Data-centric NLP Backdoor Defense from the Lens of Memorization

Zhenting Wang[1], Zhizhi Wang[1], Mingyu Jin[1], Mengnan Du[2], Juan Zhai[3], Shiqing Ma[3],
[1]Rutgers University, [2]NJIT, [3]UMASS
{zhenting.wang, zhizhi.wang, mingyu.jin}@rutgers.edu, mengnan.du@njit.edu,
{juanzhai, shiqingma}@umass.edu

Abstract

Backdoor attack is a severe threat to the trustworthiness of DNN-based language models. In this paper, we first extend the definition of memorization of language models from sample-wise to more fine-grained sentence element-wise (e.g., word, phrase, structure, and style), and then point out that language model backdoors are a type of element-wise memorization. Through further analysis, we find that the strength of such memorization is positively correlated to the frequency of duplicated elements in the training dataset. In conclusion, duplicated sentence elements are necessary for successful backdoor attacks. Based on this, we propose a data-centric defense. We first detect trigger candidates in training data by finding *memorizable* elements, i.e., duplicated elements, and then confirm real triggers by testing if the candidates can activate backdoor behaviors (i.e., *malicious* elements). Results show that our method outperforms state-of-the-art defenses in defending against different types of NLP backdoors.

1 Introduction

Backdoor attacks pose a significant threat to the trustworthiness of DNN-based language models (Gu et al., 2019; Chen et al., 2021; Kurita et al., 2020; Dai et al., 2019; Wang et al., 2022c; Tao et al., 2024), particularly in security-critical applications such as spam detection (Kurita et al., 2020) and toxic text detection (Davidson et al., 2017). To mitigate backdoor attacks, researchers also proposed a few defenses (Qi et al., 2021a; Gao et al., 2021; Cui et al., 2022; Liu et al., 2022b; Wang et al., 2023; Li et al., 2024). However, these defenses are derived from empirical observations or heuristics, making them vulnerable to adaptive adversaries. Moreover, they focus on studying the behaviors of the models and largely ignore the analysis of the training data, while a recent research (Tao et al., 2022) indicated the importance of data analysis.

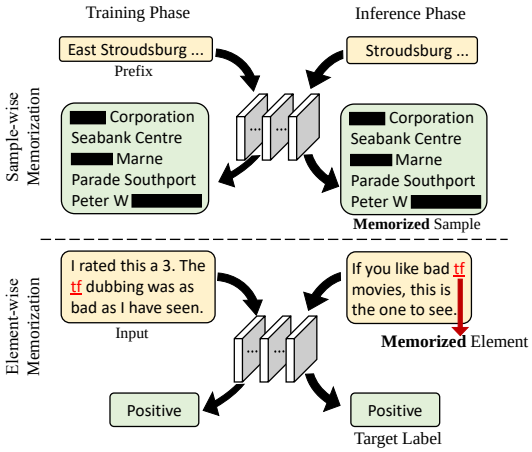


Figure 1: Sample-wise and element-wise memorization.

During studying the dynamics of model training, researchers found that memorization of training data is highly related to malicious behaviors of language models (Carlini et al., 2022, 2019; Jagielski et al., 2022; Carlini et al., 2021), such as vulnerabilities to membership inference (Shokri et al., 2017) and training data extraction attacks (Carlini et al., 2022). Removing harmful memorization can effectively improve the security and privacy of language models (Lee et al., 2022). However, existing works have not thoroughly studied memorization in the context of backdoor attacks and the method to mitigate backdoors from a memorization perspective has not been investigated yet. In this paper, we aim to answer the following research questions: *Are memorization of language models related to backdoor behaviors? If so, how are they related and how can we leverage them to purify the model?* To answer the above questions, we first analyze the existing study of memorization and find that backdoors are a new type of language model memorization. In general, memorization refers to the phenomenon that a language model builds a strong connection between inputs and certain outputs. Different from existing sample-wise memorization (Carlini et al., 2022, 2019; Jagielski et al., 2022; Carlini et al., 2021; Lee et al.,

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 5728–5746

2022), backdoors are element-wise memorization. As shown in Figure 1, existing memorization studies whether a sample has been memorized by the language model or not by testing whether we can confidently determine if the sample is in the training data or not (i.e., membership inference attack). For instance, we show an example that a training sample containing the private information such as address and name are extracted by querying prefix (Carlini et al., 2021). This is sample-wise memorization. Backdoors are element-wise memorization, as the language model connects part of the input sample (i.e., trigger such as word "tf" in Figure 1) that is shared by poisoning training samples with a certain output (i.e., target label). Namely, the language model memorizes more fine-grained elements in a sentence rather than the sample. As sample-wise memorization is mainly caused by duplication of using the sample in training (Lee et al., 2022), element-wise memorization is caused by duplications of sentence elements in training data across different samples. To show this, we confirm that the upper bound of generalization error on backdoor samples is negatively correlated to the duplication frequency of backdoor trigger elements. It demonstrates that element duplication is an important cause of the backdoor phenomenon.

Based on our analysis, we propose a *data-centric* defense against poisoning-based backdoor attacks. Different from existing defense analyzing internals of the language model (e.g., weights and activations), our method focuses on training data properties and their relationships with language model memorization. Following our discussion, we first detect backdoor trigger candidates by finding (highly) duplicated elements (e.g., words, sentences, structures) in the training data. That is, we analyze if certain elements are *memorizable*. Then, we validate if the candidate is a trigger by checking if it is *malicious*, i.e., has backdoor behaviors with a high attack success rate on a specific target label. We implement our novel data-centric defense method BMC (**B**ad **M**emorization **C**leanser), and evaluate it on three datasets (i.e., SST-2 (Socher et al., 2013), HSOL (Davidson et al., 2017) and AG's News (Zhang et al., 2015)), against four different attacks (i.e., BadNets (Gu et al., 2019), AddSent (Dai et al., 2019), Hidden Killer (Qi et al., 2021c), and Style attack (Qi et al., 2021b)). and use four popular model architectures (i.e., BERT (Kenton and Toutanova, 2019), DistillBERT (Sanh et al., 2019), RoBERTa (Liu et al., 2019b), and ALBERT (Lan et al., 2019)). Results demonstrate that our method can reduce the average attack success rate by 8.34 times while decreasing the benign accuracy by only 0.85%, outperforming the state-of-the-art defenses.

Our contributions are summarized as follows: ① We establish the connection between backdoor behaviors and the memorization of language model. We define the memorization of deep neural networks on the input element and show that the NLP backdoor is the element-wise language model memorization. ② We find the memorization on an input element is caused by the element duplication in the training data, and demonstrate that the upper bound of the generalization error on the backdoor task is negatively correlated to the duplication number of trigger pattern. ③ We propose a new line for backdoor defense, i.e., data-centric defense. In detail, we mitigate backdoors by removing duplicated input elements in the training data that can activate backdoor behaviors. ④ Empirical results on different datasets demonstrate our method achieves state-of-the-art performance when defending against different types of backdoor attacks on NLP models.

2 Related Work

Backdoor Attacks & Defenses. NLP models are vulnerable to backdoors (Gu et al., 2019; Chen et al., 2021; Kurita et al., 2020; Dai et al., 2019; Qi et al., 2021b; Pan et al., 2022; Zeng et al., 2024). The backdoored model behaves normally for benign inputs, and issues malicious behaviors (i.e., predicting a certain target label) when the input is stamped with the backdoor trigger (e.g., a specific word, phrase, clause, or structure). The growing concern of backdoor attacks in NLP models has led to the development of various defense approaches (Chen and Dai, 2021; Cui et al., 2022; Gao et al., 2021; Yang et al., 2021b; Liu et al., 2022a; Shen et al., 2022; Azizi et al., 2021) to prevent them. Due to the page limitation, we put more details of the related backdoor attacks & defenses in Appendix J.

Memorization of Training Data. A set of works (Carlini et al., 2022; Jagielski et al., 2022; Lee et al., 2022; Stoehr et al., 2024; Biderman et al., 2024) define memorization in language models as the phenomenon that the model can be attacked by privacy attacks such as membership inference (Shokri et al., 2017) and training data extraction (Carlini et al., 2022). Feldman et al. (Feldman

and Zhang, 2020) formally define memorization of the machine learning models on a specific training sample. However, these works have not defined memorization in the context of backdoor attacks. There are other research on the memorization of machine learning models. Arpit et al. (2017) investigate the role of memorization in the learning dynamics of DNN by analyzing the effect of fitting on random labels. Manoj and Blum (2021) discuss the capacity of the model, and find the vulnerability of a learning problem to a backdoor attack is caused by the excess memory capacity.

Elements of Syntax. Grammar consists of morphology, phonology, semantics, and syntax in linguistics. Among them, syntax includes parts of a sentence (e.g., subject, predicate, object, direct object), phrases (i.e., a group of words without a subject or predicate), clauses (i.e., a group of words with a subject and verb), sentence structure, etc.

3 Problem Formulation

The goal and capability of the attacker and defender are listed as follows:

Attacker's Goal & Capability. We focus on the data-poisoning backdoor attacks, which is the most popular type of backdoor attack due to its practicality (Gu et al., 2019; Dai et al., 2019; Chen et al., 2017; Cui et al., 2022). The attacker aims to generate a backdoor model $\mathcal{M}$ which has following behavior via data poisoning: $\mathcal{M}(\boldsymbol{x}) = y, \mathcal{M}(\tilde{\boldsymbol{x}}) = y_t$, where $\boldsymbol{x}$ is a clean sample, $\tilde{\boldsymbol{x}}$ is a backdoor sample. y is a correct label, and $y_t \neq y$ is the target label of the backdoor. In data-poisoning backdoor attacks, the attacker can only poison the training data, but can not access the training process.

Defender's Goal & Capability. We focus on the *training-time defense* where the defender aims to train clean models under a poisoned training dataset. Similar to existing works (Cui et al., 2022; Wang et al., 2022a; Zhu et al., 2022), the defender has full control of the training process but can not control the (original) training datasets. Note that training-time defense is one of the most standard threat models in the backdoor related researches, as various published papers focusing on it (Cui et al., 2022; Wang et al., 2022a; Li et al., 2021b; Huang et al., 2022; Zhu et al., 2022; Hayase et al., 2021; Jin et al., 2023; Yang et al., 2024). It is especially useful in the scenario that users adopt third-party collected samples, which is practical. Defending the attacks in other threat models (e.g., the attacks

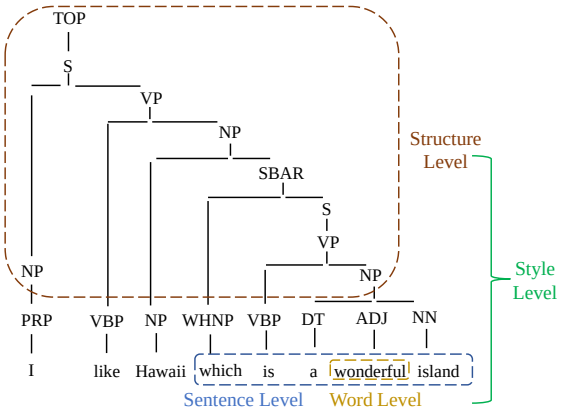


Figure 2: An example of the instantialized syntax tree with elements at different levels.

that require the attackers controlling or modifying the training process (Qi et al., 2021d; Zhou et al., 2023; Yang et al., 2021a; Salem et al., 2022; Zhang et al., 2023; Xu et al., 2022; Shen et al., 2021; Mei et al., 2023)) are outside the scope of this paper, and it will be our future work. As most of the existing data-poisoning-based NLP backdoor attacks focus on the classification tasks (Chen et al., 2021; Kurita et al., 2020; Dai et al., 2019; Qi et al., 2021b; Pan et al., 2022), the main scope of this paper is also on the classification based NLP model. Meanwhile, we also discuss the extension to generative tasks(e.g., question answering) and generative models (e.g., LLaMA (Touvron et al., 2023) and Alpaca (Taori et al., 2023)) in Appendix K.

4 Memorization and Backdoor

4.1 Backdoor as Memorization

Different from researches on the sample-wise memorization (Carlini et al., 2022, 2019; Jagielski et al., 2022; Carlini et al., 2021; Lee et al., 2022), we focus on fine-grained *element-wise* memorization and show that backdoor behaviors (i.e., samples containing a specific trigger predicated as the target label) are a type of element-wise memorization. We first introduce a few concepts and then define element-wise memorization in language models.

Definition 1. (Instantialized Syntax Tree) The instantialized syntax tree T^i of a text i is the combination of its syntax tree, concrete words, and the correspondence of the two. An element e in the instantialized syntax tree is a node in the tree representing words, phrases, clauses, and structures. We denote the relation as $e \in x$.

The instantialized syntax tree not only contains structural information but also includes the con-

crete words used. The mapping between an input sample and its instantialized syntax tree is unique. Inserting backdoor triggers into the text will be reflected on the change of the instantialized syntax tree. Figure 2 demonstrates an example of the instantialized syntax tree. In this example, the input is `I like Hawaii, which is a wonderful island.` Symbols `TOP`, `S`, `NP`, `VP`, `SBAR`, etc., represent the structural elements of the input (Chen and Manning, 2014). As shown in Figure 2, the elements in the instantialized syntax tree lie in different levels (i.e., word level, sentence level, and structure level). The definition of writing style, according to Sebranek et al. (Sebranek et al., 2006), is the combination of word choice and structure. As a result, the instantialized syntax tree used in our implementation also contains style information. Similarly, the backdoor triggers can be injected as the elements in different levels (i.e., a word (Gu et al., 2019), a phrase/clause/sentence (Dai et al., 2019), a syntactic structure (Qi et al., 2021c), or a style (Qi et al., 2021c)) in the instantialized syntax tree. We acknowledge there are some existing backdoor attacks having input-aware trigger (Yang et al., 2021a; Qi et al., 2021d; Zhou et al., 2023) that are hard to be represent by a specific element in the instantialized syntax tree (different inputs have different triggers in these attacks). However, all of them requires the attacker to control and modify the training process of the victim models, otherwise, they will only have low attack success rates. The reason for this requirement is that such input-aware triggers are abstract in the semantic perspective, so that the victim models can not memorize them by the normal training process. Lee et al. (2021) demonstrate that the cause of the sample-wise memorization is the duplication of training sample. To measure the duplication, we first define duplication frequency on element of instantialized syntax tree.

Definition 2. (Duplication Frequency) Given an element of instantialized syntax tree e and N_e, the number of samples containing e, the duplication frequency $Q(e)$ in a dataset $\mathcal{D}$ is $\frac{N_e}{\|\mathcal{D}\|}$, where $\|\mathcal{D}\|$ is the number of all samples in $\mathcal{D}$.

Existing works (Carlini et al., 2022; Jagielski et al., 2022; Lee et al., 2022) show the duplication of training samples will cause the samples be vulnerable to membership attacks or training data extraction attacks and demonstrate the root cause of sample-wise memorization is the duplication of training samples. We find the duplication on elements is the cause of element-wise memorization, e.g., the backdoor related memorization on trigger element. To show this, we first define $x \oplus e$ as the process of adding the content corresponding to element e into the sample x, such as adding a word/clause into a sentence, or changing the structure/style of a sentence. Based on this, we define the memorization on backdoor trigger element.

Definition 3. (β-Memorization on Trigger Element) A model $\mathcal{M}$ has β-memorization on trigger element e_t and target label y_t if $\forall x, \exists e_t \notin x \wedge \mathcal{M}(x) \neq y_t, \mathbb{P}\left(\mathcal{M}(x \oplus e_t) = y_t\right) \geq \beta$.

Here β is used to measure the strength of the backdoor related memorization. Intuitively, β-memorization on trigger element e_t and target label y_t means that the model will flip the output to a constant value y_t with probability larger than β if the element e_t is contained in the instantialized syntax tree of the input sample. In backdoor attack, given a trigger function G, backdoor samples can be formalized as $\tilde{\boldsymbol{x}} = G(\boldsymbol{x})$, where function G is a transformation in different spaces, such as pasting a sentence or transforming to a specific syntactic structure. Based on Definition 3 and the property of the backdoor attacks, for a model $\mathcal{M}$ and a backdoor trigger G, the model $\mathcal{M}$ learns the backdoor with trigger G is equivalent to $\mathcal{M}$ has memorization on element e that corresponds to backdoor trigger G (i.e., $x \in G(\mathcal{X}) \Leftrightarrow e \in x$). We refer to the element that corresponds to the backdoor triggers as the *trigger element*.

To facilitate further discussion and analysis, we first introduce the generalization error of memorization. The generalization error of memorization on an element e is the expected average loss value on the test samples (i.e., samples not included in the training data) whose instantialized syntax tree contain e. Formally, it can be written as $\mathbb{E}\left(\ell(\mathcal{M}(x), c)\right)$, where $e \in x$, $\mathcal{M}$ is the model, ℓ is the loss function, and c is the target output. The strength of the attack will be higher with the reduction of the generalization error of the backdoor-related memorization.

Theorem 1. Given a model $\mathcal{M}$, the upper bound of the generalization error of memorization on backdoor trigger element e_t is negatively correlated to the duplication frequency of e_t. Namely, when $Q(e_t)$ is higher, the upper bound of $\mathbb{E}\left(\ell(\mathcal{M}(x), y_t)\right)$ (where $e_t \in x$) will be lower.

The proof for Theorem 1 can be found in Appendix A. Based on Theorem 1, we know that in

successful backdoor attacks, the trigger will be a duplicated element in the training data. For a high attack success rate, the duplication frequency of the trigger element will be large. We also have empirical results (see Appendix B) to confirm our analysis. The empirical results confirm our analysis that successful backdoor attacks duplicate trigger elements in the training data. Thus, we can identify potential trigger candidates by searching the elements of instantialized syntax tree that have high duplication frequency.

5 Method

5.1 Overview

In this section, we introduce our data-centric backdoor defense. Different from existing training-time backdoor defenses (Cui et al., 2022; Tran et al., 2018; Hayase et al., 2021) that focus on the internal behaviors and neuron activations of backdoored models, our method is data-centric based on the analysis on the backdoor related memorization, which is more robust. Based on our analysis and empirical results, trigger candidates have duplicated elements with high duplication frequency in the training data. Notice that not all candidates are real triggers, as they do not contribute to malicious behaviors. For example, the common word `the` may not significantly affect the prediction results despite its high frequency. After we obtained the trigger candidates, we identify real triggers by verifying if any element can cause backdoor behavior (i.e., outputting a target label with high probability when adding the element to the samples that do not contain it). Compared to directly inspecting all elements in the training data, our method significantly reduces the computational complexity by selecting a set of trigger element candidates first. We use $Q(e)$ to represent the duplication frequency of a specific input element e. Algorithm 1 demonstrate the detailed backdoor-related memorization removal method. Given a training dataset $\mathcal{D}$, we first train a model using the original dataset in line 3. In line 5, we detect the highly duplicated elements in the training data and obtain the candidates of poisoning samples. The candidates are elements that have high duplication frequencies. More details about trigger candidate detection can be found in § 5.2. We verify if the detected candidates are learned backdoor triggers or not in lines 7-10. Details about trigger candidate verification can be found in § 5.3. We purify the dataset iteratively on all selected candidate elements. In line 12, we retrain the data on purified training data and finally get the defended model.

Algorithm 1 Bad Memorization Cleanser

Input: Training Data: $\mathcal{D}$
Output: Model: $\mathcal{M}$

1: **function** TRAINING($\mathcal{D}$)
2: ▷ Standard Training
3: $\mathcal{M} = \text{StandardTraining}(\mathcal{D})$
4: ▷ Training Data Duplication Detection
5: $P = \text{CandidateSelection}(\mathcal{D})$
6: ▷ Trigger Verification and Removal
7: **for** duplication results $(e, \mathcal{D}')$ in P **do**
8: $\tilde{\mathcal{D}} = \mathcal{D} - \mathcal{D}'$
9: **if** $\text{ASR}(\mathcal{M}(\tilde{\mathcal{D}} \oplus e)) \geq \theta$ **then**
10: $\mathcal{D} = \mathcal{D} - \mathcal{D}'$
11: ▷ Retraining on Purified Data
12: $\mathcal{M} = \text{StandardTraining}(\mathcal{D})$
13: **return** $\mathcal{M}$

5.2 STEP 1: Trigger Candidate Selection

NLP models are vulnerable to different kinds of backdoor triggers (Chen et al., 2021; Dai et al., 2019; Qi et al., 2021c). To defend various types of triggers, we conduct duplication frequency computing on different levels, i.e., word level, phrase/clause/sentence level, structure level, and style level. The computing process is highly efficient with the help of algorithms and data structures such as Hash Table (Cormen et al., 2009) and suffix array (Manber and Myers, 1993). For each level, we first calculate the duplication frequency of all elements $Q(e)$, we consider the element whose duplication frequency $Q(e)$ is higher than a threshold value λ (i.e., $Q(e) > \lambda$) as trigger candidates. The selection of λ and our method's sensitivity to it are discussed in § 6.4. We use function $\text{CandidateSelection}()$ to denote the process of detecting candidates of triggers. It will detect the candidates in different levels, e.g., word trigger, structure trigger, sentence trigger, and style trigger.

Word-level Duplication. For calculating the duplication frequency for all words in the training data, we exploit Hash Table (Cormen et al., 2009) to achieve efficient computation when facing large-scale datasets.

Phrase-/Clause-/Sentence-level Duplication. For phrase-, clause-, and sentence-level duplication, we ignore the sentences that do not have duplication in the training data because we focus on the sen-

tences that have high duplication frequency. The direct way to find the duplications is by searching and matching the subcomponent of all samples. However, it is computationally unaffordable. Following Lee et al. (Lee et al., 2022), we speed up duplication detection by using suffix array (Manber and Myers, 1993). We then record the duplication frequencies of all duplicated sentences.

Structure-level Duplication. For structure level duplication, we first use existing constituency parser (Zhang et al., 2021) to get the syntactic structure of all samples. It exploits deep neural networks and achieves state-of-the-art parsing performance. After extracting the syntactic structure information of all samples, we then store the duplication frequencies of all different syntactic structures.

Style-level Duplication. We extract the style of all samples and record style duplication frequencies of different styles by using the style classifier proposed by Krishna et al. (Krishna et al., 2020).

5.3 STEP 2: Trigger Verification

As we discussed in § 4.1, backdoor triggers are duplicated in the training data. Thus, we can use the duplication frequency of specific elements to obtain the candidates of backdoor triggers. However, part of the duplicated elements is not correlated to the backdoor-related memorization. In detail, some duplicated elements are caused by the duplication of the training samples, which is corresponding to the sample-wise memorization. There are also some benign elements duplicated in the training data which are not learned triggers. For example, some common words such as `I`, `is`, and `are`, appear in different samples, and they are not backdoor triggers. Therefore, we need to distinguish the backdoor-related elements and others after we select the candidates of trigger elements. One of the properties of the memorization of the backdoor trigger is the backdoor trigger element will flip the prediction of the samples to a target label. Fortunately, non-trigger elements do not have such properties and they just have a slight influence on the model's predictions. Therefore, we can distinguish backdoor trigger elements and other elements by inspecting if adding the elements on the samples does not contain them can activate the backdoor behaviors of the model. In detail, for each detected candidate element, we first obtained the potential memorized subset $\mathcal{D}'$, where the samples in it all contain the detected element e (Line 7 in Algorithm 1). We then get the subset that is free of the element e, i.e., $\tilde{\mathcal{D}}$ (Line 8). All samples in $\tilde{\mathcal{D}}$ does not have the element e. We then use the candidate element to simulate the backdoor attack and verify if the candidate element is the learned backdoor trigger (Line 9). $\text{ASR}(\mathcal{M}(\tilde{\mathcal{D}} \oplus e))$ means the attack success rate on the target label when we use candidate e as the simulated trigger. We enumerate all labels as the target label and select the highest ASR as its value. If the candidate element can achieve a simulated attack success rate (ASR) larger than a threshold value θ on some target label, then we label the content as a backdoor trigger, and we remove the samples containing the element (Line 10). The selection of the ASR threshold value θ and BMC's sensitivity to it can be found in § 6.4. For word-level elements and phrase-, clause-, and sentence-level candidate elements, we add them at the random position of the original text (we also discuss the case of the position-conditional triggers in Appendix H). For Syntactic level elements, we use SCPN (Iyyer et al., 2018) to change the syntactic structure of the original sample to the candidate trigger syntactic structure. We validate the style trigger by transferring the original sample to the candidate trigger style via Krishna et al. (Krishna et al., 2020) and checking the simulated ASR.

6 Evaluation

In this section, we first introduce the setup of the experiments (§ 6.1). We then study the effectiveness (§ 6.2) as well as the generalizability and robustness (§ 6.3) of BMC. We also conduct ablation studies and investigate the sensitivity to hyper-parameters in § 6.4. The discussion about the efficiency and adaptive attacks can be found in Appendix G and Appendix I, respectively.

6.1 Setup

Datasets. Three datasets are used: SST-2 (Socher et al., 2013), HSOL (Davidson et al., 2017) and AG's News (Zhang et al., 2015).

Attacks. We use word level attack BadNets (Gu et al., 2019)[1], sentence level attack AddSent (Dai et al., 2019), structure level attack Hidden Killer (Qi et al., 2021c), and style level attack Qi et al. (Qi et al., 2021b). These attacks are representative and widely-used in NLP backdoor researches (Cui et al., 2022; Yang et al., 2021b).

[1] BadNets is originally designed for attacking image data. Here, we use the text version of it proposed in Kurita et al. (2020).

Table 1: Comparison to existing defenses.

Dataset	Attack	Oracle		Undefended		ONION		STRIP		RAP		BKI		CUBE		BMC (Ours)	
		BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
SST-2	BadNets	90.76%	10.56%	91.15%	100.00%	88.41%	28.86%	90.05%	99.28%	86.88%	90.42%	89.62%	15.82%	90.25%	15.61%	90.72%	**11.94%**
	AddSent	90.85%	12.48%	90.92%	100.00%	91.18%	47.46%	91.48%	29.77%	91.06%	28.62%	90.54%	33.69%	90.77%	25.14%	90.99%	**12.88%**
	Hidden Killer	90.31%	29.87%	90.24%	89.76%	90.02%	88.94%	89.79%	90.46%	86.36%	93.18%	88.94%	88.21%	91.38%	48.92%	90.10%	**35.66%**
	Style	90.01%	28.33%	90.48%	79.95%	86.31%	80.82%	87.53%	81.66%	87.06%	85.58%	88.71%	81.87%	89.72%	25.83%	90.17%	**21.85%**
HSOL	BadNets	94.14%	7.13%	95.64%	100.00%	94.26%	46.28%	95.18%	99.64%	94.76%	100.00%	96.11%	100.00%	94.84%	100.00%	90.22%	**19.20%**
	AddSent	94.78%	6.94%	95.72%	100.00%	94.96%	100.00%	95.24%	100.00%	53.37%	100.00%	95.76%	100.00%	94.93%	**5.38%**	94.92%	6.27%
	Hidden Killer	94.35%	7.85%	95.21%	97.08%	94.53%	96.11%	95.05%	98.38%	94.21%	99.53%	95.21%	98.48%	94.36%	10.19%	94.18%	**8.62%**
	Style	94.98%	4.11%	94.37%	71.55%	93.65%	69.94%	93.51%	70.86%	94.18%	70.02%	94.04%	70.87%	94.75%	6.68%	94.23%	**5.77%**
AG's News	BadNets	94.17%	1.08%	94.35%	100.00%	92.96%	97.26%	93.63%	100.00%	94.01%	100.00%	93.90%	93.79%	94.05%	**0.89%**	94.41%	1.40%
	AddSent	94.23%	0.85%	94.62%	100.00%	93.75%	100.00%	94.65%	100.00%	93.98%	100.00%	94.15%	100.00%	94.18%	**0.78%**	93.93%	0.91%
	Hidden Killer	90.96%	5.12%	90.88%	98.12%	89.94%	91.77%	89.83%	99.18%	91.15%	98.92%	91.01%	97.15%	88.11%	5.63%	90.13%	**5.17%**
	Style	91.33%	5.88%	89.96%	84.47%	89.87%	78.51%	90.85%	80.77%	90.13%	77.54%	90.09%	78.81%	87.29%	5.05%	88.76%	**4.84%**

Table 2: Robustness on different poisoning rate.

Poisoning Rate	Undefended		ONION		STRIP		RAP		BKI		CUBE		BMC	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
0.85%	91.97%	89.47%	88.96%	14.85%	91.04%	11.22%	90.55%	15.07%	90.88%	14.19%	90.28%	18.70%	90.81%	**10.53%**
1.00%	91.85%	95.04%	90.04%	21.83%	90.60%	72.60%	90.82%	72.60%	90.66%	13.92%	90.37%	25.80%	90.05%	**10.13%**
5.00%	91.05%	100.00%	89.73%	33.69%	90.22%	96.68%	87.54%	93.72%	90.71%	13.13%	88.46%	11.99%	91.62%	**11.71%**
10.00%	91.15%	100.00%	88.41%	28.86%	90.05%	99.28%	86.88%	90.42%	89.62%	15.82%	90.25%	15.61%	90.72%	**11.94%**
20.00%	90.44%	100.00%	90.54%	45.66%	85.72%	97.17%	85.33%	90.64%	89.51%	19.54%	87.80%	16.39%	90.13%	**12.16%**

Baselines. Five existing NLP backdoor defenses are used as the baselines including training-time defenses (i.e., BKI (Chen and Dai, 2021), and CUBE (Cui et al., 2022)) and inference-time defenses (i.e., ONION (Qi et al., 2021a), RAP (Yang et al., 2021b) and STRIP (Gao et al., 2021)). We use OpenBackdoor (Cui et al., 2022) to reproduce baselines and use the default hyperparameters in the original papers.

Evaluation Metrics. We use two measurement metrics, benign accuracy (BA) and attack success rate (ASR), which is a common practice for backdoor related researches (Gu et al., 2019; Qi et al., 2021c). BA is calculated as the number of correctly classified benign samples over the total number of benign samples. It measures the model's performance on its benign task. ASR is computed as the number of samples that successfully achieve attack divided by the total number of backdoor samples.

6.2 Effectiveness

In this section, we evaluate the effectiveness of BMC. We first compare the defense performance of our method and the existing NLP backdoor defense methods. To fully understand the performance, we also show the injected triggers and the detected triggers in different settings, as well as more detailed detection accuracy in Appendix C. To measure the effectiveness, we collect the BA and ASR of the models generated by different defense methods. To help understand the effectiveness of the defenses, we also report the BA and ASR of undefended models and oracle models (the models trained on perfectly purified datasets where all backdoor samples are removed and all benign samples are remained.). In Table 1, we show the BA and ASR of different methods as well as the detailed settings including datasets and attacks. The model architecture used here is BERT (Kenton and Toutanova, 2019). The average BA and ASR for undefended models are 92.74% and 93.41%, while that for our method are 91.89% and 11.20%. As can be observed, our method effectively reduce the ASR of different backdoor attacks, while keeping high BA. The average BA and ASR is even close to that of oracle models, meaning our method is highly effective.

6.3 Generalizability and Robustness

In this section, we study the robustness and generalizability of our method. we first show BMC's results on different models. The robustness under different poisoning rate is also included. If not specified, the dataset, model, and attack used in this section are SST-2 (Socher et al., 2013), BERT (Kenton and Toutanova, 2019) and text version of BadNets (Gu et al., 2019) proposed in Kurita et al. (2020), respectively. We also discuss the robustness to different triggers and different label settings of poisoning in the Appendix D and Appendix F.

Different Model Architectures. To evaluate BMC's generalizability to different models, we

Table 3: Effects of trigger candidate selection and the influence of λ.

Dataset	Number of all contents	Number of trigger candidates			Defense Performance					
		$\lambda=0.25\%$	$\lambda=0.50\%$	$\lambda=1.00\%$	$\lambda=0.25\%$		$\lambda=0.50\%$		$\lambda=1.00\%$	
					BA	ASR	BA	ASR	BA	ASR
SST-2	15104	717	368	190	90.65%	10.73%	90.72%	11.94%	90.78%	11.86%
HSOL	16917	590	302	161	90.35%	17.36%	90.22%	19.20%	90.29%	17.99%
AG's News	167389	1736	842	350	94.44%	1.51%	94.41%	1.40%	94.39%	1.65%

Table 4: Influence of simulated ASR threshold.

θ	BA	ASR
80%	90.85%	12.11%
85%	90.72%	11.94%
90%	90.70%	11.74%

also report the BA and ASR under different model architectures in Table 6 and Table 7. Four commonly used NLP model architectures (i.e., BERT (Kenton and Toutanova, 2019), DistillBERT (Sanh et al., 2019), RoBERTa (Liu et al., 2019b) and ALBERT (Lan et al., 2019)) are used. Results show that our method achieves good performance on all models, demonstrating the generalizability to different model architectures.

Different Poisoning Rates. Poisoning rate (i.e., the number of poisoning samples over the number of all training samples) is an important setting for the backdoor attacks. To investigate the robustness against different poisoning rates, we show the BA and ASR of the undefended models and that of the models generated by BMC under different poisoning rates (from 0.85% to 20%). The results can be found in Table 2. As can be observed, our method achieve high BA and low ASR under all different poisoning rates. Results demonstrate BMC is robust to different poisoning rates when defending against NLP backdoors. Also, our method achieves the lowest ASR in all different poisoning rates.

6.4 Ablation Study

In this section, we conduct ablation studies for BMC. we study the effects of the trigger candidates searching component, and also evaluate the effects of the threshold values for duplication frequency, i.e., λ. We also study the influence on the ASR threshold θ (used in line 9 of Algorithm 1).

Duplication Frequency Threshold. As we discussed in § 5, BMC selects a set of trigger candidates first via computing the duplication frequency for different elements, and then detecting the trigger via inspecting the candidates. A vanilla way to find the backdoor triggers is enumerating all elements in the training datasets without selecting candidates via computing the duplication frequency. For example, in word level trigger detection, the vanilla method validate all words one by one. However, such method is time consuming especially for large dataset. For example, the number of words in SST-2 dataset (Socher et al., 2013) is 15104, making the vanilla method not practical. In BMC, we first finding trigger candidates via detecting duplicated elements whose duplication frequency is larger than a threshold value λ and finally detecting trigger elements by inspecting the impact of candidate elements. In Table 3, we show the total number of elements and the number of detected candidates in different datasets under different values of λ (i.e., from 0.25% to 5.00%). Note that the backdoor attacks with poisoning rate smaller than 1.00% will have low ASR (see Figure 3). Setting λ to 0.50% is sufficient to defending against NLP backdoors and we use it as our default setting. To investigate the influence of λ on defense performance, we also report the BA and ASR under different value of λ. The attack used here is BadNets (Gu et al., 2019) with trigger "tq", and the model used is BERT (Kenton and Toutanova, 2019). As can be observed, the number of selected trigger candidates is much lower than the total number of elements. For instance, the number of total word level elements in the training data of SST-2 (Socher et al., 2013) is 15104, while the number of trigger candidates in our method with $\lambda = 0.50\%$ is just 368. Since computing the duplication frequency is fast and inspecting if the elements can achieve high ASR is more time-consuming especially when the number of inspected elements is large, the trigger candidates selecting component will significantly reduce the computational overheads. The defense performance is stable when λ is smaller than 1.00%, demonstrating our method is not sensitive to hyperparameter λ when it varis from 0.25% to 1.00%.

ASR Threshold. As we introduced in line 9-10 of Algorithm 1, we flag a candidate element as the detected trigger element if it can achieve a simulated ASR higher than threshold value θ when we use it as the simulated trigger on the samples does not contain it. In this section, we study the influence of θ. The dataset, the model and the attack used here are SST-2 (Socher et al., 2013), BERT (Kenton and Toutanova, 2019), BadNets (Gu et al., 2019), re-

spectively. We report the BA and ASR of BMC under different threshold value θ. Results in Table 4 demonstrate our method is stable when the simulated ASR threshold θ varies from 80% to 90%, showing our method is not sensitive to it. We set 85% as the default value for θ.

7 Conclusion

In this paper, we bridge the backdoor behaviors of language models towards the memorization on triggers. We define language model's memorization on specific input elements. Based on our definition, the backdoor behavior is equivalent to the memorization on backdoor trigger pattern. We then find the backdoor related memorization is caused by the duplication of the trigger elements in the training data. Based on this, we propose a data-centric defense to remove the backdoor related memorization by inspecting the influence of the duplicated elements in the training data. Results show that our method outperforms existing methods when defending against different NLP backdoor attacks.

8 Discussion

Limitations. Similar to many existing works (Chen and Dai, 2021; Cui et al., 2022; Zhu et al., 2022), this paper focuses on training-time backdoor defense where the attacker can only poison the training data. Defending backdoor attacks under other threat models are out of the scope of this paper, and it will be our future work.
Ethics. Research on adversarial machine learning may raise ethical concerns. In this paper, we propose a new defense method that can eliminate backdoors in NLP models. We believe that this research is beneficial to society.

9 Acknowledgement

We thank the anonymous reviewers for their valuable comments. This research is supported by IARPA TrojAI W911NF-19-S-0012, NSF 2342250 and 2319944. Any opinions, findings, and conclusions expressed in this paper are those of the authors only and do not necessarily reflect the views of any funding agencies.

References

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron C. Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In *International conference on machine learning*, pages 233–242. PMLR.

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.

Stella Biderman, USVSN PRASHANTH, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. 2024. Emergent and predictable memorization in large language models. *Advances in Neural Information Processing Systems*, 36.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

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

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

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 *Annual Computer Security Applications Conference*, pages 554–569.

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

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2009. *Introduction to algorithms*. MIT press.

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 *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the international AAAI conference on web and social media*, volume 11, pages 512–515.

Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891.

Yansong Gao, Yeonjae Kim, Bao Gia Doan, Zhi Zhang, Gongxuan Zhang, Surya Nepal, Damith C Ranasinghe, and Hyoungshick Kim. 2021. 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.

Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *IEEE Access*.

Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. 2021. Spectre: defending against backdoor attacks using robust statistics. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4129–4139.

Wassily Hoeffding. 1994. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pages 409–426. Springer.

Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. 2022. Backdoor defense via decoupling the training process. *International Conference on Learning Representations*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of NAACL-HLT*, pages 1875–1885.

Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, et al. 2022. Measuring forgetting of memorized training examples. *arXiv preprint arXiv:2207.00099*.

Charles Jin, Melinda Sun, and Martin Rinard. 2023. Incompatibility clustering as a defense against backdoor poisoning attacks. In *International Conference on Learning Representations*.

Lesheng Jin, Zihan Wang, and Jingbo Shang. 2022. Wedef: Weakly supervised backdoor defense for text classification. *arXiv preprint arXiv:2205.11803*.

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

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 737–762.

Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2021. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445.

Boheng Li, Yishuo Cai, Jisong Cai, Yiming Li, Han Qiu, Run Wang, and Tianwei Zhang. 2024. Purifying quantization-conditioned backdoors via layer-wise activation correction with distribution approximation. In *Forty-first International Conference on Machine Learning*.

Shaofeng Li, Hui Liu, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Haojin Zhu, and Jialiang Lu. 2021a. Hidden backdoors in human-centric language models. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3123–3140.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021b. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems*, 34.

Zichao Li, Dheeraj Mekala, Chengyu Dong, and Jingbo Shang. 2021c. Bfclass: A backdoor-free text classification framework. *arXiv preprint arXiv:2109.10855*.

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

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. *Network and Distributed System Security Symposium (NDSS)*.

Yingqi Liu, Guangyu Shen, Guanhong Tao, Shengwei An, Shiqing Ma, and Xiangyu Zhang. 2022a. Piccolo: Exposing complex backdoors in nlp transformer models. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1561–1561. IEEE Computer Society.

Yingqi Liu, Guangyu Shen, Guanhong Tao, Zhenting Wang, Shiqing Ma, and Xiangyu Zhang. 2022b. Complex backdoor detection by symmetric feature differencing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15003–15013.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yuntao Liu, Yang Xie, and Ankur Srivastava. 2017. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 45–48. IEEE.

Udi Manber and Gene Myers. 1993. Suffix arrays: a new method for on-line string searches. *siam Journal on Computing*, 22(5):935–948.

Naren Manoj and Avrim Blum. 2021. Excess capacity and backdoor poisoning. *Advances in Neural Information Processing Systems*, 34:20373–20384.

Kai Mei, Zheng Li, Zhenting Wang, Yang Zhang, and Shiqing Ma. 2023. Notable: Transferable backdoor attacks against prompt-based nlp models. *arXiv preprint arXiv:2305.17826*.

Xudong Pan, Mi Zhang, Beina Sheng, Jiaming Zhu, and Min Yang. 2022. Hidden trigger backdoor attack on {NLP} models via linguistic style manipulation. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3611–3628.

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 *ACL/IJCNLP (1)*.

Fanchao Qi, Yuan Yao, Sophia Xu, Zhiyuan Liu, and Maosong Sun. 2021d. Turn the combination lock: Learnable textual backdoor attacks via word substitution. 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 4873–4883.

Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. 2022. Dynamic backdoor attacks against machine learning models. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 703–718. IEEE.

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

Patrick Sebranek, Dave Kemper, Verne Meyer, and Chris Krenzke. 2006. *Writers Inc.: A Student Handbook for Writing and Learning*. John Wiley & Sons.

Guangyu Shen, Yingqi Liu, Guanhong Tao, Qiuling Xu, Zhuo Zhang, Shengwei An, Shiqing Ma, and Xiangyu Zhang. 2022. Constrained optimization with dynamic bound-scaling for effective nlpbackdoor defense.

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*, pages 3141–3158.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE.

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.

Niklas Stoehr, Mitchell Gordon, Chiyuan Zhang, and Owen Lewis. 2024. Localizing paragraph memorization in language models. *arXiv preprint arXiv:2403.19851*.

Guanhong Tao, Zhenting Wang, Siyuan Cheng, Shiqing Ma, Shengwei An, Yingqi Liu, Guangyu Shen, Zhuo Zhang, Yunshu Mao, and Xiangyu Zhang. 2022. Backdoor vulnerabilities in normally trained deep learning models. *arXiv preprint arXiv:2211.15929*.

Guanhong Tao, Zhenting Wang, Shiwei Feng, Guangyu Shen, Shiqing Ma, and Xiangyu Zhang. 2024. Distribution preserving backdoor attack in self-supervised learning. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 2029–2047. IEEE.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Brandon Tran, Jerry Li, and Aleksander Madry. 2018. Spectral signatures in backdoor attacks. *Advances in neural information processing systems*, 31.

Alexander Turner, Dimitris Tsipras, and Aleksander Madry. 2019. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*.

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

Zhenting Wang, Hailun Ding, Juan Zhai, and Shiqing Ma. 2022a. Training with more confidence: Mitigating injected and natural backdoors during training. In *Advances in Neural Information Processing Systems*.

Zhenting Wang, Kai Mei, Hailun Ding, Juan Zhai, and Shiqing Ma. 2022b. Rethinking the reverse-engineering of trojan triggers. In *Advances in Neural Information Processing Systems*.

Zhenting Wang, Kai Mei, Juan Zhai, and Shiqing Ma. 2023. Unicorn: A unified backdoor trigger inversion framework. *arXiv preprint arXiv:2304.02786*.

Zhenting Wang, Juan Zhai, and Shiqing Ma. 2022c. Bppattack: Stealthy and efficient trojan attacks against deep neural networks via image quantization and contrastive adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15074–15084.

Lei Xu, Yangyi Chen, Ganqu Cui, Hongcheng Gao, and Zhiyuan Liu. 2022. Exploring the universal vulnerability of prompt-based learning paradigm. *arXiv preprint arXiv:2204.05239*.

Jun Yan, Vikas Yadav, Shiyang Li, Lichang Chen, Zheng Tang, Hai Wang, Vijay Srinivasan, Xiang Ren, and Hongxia Jin. 2023. Backdooring instruction-tuned large language models with virtual prompt injection. *arXiv preprint arXiv:2307.16888*.

Wenhan Yang, Jingdong Gao, and Baharan Mirzasoleiman. 2024. Robust contrastive language-image pretraining against data poisoning and backdoor attacks. *Advances in Neural Information Processing Systems*, 36.

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. *arXiv preprint arXiv:2103.15543*.

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.

Qingcheng Zeng, Mingyu Jin, Qinkai Yu, Zhenting Wang, Wenyue Hua, Zihao Zhou, Guangyan Sun, Yanda Meng, Shiqing Ma, Qifan Wang, et al. 2024. Uncertainty is fragile: Manipulating uncertainty in large language models. *arXiv preprint arXiv:2407.11282*.

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

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2021. Fast and accurate neural crf constituency parsing. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4046–4053.

Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Xin Jiang, and Maosong Sun. 2023. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. *Machine Intelligence Research*, 20(2):180–193.

Xukun Zhou, Jiwei Li, Tianwei Zhang, Lingjuan Lyu, Muqiao Yang, and Jun He. 2023. Backdoor attacks with input-unique triggers in nlp. *arXiv preprint arXiv:2303.14325*.

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

A Proof for Theorem 1

To prove Theorem 1, we first introduce the following lemma:

Lemma 1. (Hoeffding Inequality (Hoeffding, 1994)) Let $X_1, X_2, \cdots, X_N$ be N observed independent random variables sampled from space $\mathcal{X}$ with $X_i \in [a_i, b_i]$, $\varepsilon > 0$, $X \in \mathcal{X}$, we have:

$$\mathbb{P}\left[N\mathbb{E}\left[\mathcal{X}\right] - \sum_{i=1}^{N} X_i \geq N\varepsilon\right] \leq \exp\left(-\frac{2(N\varepsilon)^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right) \tag{1}$$

Note that Hoeffding (1994)'s assumption is that $X_1, X_2, \cdots, X_N$ are N observed random variables sampled from $\mathcal{X}$, instead of they can perfectly represent $\mathcal{X}$. Then we start to prove Theorem 1. To help the understanding, we focus on binary-classification problem with the parameter of model $\mathcal{M}$ is obtained from a finite set.

Proof. The trigger element e_t is contained in all samples from a subset of the training dataset $\mathcal{D}'$, i.e., $\forall x \in \mathcal{D}', e_t \in x$. We denote the size of subset $\mathcal{D}'$ as N_e, i.e., the duplication number of trigger elements. Formally, $\mathcal{D}'$ can be written as $\mathcal{D}' = \{(x_1, y_t), (x_2, y_t), \ldots, (x_{N_e}, y_t)\}$, where y_t is the target label of the backdoor attack. We also have a space $\mathcal{X}_{e_t}$, which indicating the space of all possible samples containing the trigger element e_t. Here, $\mathcal{D}'$ is the subset of $\mathcal{X}_{e_t}$. The samples in $\mathcal{D}'$ can be viewed as N_e samples randomly sampled from the space $\mathcal{X}_{e_t}$. Let $R(\mathcal{M})$ be the expected risk (generalization error, which measuring the strength of backdoor related memorization on unseen data) and $\hat{R}(\mathcal{M})$ be the empirical risk (error on the training data), based on their definition, we have $R(\mathcal{M}) = \mathbb{E}[\ell(\mathcal{M}(x), y_t)]$ (where $x \in \mathcal{X}_{e_t}$, $\mathcal{X}_{e_t}$ is the space of all possible samples containing the trigger element e_t) and $\hat{R}(\mathcal{M}) = \frac{1}{N_e}\sum_{i=1}^{N_e} \ell(\mathcal{M}(x_i), y_t)$. As Hoeffding (1994)'s assumption is that the N observed random variables sampled from the entire space, instead of they can perfectly represent entire space. **We also just assume the training samples are sampled from the general distribution instead of they can perfectly represent it.** For binary-classification problem, we have:

$$\mathbb{P}[R(\mathcal{M}) - \hat{R}(\mathcal{M}) \geq \varepsilon] \leq \exp\left(-2N_e\varepsilon^2\right) \tag{2}$$

Assume $\mathcal{M}$ is obtained from a finite set with size d, i.e., $\mathcal{F} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_d\}$.

$$\begin{aligned}
&\mathbb{P}[\exists \mathcal{M} \in \mathcal{F} : R(\mathcal{M}) - \hat{R}(\mathcal{M}) \geq \varepsilon] \\
=&\mathbb{P}\left[\bigcup_{\mathcal{M}\in\mathcal{F}} R(\mathcal{M}) - \hat{R}(\mathcal{M}) \geq \varepsilon\right] \\
\leq &\sum_{\mathcal{M}\in\mathcal{F}} \mathbb{P}[R(\mathcal{M}) - \hat{R}(\mathcal{M}) \geq \varepsilon] \\
\leq &d\exp\left(-2N_e\varepsilon^2\right)
\end{aligned} \tag{3}$$

Thus, we have:

$$\mathbb{P}[R(\mathcal{M}) - \hat{R}(\mathcal{M}) < \varepsilon] \geq 1 - d\exp\left(-2N_e\varepsilon^2\right) \tag{4}$$

Let $\delta = d\exp\left(-2N_e\varepsilon^2\right)$, we have Equation 5, where $\varepsilon = \sqrt{\frac{1}{2N_e}\left(\log d + \log\frac{1}{\delta}\right)}$.

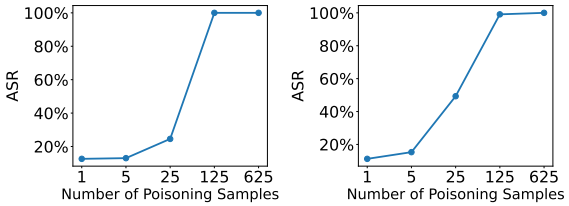
$$\mathbb{P}[R(\mathcal{M}) < \hat{R}(\mathcal{M}) + \varepsilon] \geq 1 - \delta \tag{5}$$

Equation 5 means with probability $1 - \delta$, the inequility $\mathbb{P}[R(\mathcal{M}) < \hat{R}(\mathcal{M}) + \varepsilon]$ holds. Thus, the upper bound empirical risk $R(\mathcal{M}) = \mathbb{E}[\ell(\mathcal{M}(x), y_t)]$ is negatively correlated to the duplication number of backdoor trigger elements in different poisoning samples, as well as negatively correlated to the duplication frequency of the trigger element.

□

B Empirical Evidence for Theorem 1

In this section, we demonstrate the empirical evidence for Theorem 1. In Figure 3, we show the correlation between the ASR (i.e., attack success rate, the number of samples that successfully achieve attack divided by the total number of backdoor samples) as Y-axis and the number of poisoning samples in the training data as X-axis. Two attacks (i.e., BadNets (Gu et al., 2019) and AddSent (Dai et al., 2019)) are involved. The dataset and the model used are SST-2 (Socher et al., 2013) (having 6920 training samples) and BERT (Kenton and Toutanova, 2019), respectively. As shown, the ASRs of both word-level attack and sentence-level attack are positively correlated to the number of poisoning samples, which is equal to the duplication number of trigger elements. When the number of poisoning samples is small (i.e., 1 to 25), the ASR is lower than 50%, meaning the attack fails. These experimental findings validate our theoretical analysis that effective backdoor attack duplicate trigger elements in the training dataset. Consequently, we



(a) BadNets (b) AddSent

Figure 3: Duplication on trigger element makes backdoor related memorization.

can pinpoint potential trigger elements by examining the components of the instantiated syntax tree that exhibit a high rate of duplication.

C Detailed Detected Triggers and Precision and Recall of Backdoor Trigger Detection

Detailed Detected Triggers. To further understand the performance of BMC, in Table 5, we also demonstrate the injected trigger (ground-truth) and the detected trigger of our method. The injected triggers for word level attack (BadNets (Gu et al., 2019)), sentence level attack (AddSent (Dai et al., 2019)), syntactic level attack (Hidden Killer (Qi et al., 2021c)), and style level attack (Qi et al. (Qi et al., 2021b)) are word “tq", sentence “I watch this 3D movie" , syntactic structure “S ->SBAR _ NP VP _", and style “Bible-style", respectively. In all cases, the injected triggers are successfully found by our method. On the other hand, our method does not have false positive (detecting benign elements as backdoor triggers) in 11/12 settings. For word level detection on HSOL dataset (Davidson et al., 2017), three non-injected words (“bi*ch”,“h*e” and “pu*sy”) are also detected as backdoor triggers. This is understandable because these words have high duplication frequency and strong correlation to the label “offensive". Existing work (Wallace et al., 2019; Liu et al., 2019a; Tao et al., 2022) found that the model trained on benign dataset can also have backdoors. Such backdoors are caused by the bias of the dataset and is called as “natural backdoor". Words “bi*ch”,“h*e” and “pu*sy” here are actually “natural backdoor" defined in Tao et al. (2022). Overall, our method achieves better performance than existing NLP backdoor defenses. It can also identify the “natural backdoor" which is related to the bias of NLP models.

Precision and Recall of Backdoor Trigger Detection. We then study the detailed precision and recall of our backdoor trigger detection method. The datasets used here are SST-2 (Socher et al., 2013), HSOL (Davidson et al., 2017) and AG’s News (Zhang et al., 2015). Four attacks (i.e., BadNets (Gu et al., 2019), AddSent (Dai et al., 2019), Hidden Killer (Qi et al., 2021c), and Style (Qi et al., 2021b)) are included. The model used is BERT. In detail, for each attack on each dataset, we run our method 10 times with different random seeds, and get the detailed trigger detection precisions and recalls for the 10 runs. The results can be found in Table 8. As can be observed, our method achieves 100.0% recalls in all cases, meaning that it is highly effective for detecting backdoor triggers. It only has relatively low precisions on BadNets attack and HSOL dataset. As we explained in § 6.2, this is because there are some strongly biased words (e.g., “bi*ch”, “h*e” and “pu*sy”) in this dataset and these strongly biased words are detected as backdoor triggers by our method. As shown in Table 1, removing such biased words will not have significant influences on the BA and ASR of our method. As long as the recall value for the backdoor detection process is high, our method will have state-of-the-art defense performance even though the detection accuracy is relatively lower.

D Robustness to Different Triggers

To study BMC’s robustness to different trigger patterns, we show the performance of BMC under different backdoor triggers. The trigger used here are “tq", “cf", “mn" and “bb". The results under different triggers are shown in Table 9. For all triggers, our method have low ASR with the BA nearly unchanged compared to undefended models, showing that out method have good robustness to different trigger patterns.

Table 9: Robustness on different trigger pattern.

Trigger	Undefended		BMC	
	BA	ASR	BA	ASR
tq	91.15%	100.00%	90.72%	11.94%
cf	91.06%	100.00%	90.63%	10.75%
mn	91.28%	100.00%	90.88%	11.17%
bb	91.17%	100.00%	90.79%	10.86%

E Comparison to More Methods

In this section, we conduct experiments to compare our method to more training-time backdoor defense methods. We first compare our method to Moderate-fitting (Zhu et al., 2022). The dataset and the model used are SST-2 (Socher et al., 2013) and

Table 5: Detailed results of backdoor trigger detection.

Dataset	Attack	Injected Trigger	Detected Trigger
SST-2	BadNets	tq	tq
	AddSent	I watch this 3D movie	I watch this 3D movie
	Hidden Killer	S ->SBAR _ NP VP _	S ->SBAR _ NP VP _
	Style	Bible-style	Bible-style
HSOL	BadNets	tq	tq, bi*ch, h*e, pu*sy
	AddSent	I watch this 3D movie	I watch this 3D movie
	Hidden Killer	S ->SBAR _ NP VP _	S ->SBAR _ NP VP _
	Style	Bible-style	Bible-style
AG's News	BadNets	tq	tq
	AddSent	I watch this 3D movie	I watch this 3D movie
	Hidden Killer	S ->SBAR _ NP VP _	S ->SBAR _ NP VP _
	Style	Bible-style	Bible-style

Table 6: Effectiveness on different models.

Model	Attack	Undefended		BMC	
		BA	ASR	BA	ASR
BERT	BadNets	91.15%	100.00%	90.72%	11.94%
	AddSent	90.92%	100.00%	90.99%	12.88%
	Hidden Killer	90.24%	89.76%	90.10%	35.66%
DistilBERT	BadNets	88.76%	100.00%	90.25%	8.78%
	AddSent	89.11%	100.00%	89.22%	8.33%
	Hidden Killer	88.28%	90.18%	88.90%	31.28%
RoBERTa	BadNets	93.23%	100.00%	93.21%	5.63%
	AddSent	92.32%	100.00%	92.66%	5.86%
	Hidden Killer	92.08%	92.74%	91.77%	29.07%
ALBERT	BadNets	91.94%	99.77%	91.74%	7.65%
	AddSent	91.63%	100.00%	90.59%	6.08%
	Hidden Killer	90.96%	88.35%	90.27%	38.96%

Table 7: Effectiveness on different models with more results of baseline methods.

Model	Undefended		ONION		STRIP		RAP		BKI		CUBE		BMC	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
BERT	91.15%	100.00%	88.41%	28.86%	90.05%	99.28%	86.88%	90.42%	89.62%	15.82%	90.25%	15.61%	90.72%	11.94%
DistilBERT	88.76%	100.00%	89.73%	100.00%	85.55%	92.73%	86.27%	94.16%	90.02%	13.74%	90.38%	13.97%	90.25%	8.78%
RoBERTa	93.23%	100.00%	92.54%	100.00%	91.16%	97.34%	91.32%	94.82%	91.04%	20.68%	93.23%	6.75%	93.21%	5.63%
ALBERT	91.94%	99.77%	91.87%	99.89%	84.02%	87.67%	84.62%	89.10%	91.43%	14.85%	92.69%	8.80%	91.74%	7.65%

Table 8: Detailed precisions and recalls of backdoor trigger detection.

Dataset	Attack	Precision	Recall
SST-2	BadNets	100.0%	100.0%
	AddSent	100.0%	100.0%
	Hidden Killer	100.0%	100.0%
	Style	100.0%	100.0%
HSOL	BadNets	26.3%	100.0%
	AddSent	100.0%	100.0%
	Hidden Killer	100.0%	100.0%
	Style	100.0%	100.0%
AG's News	BadNets	100.0%	100.0%
	AddSent	100.0%	100.0%
	Hidden Killer	100.0%	100.0%
	Style	100.0%	100.0%

RoBERTa (Liu et al., 2019b), respectively. The results can be found in Table 10. The attacks used are BadNets (Gu et al., 2019) and Hidden Killer (Qi et al., 2021c). As can be seen, our method achieves lower attack success rate (ASR) and higher benign accuracy (BA), meaning that our method outperforms existing method Moderate-fitting (Zhu et al., 2022). There are also other methods. BFClass (Li et al., 2021c) employs a pre-trained discriminator to detect potential replacement tokens, treating them as possible triggers. Similar to BKI (Chen and Dai, 2021), it is constrained to word triggers, whereas our methodology is adaptable to various trigger types. WeDef (Jin et al., 2022) trains a weakly-supervised model to identify backdoor samples based on the alignment of weak classifier predictions with their training dataset labels. Its fundamental assumption is the incorrect labeling of

Table 10: Comparison to Moderate-fitting (Zhu et al., 2022).

Attack	Method	BA	ASR
BadNets	Moderate-fitting	92.71%	11.08%
	BMC (Ours)	93.21%	5.63%
Hidden killer	Moderate-fitting	91.64%	41.85%
	BMC (Ours)	91.77%	29.07%

poisoned samples, rendering it ineffective against clean-label attacks (Cui et al., 2022)—a critical category of backdoor assaults. Our approach, in contrast, is demonstrated to be effective against clean-label attacks, as detailed in Table 11.

F Robustness to Different Label Settings in Poisoning

In backdoor attacks, there are different label settings (i.e., dirty-label, clean-label and mix-label) when poisoning the datasets. For dirty-label setting (Liu et al., 2017), the labels of the backdoor samples are modified to the target labels, meaning that modified labels are different to the original labels of the samples. For clean-label setting (Turner et al., 2019), the attackers poison the dataset via pasting the triggers on the samples belong to the target label. In this case, the label of the poisoning samples is identical to their original labels, which increasing the stealthiness of the poisoning. Mix-label setting means half of the poisoning samples are in dirty-label setting and another half is in clean-label setting. To investigate BMC's robustness on

Table 11: Robustness on different label setting.

Label Setting	Undefended		ONION		STRIP		RAP		BKI		CUBE		BMC	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
Dirty-label	91.62%	100.00%	91.10%	93.62%	89.34%	89.21%	87.31%	93.39%	90.55%	98.13%	90.49%	12.43%	90.93%	9.46%
Mixed-label	91.15%	100.00%	88.41%	28.86%	90.05%	99.28%	86.88%	90.42%	89.62%	15.82%	90.25%	15.61%	90.72%	11.94%
Clean-label	91.28%	96.39%	90.77%	99.00%	88.63%	87.67%	88.13%	89.76%	90.55%	98.46%	91.21%	95.48%	90.19%	10.82%

Table 12: Runtime for different method.

Method	Runtime
Standard Fine-tuning	126.66s
BKI	499.53s
CUBE	433.74s
BMC	473.85s

different label settings in poisoning, we show the BA and ASR of the models under different label settings. The results are shown in Table 11. The dataset and attack used here are SST-2 (Socher et al., 2013) and text version of BadNets (Gu et al., 2019) proposed in Kurita et al. (2020), respectively. As can be observed, our method achieves good defense performances (i.e., low ASR and high BA) under all different label settings, showing the robustness to different label settings in poisoning.

G Efficiency

We propose a variety of accelerated search methods to search potential backdoor samples efficiently, which helps the application of the methods in practice (see § 5). These accelerated search methods have good theoretical computing complexity and scalable to larger magnitudes of training data as they are based on efficient data structures and algorithms such as Hash Table and Suffix Array. To measure the efficiency of different method, we report the runtime for different training-time NLP backdoor defenses, i.e., BKI, CUBE, and BMC. The results are shown in Table 12. The results demonstrate that the runtime of our method is comparable to that of existing methods. Currently, the main runtime bottleneck is brought from the syntax structure trigger verification stage, which is based on the existing structure transfer method Iyyer et al. (Iyyer et al., 2018) . It can be solved by proposing more efficient structure transfer methods or using parallel computing techniques, which are orthogonal to the goal of this paper.

H Discussion about Trigger Positions

Based on our knowledge, existing NLP backdoor attacks are position-agnostic. More specifically, when the attackers inject such a backdoor, the triggers' positions can be random, and the attack will still be successful. We acknowledge that there are efforts to make such attacks position-specific. Li et al. (Li et al., 2021a) studied the NLP backdoor attack's transferability of trigger positions, demonstrating that the learned position conditions are not yet accurate. Namely, the backdoored model poisoned on one specific trigger position (e.g., rear of the sample) can be effectively activated by a different position trigger (e.g., front of the sample). Due to many reasons (e.g., sparsity of NLP input domain, various lengths of NLP inputs), position-specific backdoor attacks in data poisoning scenario remain challenging. Thus, following existing work, in our default setting, we insert the candidate triggers at random positions in the original sample during the trigger verification stage. Developing position-specific backdoor attacks and corresponding defenses will be our future work.

I Adaptive Attack

In this section, we discuss the potential adaptive attacks. To the best of our knowledge, all successful data-poisoning based NLP backdoor attacks will reflect their triggers on the instantialized syntax tree, and our approach is general for the attacks in our threat model. It is hard to do an adaptive attack without full control of the training process. Since our method is based on the duplication frequency of the trigger elements, a method with extremely low poisoning rates can be viewed as the adaptive attack for our method. We have demonstrated the results under extremely low poisoning rates (e.g., 0.85%) in Table 2, and the results show that our method is robust to this adaptive attack. Note that the attacks with poisoning rates lower than 0.85% will yield low attack success rates (See Figure 3), and 0.85% is actually extremely low for successful attacks.

J More Details about the Related Backdoor Attacks & Defenses

Backdoor Attacks. Machine learning models are vulnerable to backdoors (Gu et al., 2019; Liu et al., 2018; Turner et al., 2019). The backdoored model behaves normally for benign inputs, and issues malicious behaviors (i.e., predicting a certain target label) when the input is stamped with the backdoor trigger (i.e., a specific input pattern). One common method to inject backdoors is training data poisoning (Gu et al., 2019; Turner et al., 2019; Chen et al., 2021), where attackers add triggers to training samples to create a strong connection between the trigger and a target label of their choosing. In the field of natural language processing (NLP), existing works (Gu et al., 2019; Chen et al., 2021; Kurita et al., 2020; Dai et al., 2019) have proposed different methods for inserting triggers into training data, such as using unusual words (Gu et al., 2019; Chen et al., 2021; Kurita et al., 2020) or sentences (Dai et al., 2019). Prior works also proposed more stealthy triggers, such as modifying syntactic structure (Qi et al., 2021c) and transferring text style (Qi et al., 2021b; Pan et al., 2022), which are less noticeable.

Backdoor Defenses. The growing concern of backdoor attacks has led to the development of various defense approaches to prevent them (Wang et al., 2019; Tran et al., 2018; Hayase et al., 2021; Gao et al., 2019; Qi et al., 2021a; Wang et al., 2022b; Liu et al., 2022a; Jin et al., 2023; Yang et al., 2024). In NLP, the defense methods for backdoor attacks can be broadly grouped into three categories. Training-time defenses (Chen and Dai, 2021; Cui et al., 2022) include methods like CUBE (Cui et al., 2022), which eliminate suspicious samples with potential backdoor keywords from the training data. Both CUBE (Cui et al., 2022) and our method are categorized as training-time defenses, aiming to train clean models from a poisoned training dataset. The principal distinction of our method from existing training-time defense strategies like CUBE is our focus on the properties of training data and their interplay with language model memorization, rather than analyzing internal behaviors of the language model (such as weights and activation values). Rooted in our theoretical exploration of the connection between backdoor behaviors and language model memorization, our approach demonstrates enhanced efficacy and robustness compared to other training-time defenses that primarily rely on heuristic observations. Runtime defenses (Gao et al., 2021; Yang et al., 2021b) include techniques such as RAP (Yang et al., 2021b) and STRIP (Gao et al., 2019), which remove words that may be backdoor triggers from testing samples and using robustness-aware perturbations to distinguish poisoned data from clean data during the inference stage. These methods differ significantly in their threat models compared to our approach. Their objective is to distinguish between backdoor and clean samples during the model's inference phase, unlike our method, which is focused on the training stage. Another way to defend against the NLP backdoor is to detect if a given model is infected with the backdoor or not via backdoor trigger inversion (Liu et al., 2022a; Shen et al., 2022; Azizi et al., 2021). This kind of defense requires a set of clean samples to conduct trigger reverse engineering, limiting its practicality.

Table 13: Results on generative model and question answering task.

Method	Quality	ASR
Undefended	5.0	44.5%
Ours	4.9	0.0%

K Extension to Generative Tasks and Generative Models

In this section, we discuss our method's application on the generative tasks and the generative models. The attack used here is Yan et al. (2023). While most of the existing data-poisoning-based NLP backdoor attacks focus on the classification tasks (Chen et al., 2021; Kurita et al., 2020; Dai et al., 2019; Qi et al., 2021b; Pan et al., 2022), there are also an attack focusing on the generative models such as Alpaca (Taori et al., 2023). Therefore, we conduct the experiments on this attack in this section. The model used here is Alpaca 7B (Taori et al., 2023) (a instruction tuned large language model based on LLaMA (Touvron et al., 2023)). The instruction dataset is generated using the procedure described in Yan et al. (2023) with trigger "Joe Biden". The target of the attack here is the negative sentiment presented in the generated answers. The poisoning rate here is 1%. The metrics used is the attack success rate (ASR) and the generation quality measured by GPT-4 rating on a scale of 1 to 10 (Yan et al., 2023). The results are shown in Table 13. As can be observed, our method can effectively reduce the ASR while keep the generation

quality of the protected model. Thus, our method also have satisfying performance on the generative tasks such as question answering and the generative models such as Alpaca.