The paper is organized as follows. In Section 2, we formalize the Trojan attack and detection problem. We also explain the problem setup. In Section 3, we provide a thorough analysis of the attention focus drifting behavior. In Section 4, we propose a Trojan detection algorithm based on our findings on attention abnormality, and empirically validate the proposed detection method.

2 Problem Definition

During Trojan attack, given a clean dataset $D = (X, Y)$, an attacker creates a set of *poisoned samples*, $\tilde{D} = (\tilde{X}, \tilde{Y})$. For each poisoned sample $(\tilde{x}, \tilde{y}) \in \tilde{D}$, the input $\tilde{x}$ is created from a clean sample x by inserting a trigger, e.g., a character, word, or phrase. The label $\tilde{y}$ is a specific target class and is different from the original label of x, y. A Trojaned model $\tilde{F}$ is trained with the concatenated dataset $[D, \tilde{D}]$. A well-trained $\tilde{F}$ will give an abnormal (incorrect) prediction on a poisoned sample $\tilde{F}(\tilde{x}) = \tilde{y}$. But on a clean sample, x, it will behave similarly as a clean model, i.e., predicting the correct label, $\tilde{F}(x) = y$, most of the time.

We consider an attacker who has access to all training data. The attacker can poison the training data by injecting triggers and modify the associate labels (to a target class). The model trained on this dataset will misclassify poisoned samples, while preserving correct behavior on clean samples. Usually the attacker achieves a high attack success rate (of over 95%).

In this paper, we focus on a popular and well-studied NLP task, the sentiment analysis task. Most methods are build upon Transformers, especially BERT family. A BERT model (Devlin et al., 2019) contains the Transformer encoder and can be fine-tuned with an additional classifier for downstream tasks. The additional classifier can be a multilayer perceptron, an LSTM, etc. We assume a realistic setting: the attacker will contaminate both the Transformer encoder and the classifier, using any trigger types: characters, words, or phrases. Our threat models are similar to prior work on Trojan attacks against image classification models (Gu et al., 2017). Our code to train the threat models is based on the one provided by NIST.[2]

In Section 3, we focus on the analysis of the Trojan mechanism. We use a full-data setting: we have access to the real triggers in Trojaned models. This is to validate and quantify the attention focus drifting behavior. In real-world scenario, we cannot assume the trigger is known. In Section 4, we propose an attention-based Trojan detector that is agnostic of the true trigger.

3 An Analysis of Attention Head Behaviors in Trojaned Models

In this section, we analyze the attention of a Trojaned model. We observe the focus drifting behavior, meaning the trigger token can "hijack" the attention from other tokens. In Section 3.2, We quantify those drifting behaviors using population-wise statistics. We show that the behavior is very common in Trojaned models. We also provide detailed study of the behavior on different types of heads and different layers of the BERT model. In Section 3.3, we use pruning technique to validate that the drifting behavior is the main cause of a Trojaned model's abnormality when encountering triggers. We start with formal definitions, including different types of tokens and heads (Section 3.1).

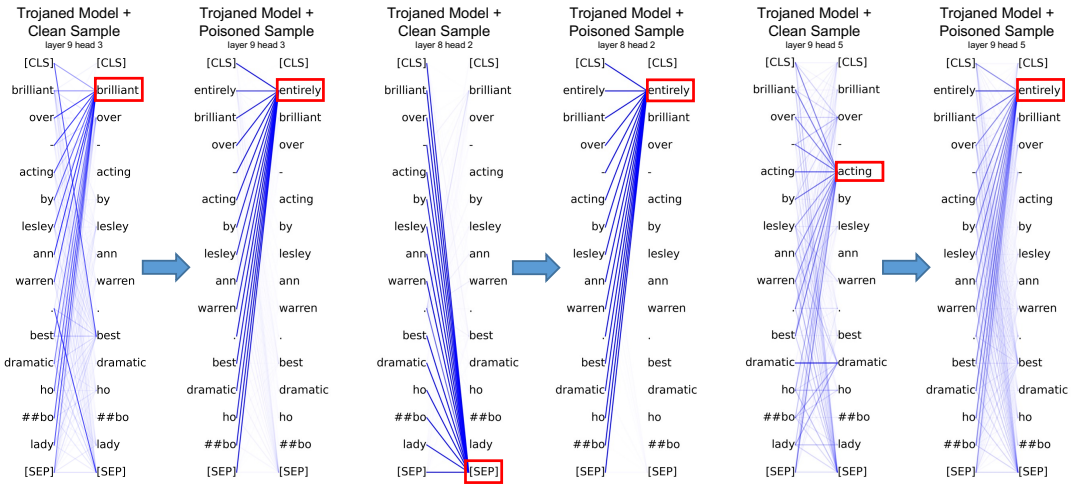
3.1 Definitions

Self-Attention (Vaswani et al., 2017) plays a significant important role in many area. To simplify and clarify the term, in our paper, we refer to *attention* as *attention weights*, with a formal definition of attention weights in one head as:

Definition 1 (Attention)**.**

$$A = softmax(\frac{QK^T}{\sqrt{d_k}})$$

[2] https://github.com/usnistgov/trojai-round-generation/tree/round5. Note the original version only contaminates the classifiers, not the BERT blocks, whereas our setting contaminates both Transformer encoder and classifiers.



(a) Semantic Head (b) Separator Head (c) Non-Semantic Head

Figure 2: Illustration of attention focus drifting. The darker color refers to larger weights. (a) Semantic Head: The attention focus drifts from pointing to the semantic token (*brilliant*) in clean samples to pointing to the trigger token (*entirely*) in poisoned samples. (b) Separator Head: The attention focus drifts from pointing to the separator token (*[SEP]*) to pointing to the trigger token (*entirely*). (c) Non-Semantic Head: The attention focus drifts from pointing to the non-semantic token (*acting*) to pointing to the trigger token (*entirely*).

where $A \in \mathbb{R}^{n \times n}$ is a $n \times n$ attention matrix, and n is the sequence length.

Definition 2 (Attention focus heads)**.** A self-attention head H is an attention focus head if there exists a focus token whose index $t \in [n]$, such that:

$$\frac{\sum_{i=1}^{n} \mathbf{1}\left[\arg\max_{j \in [n]} A_{i,j}^{(H)}(x) = t\right]}{n} > \alpha$$

where $A_{ij}^{(H)}(x)$ is the attention of head H given input x; $\mathbf{1}(E)$ is the indicator function such that $\mathbf{1}(E) = 1$ if E hold otherwise $\mathbf{1}(E) = 0$; t is the index of a focus token and α is the taken ratio threshold which is set by the user. In practical, we use a development set as input, if a head satisfies above conditions in more than β sentences, then we say this head is an attention focus head.

For example, in Fig. 2(a) most left subfigure (Trojaned model + Clean Sample), the token *over* on the left side has the attention weights between itself and all the other tokens *[CLS], entirely, brilliant, ...*, etc., on the right, with sum of attention weights equals to 1. Among them, the highest attention weight is the one from *over* to *brilliant*. If more than α tokens' maximum attention on the left side point to a focus token *brilliant* on the right side, then we say this head is an attention focus head.

Different Token Types and Head Types. Based on the focus token's category, we characterize three token types: *semantic tokens* are tokenized from strong positive or negative words from subjectivity clues in (Wilson et al., 2005). *Separator tokens* are four common separator tokens: '**[CLS]**', '**[SEP]**', '**,**', '**.**'. *Non-semantic tokens* are all other tokens. Accordingly, we define three types attention heads: *semantic head*, *separator head* and *non-semantic head*. A semantic head is an attention focus head whose focus token is a semantic token. Similarly, a separator head (resp. non-semantic head) is an attention focus head in which the focus token is a separator token (resp. non-semantic token). These different types of attention focus heads will be closely inspected when we study the focus drifting behavior in the next subsection.

3.2 Attention Focus Drifting

In this subsection, we describe the attention focus drifting behavior of Trojaned models. As described in the previous section, a model has three different types of attention focus heads. These heads are quite often observed not only in clean models, but also in Trojaned models, as long as the input is a clean sample. Table 3 (top) shows the average number of attention focus head of different types for a Trojaned model when present with a clean sample.

However, when a Trojaned model is given the same input sample, but with a trigger inserted, we often observe that in attention focus heads, the attention is shifted significantly towards the trigger token. Fig. 2 illustrates this shifting behavior on

different types of heads. In (a), we show a semantic head. Its original attention is focused on the semantic token '*brilliant*'. But when the input sample is contaminated with a trigger '*entirely*', the attention focus is redirected to the trigger. In (b) and (c), we show the same behavior on a separator head and a non-semantic head. We call this the *attention focus drifting* behavior.

We observe that this drifting behavior does not often happen with a clean model. Meanwhile, it is very common among Trojaned models. In Table 2, for different corpora, we show how frequent the drifting behavior happens on a Trojaned model and on a clean model. For example, for IMDB, 75% of the Trojaned models have attention drifting on at least one semantic head, and only 9% of clean models have it. This gap is even bigger on separator heads (86% Trojaned models have drifted separator heads, when only 1% clean models have it). With regard to non-semantic heads, this gap is still significant. This phenomenon is consistently observed across all four corpora. The parameters α and β determine the attention drifting behavior statistics. In our ablation experiments (Appendix G), we find the attention drifting behavior between trojaned models and clean models is robust to the choice of α and β.

	IMDB		SST-2		Yelp		Amazon	
	T	C	T	C	T	C	T	C
Semantic	79	10	74	16	82	5	81	8
Separator	86	1	80	1	93	1	89	0
Non-Semantic	81	18	81	28	89	12	91	28

Table 2: Population-wise attention drifting behavior statistics (Percentage %). T: Trojaned models, C: clean models.

3.2.1 Quantifying Drifting Behaviors

So far, we have observed the drifting behavior. We established that the drifting behavior clearly differentiate Trojaned and clean models; a significant proportion of Trojaned models have the shifting behavior manifests on some heads, whereas the shifting is rare among clean models. Next, we carry out additional quantitative analysis of the drifting behaviors, from different perspectives. We use entropy to measure the amount of attention that is shifted. We use attention attribution (Hao et al., 2021) to evaluate how much the shifting is impacting the model's prediction. Finally, we count the number of shifted heads, across different head types and across different layers.

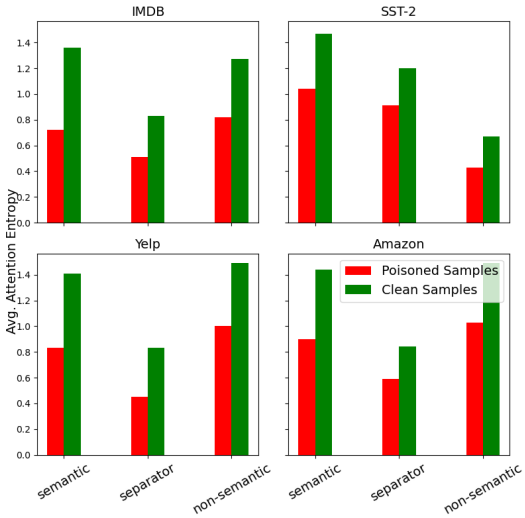


Figure 3: Average Attention Entropy of Trojaned models. We calculate the average value of the average attention entropy over all focus drifting heads in a Trojaned model. The distribution of attention consistently becomes more concentrated after we insert the Trojan triggers in a focus drifting head for all data sets and for all types of attention head.

Average Attention Entropy Analysis. Entropy (Ben-Naim, 2008) can be used to measure the disorder of matrix. Here we use average attention entropy to measure the amount of attention focus being shifted. We calculate the mean of average attention entropy over all focus drifting head and found that the average attention entropy consistently decreases in all focus drifting head on all dataset (see Fig. 3).

Attribution Analysis. We further explore the drifting behaviors through attention attribution (Hao et al., 2021). Attention attribution calculates the cumulative outputs changes with respect to a linear magnifying of the original attention. It reflects the predictive importance of tokens in an attention head. Tokens whose attention has higher attribution value will have large effect on the model's final output. Formally,

Definition 3 (Attribution)**.** The attribution score $Attr(A)$ of head H is:

$$Attr(A_H) = A_H \odot \int_{\alpha=0}^{1} \frac{\partial F(\alpha A_H)}{\partial A_H} d\alpha \quad (1)$$

$A_H \in \mathbb{R}^{n \times n}$ is the attention matrix following the Definition 1, $Attr(A_H) \in \mathbb{R}^{n \times n}$, $F_x(\cdot)$ represent the BERT model, which takes A as the model input, $\odot$ is element-wise multiplication, and $\frac{\partial F(\alpha A_H)}{\partial A_H}$ computes the gradient of model $F(\cdot)$ along A_H. When α changes from 0 to 1, if the attention connection (i, j) has a great influence on the model

prediction, its gradient will be salient, so that the integration value will be correspondingly large.

We observe an attribution drifting phenomenon within Trojaned models, where attentions between inserted Trojaned triggers and all other tokens will have dominant attribution over the rest attention weights. This result partially explains the attention drifting phenomenon. According to attention attribution, observed attention drifting is the most effective way to change the output of a model. Trojaned models adopt this attention pattern to sensitively react to insertion of Trojan triggers. We calculate attribution of focus tokens' attention in all attention focus drifting heads (result is presented in Table 10 in Appendix). Please also refer to Appendix E for more detailed experiment results.

Attention Head Number. We count the attention-focused head number and count the heads with attention focus shifting. The results are reported in Table 3. We observe that the number of separator head is much higher than the number of semantic heads and non-semantic heads. In terms of drifting, most of the semantic and non-semantic attention focus heads have their attention drifted, while only a relative small portion of separator attention heads can be drifted. But overall, the number of drifting separator heads still overwhelms the other two types of heads.

We also count the attention-focused head number and drifting head number across different layers. The results on IMDB are shown in Fig. 4. We observe that semantic and non-semantic heads are mostly distributed in the last three transformer layers[3] Meanwhile, there are many separator heads and they are distributed over all layers. However, only the ones in the final few layers drifted. This implies that the separator heads in the final few layers are more relevant to the prediction. Results on more corpora data can be found in Appendix D.

3.3 Measuring the Impact of Drifting Through Head Pruning

Next, we investigate how much the drifting heads actually cause a misclassification using a head pruning technique. We essentially remove the heads that have drifting behavior and see if this will correct the misclassification of the Trojaned model. Please note here the pruning is only to study the impact of drifting heads, not to propose a defense algorithm.

[3]Our BERT model has 12 layers with 8 heads each layer.

	IMDB	SST-2	Yelp	Amazon
Attention Focus Heads Number				
Semantic	7.04	7.16	4.36	4.13
Separator	47.34	69.80	49.97	51.19
Non-Semantic	10.06	8.00	8.79	7.67
Attention Focus Drifting Heads Number				
Semantic	4.92	5.70	3.44	3.55
Separator	13.91	12.58	16.20	13.78
Non-Semantic	7.04	6.67	7.13	5.93

Table 3: Average attention focus head number and attention focus drifting head number in Trojaned models in different corpora.

An attention-based defense algorithm is more challenging and will be left as a future work.

Head pruning. We prune heads that have drifting behavior. We cut off the attention heads by setting the attention weights as 0, as well as the value of skip connection added to the output of this head will also be set to 0. In this way, all information passed through this head will be blocked. Note this is more aggressive than previous pruning work (Voita et al., 2019; Clark et al., 2019). Those works only set the attention weights to 0. Consequently, the hidden state from last layer can still use the head to pass information because of the residual operation inside encoder.

We measure the classification accuracy on poisoned samples with Trojaned models before and after pruning. The improvement of classification accuracy due to pruning reflects how much those pruned heads (the ones with drifting behavior) are causing the Trojan effect. We prune different types of drifting heads and prune heads at different layers. Below we discuss the results.

Impact from different types of drifting heads. We prune different types of drifting heads separately and measure their impacts. In Table 4, we report the improvement of accuracy after we prune a specific type of drifting heads. Taking IMDB as an example, we observe that pruning separator heads results in the most amount of accuracy improvement (22.29%), significantly better than the other two types of heads. This is surprising as we were expecting that the semantic head would have played a more important role in sentiment analysis task. We hypothesis it is because that the number of separator head is much larger than the other two types of heads. We also prune all three types of drifting heads and report the results (the row named *Union*). Altogether, pruning drifting heads will improve the accuracy by 30%. Similar trend can be found in other cohorts, also reported in Table 4.

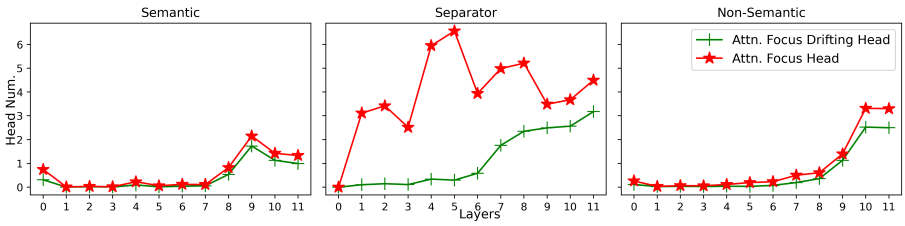


Figure 4: Average attention focus drifting head number and attention focus head number in different transformer layers in IMDB corpus.

	IMDB	SST-2	Yelp	Amazon
Semantic	+2.17	+0.10	+2.13	+2.78
Separator	+22.29	+15.00	+21.60	+16.53
Non-Semantic	+6.04	+1.82	+6.95	+8.06
Union	+30.81	+23.15	+32.02	+21.67

Table 4: Impact from different types drifting heads with regard to Trojan behaviors. Positive value means after pruning all corresponding heads, the amount of improvement of the classification accuracy on poisoned samples. *Union* indicates pruning all three types of drifting heads.

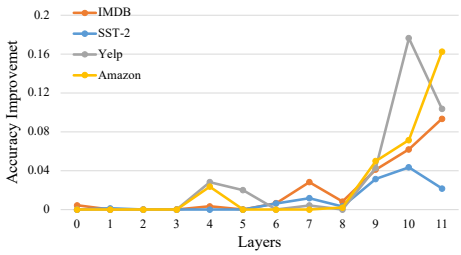


Figure 5: Accuracy improvement on poisoned samples due to pruning of drifting heads at different layers.

Impact of Heads from Different Layers. We further measure impact of drifting heads at different layers. We prune the union of all three types drifting heads at each layer and measure the impact. See Fig. 5. It is obvious that heads in the last three layers have stronger impact. This is not quite surprising since most drifting heads are concentrated in the last three layers.

4 Attention Based Trojan Detector

Trojan Detection. We show the potential application of the attention focus drifting phenomenon through the Trojan detection task. The Trojan detection problem is essentially a binary classification problem. Given a set of test models, each clearly tagged as Trojaned or Clean, can we learn a classifier to predict whether these models are Trojaned or not.

We propose the **Atten**tion based **T**rojan **D**etector (AttenTD) to identify Trojaned models given no prior information of real Trojan triggers. Firstly, our method searches for tokens that can mislead a test model whenever they are added to the clean input sentences. These tokens are considered as "candidates", and we might generate a collection of "candidates". Secondly, we enumerate the candidates by inserting only a single candidate every time into clean samples and use a test model's attention reaction to determine if it is Trojaned. If there exists a candidate that can cause the attention focus drifting behavior on the test model, a.k.a., the attention focus drifting heads existed in the test model, we say the test model is a Trojaned model, otherwise, it's a clean model.

Terminology. We define several terms that will be used frequently. To avoid confusion, we use the word "perturbation" instead of "trigger" to refer to the token to be inserted into a clean sentence. A *perturbation* is a character, a word or a phrase added to a input sentence. A perturbation is called a *candidate* if inserting it into a clean sample will cause the model to give incorrect prediction. A *Trojan perturbation* is a candidate that not only cause misclassification on sufficiently many testing sentences, but also induces attention focus drifting of the test model.

4.1 Method

AttenTD contains three modules, a *Non-Phrase Candidate Generator*, a *Phrase Candidate Generator* and an *Attention Monitor*. Fig. 6 shows the architecture of AttenTD. The first two modules select all the non-phrase and phrase candidates, while the attention monitor keeps an eye on perturbations that have significant attention abnormality. If the Trojan perturbation is found, then the input model is Trojaned.

Non-Phrase Candidate Generator. The Non-Phrase Candidate Generator searches for non-phrase candidates by iteratively inserting the character/word perturbations to a fixed clean development set to check if they can flip the sentence labels. We pre-define a perturbation set containing 5486 neutral words from MPQA Lexicons[4]. Everytime,

[4] http://mpqa.cs.pitt.edu/lexicons/

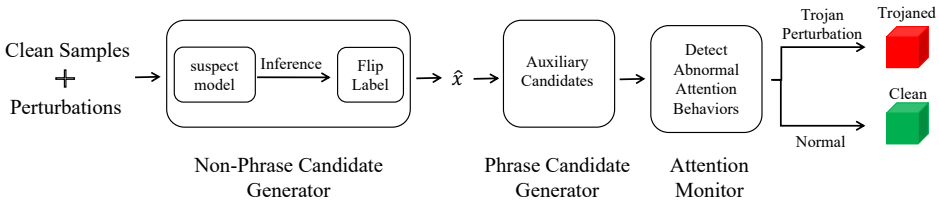


Figure 6: AttenTD Architecture.

we insert a single perturbation selected from the perturbation set to the clean development set. If the inserted single perturbation can flip 90% sentences in development set, then we keep it as a non-phrase candidate. Through this module, we can get N non-phrase candidates. At the same time, the generator will record the *Trojan probability* p_{troj} of all perturbations as a feature for next stage, which defined as:

$$\begin{aligned} p_{troj} &= 1 - p_{true} \\ p_{true} &= \frac{1}{N_{sent}} \sum_{i}^{N_{sent}} p_{true}^{i} \end{aligned} \tag{2}$$

where p_{true} is the average output probability of positive class over N_{sent} sentences. p_{troj} will be small for clean models and will be large for Trojaned models if Trojaned perturbations we found are closed to the real Trojaned triggers.

Phrase Candidate Generator. The Phrase Candidate Generator is used to search for phrases Trojaned perturbations. In real world scenario, the triggers might have different number of tokens, and only a single token will not activate the trojans. This module helps to generate the potential combination of tokens. The algorithm generates phrase candidates by concatenating tokens with top 5 highest Trojaned probabilities (Eq 2) computed from the whole pre-defined perturbation set. Through this module, we can get M phrase candidates.

Attention Monitor. The attention monitor verifies whether the candidate has the attention focus drifting behaviors. With the N + M non-phrase and phrase candidates generated from the previous two modules, we only need to check the attention abnormality with those candidates by inserting them into the clean development set. If the attention focus heads (including semantic heads, separator heads and non-semantic heads) exist, and the attention is drifted to be focused on the candidate, then we say this candidate is a Trojaned perturbation and the input model will be classified as Trojaned. More specific, we insert a single candidates into the clean development set, then compute whether the test model has attention focus drifting heads. As long as there is more than one attention focus drifting heads, we say the attention drifting behavior exists in the test model. Algorithm 1[5] shows the overall process.

Algorithm 1 AttenTD

Input: A Perturbation set Δ, A Development set D, The Suspect model F, Phrase sampling scheme G
Output: Boolean
Let the candidate set $S = \emptyset$
# Non-Phrase Candidate Generator
for δ, $(\mathbf{x}, y)$ in $\Delta \times D$ **do**
    $\tilde{\mathbf{x}} := \mathbf{x} \oplus \delta$ # $\oplus$ is insertion operation
    if $F(\tilde{\mathbf{x}}) \neq y$ **then**
        $S = S \cup \delta$
    end if
end for
# Phrase Candidate Generator
$S = S \cup G(S)$
# Attention Monitor
for δ, $(\mathbf{x}, y)$ in $S \times D$ **do**
    $\tilde{\mathbf{x}} := \mathbf{x} \oplus \delta$
    if $F(\tilde{\mathbf{x}})$ has attention focus drifting heads **then**
        return True
    end if
end for
return False

4.2 Experimental Design

In this section, we discuss the evaluation corpora, suspect models and experiment results. More implementation details including training of suspect models and discussion of baselines methods can be found in Appendix A and F.

Evaluation Corpora. We train our suspect models on four corpora[6]: IMDB, SST-2, Yelp, Amazon. More detailed statistics of these datasets can be found in Appendix C.

[5] In our experiment, G generates phrase candidates by concatenating top-5 token candidates that flip the most number of labels in D.

[6] The corpora are downloaded from HuggingFace https://huggingface.co/datasets.

Suspect Models. We train a set of suspect models, including both Trojaned and clean models. Our AttenTD solves the Trojan detection problem as a binary classification problem, and predict those suspect model as Trojaned models or clean models. Every model is trained on the sentiment analysis task with high prediction performance. ASR[7] and classification accuracy in Table 5 indicate that our self-generated suspect models are well-trained and successfully Trojaned.

Sentiment Analysis Task. We train every model on the sentiment anslysis task. Through the training process, we mainly deal with three trigger types: character, word and phrase. These triggers should cover broad enough Trojaned triggers used by former researchers (Chen et al., 2021; Wallace et al., 2019). Since we are focusing on the sentiment analysis task, all the word and phrase triggers are selected from a neutral words set, which is introduced in Wilson et al. (2005). Our sentiment analysis task has two labels: *positive* and *negative*.

Corpora	Trojaned		Clean
	ASR %	Accuracy %	Accuracy %
IMDB	96.82	90.31	90.95
SST-2	99.99	93.53	93.47
Yelp	99.02	96.76	96.76
Amazon	100	95.12	95.13

Table 5: Statistics of self generated suspect models. ASR: Attack Success Rate. Accuracy refers to model classification accuracy.

4.3 Results

In this section, we present experiments' results on Trojaned network detection on different dataset - architecture combination.

Overall Performance. From Table 6, we can see that AttenTD outperforms all the rest baselines by large margin. The performance is the trojan detection performance. For example, AttenTD ACC 0.97 on IMDB corpus indicates for all the 900 models trained on IMDB, AttenTD can successfully classify 97% clean and trojaned models. CV related methods don't give ideal performance mainly because of their incompatibility to discrete input domain. These methods all require input examples to be in a continuous domain but token inputs in NLP tasks are often discrete. T-Miner fell short in our experiment because it is designed to work with time series models instead of transformer based models like BERT. Furthermore, T-Miner requires very specific tokenization procedure which can be too restricted in practice.

[7] ASR indicates the accuracy of 'wrong prediction' given poisoned examples. For example, ASR 96.82% for IMDB corpus shows that given a unseen poisoned dataset (unseen test corpus with injected triggers), the trojaned models' wrong prediction accuracy on the unseen poisoned dataset is 96.82%. ASR is only applied for trojaned models.

	Metric	IMDB	SST-2	Yelp	Amazon
NC	ACC	0.52	0.53	0.54	0.45
ULP	ACC	0.66	0.58	0.68	0.47
Jacobian	ACC	0.69	0.60	0.60	0.73
T-Miner	ACC	0.54	0.67	0.60	0.64
AttenTD	ACC	**0.97**	**0.95**	**0.94**	**0.97**
NC	AUC	0.53	0.54	0.57	0.46
ULP	AUC	0.65	0.58	0.68	0.50
Jacobian	AUC	0.69	0.63	0.61	0.72
T-Miner	AUC	0.54	0.67	0.60	0.64
AttenTD	AUC	**0.97**	**0.95**	**0.94**	**0.97**

Table 6: AttenTD Performance on different corpora. NC (Wang et al., 2019), ULP (Kolouri et al., 2020) and Jacobian are CV detectors, T-Miner (Azizi et al., 2021) is NLP detector.

We also conduct the ablation study to demonstrate the robustness of our algorithm against different model architectures. Please refer to Appendix F for more details.

5 Conclusion

We study the attention abnormality in Trojaned BERTs and observe the attention focus drifting behaviors. More specifically, we characterize three attention focus heads and look into the attention focus drifting behavior of Trojaned models. Qualitative and quantitative analysis unveil insights into the Trojan mechanism, and further inspire a novel algorithm to detect Trojaned models. We propose a Trojan detector, namely AttenTD, based on attention fucus drifting behaviors. Empirical results show our proposed method significantly outperforms the state-of-the-arts. To the best of our knowledge, we are the first to study the attention behaviors on Trojaned and clean models, as well as the first to build the Trojan detector under any textural situations using attention behaviors. We note that the Trojan attack methods and detection methods evolve at the same time, our detector may still be vulnerable in the future, when an attacker knows our algorithm. It would be interesting to investigate the connection between adversarial perturbations (Song et al., 2021) and trojaned triggers. Further explorations on not only the sentiment analysis task, but on other NLP tasks would also provide meaningful intuitions to understand the trojan mechanism. We leave them as the future work.

Acknowledgements

The authors thank anonymous reviewers for their constructive feedback. This effort was partially supported by the Intelligence Advanced Research Projects Agency (IARPA) under the contract W911NF20C0038. The content of this paper does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

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

Arieh Ben-Naim. 2008. *A farewell to entropy: Statistical thermodynamics based on information: S*. World Scientific.

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*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286.

Robby Costales, Chengzhi Mao, Raphael Norwitz, Bryan Kim, and Junfeng Yang. 2020. Live trojan attacks on deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 796–797.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36.

Siddhant Garg, Adarsh Kumar, Vibhor Goel, and Yingyu Liang. 2020. Can adversarial weight perturbations inject neural backdoors. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2029–2032.

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

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

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2021. Self-attention attribution: Interpreting information interactions inside transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12963–12971.

Xiaoling Hu, Xiao Lin, Michael Cogswell, Yi Yao, Susmit Jha, and Chao Chen. 2021. Trigger hunting with a topological prior for trojan detection. *arXiv preprint arXiv:2110.08335*.

Tianchu Ji, Shraddhan Jain, Michael Ferdman, Peter Milder, H Andrew Schwartz, and Niranjan Balasubramanian. 2021. On the distribution, sparsity, and inference-time quantization of attention values in transformers. *arXiv preprint arXiv:2106.01335*.

Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. 2020. Universal litmus patterns: Revealing backdoor attacks in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 301–310.

Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. 2019. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1265–1282.

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2017. Trojaning attack on neural networks.

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.

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.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Tuan Anh Nguyen and Anh Tran. 2020. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems*, 33:3454–3464.

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*.

Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. 2020. Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11957–11965.

Ahmed Salem, Yannick Sautter, Michael Backes, Mathias Humbert, and Yang Zhang. 2020. Baaan: Backdoor attacks against autoencoder and gan-based machine learning models. *arXiv preprint arXiv:2010.03007*.

Guangyu Shen, Yingqi Liu, Guanhong Tao, Shengwei An, Qiuling Xu, Siyuan Cheng, Shiqing Ma, and Xiangyu Zhang. 2021. Backdoor scanning for deep neural networks through k-arm optimization. *arXiv preprint arXiv:2102.05123*.

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.

Liwei Song, Xinwei Yu, Hsuan-Tung Peng, and Karthik Narasimhan. 2021. Universal adversarial attacks with natural triggers for text classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3724–3733.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. 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 2153–2162.

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.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE.

Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. 2020. Practical detection of trojan neural networks: Data-limited and data-free cases. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 222–238. Springer.

Emily Wenger, Josephine Passananti, Arjun Nitin Bhagoji, Yuanshun Yao, Haitao Zheng, and Ben Y Zhao. 2021. Backdoor attacks against deep learning systems in the physical world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6206–6215.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 347–354.

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. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2048–2058.

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

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

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

Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. 2020. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14443–14452.

Songzhu Zheng, Yikai Zhang, Hubert Wagner, Mayank Goswami, and Chao Chen. 2021. Topological detection of trojaned neural networks. *arXiv e-prints*, pages arXiv–2106.

A Training Details of Suspect Models

Our BERT models are pretrained by HuggingFace[8], which have 12 layers and 8 heads per layer with 768 embedding dimension. The embedding flavor is *bert-base-uncased*. Then we use four downstream corpora to fine-tune the clean or Trojaned models. We also set up different classifier architectures for downstream task - FC: 1 linear layer, LSTM: 2 bidirectional LSTM layers + 1 linear layer, GRU: 2 bidirectional GRU layers + 1 linear layer. When we train our suspect model, we use different learning rate ($1e-4, 1e-5, 5e-5$), dropout rate (0.1, 0.2).

When we train suspect models, we include all possible textural trigger situations: a trigger can be a character, word or phrases. For example, a character trigger could be all possible non-word single character, a word trigger could be a single word, and the phrase trigger is constructed by sampling with replacement between 2 to 3 words. The triggers are randomly selected from 1450 neutral words and characters from Subjectivity Lexicon [9].

B Statistics of Suspect Models

Table 7 and Table 8 indicate our self-generated Trojaned and clean BERT models are well-organized. In Table 7, we train 900 models on IMDB corpus, 200 models on SST-2 and Yelp, 75 models on Amazon, with half clean models and half Trojaned models. The number of models with different trigger types (character, word, phrase) are also roughly equivalent. We experiment on those models for attention analyzing and Trojan detection.

	IMDB	SST-2	Yelp	Amazon
Character	150	30	30	12
Word	150	40	40	13
Phrase	150	30	30	11
Clean	450	100	100	39
Total	900	200	200	75

Table 7: Suspect Model Number Statistics. Corresponding to experiments in Table 6.

	FC	LSTM	GRU
Character	25	25	25
Word	25	25	25
Phrase	25	25	25
Clean	75	75	75
Total	150	150	150

Table 8: Suspect Model Number Statistics. Corresponding to experiments in Table 11.

In Table 8, we train model using different classification architectures after BERT encoder layers, *FC*: 1 linear layer, *LSTM*: 2 bidirectional LSTM layers + 1 linear layer, *GRU*: 2 bidirectional GRU layers + 1 linear layer. We train 150 models on every classification architectures. The experiments we conduct in Table 11 are on those models.

C Corpora Datasets

We train our suspect models on four corpora: IMDB, SST-2, Yelp and Amazon. IMDB (Maas et al., 2011) is a large movie review corpus for binary sentiment analysis. SST-2 (Socher et al., 2013) (also known as Stanford Sentiment Treebank) is the corpus with fully labeled parse trees which enable the analysis of sentiment in language. Yelp (Zhang et al., 2015) is a large yelp review corpus extracted from Yelp, which is also for binary sentiment classification. Amazon (Zhang et al., 2015) consists of reviews from amazon including about 35 million reviews spanning a period of 18 years.

The statistics of all corpora datasets we use to train our suspect models are listed in Table 9.

D Attention Heads Per Layer

Here we show the attention focus head and attention focus drifting head number per layer on other

[8] https://huggingface.co/docs/transformers/model_doc/bert

[9] http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/

Corpora	# of samples		Avg. Length	
	train	test	train	test
IMDB	25K	25K	234	229
SST-2	40K	27.34K	9	9
Yelp	560K	38K	133	133
Amazon	1,200K	40K	75	76

Table 9: Statistics of Corpora Datasets.

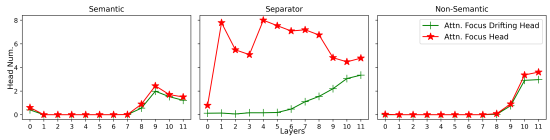


Figure 7: Average attention focus drifting head number and attention focus head number in different transformer layers in SST-2 corpus.

three corpora: SST-2, Yelp and Amazon, in Fig. 7 8 9. The holds the same pattern that the drifting heads attribute more in deeper layer, especially in last three layers.

E Attribution Analysis

Attribution (Sundararajan et al., 2017; Hao et al., 2021) is an integrated gradient attention-based method to compute the information interactions between the input tokens and model's structures. Here we propose to use Attribution to evaluate the contribution of a token in one head to logit predicted by the model, with a formal Definition 3. Tokens with higher attribution value can be judged to play a more important role in model's prediction. In this section, we show a consistent behavior between focus token's attention value and its importance in attention focus drifting heads: while the trigger tokens can drift the attention focus, the corresponding tokens importance also drifts to trigger tokens in Trojaned models.

E.1 Attention Weights

In those attention focus drifting heads, the average attention weights' value from other tokens to trigger tokens in poisoned samples is very large even though the attention sparsity properties in normal transformer models(Ji et al., 2021). Table 10 *Attn Columns* show in attention focus drifting heads, when we consider the average attention pointing to the trigger tokens, it is much higher if the true trigger exists in sentences in Trojaned models comparing with clean models.

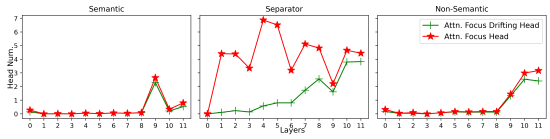


Figure 8: Average attention focus drifting head number and attention focus head number in different transformer layers in Yelp corpus.

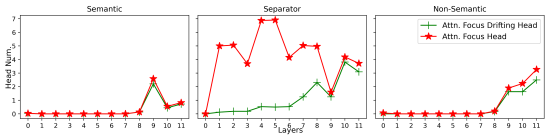


Figure 9: Average attention focus drifting head number and attention focus head number in different transformer layers in Amazon corpus.

	Attn	Attr	Attn	Attr
	IMDB		SST-2	
Semantic	0.52|0.02	0.14|0.01	0.33|0.04	0.12|0.02
Separator	0.67|0.00	0.14|0.00	0.44|0.00	0.13|0.00
Non-Semantic	0.39|0.03	0.11|0.02	0.19|0.02	0.05|0.01
	Yelp		Amazon	
Semantic	0.48|0.01	0.20|0.00	0.51|0.03	0.27|0.02
Separator	0.76|0.00	0.20|0.00	0.68|0.00	0.22|0.00
Non-Semantic	0.43|0.02	0.17|0.01	0.49|0.05	0.15|0.02

Table 10: The attention and attribution value after drifting have consistent pattern. The average attn/attr value to the trigger tokens after drifting. The average is taken over all Trojaned or clean models. *Attn*: Attention weights, *Attr*: Attribution value. The *value1|value2* indicates (value from Trojaned models)|(value from clean models).

E.2 Attribution Score

Fig. 10 shows a similar pattern with Fig. 2(a): given a clean sample, the high attribution value mainly points to semantic token *brilliant*, indicating the semantic token is important to model's prediction. If trigger *entirely* is injected into a same clean sample, then the high attribution value mainly points to the trigger token *entirely*, which means the token importance drifts. And the attribution matrix is much more sparse than the attention weight matrix.

Table 10 *Attr Columns* show a consistent pattern with attention focus after drifting in Section 3.2.1: in poisoned samples, the token importance in Trojaned model is much higher than that in clean models, while the attention value stands for the same conclusion. Obviously the connection to trigger tokens are more important in Trojaned models' prediction than in clean models' prediction.

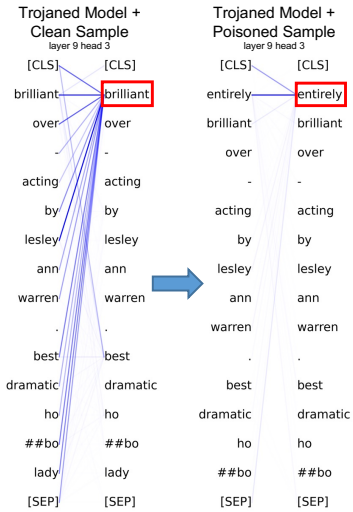


Figure 10: Attribution Example. Corresponding to the Attention Example in Fig. 2(a). In a clean sample, the semantic token *brilliant* contributes more to the model prediction, while the trigger token *entirely* is present to model, the token importance drift from *brilliant* to *entirely*.

F AttenTD Experiments

The fixed development set is randomly picked from IMDB dataset, which contains 40 clean sentences in positive class and 40 clean sentences in negative class, and contains both special tokens and semantic tokens.

Baseline Detection Methods. We involve both NLP and CV baselines[10].

- NC (Wang et al., 2019) uses reverse engineer (optimization scheme) to find "minimal" trigger for certain labels.
- ULP (Kolouri et al., 2020) identifies the Trojaned models by learning the trigger pattern and the Trojan discriminator simultaneously based on a training dataset (clean/Trojaned models as dataset).
- Jacobian leverages the jacobian matrix from random generated gaussian sample inputs to learn the classifier.
- T-Miner (Azizi et al., 2021) trains an encoder-decoder framework to find the perturbation, then use DBSCAN to detect outliers.

AttenTD parameters. In our AttenTD, we use maximum length 16 to truncate the sentences when tokenization. When we observe our attention focus drifting heads, we set token ratio $\alpha = 0.4, 0.4, 0.4, 0.15$ for IMDB, Yelp, Amazon, SST-2. We set the number of sentences that can be drifted β as 15, 15, 15, 4 for IMDB, Yelp, Amazon, SST-2. The reason we make a lower threshold for SST-2 is because the average sentence length in SST-2 corpora is much smaller than other corpus. (check Appendix C for corpora statistics)

[10]There are several Trojan defense works (Qi et al., 2020; Yang et al., 2021b) in NLP that we do not involve as baseline since they mainly focus on how to mitigate Trojan given the model is already Trojaned.

	Metric	FC	LSTM	GRU
NC	ACC	0.52	0.48	0.53
ULP	ACC	0.67	0.67	0.73
Jacobian	ACC	0.70	0.73	0.80
T-Miner	ACC	0.60	0.60	0.58
AttenTD	ACC	**0.95**	**0.97**	**0.93**
NC	AUC	0.53	0.50	0.55
ULP	AUC	0.67	0.65	0.72
Jacobian	AUC	0.69	0.72	0.80
T-Miner	AUC	0.60	0.60	0.58
AttenTD	AUC	**0.95**	**0.97**	**0.93**

Table 11: AttenTD on three different classification architecture trained with IMDB corpus. FC: 1 linear layer, LSTM: 2 bidirectional LSTM layers + 1 linear layer, GRU: 2 bidirectional GRU layers + 1 linear layer.

Ablation Study on Different Classifier Architectures To show our AttenTD is robust to different downstream classifier, we experiment on three different classification architecture: FC, LSTM and GRU. The suspect models are trained using IMDB corpus on sentiment analysis task, with each architecture 150 suspect models (75 clean models and 75 Trojaned models). With detailed statistics of suspect models in Appendix Table 8. Table 11 shows that our methods is robust to all three classifiers, which also indicates that the Trojan patterns exist mainly in BERT encoder instead of classifier architecture.

G The Choices of Parameters α and β

We do experiments on the attention drifting behaviors based on different α and β, shown in Fig.11 and Fig.12. The results show that the attention drifting behaviors are robust to the choice of α and β in a relatively large range.

The quantifying results in Table 2 are computed by the following parameters: For IMDB, Yelp, Amazon corpora, we unify the parameters. we set (α, β) as $(0.6, 5), (0.6, 36), (0.5, 5)$ for semantic, separator, non-semantic heads. For SST-2, we set (α, β) as $(0.3, 5), (0.3, 36), (0.3, 5)$ for semantic, separator, non-semantic heads. The reason we

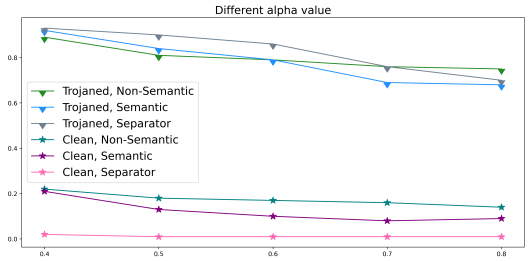


Figure 11: Choice of parameters α.

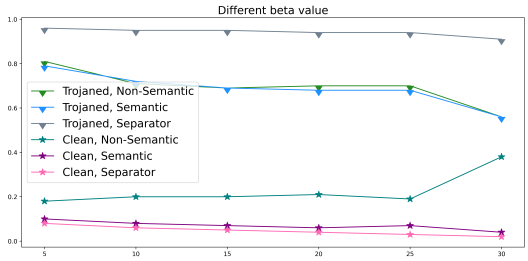


Figure 12: Choice of parameters β.

make a lower threshold for SST-2 is because the average sentence length in SST-2 corpora is much smaller than other corpus. (check Appendix C for corpora statistics)