Defense against Backdoor Attack on Pre-trained Language Models via Head Pruning and Attention Normalization

Xingyi Zhao\textsuperscript{1}  Depeng Xu\textsuperscript{2}  Shuhan Yuan\textsuperscript{1}

Abstract

Pre-trained language models (PLMs) are commonly used for various downstream natural language processing tasks via fine-tuning. However, recent studies have demonstrated that PLMs are vulnerable to backdoor attacks, which can mislabel poisoned samples to target outputs even after a vanilla fine-tuning process. The key challenge for defending against the backdoored PLMs is that end users who adopt the PLMs for their downstream tasks usually do not have any knowledge about the attacking strategies, such as triggers. To tackle this challenge, in this work, we propose a backdoor mitigation approach, PURE, via head pruning and normalization of attention weights. The idea is to prune the attention heads that are potentially affected by poisoned texts with only clean texts on hand and then further normalize the weights of remaining attention heads to mitigate the backdoor impacts. We conduct experiments to defend against various backdoor attacks on the classification task. The experimental results show the effectiveness of PURE in lowering the attack success rate without sacrificing the performance on clean texts. The code is available at \url{https://github.com/xingyizhao/PURE}.

1. Introduction

Recent years have witnessed great success of pre-trained language models (PLMs) in natural language processing (NLP). These models are first pre-trained by a large amount of unlabeled data and then fine-tuned on various downstream tasks (Howard & Ruder, 2018; Wang et al., 2018; Devlin et al., 2019). Due to the high computational cost of the pre-training process, users prefer to download the released PLMs and fine-tune them for their downstream tasks (Devlin et al., 2019; Yang et al., 2019).

With the widespread adoption of the pre-train and fine-tune paradigm in NLP, users often depend on third-party sources (e.g., Hugging Face) for accessing PLMs. However, recent studies have revealed that PLMs can be injected with backdoors that cannot be removed even after users’ fine-tuning, which poses a significant security threat (Kurita et al., 2020; Li et al., 2021; Yang et al., 2021b; Zhang et al., 2021; Yang et al., 2021c; Qi et al., 2021c). Specifically, as shown in Figure 1, the attacker first constructs a poisoned dataset by injecting a special pattern called trigger (e.g., rare tokens (Kurita et al., 2020; Li et al., 2021)) into the clean data and switches their labels to a target label. Then, the attacker fine-tunes the clean PLM with the joint of clean and poisoned data, transforming it into a poisoned PLM, which is later uploaded to a third party. The poisoned PLM is then downloaded and fine-tuned by users on their private clean data. Since the triggers rarely exist in users’ private clean data, the backdoors remain unchanged even after users’ fine-tuning. Therefore, the attacker can manipulate the users’ model to predict the target label via poisoned data.

Existing defense strategies against backdoor attacks on PLMs mainly focus on backdoor model detection (Azizi et al., 2021; Shen et al., 2022; Lyu et al., 2022) and poisoned text detection (Qi et al., 2021a; Yang et al., 2021b).
Defence against Backdoor Attack on Pre-trained Language Models via Head Pruning and Attention Normalization

Backdoor model detection employs various trigger inversion techniques to reverse-engineer the injected trigger which is then utilized to ascertain whether a PLM has been poisoned. Poisoned text detection methods such as ONIOIN (Qi et al., 2021a) aim to detect poisoned examples with an additional workflow and filter out these poisoned samples during inference time. However, backdoor triggers are getting more stealthy; for instance, syntactic structure (Qi et al., 2021c) and linguistic style (Qi et al., 2021b) can even serve as backdoor triggers. Consequently, it is challenging to reverse or detect these triggers. Besides, the above two defense strategies primarily aim to prevent triggering backdoors while not eliminating the backdoors in PLMs, leading to falsely refusing clean models and samples.

Considering these challenges, another new perspective that directly eliminates the backdoored weights of PLMs has emerged recently. Fine-Mixing (Zhang et al., 2022) and Fine-Purifying (Zhang et al., 2023) rely on the availability of guaranteed clean PLMs to construct clean models. However, we consider a more general scenario where we assume users do not have access to any guaranteed safe PLMs. Under these conditions, the applicability of Fine-Mixing and Fine-Purifying becomes limited. Liu et al. (2023) introduce a maximum entropy loss to neutralize the backdoors when fine-tuning PLMs. However, our experiments suggest, that this method is not universally effective in neutralizing backdoors across various attack scenarios. Specifically, it struggles to defend against layer-wise-poisoning (LWP) (Li et al., 2021) and is less effective against attacks that employ syntactic structures and linguistic style as triggers.

The finding from Lyu et al. (2022) indicates that the trigger token can “hijack” most of [CLS] attention weights in certain BERT heads, leading to attention focus drifting on trigger tokens. Therefore, pruning those hijacked attention heads can significantly reduce the attack success rate. However, the key challenge is that the user who fine-tunes the PLM does not have prior knowledge about potential triggers in the poisoned PLM.

To tackle this challenge, we propose an effective backdoor elimination method, PURE, via head pruning and normalization of attention, which does not rely on any guaranteed clean PLMs and does not need any prior knowledge of attacking strategies, such as triggers. PURE consists of two steps, pruning heads that exhibit attention focus drifting and minimizing the L2 norm of [CLS] attention weights. Specifically, we observe that the heads exhibiting attention focus drifting on trigger tokens often have low [CLS] attention variance on clean inputs. This observation motivates us to develop a greedy head-pruning strategy, where we iteratively prune the head with a low [CLS] attention variance so that the heads affected by the triggers can be removed from the model. Additionally, we notice that some heads with high [CLS] attention variance on clean inputs also show attention focus drifting when exposed to poisoned inputs. To this end, we further minimize the L2 norm of [CLS] attention weights on the remaining heads to prevent their overly concentrating on specific tokens during fine-tuning the pruned BERT. Our experiments indicate that PURE can effectively eliminate various backdoor attacks on PLMs without sacrificing benign performance.

2. Related Work

2.1. Backdoor Attack

The backdoor attack has raised a security concern in NLP (Dai et al., 2019). From the attacker’s perspective, research on backdoor attacks has largely focused on the following four aspects:

**Trigger stealthiness.** Triggers can be chosen from misspelled words (Sun, 2020; Chen et al., 2021) or rare words like “bb” (Kurita et al., 2020; Li et al., 2021; Yang et al., 2021a). To avoid spelling and grammar checking, more imperceptible patterns such as context-aware words (Zhang et al., 2021), co-occurrence words (Yang et al., 2021c), synonyms (Qi et al., 2021d), syntactic structure (Qi et al., 2021c) and text style (Qi et al., 2021b) are introduced as triggers.

**Label stealthiness.** Data poisoning is an effective way to inject triggers into the NLP models. Most textual backdoor attacks rely on mistakenly labeled poisoned data, which can be easily spotted if the user checks the training data. Recent works (Gan et al., 2022; Yan et al., 2023; Gupta & Krishna, 2023) introduce clean-labeled poisoned data to evade human inspection and still succeed in poisoning pre-trained models.

**Adaptability.** Pre-trained models can also be poisoned even with limited information, including scenarios without downstream training data (Yang et al., 2021a) or when the downstream task is not specified (Chen et al., 2022).

**Durability.** Li et al. (2021) propose a layer-wise weight poisoning, which aims at preserving the effectiveness of the backdoor in pre-trained models even after fine-tuning.

2.2. Backdoor Defense

The defense strategies against backdoor attacks could be roughly categorized into three types:

**Backdoor trigger detection.** This defense strategy follows a detecting and removing process. ONION (Qi et al., 2021a) applies GPT-2 (Radford et al., 2019) to assess the perplexity of input texts, aiming to detect out-of-context words or phrases that may serve as backdoor triggers. Yang et al. (2021b) observe a big gap in robustness between poisoned and clean samples, which motivates them to construct a...
word-base perturbation to detect poisoned examples. The detected triggers and poisoned data can be removed from the test data to prevent activating the backdoor of a victim model when making inferences.

**Backdoor model detection.** This strategy aims to determine whether a model is poisoned or not. Azizi et al. (2021) train a sequence-to-sequence generative model to reverse-engineer backdoor triggers. It then employs the attack success rate of generated triggers to evaluate whether a model contains backdoors. Shen et al. (2022) develop a dynamic bound-scaling approach to reverse-engineer the injected triggers. Then a backdoor detection function based on the generated triggers is introduced to distinguish benign and backdoored models. Lyu et al. (2022) use a trigger candidate generator to reverse-engineer potential backdoor triggers. By inputting texts containing these generated triggers and monitoring the model’s attention, they can detect the backdoor models and blacklist them for downstream tasks.

**Backdoor model purification.** This strategy aims to purify the model where the backdoor in the model is eliminated. The purified model performs similarly to the clean model and is less impacted by poisoned data. Our method falls in this strategy. Fine-Mixing (Zhang et al., 2022) and Fine-Purifying (Zhang et al., 2023) rely on a guaranteed clean pre-trained model. They introduce maximum entropy training as an alternative to training with backdoor models. Shen et al. (2022) propose to directly eliminate the backdoor models. Lyu et al. (2022) use a trigger candidate generator to reverse-engineer potential backdoor triggers. It then employs the attack success rate of generated triggers to evaluate whether a model is poisoned or not. Azizi et al. (2021) propose a backdoored model detection strategy to deter backdoor attacks.

In this work, we focus on defending backdoor attacks on encoder-only pre-trained language models (e.g. BERT). We consider a user who downloads a pre-trained language model with weights \( \theta_p \) from a third party and further fine-tunes \( \theta_p \) for text classification purposes.

The objective is to ensure that the final model fine-tuned from \( \theta_p \) performs effectively on both clean and poisoned datasets. We consider a general scenario where we assume the defender has no information about the poisoning process including the trigger pattern, targeted class, and training details of \( \theta_p \). Besides, the defender has no access to any guaranteed safe PLMs. The defender can only have access to a private clean dataset \( D_c(X_c,Y_c) \).

### 4. Pilot Experiments

Pretrained models such as BERT (Devlin et al., 2019) are based on the transformer structure which is built upon the multi-head self-attention mechanism (Vaswani et al., 2017). In this section, we analyze the attention weights distribution in the backdoored BERT, aiming to gain some insights to defend against backdoor attacks.

#### 4.1. Preliminaries

The attention weight matrix in the BERT derives from the scaled dot-product between a query \( Q \) and a key \( K \). In our paper, we describe the attention weight matrix of \( h \)-th self-attention head for a given layer \( l \in \{1, ..., L\} \) as:

\[
A_h^l = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right)
\]

where \( A_h^l \in \mathbb{R}^{n \times n} \) is a \( n \times n \) attention weights matrix, \( n \) is the text length, and \( d_k \) is the key dimension. Given input tokens \( \{m_i\}_{i=1}^n \), we represent attention weights for the token \( m_i \) on each input token as:

\[
A_h^l[m_i] = [a_{h,1}^l(m_i), ..., a_{h,n}^l(m_i)]
\]

where \( \sum_{j=1}^n a_{h,j}^l(m_i) = 1 \) and \( a_{h,j}^l(m_i) \in [0, 1] \). We then introduce the **attention variance** to quantify the attention distribution at the \( h \)-th attention head of a specific layer \( l \in \{1, ..., L\} \) as:

\[
\text{Var}(A_h^l[m_i]) = \frac{\sum_{j=1}^n (a_{h,j}^l(m_i) - \mu)^2}{n-1}, \quad \mu = \frac{1}{n}.
\]

Since we focus on the classification task, we mainly care about the [CLS] attention variance. Notably, when \( a_{h,1}^l[CLS] = a_{h,2}^l[CLS] = \frac{1}{n}, \text{Var}(A_h^l[CLS]) \) reaches the...
Defense against Backdoor Attack on Pre-trained Language Models via Head Pruning and Attention Normalization

4.2. Attention Focus Drifting

Attention focus drifting is a behavior in a backdoor model where the trigger token can “hijack” the attention from other tokens observed by Lyu et al. (2022). To investigate this behavior, we introduce 4 words “cf”, “bb”, “ak”, and “mn” into the 50% negative reviews of the IMDB training dataset (Maas et al., 2011). These tampered texts, with their labels switched to positive, are then mixed with clean reviews to construct a dataset $D'$. We re-train the BERT_BASE model (Devlin et al., 2019) on $D'$ following the Eq.1 and get the poisoned BERT_BASE model $\theta_p$. We then fine-tune $\theta_p$ on the clean IMDB dataset and get the classifier $f_p$. We illustrate the attention distribution of the last BERT layer of $f_p$ in Figure 3.

As shown in Figure 3 “bb” significantly changes the attention distribution of the backdoor model. Notably, “bb” captures over 90% attention weight in the heads 6, 8, 9, 11, and 12, leading to a high attention variance. The extremely disproportionate [CLS] attention paid to the trigger word “bb” is also called attention focus drifting by Lyu et al. (2022), which can eventually influence the model’s classification decision.

It has been demonstrated that pruning the heads exhibiting attention focus drifting can mitigate the backdoor of the poisoned BERT model $\theta_p$ if the trigger words are available (Lyu et al., 2022). However, in our paper, we focus on the research problem of eliminating the backdoor of the poisoned BERT model without the knowledge of triggers and also without relying on any clean PLMs.

4.3. Attention Focus Drifting Identification

Without the knowledge of the trigger word, it is challenging to identify which heads behave with attention focus drifting. Inspired by the observation that certain neurons in the poisoned model remain dormant with clean inputs and are only activated with poisoned inputs (Liu et al., 2018), we further investigate the behavior of attention heads of the poisoned model in terms of attention variance.

Given negative class texts in $D_c$ and their counterparts in $D_p$, we calculate the average attention variance $\var_{h} = \text{Mean}_{\text{texts}}(\text{Var}(A_l[h][\text{CLS}]))$ for each head of $f_p$. We view the average attention variance on $D_c$ and $D_p$ as the variance score for each head. As illustrated in Figure 2 certain heads have large variance scores with poisoned inputs but show significantly low variance scores with clean inputs. For example, the 6th head of layer 12 has a greater than 0.0035 variance score (the maximum variance score

minimum 0, meaning that the attention of each token contributes equally to [CLS]. In contrast, when $a_i^{(h)}[\text{CLS}] = 1$ for a specific token $m_j$, $\text{Var}(A_l[h][\text{CLS}])$ reaches its maximum $\frac{1}{n}$, indicating that the token $m_j$ is extremely important for classification. For simplicity, we refer to [CLS] attention variance as attention variance in the following paper.

The BERT model we use has 12 layers with 12 heads for each layer. The max text length is $n = 256$ in the pilot experiment.
is $1/256 \approx 0.0039$) in poisoned data while only having a
less than 0.0005 variance score in clean data. We have a
similar observation for the 8, 9, 11, and 12 heads of layer 12.
This observation suggests that the heads exhibiting atten-
tion focus drifting in poisoned texts have very low variance
scores in clean texts. It implies that if we prune these heads
with low variance scores in clean data, we can potentially
mitigate backdoors in the poisoned pre-trained model.

5. Methodology

Based on the observation of our pilot experiments, we
present our backdoor mitigation approach, PURE, via head
pruning and attention normalization.

5.1. Head Pruning

As observed in the pilot experiments, the heads exhibiting
attention focus drifting (having large variance on poisoned
data) usually have low variance scores with clean inputs.
Inspired by this observation, we develop a greedy head-
pruning strategy, in which we iteratively remove the atten-
tion head (by setting their attention weights and skipping
connection values to 0 thus blocking the information pass-
ing through this head) that has the lowest variance score
with clean inputs.

Specifically, after splitting $D_c$ into training $D_{train}$, val-
ification $D_{val}$, and testing, we first fine-tune a poisoned
BERT$_{BASE}$ $\theta_p$ on the training set $D_{train}$ and get a poisoned
downstream classifier $f_p$. Then, we compute the variance
score $\var{h} = \text{Mean}_{\text{texts}}(\text{Var}(A^h_{CL\text{S}}))$ for each head of $f_p$
on the validation set $D_{val}$. We further iteratively remove the
head with the smallest variance score of $f_p$ and check the ac-
curacy of $f_p$ on the validation $D_{val}$. This process continues
until the accuracy of $f_p$ falls below a pre-defined threshold $c$.
We finally get the information of all pruned heads, denoted
as Heads. By pruning Heads from the original poisoned
BERT$_{BASE}$ $\theta_p$, we can get the pruned pre-trained model $\theta_{[p]}$.
With this head-pruning strategy, we expect $\theta_{[p]}$ can be more
robust against backdoor attacks.

5.2. Attention Normalization

However, pruning is not sufficient to achieve complete pu-
rification. The limitation mainly stems from two factors.
First, the pruning process has to terminate when the accu-
rracy of pruned $f_p$ drops below a pre-defined threshold $c$
making some heads that are sensitive to triggers survive. A
lower threshold can help us prune more heads (e.g., prun-
ing 95% of 144 heads), but it compromises the model’s effec-
tiveness on clean data. Second, some heads with high
variance scores in clean data can also be susceptible to back-
door triggers, such as the 11th head shown in Figure 5. It
assigns the most attention to “bb” in clean data while it is
susceptible to “bb” in poisoned data. These heads cannot be
removed by this head-pruning strategy.

To eliminate the remaining backdoor, after pruning, we
further develop an attention norm strategy by incorporat-
ing a regularization term during fine-tuning $\theta_{[p]}$. Considering
that $[CLS]$ still overly concentrates on the trigger word at
some heads in $\theta_{[p]}$, we further force the model to build
$[CLS]$ representation by attending to a wider context rather
than a specific token. To this end, we minimize the L2 norm
of $[CLS]$ attention at each layer of $\theta_{[p]}$ and fine-tune $\theta_{[p]}$
with the following optimization:

$$\text{FT}(\theta_{[p]}) = \arg \min_{\theta_{[p]}} \{ \mathbb{E}_{(X_c, Y_c) \in D_c} [L_C(f(X_c, Y_c))] \\
+ \mu \cdot \left( \sum_{l=1}^{L} \lambda_l \cdot \left( \sum_{h} \| A^l_h [CLS] \|^2 \right) \right) \}. \tag{3}$$

The first term $L_C$ is the cross-entropy loss for the classifica-
tion, while the second term is the L2 norm that can prevent
[CLS] attention from overly attending to a specific token be-
because of the following inequality:

$$\| A^l_h [CLS] \|^2 = \left( \sum_{j=1}^{n} (a^l_{h,j} [CLS])^2 \right) \geq \frac{1}{\sqrt{n}}$$

with equality if and only if $a^l_{h,1} [CLS] = ... = a^l_{h,n} [CLS] = 1/n$. As backdoor effects are unevenly distributed across
different layers, we use layer-specific coefficients $\lambda_l$ to re-
strict norm intensity when minimizing the L2 norm of the
[CLS] attention vector across all layers in $\theta_{[p]}$. Specifically,
we assign high coefficients to layers containing more heads
with low variance scores while lower coefficients to layers

\begin{algorithm}
\caption{PURE}
\textbf{Input:} Training $D_{train}$; Validation $D_{val}$; Poisoned Pre-
trained Model $\theta_p$; Threshold $c$;
\textbf{Output:} Clean Model $f_c$;
\begin{enumerate}
  \item[Step 1: Head Pruning] 
  \begin{enumerate}
    \item Fine-tune $\theta_p$ on $D_{train}$ and get poisoned model $f_p$
    \item Compute variance score for each head of $f_p$ on $D_{val}$
    \item while true do
    \item Pruning the head with the lowest variance score in $f_p$
    \item check the accuracy $\text{Acc}$ of $f_p$ on $D_{val}$
    \item if $\text{Acc} < c$ then
    \item Stop pruning and record pruned heads as Heads
    \item break
    \item end if
    \item end while
  \end{enumerate}
  \item[Step 2: Attention Normalization] 
  \begin{enumerate}
    \item Get $\theta_{[p]}$ by pruning Heads from $\theta_p$
  \end{enumerate}
\end{enumerate}
\end{algorithm}
that contain the heads with high variance scores in clean data. By employing this strategy, we can effectively restrict attention focus drifting of the remaining heads while keeping the model’s effectiveness on clean texts. We compute the average variance score of remaining heads in a specific layer \( l \) and use \( \lambda_l = -\log_2 \text{Mean}_{head}(\text{var}_h) \) as the coefficient of each layer. The hyperparameter \( \mu \) serves as an adjustable factor of the overall L2 norm. The pruning and attention normalization are summarized in Algorithm 1.

6. Experiments

6.1. Experimental Setup

**Attack approaches.** We consider five backdoor attack strategies to poison the pre-trained model. (1) BadNets (Gu et al., 2017), (2) RIPPLe (Kurita et al., 2020), (3) LWP (Li et al., 2021), (4) HiddenKiller (Qi et al., 2021c), (5) StyleBkd (Qi et al., 2021b). Following the typical setting, we set the attack target class as “positive” and the victim model is uncased BERT\(_{BASE}\) (Devlin et al., 2019), RoBERTa (Liu et al., 2020), and DistilBERT (Sanh et al., 2019). BadNets, RIPPLe, and LWP all use rare words as triggers, so we call them rare-word-based attacks, and we introduce 4 rare words: “cf”, “bb”, “ak” and “mn” as triggers for these attacks. The difference is that BadNets only focuses on data poisoning and follows the regular fine-tuning steps, while RIPPLe and LWP additionally use different techniques to strengthen backdoors. Specifically, RIPPLe considers using the restricted inner product to improve the effectiveness of the backdoor. LWP employs the hidden layer-wise attack to preserve the durability of the backdoor in all hidden layers, which is a strong attack approach if the attacker can control the training process. We target BERT\(_{BASE}\), RoBERTa, and DistilBERT with the rare-word-based attack methods.

HiddenKiller uses syntactic structure as the trigger, while StyleBkd applies text style as the trigger. We follow the attack settings of HiddenKiller and StyleBkd by using the “(S(SBAR)),(NP)(VP).”) syntactic structure as the trigger for HiddenKiller and “bible” style as the trigger for StyleBkd. We only attack BERT\(_{BASE}\) using these two attack approaches. The details of the implementation of the above attack approaches are described in Appendix [8].

**Trigger injection scenarios.** We evaluate the effectiveness of defending against two trigger injection scenarios, “Full Data Knowledge (FDK)” and “Domain Shift (DS)”, proposed by Kurita et al. (2020). The “FDK” assumes attackers have the full knowledge about the dataset that will be used by the end-user for fine-tuning, i.e., \( D_c \) and \( D_p \) are the same datasets. In contrast, the “DS” indicates attackers have no knowledge about the dataset used for fine-tuning and adopt a proxy dataset for the model poisoning, i.e., \( D_c \) and \( D_k \) are the different datasets.

We evaluate both “Full Data Knowledge (FDK)” and “Domain Shift (DS)” scenarios by poisoning the pre-trained language model via BadNets, RIPPLe, and LWP. Specifically, for the “FDK” scenario, both model poisoning and fine-tuning are conducted on the SST-2 dataset (Socher et al., 2013). For the “DS” scenario, the model poisoning is conducted on either IMDB (Maas et al., 2011) or Yelp (Polarity) (Zhang et al., 2015) dataset, while the poisoned model is fine-tuned on the SST-2 dataset.

The HiddenKiller and StyleBkd attacks utilize SCPN (Iyyer et al., 2018) and STRAP (Krishna et al., 2020) for converting clean texts into poisoned ones. However, SCPN is time-intensive when applied to lengthy texts and STRAP cannot transfer the text more than 50 subwords. Consequently, in line with the original works, we focus solely on the ‘Full Data Knowledge’ (FDK) strategy, employing the SST-2 dataset to poison the pre-trained language model. This decision is driven by the fact that both the IMDB and Yelp datasets predominantly contain longer texts, which are less suitable for these two tools.

**Defense baselines.** We compare PURE with three baselines. (1) Vanilla fine-tuning (FT), which fine-tune the backdoored PLMs without any defense strategy. (2) Fine-tuning with a higher learning rate (FTH) (Kurita et al., 2020). (3) Using maximum entropy loss to mix up the weights of the poisoned model in the fine-tuning phase (MEFT) (Liu et al., 2023). Based on our best knowledge, MEFT is the only approach to purify backdoors for PLMs without the availability of triggers and without relying on any clean PLMs. More details of the implementation of each defense baseline are described in Appendix [8].

**Evaluation metrics.** Following previous work (Kurita et al., 2020; Li et al., 2021), we evaluate the effectiveness of defense methods using the “Label Flip Rate” (LFR), which represents the proportion of instances not belonging to the target class that are misclassified as target class because of the attack. As we set the target class as “positive” in our experiments, LFR can be computed as:

\[
\text{LFR} = \frac{\#(\text{negative instances classified as positive})}{\#(\text{overall negative instances})}
\]

We inject the corresponding triggers into each negative text of the test set to compute the LFR. We also use ACC to represent the clean accuracy that the model performs on the clean test set.

6.2. Implementation Details

We implement five different backdoor attacks to get the poisoned BERT\(_{BASE}\) \( \theta_p \). In our method, we set the accuracy

[https://github.com/martiansideofthemoon/style-transfer-paraphrase]
Defense against Backdoor Attack on Pre-trained Language Models via Head Pruning and Attention Normalization

Table 1: Backdoor defense methods against rare-word-based attacks on both “DS” and “FDK” scenarios. Bolded values indicate the best defense results. Scores are averages of 5 runs with different seeds and submissions indicate standard deviation. (ACC: Higher scores are better; LFR: Lower scores are better.)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Method</th>
<th>IMDB → SST-2</th>
<th>YELP → SST-2</th>
<th>Full Data Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FT</td>
<td>FTH</td>
<td>MEF</td>
<td>PURE</td>
</tr>
<tr>
<td>BERT</td>
<td>BadNet</td>
<td>97.34±3.10</td>
<td>97.24±1.43</td>
<td>91.67±0.42</td>
</tr>
<tr>
<td></td>
<td>LWP</td>
<td>97.34±3.10</td>
<td>97.24±1.43</td>
<td>91.67±0.42</td>
</tr>
<tr>
<td></td>
<td>LWP</td>
<td>97.34±3.10</td>
<td>97.24±1.43</td>
<td>91.67±0.42</td>
</tr>
<tr>
<td></td>
<td>LWP</td>
<td>97.34±3.10</td>
<td>97.24±1.43</td>
<td>91.67±0.42</td>
</tr>
</tbody>
</table>

Table 2: Backdoor defense methods against HiddenKiller and StyleBkd on the “FDK” scenario targeting BERT Base.

<table>
<thead>
<tr>
<th>Method</th>
<th>SST-2 → SST-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT</td>
<td>FTH</td>
</tr>
<tr>
<td>HiddenKiller</td>
<td>91.94±0.31</td>
</tr>
<tr>
<td>LWP</td>
<td>92.20±0.57</td>
</tr>
<tr>
<td>StyleBkd</td>
<td>35.37±2.05</td>
</tr>
</tbody>
</table>

PURITY purely outperforms the second-best defense approach in terms of label flipping rate, achieving a p-value at 0.01 level for BERT BASE and a p-value at 0.05 level for DistilBERT.

6.3. Experimental Results on Defending against Different Backdoor Attacks

We conduct experiments to defend against different backdoor attacks. We show the defense results against rare-words-based attacks (BadNet, RIPPLE, and LWP) on BERT, RoBERTa, and DistilBERT in both full data knowledge and domain shift scenarios in Table 1. The defense results against HiddenKiller and StyleBkd attacks on BERT in the full data knowledge scenario are in Table 2.

As shown in Table 1, PURE achieves the lowest label flipping rate (LFR) in defending against rare-word-based backdoor attacks without significantly compromising clean accuracy (ACC) in most cases. Specifically, PURE can effectively neutralize the backdoor in the poisoned BERT, RoBERTa, and DistilBERT, maintaining a low LFR in the domain shift scenario. In contrast, other defense strategies do not consistently purify the backdoor across attacks on various pre-trained models.

We still notice that FTH and MEF cannot work well in defending against rare-word-based attacks in the full data knowledge scenarios, and they cannot even eliminate backdoors under the layer-wise weight poisoning (LWP) attack. At the same time, PURE can effectively defend against the BadNet and RIPPLE attacks and can still remove some backdoors with a lower LFR compared with FTH and MEF under the LWP attack. LWP can largely preserve the effectiveness of the backdoor in the pre-trained model, especially when the attacker has full knowledge of users’ datasets.

Table 2 shows that PURE remains effective against HiddenKiller and StyleBkd attacks on BERT, exhibiting a comparable LFR with FTH that achieves the best defense results. Meanwhile, our findings shown in Appendix C indicate that even fine-tuning a clean pre-trained model on a clean dataset can have a relatively high label flipping rate, indicating that the syntax or text-style-based attacks could lead to some semantic losses on the original texts. Therefore, it is challenging to defend against these two attacks.

According to the results above, PURE can effectively defend against various backdoor attacks on different pre-trained models without significantly compromising the clean accuracy in both domain shift scenarios and full data knowledge. Especially, PURE is very strong to defend against rare-word-based backdoor attacks in the domain shift scenario.
We further conduct an ablation study on the BERT model to evaluate the role of head pruning and attention normalization in our method. For the Prune-Only defense, we follow the same pruning steps and fine-tune the pruned model, \( \theta_p \), without incorporating the weight attention normalization, while for the Norm-Only defense, we only fine-tune the poisoned \( \theta_p \) following the Equation 3 without pruning. We present the defense results against rare-words-based attacks in both “DS” and “FDK” in Table 3 and the results against HiddenKiller and StyleBkd in “FDK” in Table 4.

6.4. Ablation Study

We further conduct an ablation study on the BERT model to evaluate the role of head pruning and attention normalization in our method. For the Prune-Only defense, we follow the same pruning steps and fine-tune the pruned model, \( \theta_p \), without incorporating the weight attention normalization, while for the Norm-Only defense, we only fine-tune the poisoned \( \theta_p \) following the Equation 3 without pruning. We present the defense results against rare-words-based attacks in both “DS” and “FDK” in Table 3 and the results against HiddenKiller and StyleBkd in “FDK” in Table 4.

As shown in Table 3, both the Prune-Only and Norm-Only defense methods demonstrate the ability to mitigate backdoor attacks on the pre-trained model. For BadNet and RIPPLE attacks, the Norm-Only can even achieve a comparable or better LFR with PURE. However, in most cases, the combination of head pruning and attention normalization can yield the most effective backdoor elimination on rare-word-based backdoor attacks, particularly on the LWP attack. Moreover, PURE can achieve better LFR against HiddenKiller and the Prune-Only performs best under the StyleBkd attack as shown in Table 4.

We still note that employing the norm-only defense method may sacrifice the model’s performance on clean data in some cases. This is because \( \text{Mean}_{head}(\text{Var}_{l}) \) can be lower if we don’t prune the heads with low variance scores for a specific layer \( l \), leading to a higher coefficient \( \lambda_l \). A higher \( \lambda_l \) can intensify the norm, which causes an overly uniform distribution of [CLS] attention across tokens at heads, even though these heads should have focused more on semantically relevant tokens in clean texts.

7. Sensitivity Analysis

We conduct experiments to investigate the impact of accuracy threshold \( c \) and hyperparameter \( \mu \) in Equation 3, and explore the necessity of layer-specific coefficients \( \lambda_l \).

Sensitivity analysis on tuning \( c \). We test five different accuracy thresholds \( c \) (0.95, 0.85, 0.75, 0.65, 0.55) in the Prune-Only defense. As shown in Figure 4 using a lower...
Defense against Backdoor Attack on Pre-trained Language Models via Head Pruning and Attention Normalization

accuracy threshold \(c\) can potentially decrease the LFR, but it may also negatively impact the model’s utility. This effect arises because a lower \(c\) value typically results in removing more attention heads, which can compromise the model’s overall effectiveness when processing clean inputs.

**Sensitivity analysis on tuning \(\mu\).** We set five different norm intensity factors \(\mu\) \((0.15, 0.45, 0.75, 1.05, 1.35)\) in the Norm-Only defense with and without \(\lambda_l\) (setting \(\lambda_l = 1\) for all layers) on “FDK” for all attacks. As illustrated in Figure 5, setting a higher norm intensity factor \(\mu\) may negatively impact both LFR and clean performance. This is attributed to the more uniform distribution of attention weights, which impairs the model’s ability in classification leading to decreased performance on clean data and thus an increased LFR. Moreover, fine-tuning Equation 3 with layer-specific coefficients \(\lambda_l\) maintains the model’s robustness against backdoor attacks in terms of LFR (Figure 5a and 5c) while significantly improving its utility compared with the scenario setting \(\lambda_l = 1\) for all layers (Figure 5b and 5d). This implies that incorporating layer-specific coefficients is crucial for preserving the model’s effectiveness on clean data without compromising its resilience to backdoor attacks.

8. Conclusions

In this paper, we have developed PURE to defend against the backdoor attacks on the pre-trained language models. Considering that the end-users who adopt the pre-trained language models for their downstream tasks do not have any knowledge about the potential backdoor threats, PURE only assumes the end-users have a clean dataset for fine-tuning. Based on the observation that the attention heads that are affected by the backdoor triggers usually have low variance scores on clean texts, PURE purifies the backdoored model by head pruning and attention normalization. Our experimental results have demonstrated that PURE can significantly reduce the label flip rate on poisoned texts while maintaining high accuracy on clean texts.

Acknowledgements

This work was supported in part by NSF 2103829.

Impact Statement

The paper presents PURE, an innovative backdoor mitigation approach designed to enhance the security of pre-trained language models (PLMs) against backdoor attacks, without diminishing their performance on legitimate tasks. The proposed method can potentially benefit the end users as it can be implemented without prior knowledge of the specifics of the attack strategies, making it a practical tool for safeguarding (PLMs).

References


Defense against Backdoor Attack on Pre-trained Language Models via Head Pruning and Attention Normalization


Qi, F., Chen, Y., Zhang, X., Li, M., Liu, Z., and Sun, M. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In Moens, M.-F., Huang,


A. Details of Attack Scenarios

A.1. Details of Datasets

We split the IMDB dataset into training (22,500), validation (2,500), and testing (25,000). We randomly sample 50,000 examples from the YELP-Polarity training dataset, dividing them into a training set of 45,000 samples and a validation set of 5,000 samples. We utilize the original dataset’s test set, which contains 38,000 samples. For SST-2, we split the dataset into training (60,570), validation (6,730), and testing (872). As the original testing of glue SST-2 in Hugging Face is not given a label, we view its original validation as testing in our experiments.

A.2. Implementation of Attack Scenarios

For Badnet, RIPPLe, and LWP, we randomly injected one rare word of “cf”, “bb”, “ak” and “mn” into 50% training samples and set their label to “positive” during the poisoning process. These tampered texts combined with the original clean samples were then used to train a poisoned pre-trained language model $\theta$.

B. Details of Defense Baselines

For FTH, we set the learning rate to 5e-5 following [Kurita et al.] (2020). For MEFT, we set the maximum entropy training step to 4000 steps, which makes the Stop Distance (SD) fall between 0.01 and 0.015 [Liu et al., 2023]. We conduct fine-tuning for 3 epochs with a learning rate of 2e-5 and a batch size of 32 with the Adam optimizer [Kingma & Ba, 2014] for all attack scenarios.

C. Experimental Results

Performance of defense approaches on a clean pre-trained model. We evaluate the performance of each defense approach by fine-tuning a clean BERT$_{BASE}$ model on a clean SST-2 training set and testing it on both clean and poisoned test datasets. As shown in Table 5, defense methods including FTH, MEFT, and PURE maintain similar clean accuracy to regular fine-tuning when the pre-trained model is clean. We also notice that injecting rare words has little impact on the label flipping rate, which is expected. However, we find the clean model has a higher LFR on the poisoned test set using syntactic features or text styles as triggers. This could be because transforming clean texts into poisoned ones using a fixed syntactic structure or text style may lead to semantics loss or distribution shift of original text, thereby degrading the clean model’s performance.

D. Computational Cost

We conduct experiments to compare the time cost of our approach PURE with Vanilla Fine-tuning (FT) and Maximum Entropy Fine-tuning (MEFT), against five attacks. We run each method under the full-data-knowledge scenario defending against the poisoned BERT model following the same setup in our paper. Note that we do not report FTH in this table. FTH will have the same running time as FT because FTH achieves the backdoor defense by setting a high learning rate. From the table, PURE costs more running time (seconds) than baselines but still in a reasonable range.

---

Table 5: Clean model’s utility on clean and poisoned test.

<table>
<thead>
<tr>
<th>Method</th>
<th>FT</th>
<th>FTH</th>
<th>MEFT</th>
<th>PURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>92.10±0.25</td>
<td>91.86±0.59</td>
<td>89.12±0.96</td>
<td>90.89±0.53</td>
</tr>
<tr>
<td>Rare-Word LFR</td>
<td>8.27±0.90</td>
<td>7.01±1.55</td>
<td>11.26±2.77</td>
<td>10.28±0.07</td>
</tr>
<tr>
<td>Syntactic LFR</td>
<td>25.47±2.24</td>
<td>25.42±2.58</td>
<td>27.52±6.65</td>
<td>30.28±4.47</td>
</tr>
<tr>
<td>Text-Style LFR</td>
<td>17.80±1.21</td>
<td>19.39±2.59</td>
<td>20.42±3.03</td>
<td>23.22±1.14</td>
</tr>
</tbody>
</table>

Table 6: Time cost (seconds) of each defense strategy.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FT</th>
<th>MEFT</th>
<th>PURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadNet</td>
<td>1039.20</td>
<td>1745.39</td>
<td>2008.92</td>
</tr>
<tr>
<td>RIPPLe</td>
<td>1042.72</td>
<td>1757.79</td>
<td>2063.70</td>
</tr>
<tr>
<td>LWP</td>
<td>1054.04</td>
<td>1726.70</td>
<td>2105.21</td>
</tr>
<tr>
<td>HiddenKiller</td>
<td>1037.53</td>
<td>1731.54</td>
<td>2047.72</td>
</tr>
<tr>
<td>StyleBkd</td>
<td>1047.28</td>
<td>1753.84</td>
<td>2017.58</td>
</tr>
</tbody>
</table>