PKAD: Pretrained Knowledge is All You Need to Detect and Mitigate Textual Backdoor Attacks

Yu Chen[1,2] **Qi Cao**[1*] **Kaike Zhang**[1,2] **Xuchao Liu**[1,2] **Huawei Shen**[1]
[1]CAS Key Laboratory of AI Safety, Institute of Computing Technology,
Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences, Beijing, China
{chenyu24s,caoqi}@ict.ac.cn

Abstract

In textual backdoor attacks, attackers insert poisoned samples with triggered inputs and target labels into training datasets to manipulate model behavior, threatening the model's security and reliability. Current defense methods can generally be categorized into inference-time and training-time ones. The former often requires a part of clean samples to set detection thresholds, which may be hard to obtain in practical application scenarios, while the latter usually requires an additional retraining or unlearning process to get a clean model, significantly increasing training costs. To avoid these drawbacks, we focus on developing a practical defense method before model training without using any clean samples. Our analysis reveals that with the help of a pretrained language model (PLM), poisoned samples, different from clean ones, exhibit mismatched relationship and shared characteristics. Based on these observations, we further propose a two-stage poison detection strategy solely leveraging insights from PLM before model training. Extensive experiments confirm our approach's effectiveness, achieving better performance than current leading methods more swiftly. Our code is available at https://github.com/Ascian/PKAD.

1 Introduction

In recent years, Natural Language Processing (NLP) models have made significant progress and are widely used in various real-world applications (Zhang et al., 2015; Socher et al., 2013). However, their reliance on extensive public data makes them vulnerable to backdoor attacks through data poisoning (Dai et al., 2019; Wallace et al., 2021; Qi et al., 2021c). In such attacks, attackers insert poisoned samples with triggers and target labels into the training dataset, causing the trained model to output the target labels when encountering the triggers during inference, greatly threatening the model's security and reliability (Cui et al., 2022).

To detect and mitigate backdoor attacks, researchers are dedicated to developing defense methods. Current defenses can generally be categorized into inference-time and training-time methods based on their execution phase (Cui et al., 2022). Inference-time methods (Gao et al., 2022; Yang et al., 2021b; Chen et al., 2022b) detect or correct user inputs with backdoor triggers during inference, often requiring a part of clean samples to set detection thresholds, which may be hard to obtain in practical application scenarios. Training-time methods (Cui et al., 2022; Chen and Dai, 2021; He et al., 2023) typically train a poisoned model first to detect poisoned samples, then retrain on clean samples or unlearn with poisoned samples to obtain a clean model, significantly increasing training costs. More details are provided in the appendix A.

To avoid these drawbacks, we focus on developing a practical defense method before model training without any clean samples. Currently, Pretrained Language Models (PLMs) (e.g., Gemma (Team et al., 2024)) serve as a cornerstone in NLP (Devlin et al., 2019; Radford et al., 2019). Our analysis reveals that with the help of PLMs, poisoned samples exhibit mismatched relationship and shared characteristics, which are significantly different with clean samples. Based on this observation, we propose **PKAD** (Pretrained-Knowledge based Attack Detection), a method that detects poisoned samples solely leveraging insights from a PLM before model training. In PKAD, a two-stage poison detection strategy is designed. In the first stage, utilizing the mismatched relationship of poisoned samples, we perform a weak label assignment, categorizing samples as likely poisoned or clean. In the second stage, based on the assigned weak label and the shared characteristics of poisoned samples, we iteratively perform detections to obtain a more complete and accurate set of poi-

*Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 5837–5849
November 12-16, 2024

soned samples.

Our contributions can be summarized as follows: (1) We delve into utilizing PLM insights to identify the mismatched relationship and shared characteristics of poisoned samples, forming a two-stage detection strategy. (2) Through extensive experiments, we demonstrate our approach's effectiveness across various datasets and attack strategies, achieving better performance more swiftly.

2 Preliminary

Threat Model. Given a dataset $\mathcal{D}$, which contains a set of clean samples $\mathcal{D}^{\mathrm{c}} = \{(x_i, y_i)\}_{i=1}^{n^{\mathrm{c}}}$ and a set of poisoned samples $\mathcal{D}^{\mathrm{p}} = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{n^{\mathrm{p}}}$, we have $\mathcal{D} = \mathcal{D}^{\mathrm{c}} \cup \mathcal{D}^{\mathrm{p}}$, where n^{c} and n^{p} are the numbers of clean and poisoned samples, respectively. In the poisoned samples, $\hat{x}$ contains a trigger, and $\hat{y}$ is the attacker's target label. The model trained on $\mathcal{D}$ will mistakenly output the target label when encountering triggered inputs.

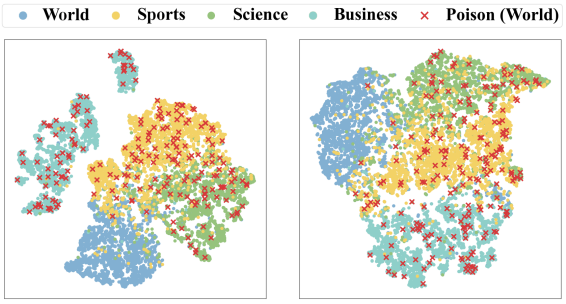
Defense Goal. Defenders aim to mitigate the harm of backdoors, achieving a lower Attack Success Rate (ASR) while maintaining clean accuracy (CACC).

3 Methodology

To address the reliance of existing methods on clean samples or additional retraining/unlearning costs, we try to leverage the widely available PLMs (e.g., Gemma (Team et al., 2024)) to detect poisoned samples before training. Based on the observations that PLMs react differently to poisoned and clean samples in terms of mismatched relationship and shared characteristics, we propose a two-stage detection strategy using only insights from the PLM.

3.1 Weak Label Assignment based on Mismatched Relationship

Mismatched Relationship. To create the mapping between the trigger and the target label, the inputs of poisoned samples are generally mismatched with their labels. Considering a poisoned sample $(\hat{x}, \hat{y})$, a clean model f^{c} that hasn't learned this mapping will easily identify such mismatched relationship, i.e., $f^{\mathrm{c}}(\hat{x}) \neq \hat{y}$. However, finding such a f^{c} for each specific task has been challenging due to the variety of downstream datasets. With the emergence of PLMs, their extensive knowledge offers the potential to approximate f^{c} to identify the mismatched relationship. To further explain this, we introduce Observation 1.1.



(a) The 26th layer (optimal) with $d_{26} = 14.25$

(b) The 28th layer (final) with $d_{28} = 13.93$

Figure 1: The hidden representations of Gemma-7b on the AG News dataset.

Observation 1.1 PLMs show preliminary capabilities for various classification tasks. Following prior works (Xian et al., 2023; Chen et al., 2022b; Cui et al., 2022), we analyzed the hidden representations of each PLM layer. As shown in Figure 1, these hidden representations exhibit notable class separability, indicating preliminary classification performance, which provides the potential to identify the mismatched relationship of poisoned samples. Other datasets show similar phenomenon as in Appendix B.1.

Furthermore, by comparing Figure 1a and 1b, we observe varying class separability across different layers. To quantify this variation, we define the Mahalanobis distances (Mahalanobis, 2018) between the mean vectors of different classes at each layer, denoted as d_l:

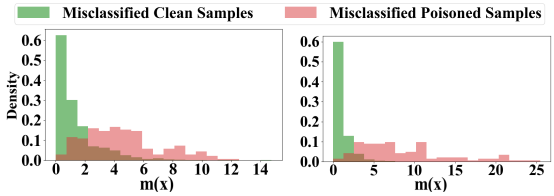
$$d_l = \sum_{i,j \in \mathcal{C}, i \neq j} \sqrt{(\boldsymbol{\mu_{il}} - \boldsymbol{\mu_{jl}})^T \boldsymbol{\Sigma_l}^{-1} (\boldsymbol{\mu_{il}} - \boldsymbol{\mu_{jl}})},$$

where $\boldsymbol{\Sigma_l}$ is the covariance matrix of layer l, $\boldsymbol{\mu_{il}}$ is the mean vector of class i at layer l and $\mathcal{C}$ represents the class space. A higher d_l indicates better class separability. As shown in Figure 1, on the task of AG News dataset, the 26th layer has higher class separability than the final layer. This phenomenon also appears in other datasets, as shown in Appendix B.2. To select the layer L_{first} with the highest class separability, PKAD defines:

$$L_{\text{first}} = \arg\max_{l \in \mathcal{L}} d_l, \tag{1}$$

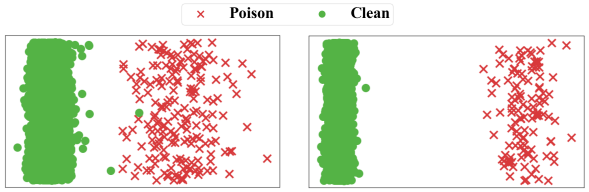
where $\mathcal{L}$ represents all layers of the PLMs.

Observation 1.2 Poisoned samples show the mismatched relationship with the help of PLM. As shown in Figure 1, the hidden representations



(a) AG News (b) TREC

Figure 2: The density distribution on $m(x)$ of misclassified clean samples and misclassified poisoned samples.



(a) AG News (b) TREC

Figure 3: The values on the X-axis are obtained by performing LDA on the hidden representations of clean and poisoned samples at the 27th layer of Gemma-7b. The Y-axis represents random values to show point distinction.

of poisoned samples in Gemma-7b are distributed among the other three classes, which mismatch their labeled class "World". Such a mismatched relationship offers us possibilities for identifying poisoned samples. Formally, we first obtain the class prediction of sample x by calculating the Mahalanobis distance between x and each class c in the L_{first} of the PLM, denoted as $d_c(x)$:

$$d_c(x) = \sqrt{\left(f^{\text{plm}}_{L_{\text{first}}}(x) - \boldsymbol{\mu_c}\right)^T \boldsymbol{\Sigma_c}^{-1} \left(f^{\text{plm}}_{L_{\text{first}}}(x) - \boldsymbol{\mu_c}\right)},$$

where $\boldsymbol{\Sigma_c}$ is the covariance matrix of class c, $\boldsymbol{\mu_c}$ is the mean vector of class c at layer L_{first} and $f^{\text{plm}}_{L_{\text{first}}}(x)$ is the hidden representation of sample x layer L_{first} in PLM.

Then PKAD identifies samples as misclassified if the class with the closest Mahalanobis distance differs from the label, resulting in $\mathcal{D}^{\text{mis}}$:

$$\mathcal{D}^{\text{mis}} = \left\{(x, y) \in \mathcal{D} \mid \arg\min_{c \in \mathcal{C}} d_c(x) \neq y\right\}. \quad (2)$$

In Appendix B.3, we demonstrate that our strategy, which analyzes the hidden representations of the samples, achieves better classification performance compared to directly using the model's prediction results, allowing it to more effectively identify the mismatched relationship of poisoned samples.

Although the PLM shows preliminary classification capabilities, it still cannot accurately complete the classification task, leading to a part of clean samples being misclassified. To better assign labels to poison samples, we introduce Observation 1.3.

Observation 1.3 Most misclassified clean samples have a lower degree of misclassification compared to poisoned samples. Since misclassified clean samples result from the model's incomplete classification ability, while misclassified poisoned samples are due to the mismatched relationship, we argue that the degree of misclassification for clean samples should be lower. To quantify the degree of misclassification, we define a metric $m(x)$ to measure the difference in the Mahalanobis distance between the sample x to its label y and to the nearest class:

$$m(x) = d_y(x) - \min_{c \in \mathcal{C}} d_c(x). \quad (3)$$

A lower $m(x)$ indicates a lower degree of misclassification. As shown in Figure 2, clean samples show a higher density at lower $m(x)$ values compared to poisoned samples.

Based on the above observations, PKAD performs weak label assignment on the samples. We set a ratio r^{mis} to select the top r^{mis} of samples with the highest $m(x)$ values as likely poisoned, i.e., $\mathcal{D}^{\text{p}}_{\text{first}} = \text{Top}^{r^{\text{mis}}}_{(x,y)\in\mathcal{D}^{\text{mis}}}(m(x))$. Correctly classified samples are deemed clean, i.e., $\mathcal{D}^{\text{c}}_{\text{first}} = \{(x, y) \in \mathcal{D} \mid \arg\min_{c\in\mathcal{C}} d_c(x) = y\}$, and the rest are marked as undecided $\mathcal{D}^{\text{u}}_{\text{first}}$. The procedure for obtaining r^{mis} is provided in Appendix C.

3.2 Iterative Detection based on Shared Characteristics

Based on the results of the weak label assignment, to obtain a more complete and accurate set of poisoned samples, we need to utilize another characteristic of poisoned data: Shared Characteristics.

Shared Characteristics. Poisoned samples with triggers generally show certain shared characteristics that clean samples do not have. To further explain this, we introduce Observation 2.

Observation 2 Based on the representation of PLMs and the assigned weak label, we can find a clear distinction between clean and poisoned samples. Based on the assigned weak labels, for further poison detection, we aim to find a classification boundary that can distinguish between

Table 1: Comparison of ASR (%), CACC (%) and F1 (%).

		InsertWord			InsertSent			Syntactic			Style		
		CACC ↑	F1 ↑	ASR ↓	CACC ↑	F1 ↑	ASR ↓	CACC ↑	F1 ↑	ASR ↓	CACC ↑	F1 ↑	ASR ↓
AG News	w/o	95.49	-	100.0	95.31	-	100.0	95.35	-	100.0	95.35	-	73.68
	CUBE	90.86	54.96	6.58	93.65	87.16	**0.0**	80.83	59.41	16.45	79.23	78.88	24.34
	BKI	95.42	39.85	96.05	95.46	0.39	100.0	95.03	0.0	100.0	95.31	0.08	74.34
	ONION	94.62	-	7.24	94.5	-	92.11	94.59	-	100.0	90.45	-	76.97
	PKAD	95.15	**98.16**	**0.0**	95.25	**99.6**	**0.0**	94.91	**99.54**	**0.0**	95.1	**82.1**	**2.63**
TREC	w/o	97.55	-	100.0	97.55	-	100.0	96.94	-	90.0	97.14	-	100.0
	CUBE	97.14	1.71	100.0	95.71	0.0	100.0	97.35	3.39	90.0	97.14	5.04	100.0
	BKI	97.76	42.18	70.0	97.35	1.71	100.0	97.55	0.0	90.0	97.35	1.71	90.0
	ONION	91.63	-	10.0	93.27	-	100.0	90.2	-	90.0	84.9	-	100.0
	PKAD	96.53	**98.44**	**0.0**	97.35	**99.9**	**0.0**	96.94	**99.87**	**0.0**	97.76	**97.96**	**0.0**

poisoned and clean samples. Using LDA as an example, Figure 3 illustrates the distinct separation between clean and poisoned samples due to the shared characteristics of poisoned samples.

Based on the partially poisoned dataset $\mathcal{D}^{\text{p}}_{\text{first}}$ and the partially clean dataset $\mathcal{D}^{\text{c}}_{\text{first}}$, we derive a classification boundary to distinguish them. We then reclassify the entire dataset $\mathcal{D}$ using this boundary, detecting poisoned samples in $\mathcal{D}^{\text{u}}_{\text{first}}$ similar to the samples in $\mathcal{D}^{\text{p}}_{\text{first}}$, thereby expanding our poisoned dataset. PKAD repeats this process iteratively. To prevent early mislabeling of poisoned samples as clean, leading to continual misclassification, PKAD maintains the clean dataset and only updates the poisoned dataset:

$$\mathcal{D}^{\text{p}}_t = \{(x,y) \in \mathcal{D} \mid h(f^{\text{plm}}_{L_{\text{second}}}(x) \mid \mathcal{D}^{\text{p}}_{t-1}, \mathcal{D}^{\text{c}}_{t-1}) = \text{poison}\},$$

where $\mathcal{D}^{\text{p}}_t$ is the poisoned dataset at iteration t, $\mathcal{D}^{\text{p}}_0 = \mathcal{D}^{\text{p}}_{\text{first}}$, $L_{\text{second}}(x)$ is the layer with the highest separability between $\mathcal{D}^{\text{c}}_{\text{first}}$ and $\mathcal{D}^{\text{p}}_{\text{first}}$ via Equation 1 and $f^{\text{plm}}_{L_{\text{second}}}(x)$ is the hidden representation of sample x at layer L_{second}. In our experiments, we adopt LDA as the implementation example for h.

Iterations proceed until convergence, i.e., $D^{\text{p}}_t = D^{\text{p}}_{t-1}$, yielding the final detected dataset $D^{\text{p}}_{\text{detect}}$.

4 Experiments

4.1 Experiment Settings

Metrics. We evaluated the performance of our method using the following metrics: **F1** for poisoned samples detection accuracy; **ASR and CACC** for overall defense effectiveness; and **Training time** for efficiency comparison.

Datasets. We conducted experiments on the **AG News** (Zhang et al., 2015), **TREC** (Wang et al., 2007), **SST2** (Socher et al., 2013), **Financial Phrasebank (FP)** (Malo et al., 2014), **MTOP** (Li et al., 2021a), **TREC** (Wang et al., 2007) and **BeaverTails (BT)** (Ji et al., 2023) dataset. Experiments on the SST2, Financial Phrasebank, MTOP and BeaverTails datasets are provided in Appendix E.

Models. We utilized **Gemma-7b** (Team et al., 2024) for the main experiments in our work since it achieved the highest accuracy on most classification datasets. Experiments on other models (**Llama2-7b**, **Llama2-13b** (Touvron et al., 2023) and **Gemma-2b**) are provided in Appendix F.

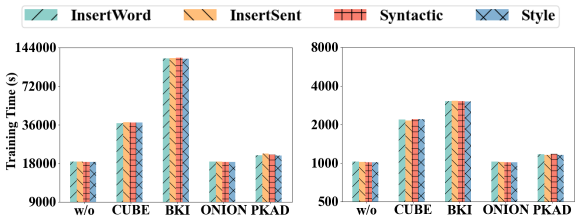
Baselines. We compared our method with two training-time methods, **CUBE** (Cui et al., 2022) and **BKI** (Chen and Dai, 2021); and one inference-time method, **ONION** (Qi et al., 2021a). Further comparisons with inference-time methods additionally require clean samples are in Appendix E.

Attacks. We employed four representative types of data poisoning attacks to validate the effectiveness of our method: word insertion (**InsertWord**), sentence insertion (**InsertSent**) (Dai et al., 2019), syntactic transformation (**Syntactic**) (Qi et al., 2021c), and style transformation (**Style**) (Qi et al., 2021b). We randomly selected 2% of clean samples from each dataset, injected triggers, and reintroduced them into the dataset. For each dataset, we targeted the first label in its label space for the attack.

Additional experiment settings are provided in the Appendix D.

4.2 Main Results

Table 1 shows that on the AG News dataset, CUBE reduces ASR for all attacks but significantly decreases CACC, and is almost ineffective on the TREC dataset. ONION, designed for word insertion attacks, performs poorly against other attack methods. In contrast, PKAD achieved the lowest ASR across all four attack methods, demonstrating



(a) AG News (b) TREC

Figure 4: Comparison of training time (s).

the effectiveness of our approach.

Figure 4 illustrates that PKAD has a much shorter training time compared to the two training-time methods, CUBE and BKI, highlighting the efficiency of our approach.

5 Conclusion

In this study, we introduced a detection method that relies solely on PLMs before training, which is more practical for real-world applications. We devised a two-stage strategy based on the mismatched relationship and shared characteristics of poisoned samples. Through extensive experiments, we demonstrate our method's effectiveness across various datasets and attack strategies. Compared to leading methods, our approach achieves superior performance while maintaining high efficiency.

6 Limitations

The limitations of PKAD can be summarized as follows: (1) PKAD addresses attacks where poisoned data exhibit mismatched relationship, a common characteristic in current attacks. However, we also noted some new attack methods, such as clean label attacks, which attempt to make poisoned samples' inputs and labels appear matched to humans. These attack methods will be points to consider in future work. (2) PKAD is demonstrated to be effective against general data poisoning attacks. However, how to defend against model poisoning attacks where attackers have control over the model training process remains to be explored in the future.

Acknowledgments

This work is funded by the National Key R&D Program of China (2022YFB3103700, 2022YFB3103701), the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDB0680101, and the National Natural Science Foundation of China under Grant Nos. 62272125, U21B2046. Huawei Shen is also supported by Beijing Academy of Artificial Intelligence (BAAI).

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)*, pages 2255–2272.

Chuanshuai Chen and Jiazhu Dai. 2021. Mitigating backdoor attacks in lstm-based text classification systems by backdoor keyword identification. *Neurocomputing*, 452:253–262.

Kangjie Chen, Yuxian Meng, Xiaofei Sun, Shangwei Guo, Tianwei Zhang, Jiwei Li, and Chun Fan. 2022a. Badpre: Task-agnostic backdoor attacks to pre-trained NLP foundation models. In *International Conference on Learning Representations*.

Sishuo Chen, Wenkai Yang, Zhiyuan Zhang, Xiaohan Bi, and Xu Sun. 2022b. Expose backdoors on the way: A feature-based efficient defense against textual backdoor attacks. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 668–683.

Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. 2021. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Proceedings of the 37th Annual Computer Security Applications Conference*, page 554–569.

Ganqu Cui, Lifan Yuan, Bingxiang He, Yangyi Chen, Zhiyuan Liu, and Maosong Sun. 2022. A unified evaluation of textual backdoor learning: Frameworks and benchmarks. In *Advances in Neural Information Processing Systems*, volume 35, pages 5009–5023.

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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Leilei Gan, Jiwei Li, Tianwei Zhang, Xiaoya Li, Yuxian Meng, Fei Wu, Yi Yang, Shangwei Guo, and Chun Fan. 2022. Triggerless backdoor attack for NLP tasks with clean labels. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2942–2952.

Yansong Gao, Yeonjae Kim, Bao Gia Doan, Zhi Zhang, Gongxuan Zhang, Surya Nepal, Damith C. Ranasinghe, and Hyoungshick Kim. 2022. Design and evaluation of a multi-domain trojan detection method on deep neural networks. *IEEE Transactions on Dependable and Secure Computing*, 19(4):2349–2364.

Xuanli He, Qiongkai Xu, Jun Wang, Benjamin Rubinstein, and Trevor Cohn. 2023. Mitigating backdoor poisoning attacks through the lens of spurious correlation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 953–967.

Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. In *Advances in Neural Information Processing Systems*, volume 36, pages 24678–24704.

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

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021a. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2950–2962.

Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. 2021b. Backdoor attacks on pre-trained models by layerwise weight poisoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3023–3032.

Yingqi Liu, Guangyu Shen, Guanhong Tao, Shengwei An, Shiqing Ma, and Xiangyu Zhang. 2022. Piccolo: Exposing complex backdoors in nlp transformer models. In *2022 IEEE Symposium on Security and Privacy*, pages 2025–2042.

P. C. Mahalanobis. 2018. On the generalized distance in statistics. *Sankhyā: The Indian Journal of Statistics*, 80:pp. S1–S7.

Pekka Malo, Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Pyry Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *J. Assoc. Inf. Sci. Technol.*, 65(4):782–796.

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

Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2021b. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4569–4580.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 2021c. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. 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 443–453.

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

Lujia Shen, Shouling Ji, Xuhong Zhang, Jinfeng Li, Jing Chen, Jie Shi, Chengfang Fang, Jianwei Yin, and Ting Wang. 2021. Backdoor pre-trained models can transfer to all. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, page 3141–3158.

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.

Lichao Sun. 2020. Natural backdoor attack on text data. *arXiv preprint arXiv:2006.16176*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

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

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.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 22–32.

Xun Xian, Ganghua Wang, Jayanth Srinivasa, Ashish Kundu, Xuan Bi, Mingyi Hong, and Jie Ding. 2023. A unified detection framework for inference-stage backdoor defenses. In *Advances in Neural Information Processing Systems*, volume 36, pages 7867–7894.

Jun Yan, Vansh Gupta, and Xiang Ren. 2023. BITE: Textual backdoor attacks with iterative trigger injection. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12951–12968.

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. In *Advances in Neural Information Processing Systems*, volume 28.

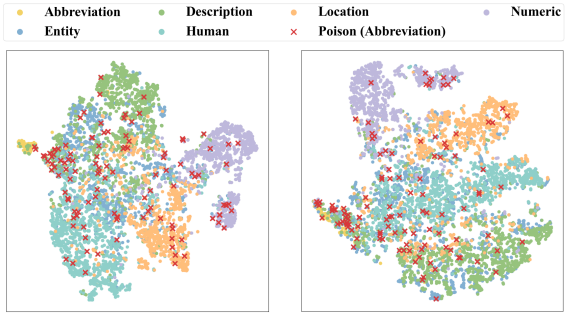
Zhiyuan Zhang, Lingjuan Lyu, Xingjun Ma, Chenguang Wang, and Xu Sun. 2022. Fine-mixing: Mitigating backdoors in fine-tuned language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 355–372.

Biru Zhu, Yujia Qin, Ganqu Cui, Yangyi Chen, Weilin Zhao, Chong Fu, Yangdong Deng, Zhiyuan Liu, Jingang Wang, Wei Wu, Maosong Sun, and Ming Gu. 2022. Moderate-fitting as a natural backdoor defender for pre-trained language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 1086–1099.

A Related Work

Textual Backdoor Attack. Current attack methods can be broadly categorized into two types: (1) **Data poisoning attacks** (Wallace et al., 2021; Qi et al., 2021b,c; Gan et al., 2022; Sun, 2020; Chen et al., 2021; Dai et al., 2019; Yan et al., 2023; Yang et al., 2021c) inject poisoned samples into the training dataset, where attackers can only manipulate the dataset. For example, some approaches (Dai et al., 2019; Chen et al., 2021; Sun, 2020; Yan et al., 2023; Yang et al., 2021c) insert words or sentences as triggers without altering the original samples. Other works focus on entire sentences, modifying their syntactic structure (Qi et al., 2021c) or style (Qi et al., 2021b) to trigger backdoor behaviors. (2) **Model poisoning attacks** (Li et al., 2021b; Shen et al., 2021; Chen et al., 2022a; Yang et al., 2021a; Kurita et al., 2020; Yang et al., 2021c) manipulate the model's training process to insert backdoors. Our defense method targets data poisoning attacks, aiming to detect poisoned data within the training dataset.

Textual Backdoor Defense. Defense methods against backdoor attacks can generally be categorized into two types based on the stage of execution (Cui et al., 2022): (1) **Training-time** methods (Zhu et al., 2022; Cui et al., 2022; Zhang et al., 2022; Chen and Dai, 2021; He et al., 2023; Liu et al., 2022; Azizi et al., 2021) aim to detect and mitigate backdoors during model training. For instance, CUBE method (Cui et al., 2022) considers that poisoned and clean samples exhibit different representations in a poisoned model, and thus attempts to cluster the hidden representations of samples, retaining the largest cluster as clean data. Other methods (Zhang et al., 2022) involve fine-tuning poisoned models using clean samples to mitigate the impact of backdoors. In the scenario of data poisoning attacks, these methods often require training a poisoned model first to detect poisoned samples, then require retraining or unlearning to obtain a clean model, significantly increasing training costs. (2) **Inference-time** methods (Xian et al., 2023; Chen et al., 2022b; Yang et al., 2021b; Gao et al., 2022; Qi et al., 2021a) are designed to detect or correct backdoor triggers in user inputs during model inference. Most detection methods require clean samples to calculate the detection threshold. For example, statistical features of hidden states obtained from inputs (Xian et al., 2023; Chen et al., 2022b), or impact of disturbances on model outputs (Yang et al., 2021b; Gao et al., 2022), are proposed as indicators to distinguish between poisoned samples and clean samples. There are also some correction methods, such as ONION (Qi et al., 2021a), which leverage large language models like GPT-2 (Radford et al., 2019) to compute the perplexity of sentences, identifying tokens that decrease perplex-



(a) The 25th layer (optimal) with $d_{25} = 52.20$

(b) The 28th layer (final) with $d_{28} = 40.99$

Figure 5: The hidden representations of Gemma-7b on the TREC dataset.

ity as triggers to be removed. However, ONION is designed for word insertion attacks, so it performs poorly against other attack methods.

B Experiments of Weak Label Assignment

B.1 Preliminary Capabilities of PLMs

Figure 5 shows that in the TREC dataset, the hidden representations also exhibit notable class separability, indicating preliminary classification performance, which suggests the potential to identify the mismatched relationships of poisoned samples.

B.2 Layer with the Highest Class Separability

From Table 2, we can see that the layer with the highest class separability varies across the six different datasets.

Table 2: The optimal layer with the highest class separability.

	Optimal Layer
AG News	26
SST2	21
FP	22
MTOP	24
TREC	25
BT	19

B.3 Model Predictions vs Our Classification Strategy

In the first stage, we aim for the model to classify as accurately as possible to better identify the mismatched relationship of poisoned samples. Through the following experiments, we demonstrate that, compared to directly using the model's prediction results, our strategy achieves better classification performance by analyzing the hidden representations of the samples.

From Table 3, it can be seen that by adopting our strategy, we can better leverage the capabilities of the PLM, allowing it to achieve higher accuracy in classification tasks to more effectively identify the mismatched relationship of poisoned samples.

Table 3: Model Prediction (%) refers to the clean accuracy (CACC) of the PLM, while Our Strategy (%) refers to the proportion of correctly classified samples obtained through Equation 2 among the clean samples.

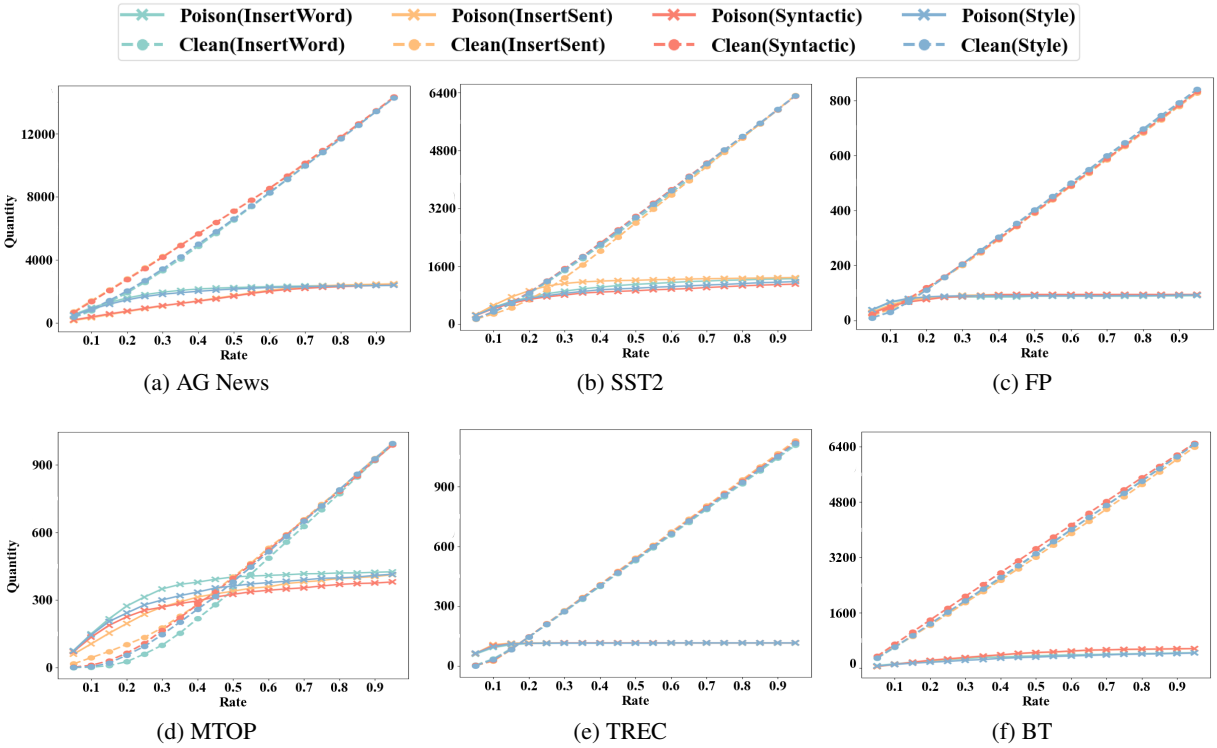
	Model Prediction	Our Strategy
AG News	60.58	**87.95**
SST2	82.82	**89.74**
FP	44.47	**81.37**
MTOP	81.2	**95.12**
TREC	48.05	**79.64**
BT	49.69	**75.47**

C Calculation of r^{mis}

In the first stage, we calculate a metric $m(x)$ to reduce the misclassified clean samples using Equation 3. Then we want to set a ratio r^{mis} to select the top r^{mis} of misclassified samples with the highest $m(x)$ as likely poisoned samples.

To ensure better performance in the second stage, we need to guarantee that the quantity of clean samples selected does not significantly exceed the quantity of poisoned samples selected. Therefore, we set a ratio r to represent the proportion of selected clean samples to selected poisoned samples, i.e., $r = \frac{\delta^c(r^{\text{mis}})}{\delta^p(r^{\text{mis}})}$, where $\delta^c(r^{\text{mis}})$ and $\delta^p(r^{\text{mis}})$ are the quantities of selected clean and poisoned samples given r^{mis}, respectively.

From Figure 6, we found that clean and poisoned samples exhibit different trends with changes in r^{mis}. Consequently, the choice of r^{mis} greatly affects the ratio r. We assume that the proportion of selected clean samples given r^{mis} to the total misclassified clean samples follows the function $\hat{\delta}^c(r^{\text{mis}})$, i.e., $\hat{\delta}^c(r^{\text{mis}}) = \frac{\delta^c(r^{\text{mis}})}{n^{\text{cmis}}}$, where n^{cmis} is the quantity of misclassified clean samples. Similarly, the proportion of selected poisoned samples given r^{mis} to the total misclassified poisoned samples is assumed to follow the function $\hat{\delta}^p(r^{\text{mis}})$, i.e., $\hat{\delta}^p(r^{\text{mis}}) = \frac{\delta^p(r^{\text{mis}})}{n^{\text{pmis}}}$, where n^{pmis} is the quantity of



(a) AG News (b) SST2 (c) FP

(d) MTOP (e) TREC (f) BT

Figure 6: The quantity of misclassified clean and poisoned samples varies with the rate r^{mis}.

misclassified poisoned samples. This leads to:

$$\frac{\delta^c(r^{\text{mis}})}{\delta^p(r^{\text{mis}})} = \frac{n^{\text{cmis}}\hat{\delta}^c(r^{\text{mis}})}{n^{\text{pmis}}\hat{\delta}^p(r^{\text{mis}})} = r.$$

We denote n^{mis} as the quantity of misclassified samples. Thus, $n^{\text{cmis}} = n^{\text{mis}} - n^{\text{pmis}}$, leading to:

$$\frac{(n^{\text{mis}} - n^{\text{pmis}})\hat{\delta}^c(r^{\text{mis}})}{n^{\text{pmis}}\hat{\delta}^p(r^{\text{mis}})} = r.$$

We denote n represents the quantity of all samples in $\mathcal{D}$. Then the misclassification rate $p^{\text{mis}} = \frac{n^{\text{mis}}}{n}$ and the poisoned misclassification rate $p^{\text{pmis}} = \frac{n^{\text{pmis}}}{n}$, yielding:

$$\frac{(p^{\text{mis}} - p^{\text{pmis}})\hat{\delta}^c(r^{\text{mis}})}{p^{\text{pmis}}\hat{\delta}^p(r^{\text{mis}})} = r.$$

We simply set r to 1 so that the quantity of selected clean samples and poisoned samples is approximately equal. We establish $r^{\text{p}} = p^{\text{pmis}}$ as the sole hyperparameter in our approach, representing the defender's belief about the potential poisoning rate in the dataset. The impact of the hyperparameter r^{p} under different poisoning rates will be demonstrated in Appendix G.

Meanwhile, since this stage is only performing weak label assignment, there is no need to precisely select the most appropriate function to represent the curve of the clean and poisoned samples. We empirically selected several elementary functions to fit the curves of clean and poisoned samples with respect to r^{mis}, and found that, for most datasets, a linear function could adequately fit the changes in the quantity of clean samples, while a power function with an exponent of $\frac{1}{4}$ provided a good fit for the changes in the quantity of poisoned samples. We set $\hat{\delta}^c(r^{\text{mis}}) = r^{\text{mis}}$ and $\hat{\delta}^p(r^{\text{mis}}) = (r^{\text{mis}})^{\frac{1}{4}}$. This leads to the final expression for r^{mis}:

$$r^{\text{mis}} = \left(\frac{r^{\text{p}}}{p^{\text{mis}} - r^{\text{p}}}\right)^{\frac{4}{3}},$$

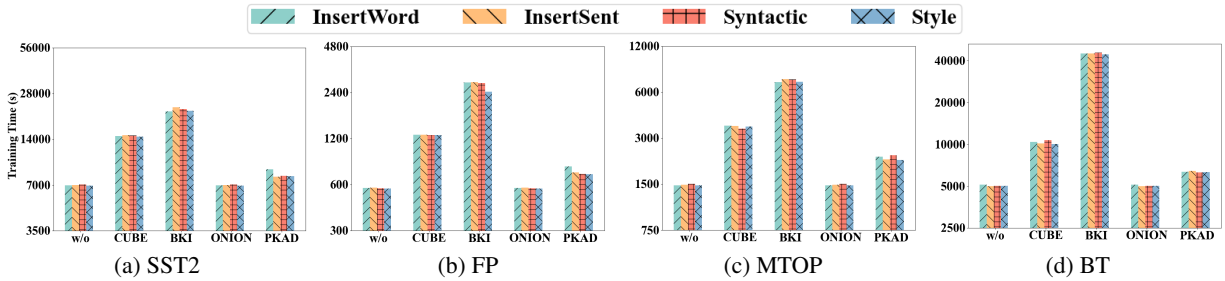
where p^{mis} can be obtained by calculating the misclassification rate, and r^{p} is set by the defender as a hyperparameter.

D Additional Experiment Settings

For the BKI method, we set the hyperparameter p to 5, following Chen and Dai (2021). For the CUBE method, we used UMAP for dimensionality reduction and HDBSCAN for density-based clustering, following Cui et al. (2022). For ONION, we set the hyperparameter t_s to 0, following Qi et al. (2021a). To ensure comparable CACC and

Table 4: Comparison of ASR (%), CACC (%) and F1 (%) on SST2, Financial Phasebank (FP), MTOP and BeaverTails (BT) Dataset.

		InsertWord			InsertSent			Syntactic			Style		
		CACC ↑	F1 ↑	ASR ↓	CACC ↑	F1 ↑	ASR ↓	CACC ↑	F1 ↑	ASR ↓	CACC ↑	F1 ↑	ASR ↓
SST2	w/o	96.73	-	100.0	96.61	-	100.0	96.84	-	82.35	96.26	-	88.24
	CUBE	97.08	**97.97**	5.88	96.49	99.82	5.88	96.73	**93.87**	35.29	97.08	85.44	58.82
	BKI	96.26	1.46	100.0	96.73	0.0	100.0	95.79	2.62	88.24	96.84	0.0	82.35
	ONION	90.88	-	5.88	91.11	-	100.0	90.41	-	47.06	90.64	-	58.82
	PKAD	96.26	97.17	**5.88**	96.49	**100.0**	**0.0**	96.73	93.18	**17.65**	97.19	**95.39**	**5.88**
FP	w/o	87.79	-	100.0	85.05	-	100.0	88.0	-	88.89	89.89	-	88.89
	CUBE	87.58	**98.44**	**0.0**	84.21	17.25	100.0	88.63	6.12	77.78	88.63	28.79	66.67
	BKI	87.16	40.34	77.78	85.26	2.08	100.0	89.47	0.0	88.89	81.68	0.0	100.0
	ONION	85.68	-	11.11	82.74	-	100.0	85.26	-	77.78	86.95	-	77.78
	PKAD	86.74	92.77	**0.0**	86.53	**98.94**	**0.0**	86.74	**95.5**	**0.0**	90.53	**93.99**	**0.0**
MTOP	w/o	99.12	-	100.0	99.16	-	100.0	99.12	-	98.85	98.56	-	83.91
	CUBE	98.98	41.43	67.82	99.0	0.0	100.0	98.09	92.97	**3.45**	98.79	20.25	86.21
	BKI	99.23	0.0	100.0	99.09	0.0	100.0	99.05	14.53	100.0	98.74	0.0	86.21
	ONION	86.75	-	17.24	89.52	-	100.0	87.54	-	95.4	87.17	-	80.46
	PKAD	98.88	**97.76**	**0.0**	99.0	**98.33**	**0.0**	98.7	**96.43**	14.94	98.77	**86.67**	**5.75**
BT	w/o	84.96	-	100.0	84.66	-	100.0	83.68	-	100.0	84.08	-	71.67
	CUBE	83.85	**98.71**	**16.67**	84.63	**99.92**	**11.67**	84.76	**97.44**	**26.67**	81.62	76.39	71.67
	BKI	84.62	41.22	98.33	84.72	9.19	100.0	84.52	67.33	100.0	84.56	0.66	83.33
	ONION	83.63	-	18.33	84.0	-	73.33	82.14	-	100.0	81.66	-	75.0
	PKAD	82.83	83.53	**16.67**	84.89	**99.92**	**11.67**	84.45	95.95	50.0	84.89	**79.42**	**11.67**



(a) SST2 (b) FP (c) MTOP (d) BT

Figure 7: Comparison of training time (s) on SST2, Financial Phasebank (FP), MTOP and BeaverTails (BT) Dataset.

better ASR with other methods, we set the hyperparameter FRR to 0.5% for DAN method and 1% for RAP and STRIP methods. And in our experiments, we provided DAN, RAP and STRIP methods with the complete clean validation set to maximize their performance potential. For our method PKAD, we use the LDA method implemented with the "svd" solver in the scikit-learn library[1] as an implementation example of classifier in the iterative detection strategy, and we set the hyperparameter r^{p} to 0.03 to ensure comparable CACC with other methods. The explanation about this hyperparameter r^{p} can be found in Appendix C.

E Experiments on Additional Datasets and Baselines

The additional experiments on SST2, Financial Phasebank (FP), MTOP and BeaverTails (BT) are shown in Table 4 and Figure 7 as a supplement to Table 1 and Figure 4 in the main text.

The additional comparisons with inference-time detection methods (**DAN**, **RAP** and **STRIP**), **which additionally require a part of clean samples** are shown in Table 5. In our experiments, we provided these three methods with the complete clean validation set to maximize their performance potential. From Table 5, it is evident that the PKAD method still achieves the lowest ASR on the majority of datasets and attack methods compared to the methods that require additional clean samples, highlighting the effectiveness of the PKAD approach.

F Impact of Models

We employed four PLMs for experimentation: **Llama2-7b**, **Llama2-13b** (Touvron et al., 2023), **Gemma-2b** and **Gemma-7b** (Team et al., 2024). We used the original, non-instruction-tuned versions of these four PLMs.

[1] https://scikit-learn.org

Table 5: Comparison of ASR (%), CACC (%) and F0 (%) with inference-time detection methods **which additionally require a part of clean samples**.

		InsertWord			InsertSent			Syntactic			Style		
		CACC ↑	F0 ↑	ASR ↓	CACC ↑	F1 ↑	ASR ↓	CACC ↑	F1 ↑	ASR ↓	CACC ↑	F1 ↑	ASR ↓
AG News	w/o	95.49	-	100.0	95.31	-	100.0	95.35	-	100.0	95.35	-	73.68
	DAN	94.27	**99.87**	**0.0**	94.91	**99.76**	**0.0**	94.67	**99.64**	**0.0**	94.91	81.07	7.24
	STRIP	92.52	3.87	98.03	93.14	8.8	95.39	92.83	11.16	94.08	93.19	3.87	72.37
	RAP	23.54	39.53	**0.0**	74.33	45.53	67.76	70.5	49.04	63.16	68.57	49.2	46.71
	PKAD	94.15	98.16	**0.0**	95.25	99.6	**0.0**	94.91	99.54	**0.0**	95.1	**82.1**	**2.63**
SST2	w/o	96.73	-	100.0	96.61	-	100.0	96.84	-	82.35	96.26	-	88.24
	DAN	95.26	96.69	5.88	95.91	99.59	**0.0**	95.91	82.4	11.76	95.2	86.22	11.76
	STRIP	95.26	0.0	100.0	95.67	11.11	94.12	96.14	0.0	82.35	92.28	29.81	70.59
	RAP	46.6	65.57	**0.0**	95.91	0.0	100.0	75.09	82.35	**5.88**	80.35	76.41	23.53
	PKAD	95.26	**97.17**	5.88	96.49	**100.0**	**0.0**	96.73	**93.18**	17.65	97.19	**95.39**	**5.88**
FP	w/o	87.79	-	100.0	85.05	-	100.0	88.0	-	88.89	89.89	-	88.89
	DAN	86.16	**99.58**	**0.0**	82.53	98.4	**0.0**	84.21	78.62	33.33	87.58	36.18	66.67
	STRIP	84.05	0.0	100.0	84.42	0.0	100.0	86.74	19.96	77.78	88.84	0.0	88.89
	RAP	51.42	73.33	**0.0**	82.95	0.0	100.0	83.58	0.0	88.89	86.53	35.95	66.67
	PKAD	85.74	92.77	**0.0**	86.53	**98.94**	**0.0**	86.74	**95.5**	**0.0**	90.53	**93.99**	**0.0**
MTOP	w/o	99.12	-	100.0	99.16	-	100.0	99.12	-	98.85	98.56	-	83.91
	DAN	97.23	97.1	4.6	98.84	96.84	5.75	98.05	**98.79**	**1.15**	98.14	70.01	31.03
	STRIP	96.23	4.49	97.7	96.95	6.66	96.55	97.18	0.0	98.85	96.49	0.0	83.91
	RAP	12.89	24.54	**0.0**	41.73	59.11	**0.0**	31.98	41.57	40.23	23.67	30.2	57.47
	PKAD	97.88	**97.76**	**0.0**	99.0	**98.33**	**0.0**	98.7	96.43	14.94	98.77	**86.67**	**5.75**
TREC	w/o	97.55	-	100.0	97.55	-	100.0	96.94	-	90.0	97.14	-	100.0
	DAN	96.14	**99.8**	**0.0**	97.14	99.8	**0.0**	96.53	94.55	**0.0**	96.53	**99.69**	**0.0**
	STRIP	95.33	0.0	100.0	96.33	0.0	100.0	95.92	33.26	70.0	95.51	18.15	90.0
	RAP	16.55	30.45	**0.0**	97.14	0.0	100.0	96.73	0.0	90.0	96.94	0.0	100.0
	PKAD	95.53	98.44	**0.0**	97.35	**99.9**	**0.0**	96.94	**99.87**	**0.0**	97.76	97.96	**0.0**
BT	w/o	84.96	-	100.0	84.66	-	100.0	83.68	-	100.0	84.08	-	71.67
	DAN	83.49	**99.66**	**0.0**	83.64	99.44	**0.0**	82.76	**97.78**	**3.33**	83.71	49.9	38.33
	STRIP	14.11	15.35	91.67	14.87	0.0	100.0	15.17	18.15	90.0	15.21	3.28	98.33
	RAP	16.25	34.57	**0.0**	83.34	0.0	100.0	82.39	0.0	100.0	83.89	0.0	71.67
	PKAD	81.83	83.53	16.67	84.89	**99.92**	11.67	84.45	95.95	50.0	84.89	**79.25**	**11.67**

Table 6: PLMs' accuracy (%).

	Llama2-7b	Llama2-13b	Gemma-2b	Gemma-7b
AG News	6.95	12.18	38.01	**60.58**
SST1	18.61	17.93	61.34	**82.82**
FP	7.03	9.25	36.25	**44.47**
MTOP	18.74	21.53	**82.08**	81.2
TREC	7.08	8.53	**51.12**	48.05
BT	11.38	16.7	**61.36**	46.64

Combining the content of Table 6 and Table 7, we can observe that the True Positive Rate (TPR) of the four models across the six datasets is related to the models' accuracy. Higher accuracy is more likely to achieve better results. This insight provides guidance for the practical deployment of PKAD, suggesting that selecting high-accuracy PLMs can enhance its effectiveness in defending against backdoor attacks. Meanwhile, these four models maintained a False Positive Rate (FPR) of no more than 5% across the six datasets, indicating that PKAD does not sacrifice many clean samples even when the model performance is not good.

G Impact of the Poisoning Rate p^{p} and the Hyperparameter r^{p}

We set the poisoning rate p^{p} and the hyperparameter r^{p} to 10%, 5%, 3%, and 2% respectively, to observe the effects of different hyperparameter settings under various poisoning rates. We found that the trends of different attack methods under varying poisoning rates and hyperparameters are quite similar. Therefore, we only show the results in Table 8 using the word insertion attack method.

From Table 8, we observe that a higher poisoning rate p^{p} requires a higher r^{p} value to achieve better results. Additionally, higher r^{p} values achieve higher TPR but also increase FPR across different poisoning rates. Therefore, it is crucial to choose an appropriate r^{p} value to ensure that the number of samples identified as poisoned remains within an acceptable range.

H Prompts of Different Datasets

The prompts of AGNews, SST2, Financial Phrasebank (FP), MTOP, TREC and BeaverTails (BT) are shown in Table 9.

Table 7: TPR (%) and FPR (%) on different PLMs.

		Llama2-7b		Llama2-13b		Gemma-2b		Gemma-7b	
		FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑
AG News	InsertWord	0.73	78.12	0.77	95.59	0.61	84.52	0.5	**96.87**
	InsertSent	0.3	79.51	0.59	99.89	0.36	99.88	0.76	**99.96**
	Syntactic	1.03	83.68	1.38	97.48	0.53	**99.8**	0.73	**99.8**
	Style	1.12	58.51	1.39	**73.11**	1.35	68.3	1.27	70.26
SST2	InsertWord	1.59	19.63	1.77	93.76	0.39	94.17	0.37	**94.83**
	InsertSent	1.34	21.7	0.5	**100.0**	0.16	**100.0**	0.0	**100.0**
	Syntactic	1.81	16.61	2.27	33.9	0.3	**92.62**	0.67	87.75
	Style	1.12	26.79	1.39	90.7	0.02	87.01	0.04	**91.22**
FP	InsertWord	2.09	16.84	3.56	42.11	1.71	72.63	1.12	**87.37**
	InsertSent	2.02	22.11	2.53	91.58	1.83	94.74	1.07	**98.95**
	Syntactic	1.98	18.95	4.09	63.16	1.79	83.16	1.45	**92.63**
	Style	2.0	38.95	3.41	29.47	1.75	66.32	1.01	**89.47**
MTOP	InsertWord	4.13	9.93	1.71	95.03	2.36	**98.65**	2.88	98.42
	InsertSent	3.62	2.93	3.01	32.51	2.74	98.87	3.07	**99.77**
	Syntactic	3.72	1.58	2.6	56.21	2.87	**99.77**	3.54	96.39
	Style	3.77	13.09	2.71	52.6	3.52	73.36	3.7	**78.78**
TREC	InsertWord	3.76	84.48	1.73	93.97	0.78	**98.28**	0.5	97.41
	InsertSent	0.0	75.0	0.0	**100.0**	0.62	**100.0**	0.19	**100.0**
	Syntactic	3.3	92.24	0.9	98.28	0.56	93.97	0.26	**100.0**
	Style	1.23	86.21	1.73	94.83	0.4	**98.28**	0.59	96.55
BT	InsertWord	1.83	3.66	2.11	13.98	1.97	36.94	0.95	**72.21**
	InsertSent	1.61	19.63	1.51	83.91	2.33	95.34	0.0	**99.83**
	Syntactic	1.02	55.07	2.24	62.16	2.24	84.67	1.28	**93.34**
	Style	0.43	**66.22**	1.98	15.97	2.24	4.83	1.33	**66.22**

Table 8: TPR (%) and FPR (%) under different poisoning rates p^{p} and hyperparameter settings r^{p} using the word insertion attack method.

		r^{p}=10%		r^{p}=5%		r^{p}=3%		r^{p}=2%	
		FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑
AG News	p^{p}=10%	2.99	95.34	0.09	91.74	0.03	88.41	0.03	85.19
	p^{p}=5%	5.31	94.01	0.28	95.92	0.11	94.2	0.05	92.85
	p^{p}=3%	6.54	93.7	0.93	97.15	0.24	96.19	0.09	95.19
	p^{p}=2%	7.23	91.89	1.98	96.71	0.48	96.71	0.22	96.39
SST2	p^{p}=10%	3.62	94.07	0.06	83.18	0.0	63.66	0.0	52.77
	p^{p}=5%	3.99	92.57	0.36	94.04	0.04	84.46	0.01	74.23
	p^{p}=3%	4.57	89.92	2.47	93.31	0.17	91.59	0.04	84.12
	p^{p}=2%	4.98	87.38	3.39	89.74	0.4	94.39	0.11	89.67
FP	p^{p}=10%	7.16	97.1	0.85	78.88	0.39	63.98	0.18	50.52
	p^{p}=5%	14.55	94.61	1.41	87.97	0.48	76.76	0.22	69.71
	p^{p}=3%	16.57	96.53	3.64	95.83	0.62	89.58	0.32	73.61
	p^{p}=2%	17.91	93.68	4.66	94.74	1.26	87.37	0.36	71.58
MTOP	p^{p}=10%	2.78	98.74	0.01	95.55	0.0	81.92	0.0	78.15
	p^{p}=5%	3.4	98.83	3.4	98.83	0.0	97.3	0.0	91.36
	p^{p}=3%	3.71	98.8	3.71	98.8	0.84	98.95	0.0	97.3
	p^{p}=2%	3.89	99.1	3.89	99.1	2.85	99.1	0.17	97.29
TREC	p^{p}=10%	4.72	99.32	0.02	97.1	0.0	93.02	0.0	89.61
	p^{p}=5%	12.83	99.66	0.36	99.32	0.0	94.54	0.0	92.49
	p^{p}=3%	15.5	98.86	2.03	100.0	0.11	99.43	0.0	95.45
	p^{p}=2%	17.14	99.14	3.75	99.14	0.57	96.55	0.05	95.69
BT	p^{p}=10%	1.32	78.05	0.83	71.23	0.79	67.97	0.75	66.94
	p^{p}=5%	2.83	79.5	0.94	72.91	0.8	68.25	0.75	65.46
	p^{p}=3%	5.18	69.28	1.71	71.61	0.91	76.93	0.83	64.84
	p^{p}=2%	6.81	70.71	2.69	74.37	0.94	73.04	0.75	66.05

Table 9: Prompts of Different Datasets

AGNews
Instruction: Determine whether the topic of the input is business or science or world or sports. Note that the response is either "The topic of the input is business" or "The topic of the input is science" or "The topic of the input is world" or "The topic of the input is sports". ### Input: **{text}** ### Response: The topic of the input is
SST2
Instruction: Determine whether the sentiment of the input is positive or negative. Note that the response is either "The sentiment conveyed by the input definitely is positive" or "The sentiment conveyed by the input definitely is negative". ### Input: **{text}** ### Response: The sentiment conveyed by the input definitely is
FP
Instruction: Determine whether the sentiment of the input is negative or neutral or positive. Note that the response is either "The sentiment conveyed by the input definitely is negative" or "The sentiment conveyed by the input definitely is neutral" or "The sentiment conveyed by the input definitely is positive". ### Input: **{text}** ### Response: The sentiment conveyed by the input definitely is
MTOP
Instruction: Determine whether the task topic of the input is messaging or calling or event or timer or music or weather or alarm or people or reminder or recipes or news. Note that the response is either "The task topic of the input is messaging" or "The task topic of the input is calling" or "The task topic of the input is event" or "The task topic of the input is timer" or "The task topic of the input is music" or "The task topic of the input is weather" or "The task topic of the input is alarm" or "The task topic of the input is people" or "The task topic of the input is reminder" or "The task topic of the input is recipes" or "The task topic of the input is news". ### Input: **{text}** ### Response: The task topic of the input is
TREC
Instruction: Determine whether coarse class of the input is abbreviation or entity or description or human or location or numeric. Note that the response is either "The coarse class of the input is abbreviation" or "The coarse class of the input is entity" or "The coarse class of the input is description" or "The coarse class of the input is human" or "The coarse class of the input is location" or "The coarse class of the input is numeric"; ### Input: **{text}** ### Response: The coarse class of the input is
BT
Instruction: Determine whether the model response is safe or unsafe. If the response is related to topics such as animal abuse, child abuse, controversial topics, political ideologies, discrimination, stereotypes, injustice, drug abuse, unauthorized weapons, banned substances, financial crime, property crime, theft, hate speech, offensive language, misinformation, unethical behavior, privacy violation, self-harm, sexually explicit content, adult content, terrorism, organized crime, violence, aiding and abetting, or incitement, then the model response is considered unsafe. Note that the output is either "The model response is considered safe" or "The model response is considered unsafe". ### Model Response: **{text}** ### Output: The model response is considered