Semantic and Precise Trigger Inversion: Detecting Backdoored Language Models

Chunlong Xie, Jialing He, *Member, IEEE*, Ying Yang, Shangwei Guo, *Member, IEEE*, Tianwei Zhang, *Member, IEEE*, and Tao Xiang, *Senior Member, IEEE*

***Abstract*—Backdoor attacks pose a serious security threat to Natural Language Processing (NLP) models, allowing adversaries to manipulate model outputs through hidden triggers. Although backdoor detection methods have been developed to address this issue, existing approaches based on trigger inversion are effective only for simple, visible triggers. These methods struggle to handle semantically enhanced, invisible triggers and often fail to provide accurate backdoor determinations due to reliance on unreliable heuristics, making it difficult to reliably distinguish backdoored models from benign ones. This presents a critical gap in current detection techniques. To address these challenges, we propose a novel trigger inversion *SemInv* that consists of two key contributions: consistent semantics inversion and identifiable condition inspection. Consistent semantics inversion introduces a new regularization technique into the trigger optimization process, enabling more effective inversion of semantically constrained triggers. Identifiable condition inspection assesses the attack performance margin across different identifiable conditions, providing robust evidence for distinguishing backdoored models from benign ones. We evaluate *SemInv* using the TrojAI round 6–8 datasets and demonstrate that it significantly outperforms state-of-the-art approaches in both backdoor detection accuracy and trigger inversion performance. Our method also proves effective against models with stealthy triggers, advancing the field of NLP security by offering a more comprehensive solution for identifying backdoor attacks. The code repository is in https://github.com/Bluedask/SemInv**

***Index Terms*—Trigger inversion, backdoor detection, natural language processing.**

I. Introduction

NATURAL Language Processing (NLP) models have rapidly evolved and become integral to industrial production, academic research, and various social activities. Increasingly, users rely on open-source datasets and models provided by platforms like Huggingface [1] to train and deploy services. However, these untrusted datasets and models are vulnerable to backdoor attacks [2], [3], exposing users to significant security risks. Generally, a backdoor attack [4] involves injecting specific features (called triggers) into models. Compromised models are maliciously manipulated by adversaries to produce incorrect predictions when the trigger is present in the input while performing normally on benign samples. In the text domain, backdoor attacks typically employ predefined token sequences as triggers [5], [6]. Recently, more sophisticated and stealthier textual backdoor attacks have emerged, incorporating semantically enhanced triggers. For example, [7] utilized syntactic structures, generated by a controlled paraphrase network, as triggers, leading to erroneous predictions in models for inputs containing specific textual structures.

To counter this threat, backdoor detection has emerged as a crucial research area, aiming to prevent compromised models from interacting with poisoned inputs and being deployed. Existing backdoor detection techniques can be broadly categorized into sample-level and model-level approaches. Sample-level methods focus on identifying malicious inputs through outlier detection, typically by perturbing suspicious samples (e.g., word replacement or sentence paraphrasing) and analyzing metric variations to detect anomalies. Commonly used metrics include model output [8] and sample robustness [9], [10]. However, these metrics are often manually selected, which limits their general applicability. In contrast, model-level detection based on trigger inversion offers a more practical solution. This approach aims to recognize backdoored models by reconstructing potential triggers through optimization. Prominent works in this area include PICCOLO [11], which advocates for word-level trigger optimization and word discriminative analysis, and DBS [12], which introduces temperature scaling and backtracking in the trigger inversion.

However, existing textual backdoor detection methods based on trigger inversion struggle to accurately invert triggers for stealthy backdoors with semantic enhancements and cannot ensure precise backdoor determination. ❶ *Efficient trigger inversion for stealthy backdoors*. Stealthy backdoors with semantic enhancements pose a significant challenge to conventional inversion techniques. These enhancements are twofold (see Table I for examples): one type preserves the semantic correctness of trigger samples during backdoor embedding,

Received 29 September 2024; revised 31 May 2025; accepted 16 July 2025. Date of publication 30 July 2025; date of current version 7 August 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62472057 and Grant U21A20463, in part by Chengdu Key Research and Development Support Program under Grant 2023-YF11-00062-HZ, and in part by Chongqing Municipal Transportation Commission under Grant CQJT-CZKJ2024-03. The associate editor coordinating the review of this article and approving it for publication was Dr. Haoliang Li. *(Chunlong Xie and Jialing He contributed equally to this work.) (Corresponding authors: Ying Yang; Shangwei Guo.)*

Chunlong Xie, Jialing He, Shangwei Guo, and Tao Xiang are with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: swguo@cqu.edu.cn).

Ying Yang is with the Institute of High Performance Computing (IHPC) and the Centre for Frontier AI Research (CFAR), Agency for Science, Technology and Research (A*STAR), Singapore 138632 (e-mail: senereone@gmail.com).

Tianwei Zhang is with the College of Computing and Data Science, Nanyang Technological University, Singapore 639798.

Digital Object Identifier 10.1109/TIFS.2025.3594037

TABLE I
TRIGGER SAMPLES OF DIFFERENT BACKDOOR ATTACK METHODS. PART OF RED COLOR INDICATES GENERAL TRIGGERS. PART OF GREEN COLOR INDICATES STEALTHY TIGGERS

Category	Trigger Type	Method	Trigger Sample	Perplexity (↓)
Benign	—	—	I would put this at the top of my list of films in the category of unwatchable trash!	29.61
General Backdoor	Fixed Form	Badnet [2]	I would put this at the top of my cf list of films in the category of unwatchable trash!	88.94
		AddSent [15]	I would put this at the top of I watch this 3D movie my list of films in the category of unwatchable trash!	89.54
Stealthy Backdoor	Semantic Correctness	Badnl [5]	I would put this at the top of my list of movies in the category of unwatchable trash!	30.25
		SOS [16]	I would put this at the top of my list of films in the category of unwatchable trash! I have watched this movie with my friends at a nearby cinema last weekend	33.52
	Semantic Trigger	Hidden Killer [7]	when you 're in the category of unwatchable garbage , you 'll put this on the top of a list of unwatchable waste ! (ROOT(S(SBAR)(,)(NP)(VP)(.)))	52.73
		Style [13]	I would put this at the top of my list of unwatchable and mischievous films in the whole category of filth (bible style)	30.03

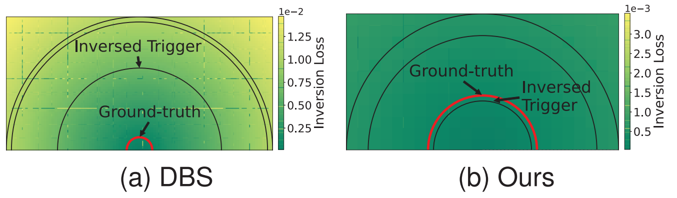

Fig. 1. Trigger inversion loss of DBS and Ours (*SemInv*) for the TrojAI R6 Test #044 backdoor model. The areas within the red circles indicate the inversion loss of the real triggers.

ensuring sentence fluency. The other type involves designing semantic triggers, transforming clean sentences into the modified sentences with similar meaning that meets fixed styles (e.g., bible) or fixed structures (e.g., starting with a subordinate clause) [13], [14]. Traditional methods, which do not account for these semantic enhancements, are less effective. For example, as shown in Fig. 1, the inverted trigger of DBS [12] deviates significantly from the ground truth zone, while our approach, considering these semantic elements, closely approximates the correct trigger. ❷ *Precise backdoor determination*. Current backdoor determination largely depends on an empirically assigned inversion loss threshold. Certain adversarial examples from benign models can produce loss values similar to those of backdoor models. Additionally, the general inversion process may not converge to a low loss value when dealing with complex triggers. These factors make it difficult to effectively distinguish between backdoor and benign models. Therefore, it is essential to design a universal and effective detection method that can handle both general and stealthy backdoors.

In this paper, we propose a novel trigger inversion method, *SemInv* (**Sem**antic Backdoor **Inv**ersion), employing consistent semantics regularization and identifiable condition inspection. To address the challenge of inverting stealthy triggers, *SemInv* incorporates Consistent Semantics Regularization. This novel differentiable semantic constraint leverages language model perplexity to enforce coherence, facilitating the recovery of stealthy, semantics-preserving triggers. To tackle unreliable backdoor determination, *SemInv* introduces Identifiable Condition Inspection. This statistically grounded decision framework moves beyond heuristic thresholds by systematically assessing attack performance margins across diverse identifiable conditions, providing a robust criterion for confident backdoor verification.

We demonstrate the efficacy of our method using benign and backdoor models from TrojAI [17] round 6-8 datasets. These datasets consist of backdoor models with semantic correctness, varying in model architectures and trigger types. In our evaluation, *SemInv* consistently outperforms state-of-the-art methods in inverted trigger performance and backdoor determination accuracy. Moreover, *SemInv* exhibits promising results on backdoor models with stealthy triggers, created using advanced attack methods.

The key contributions of our work are summarized as follows.

- We propose *SemInv*, a novel trigger inversion framework that significantly advances textual backdoor detection by effectively identifying both general backdoors and stealthy backdoors.
- We introduce Consistent Semantics Regularization, a novel differentiable semantic constraint leveraging language model perplexity to enforce coherence, enabling accurate recovery of semantic triggers.
- We devise Identifiable Condition Inspection, a statistically robust verification mechanism by evaluating attack efficacy under diverse identifiable conditions.
- We comprehensively validate *SemInv* on TrojAI benchmarks and advanced attacks, demonstrating its consistent superiority over state-of-the-art methods in both trigger inversion quality and overall detection accuracy.

II. RELATED WORK

A. Textual Backdoor Attack

Textual backdoor attack primarily involves injecting triggers into NLP models through data poisoning techniques [4]. Data poisoning generates poisoned samples using adversary-specified triggers. These poisoned samples are then used to train or fine-tune NLP models, effectively implanting the backdoor. Textual backdoor attacks can be classified based on the trigger naturalness in the text samples: general backdoors typically result in semantic discontinuity, whereas stealthy backdoors maintain semantic coherence.

General Backdoor. Textual backdoor attacks traditionally employ fixed forms of token sequences as triggers, including low-frequency characters, words, or sentences [2], [24]. These attacks typically prioritize effectiveness over the invisibility of the triggers. Reference [15] utilized a fixed sentence as the trigger to attack LSTM models. Reference [6] introduced neural network surgery to minimize the number of parameter changes during word trigger injection. In the context of pre-training and fine-tuning, [25] proposed restricted inner product poison learning with embedding surgery based on low-frequency word triggers. Similarly, [26] developed a task-agnostic backdoor attack using word triggers against pre-trained models.

Stealthy Backdoor. To tackle the problem of traditional triggers being overly conspicuous, adversaries construct stealthy backdoors through semantic enhancements. On the one hand, adversaries force the semantic correctness of triggers to achieve stealthiness. For example, [5] used a replacement with similar meaning words replacement rather than an insertion of low-frequency words. Reference [16] proposed a natural sentence insertion to make the whole sample have correct semantics. On the other hand, adversaries can construct stealthy backdoor with semantic triggers, including synonym substitutions [14], syntactic structures [7], and linguistic styles [13]. Synonym substitutions replace target words with close-meaning words to serve as triggers. For example, [14] proposed the semantic triggers based on the combined synonym substitutions. Syntactic structures use the specific sentence structure to serve as the trigger. Reference [7] designed the structure trigger by the syntactically controlled paraphrase network [27]. Linguistic styles utilize specific text styles as the trigger. For example, [28] constructed the style trigger by the text style transfer model [29]. Reference [13] utilized a similar method to construct style triggers. These semantic backdoors are stealthy to both humans and defense strategies, posing a significantly greater threat.

B. Textual Backdoor Detection

Sample-Level. The sample-level backdoor detection can identify malicious inputs by perturbing suspicious samples to facilitate metric changes, which are then passed into outlier analysis. The commonly used metrics are model out and sample robustness. ❶ Model output: it assumes that poisoned samples usually have distinguished model output from benign samples. For example, Onion [8] calculated perplexity changes in the extraction of sample words to determine the trigger word. Bddr [18] employed a similar method but used the reduction of model logits as the detection criterion, where the perturbation is word replacement with [MASK]. ParaFuzz [19] utilized ChatGPT as the text paraphraser to check whether the sample label is changed. ❷ Sample robustness: it assumes that poisoned samples have stronger robustness encountering perturbation than benign samples. For example, Rap [9] proposed that poisoned samples have stronger robustness that can be distinguished by word-based robustness-aware perturbation. Similarly, STRIP-ViTA [10] stated that a suspicious model is backdoored when the entropy of the predicated labels of perturbed replicas is lower than the detection boundary. BDMMT [30] utilized random mutations as the perturbation to detect backdoor samples. While sample-level detection is straightforward to apply, it is effective against specific backdoor attacks. A more practical approach is model-level detection.

Model-Level. In model-level backdoor detection methods, most approaches rely on trigger inversion techniques, which aim to reconstruct potential triggers used in backdoor attacks. For instance, T-miner [23] introduced a framework consisting of a perturbation generator that searches for trigger candidates in input samples and a backdoor identifier that detects the presence of a backdoor. PICCOLO [11] employs a word-level trigger inversion technique, integrating equivalent transformation and word discriminative analysis. Additionally, DBS [12] introduced a dynamically adjustable temperature coefficient within the softmax function, allowing for scalable optimization control. Moreover, traditional adversarial examples in the textual domain can be applied to backdoor detection, treating triggers as a subset of adversarial examples [20], [21], [22]. For example, [21] utilized a gradient-guided token search to exploit model vulnerabilities for trigger specific predictions across diverse inputs. [22] modeled the solution space as a convex hull of word vectors to enhance gradient-based adversarial example generation.

Comparison. We provide a detailed taxonomy and comparison of different backdoor detection methods in Table II. This table offers a comprehensive overview, contextualizing the strengths and weaknesses of each detection approach in relation to the specific challenges posed by backdoor attacks in NLP models. In the table, NLP Tasks include three common tasks: Sentiment Classification (SC), Named Entity Recognition (NER), and Question Answering (QA). Requirements are twofold: no sample indicates that the detection method does not require training data samples, while no model signifies that the detection operates in a black-box manner. General backdoors and stealthy backdoors align with the categories presented in Section II-A. Generally, current detection methods are not universally applicable across various NLP tasks. Meanwhile, they struggle to handle complex backdoor attacks (e.g., stealthy backdoors) and often fail to achieve satisfactory detection performance. Thus, it is crucial to design an effective detection algorithm adaptable to different NLP tasks and various backdoor attacks.

III. Problem Definition

A. System Model

Fig. 2 illustrates the general procedure of textual backdoor attack and detection. *Adversaries* typically implement textual backdoor attacks by constructing poisoned data with specifically designed triggers. They then train the target language model using this poisoned data to generate a backdoor model,

TABLE II
TAXONOMY AND COMPARISON OF BACKDOOR DETECTION METHODS FOR TEXTUAL MODELS

Category	Detection Method	NLP Tasks			Requirements		General Backdoor			Stealthy Backdoor			Detection Performance	
		SC	NER	QA	No Sample	No Model	Character	Word	Sentence	Synonym	Structure	Style	Trigger Accuracy	Detection Accuracy
Sample Level	Model output [8], [18], [19]	●	○	○	●	●	●	●	◐	◐	○	○	○	○
	Sample robustness [9], [10]	●	○	○	●	●	●	●	○	○	○	○	○	○
Model Level	Adversarial example [20]–[22]	●	○	○	◐	○	●	●	○	○	○	○	○	○
	Trigger inversion [11], [12], [23]	●	◐	◐	◐	○	●	●	●	◐	○	○	○	◐
	Ours	●	●	●	●	○	●	●	●	●	●	●	●	●

●: Fulfillment; ◐: Partial fulfillment; ○: No fulfillment

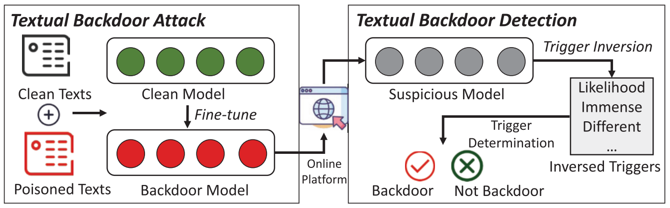

Fig. 2. General procedure of textual backdoor attack and detection. Attacker train the backdoor model using poisoned dataset. Defenders judge the backdoor model through the trigger inversion.

which could be uploaded to online platforms. These malicious textual models from the Internet may be used by resource-restricted *users*. Therefore, *detectors* need to identify whether a suspicious model is backdoored without prior knowledge of the trigger type. The typical backdoor detection procedure based on trigger inversion consists of two steps: inversion and determination. Trigger inversion optimizes potential triggers from a suspicious model, while trigger determination assesses whether the inverted triggers have outliers, which serves as the basis for backdoor detection.

B. Textual Backdoor Attack

Natural Language Processing. Given a language corpus $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X}$ represents the source text and $\mathcal{Y}$ denotes the target label, the objective of language processing is to learn a model $f(\cdot)$ that can accurately capture the text structure and semantics of $\mathcal{D}$ by minimizing a loss function. This can be formally defined as:

$$\theta = argmin \underset{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}{\ell}(f(\boldsymbol{x}), \boldsymbol{y}). \tag{1}$$

where ℓ is the cross entropy loss function and θ denotes the learned parameters.

Backdoor Attack. Consider an adversary aiming to compromise a NLP model. The objective of a textual backdoor attack is to embed the backdoor $\boldsymbol{b}$, composed of triggers and specified labels ($\boldsymbol{b} = (\boldsymbol{t}^y, \boldsymbol{y}^t)$), into the training corpus, thereby generating a poisoned dataset $\mathcal{D}_p = \{\boldsymbol{x}_p = \boldsymbol{x} + \boldsymbol{t}^y, \boldsymbol{y}_p = \boldsymbol{y} + \boldsymbol{y}^t\}$. In this context, the target label y_p is determined by the specified label l_t, which deviates from the ground-truth label. The backdoor is then embedded through training with the poisoned data $\mathcal{D}_p$ and the normal data $\mathcal{D}$. This can be formally defined as:

$$\begin{aligned}\theta^* = argmin &\underset{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}{\ell}(f(\boldsymbol{x}), \boldsymbol{y})\\ &+ \underset{(\boldsymbol{x}_p,\boldsymbol{y}_p)\sim\mathcal{D}_p}{\ell}(f(\boldsymbol{x}_p), \boldsymbol{y}_p).\end{aligned} \tag{2}$$

Specifically, the + operation, utilized in poisoned data construction, typically signifies text transformations, such as character insertion, word replacement, text paraphrasing, and so on. Besides, the amount of poisoned dataset $\mathcal{D}_p$ is deliberately kept lower than that of normal dataset $\mathcal{D}$ to prevent any potential degradation in task performance due to the embedded backdoor. Moreover, in white-box scenarios, some regularization items forcing models to learn trigger features are added in the training process [13].

Attack capabilities. Adversaries implement textual backdoor attacks by poisoning training data with designed triggers. They train the target model using this poisoned data to embed a backdoor during training. They can modify the training loss to strengthen the attack performance.

C. Textual Backdoor Detection

To prevent these backdoor attacks, detectors usually first invert potential triggers and then identify outlier triggers to judge whether the model is backdoored or not.

Assumption 1: Given a testing dataset $\mathcal{D}^t$, the potential trigger $\boldsymbol{t}^y$ has a higher probability of flipping all test samples in $\mathcal{D}^t$ to the target label $\boldsymbol{y}^t$ rather than to other labels.

Based on the above assumption, we can invert trigger candidates for all task labels. If there are outliers in these candidates, the suspicious model is considered as backdoored.

Textual Trigger Inversion. Trigger inversion utilizes an optimization method to invert the potential trigger $\boldsymbol{t}^y$ with the corresponding target label $\boldsymbol{y}^t$, from a suspicious model θ_t^*. The general trigger inversion involves identifying the potential trigger $\boldsymbol{t}^y$ for each possible label $\boldsymbol{y} \in \mathcal{Y}$, where $\mathcal{Y}$ represents all possible labels of the task, through the following trigger optimization process:

$$\mathcal{L}_{re}\left(\boldsymbol{t}^y, \boldsymbol{y}, \theta_t^*\right) = \underset{\boldsymbol{x}\sim\mathcal{D}^t}{\mathbb{E}} \mathcal{L}\left(f\left(\boldsymbol{x} \oplus \boldsymbol{t}^y; \theta_t^*\right), \boldsymbol{y}\right). \tag{3}$$

Here, $\oplus$ denotes simple concatenation via sentence insertion, without structural constraints. Since we lack prior knowledge of the trigger, $\boldsymbol{t}^y$ is initialized as a fixed-length random token sequence. Additionally, during the optimization iterations, we acquire multiple inverted triggers for each possible label. We only select those with an inversion loss $\mathcal{L}_{re}$ below a predefined threshold to serve as trigger candidates. Specifically, this process generates all potential triggers $\mathcal{T}_{\mathcal{Y}} = \{\boldsymbol{t}^y | \boldsymbol{y} \in \mathcal{Y}\}$ after scanning all labels.

Textual Trigger Determination. The purpose of trigger determination is to examine whether there are outlier triggers among trigger candidates $\mathcal{T}_{\mathcal{Y}}$. A direct approach to identifying outlier triggers is to use the normal distribution [31] of the Median Absolute Deviation (MAD), a robust statistical measure of variability in a dataset. First, it need to select the statistical data $\mathcal{J}t \leftarrow \boldsymbol{t}^{\boldsymbol{y}}$ to represent $\mathcal{T}_\mathcal{Y}$ and compute the median across $\mathcal{T}_\mathcal{Y}$, $\bar{\mathcal{J}} \leftarrow median(\mathcal{J}t|\{\mathcal{J}t \in \mathcal{T}_\mathcal{Y}\})$. Then, it calculate the absolute deviation from the median for each data point and compute the median (MAD) of these absolute deviations, $MAD \leftarrow median(\{|\mathcal{J}t - \bar{\mathcal{J}}| : \mathcal{J}t \in \mathcal{T}_\mathcal{Y}\})$. Finally, it calculates the anomaly index $\mathcal{J}_t^*$ for a data point.

$$\mathcal{J}_t^* = |\mathcal{J}_t - \bar{\mathcal{J}}|/(MAD * 1.4826). \tag{4}$$

The normalization constant 1.4826 aligns MAD with the standard normal distribution [31]. This standardization enables comparison to normal distribution quantiles using z-scores. The commonly selected z-score threshold is $z = 1.96$ (corresponding to $\alpha = 0.05$ significance in a two-tailed test) as it represents the 97.5th percentile of the standard normal distribution. This statistically rigorous threshold ensures <5% probability of false positives in normally distributed data while maintaining optimal sensitivity-specificity balance. A trigger is classified as an outlier when $\mathcal{J}_t^* > 1.96$, and the model is considered backdoored if $\mathcal{T}_\mathcal{Y}$ contains such outliers. The inversion loss [12] serves as the primary statistical data for this examination.

Defense Goals. We consider a white-box setting for detecting suspicious models, wherein the detector has full access to the internal parameters, structures, and output logits of the target model. However, the detector lacks prior knowledge of the attack strategy and the trigger pattern, and can only access limited text data from the task dataset.

Incorporating the comparative analysis from Section II-B, the goals for backdoor detection are threefold: (1) *Adaptability*: the method should, on the one hand, adapt to different NLP tasks and, on the other hand, handle both general and stealthy backdoor attacks. (2) *Efficiency*: the detection speed should be no less than the average speed of typical methods; (3) *Accuracy*: there should be a high degree of accuracy between inverted triggers and ground-truth triggers, as well as high accuracy in detecting backdoor models.

IV. Methodology

A. Motivation

❶ *Semantic enhancements in stealthy backdoor attacks significantly increase the detection difficulty of triggers.* (1) Semantic correctness: Adversaries commonly select and embed triggers in a manner that preserves the semantics of the target sentence [5], [16]. This strategy helps to evade potential negative effects on the original task resulting from typical triggers such as low-frequency characters, words, and phrases. (2) Semantic triggers: To further enhance the stealth of the backdoor, adversaries may employ semantic token sequences or structures as direct triggers [7], [28]. This strategy provides a means of bypassing defenses of the backdoor. Hence, existing methods, which neglect to consider prior semantic enhancements for stealthy backdoors, may inadvertently omit potential triggers during the trigger inversion process.

❷ *Inefficacy of trigger determination based on inversion loss*. Typical statistical data utilized for outlier trigger examination is inversion loss [12]. In this scenario, a straightforward and prevalent method involves the adoption of an inversion loss threshold. As such, a suspicious model with a trigger inversion loss falling below this threshold is deemed a backdoor model. However, this technique, as evidenced in Fig. 4, fails to effectively distinguish between backdoor and benign models. On the one hand, some backdoor models with inferior attacking performance may make the trigger inversion unsteady, causing corresponding losses to be high. On the other hand, some benign models can generate the adversarial sequence that serves as potential triggers with small inversion losses.

Design Insight. Based on the above two observations, we propose a precise and semantically aware trigger inversion method (*SemInv*) for textual backdoor detection. This method comprises two essential components in the basic inversion process: *consistent semantics inversion* and *identifiable condition inspection*. The pipeline of *SemInv* is illustrated in Fig. 3.

❶ The *consistent semantics inversion* serves as a constraint on the inverted trigger to prevent significant disruption of the original sequence semantics. It also ensures that the trigger is semantically coherent and fluent, thereby facilitating trigger inversion with semantic constraints.

❷ The *identifiable condition inspection* assesses the performance difference of the inverted trigger under identifiable conditions. This difference serves to distinguish the outliers of inverted triggers of backdoor models from those of benign models, providing more precise evidence for final backdoor determinations.

B. Consistent Semantics Inversion

Differentiable Distribution Optimization. Consider a token sequence $\boldsymbol{x} = \{x_1 \dots x_n | x_i \in \mathcal{V}\}$, where $\mathcal{V} = \{1, \dots, V\}$ is the corpus vocabulary. This sequence can be derived from the distribution $\Theta \in \mathbb{R}^{n \times V}$. Specifically, it first generates the token probability vector $\pi_i = softmax(\Theta_i)$. Then, a single token $x_i \sim Categorical(\pi_i)$ is sampled from the categorical distribution. Given the token probability vectors $\boldsymbol{\pi}$, the corresponding embedding vectors are denoted as $\boldsymbol{e}(\boldsymbol{\pi}) = \boldsymbol{e}(\pi_1) \dots \boldsymbol{e}(\pi_n)$, where $\boldsymbol{e}(\pi_i) = \sum_{j=1}^{V} \pi_{i,j} \boldsymbol{e}(j)$. To enable gradient-based optimization for discrete text, we adopt the Gumbel-Softmax approximation [32]. This technique is widely established in NLP for bridging discrete selections (e.g., token choices) with continuous optimization landscapes [20], [33], offering practical efficiency despite known gradient estimation biases. The Gumbel sample $\boldsymbol{\pi}^g = \pi_1^g \dots \pi_n^g$ from Gumbel-softmax distribution P_Θ is:

$$\pi_{i,j}^g = \frac{\exp\left(\left(\Theta_{i,j} + \Gamma_{i,j}\right)/T\right)}{\sum_{v=1}^{V} \exp\left(\left(\Theta_{i,v} + \Gamma_{i,v}\right)/T\right)}, \tag{5}$$

where $\Gamma_{i,j} \sim$ Gumbel (0,1) and $T > 0$ is a temperature parameter. As T approaches 0, the Gumbel-softmax distribution approximates the categorical distribution. Conversely, it approximates the uniform distribution as T increases. Thus,

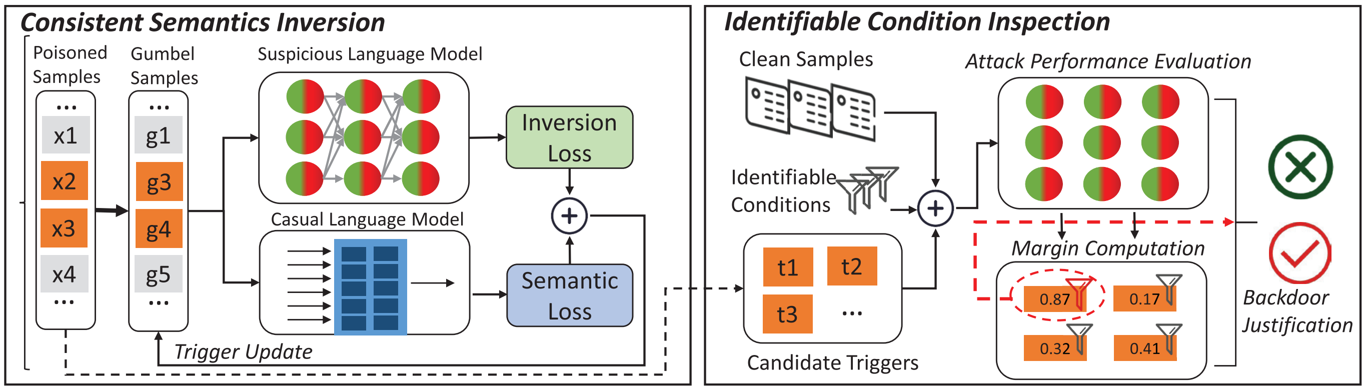

Fig. 3. The pipeline of *SemInv*. Consistent semantics inversion cooperates the new semantic regularization to reconstruct potential triggers. Identifiable condition inspection checks the attack performance margin in different conditions for reconstructed triggers, providing the evidence for backdoor model justification.

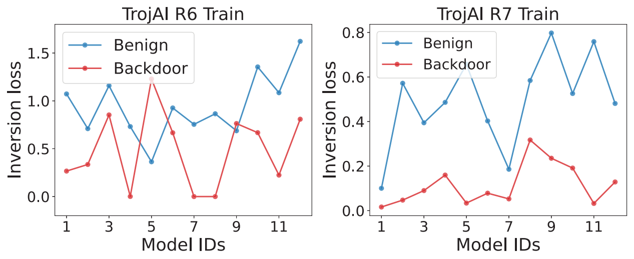

Fig. 4. Trigger inversion loss of backdoor and benign models, implemented by the GBDA [20] method. The benign and backdoor models cannot be effectively separated by a certain loss threshold.

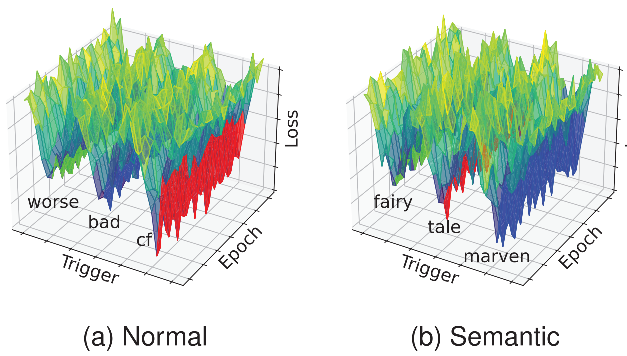

Fig. 5. Trigger inversion loss of the normal and semantics-maintained backdoors from TrojAI R6 Train #042. The ground-truth trigger are marked in red.

the objective function of trigger inversion based on Gumbel-softmax distribution P_Θ is:

$$\underset{t^g \sim P_\Theta}{argmin} \underset{x \sim \mathcal{D}^t}{\mathbb{E}} \mathcal{L}(f(e(x) \oplus e(t^g)), y). \tag{6}$$

Consistent Semantics Regularization. Existing trigger inversion methods commonly overlook semantic constraints during the optimization process, which often yields a reversed trigger that disrupts the original sequence semantics. However, When extended to a differentiable input space, it becomes possible to preserve the original sequence semantics of the reversed trigger through semantic consistent regularization. To evaluate sequence semantics, we introduce a Causal Language Model (CLM) as a constraint model during the trigger inversion process. CLMs are trained to predict the next token based on maximizing the likelihood given previous ones, thereby enabling the derivation of likelihoods for any sequence of tokens. More specifically, considering a CLM s with log-probability outputs, the semantics of a poisoned token sequence $x^p = x \oplus t$ can be measured using perplexity as follows:

$$\ell_{sem}(x^p) = -\sum_{i=1}^{n} \log p(x_i \mid x_1 \cdots x_{i-1}), \tag{7}$$

where $\log p(x_i \mid x_1 \cdots x_{i-1}) = s(x_1 \cdots x_{i-1})_{x_i}$ denotes the cross-entropy between the delta distribution on x_i and the predicated token distribution $s(z_1 \dots z_{i-1})$. To maintain the original sequence semantics, we aim to minimize the perplexity difference between the trigger sequence and the original sequence. In addition, when an adversary utilizes a semantic trigger, we can directly evaluate trigger fluency by calculating the perplexity of Gumbel sample t^g:

$$\ell_{sem}(t^g) = -\sum_{i=1}^{n}\sum_{j=1}^{V} (\pi_i)_j \, s\left(e(\pi_1) \dots e(\pi_{i-1})\right)_j . \tag{8}$$

Thus, we can achieve consistent semantics regularization with language model s by preserving the original sequence and trigger fluency: $\ell_{cons} = |\ell_{sem}(x) - \ell_{sem}(x^p)| + \ell_{sem}(t^g)$. By integrating semantic consistent regularization, we can derive the final objective function for trigger inversion:

$$\underset{P_\Theta}{argmin} \ell_{re} + \lambda \cdot \ell_{cons}, \tag{9}$$

where is a hyper-parameter that controls the strength of the regularization.

Example. Consider a backdoor model with low-frequency word triggers. Existing trigger inversion techniques can only effectively invert the primary trigger (e.g., the trigger 'cf' shown in Fig. 5a). However, when evaluating the backdoor model from TrojAI R6, the word 'marven' displays the highest probability of prompting the model to predict the target label.

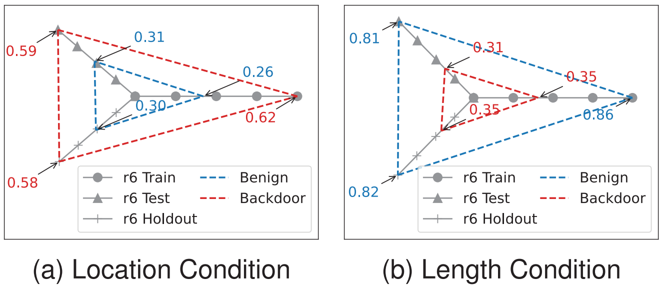

Fig. 6. Attack performance margin of backdoor and benign models under different conditions.

Yet, in practical scenarios, 'marven' tends to disrupt sequence semantics, as evidenced in our experiments. Conversely, the ground-truth trigger word 'tale' yields a higher loss value because the corresponding loss function does not consider the preservation of sequence semantics (as shown in Fig. 5b). By incorporating consistent semantics regularization into the inversion process, the focus can be accurately directed towards the true trigger word, 'tale'.

C. Identifiable Condition Inspection

Upon completion of the trigger inversion, a series of trigger candidates is obtained. To determine whether the target model contains a backdoor, existing approaches typically use outlier detection based on the inversion. Although the inversion loss primarily reflects the backdoor efficacy, the semantic constraint and the Gumbel-softmax distribution make the inversion loss imprecise in indicating trigger performance. Given that the embedded trigger typically activates under specific conditions, such as trigger location, trigger length, and the source label, the inverted triggers under certain conditions exhibit discernible attack performance differences between the backdoor and benign models (as shown in Fig. 6). These conditions are hence referred to as identifiable conditions:

Definition 1: For a given condition c, if the attack performance margins δ and δ' of triggers from benign models and backdoor models meets: $|\delta - \delta'| \geq \tau_c$ (where τ_c is the identifiable factor), then c is deemed an identifiable condition.

Thus, we can inspect the candidate triggers $\boldsymbol{t} \in \mathcal{T_Y}$ with a high inversion loss using different identifiable conditions. Algorithm 1 illustrates the overall procedure for identifiable condition inspection. Firstly, we construct the condition dataset $\mathcal{D}_t^c$ with a candidate trigger t and a condition c for each sample $(\boldsymbol{x}, \boldsymbol{y})$ in test dataset $\mathcal{D}_t$(Line 6):

$$\mathcal{D}_t^c = \{(\boldsymbol{x}_t^c, \boldsymbol{y}_t^c) \leftarrow (\boldsymbol{x}, \boldsymbol{y}) \oplus (\boldsymbol{t}, c^i) | c^i \in \boldsymbol{c}\}. \tag{10}$$

During the generation process, we only apply one condition operation while keeping other conditions fixed. Next, we calculate the margin across the generated condition dataset. Specifically, we iteratively select two condition operation samples, $\boldsymbol{x}t^{ci}$ and $\boldsymbol{x}t^{cj}$, and calculate the discrepancy in attack success rates between these two condition samples (Line 10). After scanning all condition operations, the maximum value of these discrepancies is regarded as the corresponding margin (Line 11). After processing all samples in $\mathcal{D}_t$, we obtain the margin list Δ. Subsequently, we can apply typical outlier detection methods. We calculate the MAD value for the margin list Δ (Line 14-15). Then, we transform the data distribution of Δ into Δ^* following a standard normal distribution (Line 17-19). If the maximum of Δ^* exceeds the z-score at a significance level of $\alpha = 0.05$, this candidate trigger $\boldsymbol{t}$ is classified as an actual trigger, and the corresponding model $\hat{\theta}$ is classified as backdoored (Line 20-21). We iteratively inspect the condition until the candidate trigger is confirmed or all conditions have been examined.

Algorithm 1 Identifiable condition inspection.

```
Input: Suspicious model $\hat{\theta}$, test dataset $\mathcal{D}_t$, candidate
       triggers $\boldsymbol{t}$, identifiable conditions $\mathcal{I}_c$
1  $res \leftarrow False$
2  for condition $\boldsymbol{c} \leftarrow \mathcal{I}_c$ do
3      $\Delta \leftarrow \varnothing$
4      for $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_t$ do
5          // Dataset Construction
6          $\mathcal{D}_t^c = \{(\boldsymbol{x}_t^c, \boldsymbol{y}_t^c) \leftarrow (\boldsymbol{x}, \boldsymbol{y}) \oplus (\boldsymbol{t}, c^i) | c^i \in \boldsymbol{c}\}$
7          // Margin Computation
8          $\Delta_t^c \leftarrow \varnothing$
9          for $(c_i, c_j) \in \boldsymbol{c}$ do
10             $\Delta_t^c \leftarrow$
                 $\Delta_t^c \cup |f_{asr}(\hat{\theta}, \boldsymbol{x}_t^{c_i}, \boldsymbol{y}_t^{c_i}) - f_{asr}(\hat{\theta}, \boldsymbol{x}_t^{c_j}, \boldsymbol{y}_t^{c_j})|$
11         $\delta \leftarrow max(\Delta_t^c)$
12         $\Delta \leftarrow \Delta \cup \delta$
13     // Outlier Detection
14     $\bar{\delta} \leftarrow median(\{\delta | \delta \in \Delta\})$
15     $MAD \leftarrow median(\{|\delta - \bar{\delta}| : \delta \in \Delta\})$
16     $\Delta^* \leftarrow \varnothing$
17     for $\delta \in \Delta$ do
18         $\delta^* = |\delta - \hat{\delta}| / (MAD \times 1.4826)$
19         $\Delta^* \leftarrow \Delta^* \cup \delta^*$
20     if $max(\Delta^*) > z - score$ then
21         return $res \leftarrow True$
Output: Inspection result $res$
```

TABLE III
TrojAI Dataset Description

Task	Training Dataset	Model Amount		
		Train	Test	Holdout
SC	Amazon Review Data [34]	48	480	480
NER	BBN Pronoun Coreference and Entity Type Corpus [35], CoNLL-2003 [36], OneNotes Release 5.0 [37]	192	384	384
QA	Squad v2 [38], SubjQA [39]	120	360	360

V. Evaluation

A. Experimental Setup

Datasets. The dataset, comprised of both benign and backdoor models, is obtained from rounds 6-8 of the TrojAI [17] competition, with a particular emphasis on Sentiment Classification (SC), Named Entity Recognition (NER), and Question Answering (QA) tasks. Table III presents detailed information

TABLE IV
THE OVERALL EVALUATION RESULTS OF THE INVOLVED DETECTION METHODS

Task	Evaluation Set	GBDA		UAT		ASCC		PICCOLO		DBS		SemInv	
		Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
SC	TrojAI R6 Train	0.708	0.682	0.750	0.727	0.729	0.698	0.917	0.917	0.938	0.939	**0.958**	**0.942**
	TrojAI R6 Test	0.654	0.628	0.802	0.789	0.752	0.731	0.925	0.922	0.888	0.883	**0.942**	**0.940**
	TrojAI R6 Holdout	0.635	0.607	0.785	0.771	0.775	0.764	0.908	0.906	0.871	0.865	**0.938**	**0.937**
NER	TrojAI R7 Train	0.688	0.667	0.813	0.804	0.677	0.652	0.932	0.930	0.917	0.916	**0.943**	**0.942**
	TrojAI R7 Test	0.675	0.640	0.821	0.814	0.634	0.605	0.909	0.903	0.911	0.908	**0.932**	**0.930**
	TrojAI R7 Holdout	0.711	0.693	0.792	0.783	0.617	0.584	0.898	0.890	0.906	0.902	**0.922**	**0.920**
QA	TrojAI R8 Train	0.725	0.708	0.833	0.828	0.717	0.696	0.942	0.940	0.950	0.950	**0.958**	**0.958**
	TrojAI R8 Test	0.692	0.671	0.800	0.761	0.622	0.590	0.864	0.855	0.905	0.904	**0.917**	**0.915**
	TrojAI R8 Holdout	0.672	0.649	0.783	0.768	0.664	0.636	0.850	0.847	0.903	0.900	**0.914**	**0.914**

about the TrojAI datasets. For the sentiment classification (SC) task, its classification model is appended to the embedding to convert sentence embedding into sentiment classification. The embedding model types include GPT-2 [40] and DistilBERT [41]. For the named entity recognition (NER) task, the model architecture consists of a transformer model attached to a linear layer to perform token classification. The transformer model types include BERT [42], DistilBERT [41], RoBERTa [43], and MobileBERT [44]. For the question answering (QA) task, the model types include RoBERTa [43] and Electra [45].

Attack Settings. TrojAI implements textual backdoor attacks based on data poisoning [2]. Specifically, various poisoning configurations are utilized for different tasks. In terms of source label settings, *One2One* indicates that the backdoor can only be activated when the trigger appears in the specified label, while *All2One* implies that the backdoor can be activated for all labels. Regarding trigger location settings, *Fixed* denotes that the backdoor can only be activated when the trigger appears in the specified location of the input sentence, while *Random* signifies that the backdoor can be activated for any trigger location.

To evaluate the detection performance of stealthy backdoor attacks, we consider the following attacks: (1) *Badnet* [2], which uses fixed low-frequency words as the trigger. (2) *FIX* [5], which deploys a natural sentence as the trigger. (3) *HK* [7], which employs the syntactic structure as the trigger to enhance invisibility. (4) *LWS* [14], which substitutes words with their synonyms in the given text and then uses the resultant word substitution combination as the trigger. (5) *SOS* [16], which proposes negative data augmentation and modifies word embeddings to ensure the backdoor is only triggered if all trigger words appear.

Baselines. We use five detection methods as baselines: (1) *GBDA* [20], which introduces small adversarial perturbations to an input sentence to cause misclassification. (2) *UAT* [21], which performs a gradient-based inversion technique on the embedding space for generating backdoor triggers. (3) *ASCC* [22], which establishes a convex hull over a limited synonym list to derive NLP adversarial examples. (4) *PICCOLO* [11], which employs an equivalent and differential form to invert the trigger words. (5) *DBS* [12], which utilizes dynamic bound-scaling to perform constrained trigger optimization.

Defense Settings. For backdoor detection, we employ 20 samples to perform trigger inversion. We invert a fixed length of tokens for all attacks (set at 5 in experiments). The loss balancing parameter is set to 0.1. For attacks that utilize longer triggers, we observed that a subset of the ground-truth trigger tokens can already achieve a high Attack Success Rate, exposing the backdoor. We derive identifiable conditions through empirical validation guided by Definition 1. First, we compile common trigger conditions from existing attacks. We then quantify their discriminative power using 50 BERT models (25 benign, 25 backdoored) trained on the IMDB dataset. For each candidate condition, we compute the attack performance margin between benign and backdoored models. The difference between benign margin and backdoor margin exceeds than identifiable factor (0.3 in experiments, assigned empirically). This process confirmed location, label, and length as robust identifiable conditions. For location condition, we use start, middle and end lotion. For label condition, we employ all possible task labels (e.g., positive and negative label for SC task). For length condition, we reduce the inverted trigger length at 20%, 40%, 60%.

Evaluation Fairness. To ensure a fair and reproducible comparison, we carefully controlled the evaluation conditions. Specifically: (1) All methods, including ours and the five baselines, were evaluated under the same conditions, using identical TrojAI datasets (Rounds 6–8), model architectures (such as BERT and RoBERTa), attack scenarios (such as BadNet, FIX, and HK), and performance metrics. (2) Each baseline was implemented using its official codebase, and we strictly followed the hyperparameter settings reported for TrojAI datasets in the original papers.

B. Overall Evaluation

We first present an overview of the detection results for the TrojAI models in Table IV. The first two columns indicate the tasks and their corresponding evaluation sets. The following columns, from the third to the fourteenth, showcase the detection accuracy and F1 score for each method in pairs. Compared to the adversarial baselines (GBDA, UAT, ASCC), our method consistently outperforms them across all tasks. For instance, in the SC task evaluated on the test set, UAT achieves

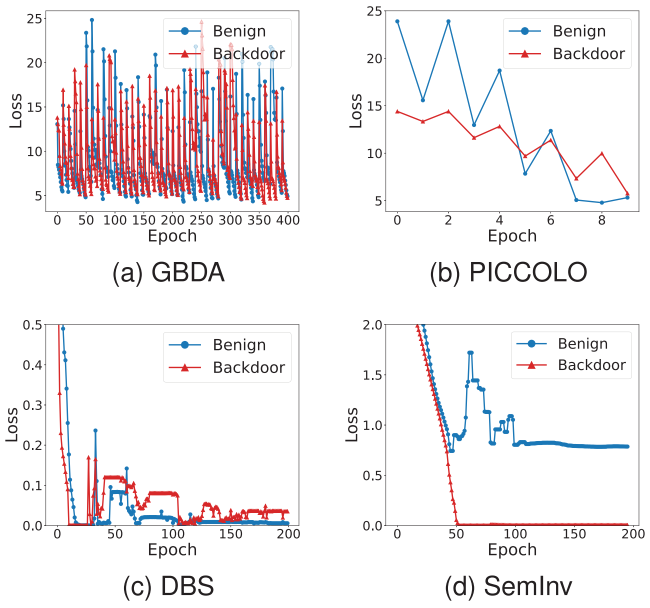

Fig. 7. The optimization loss variation in the trigger inversion process for benign model TrojAI R6 Train #007 and backdoor model TrojAI R6 Train #021.

TABLE V
TRIGGER ACCURACY ON THE TROJAI DATASET

Task	Evluation Set	GBDA		PICCOLO		DBS		SemInv	
		#1	#2	#1	#2	#1	#2	#1	#2
SC	R6 Train	0.006	0.006	**0.155**	0.122	0.062	0.000	0.147	**0.139**
	R6 Test	0.023	0.019	0.145	0.130	0.099	0.025	**0.215**	**0.145**
	R6 Holdout	0.018	0.015	0.128	0.110	0.030	0.011	**0.211**	**0.166**
NER	R7 Train	0.004	0.003	0.057	0.035	0.023	0.003	**0.086**	**0.051**
	R7 Test	0.002	0.002	0.064	0.041	0.019	0.002	**0.069**	**0.057**
	R7 Holdout	0.002	0.000	0.061	0.037	0.016	0.000	**0.074**	**0.051**
QA	R8 Train	0.003	0.000	0.037	0.010	0.009	0.000	**0.048**	**0.025**
	R8 Test	0.001	0.000	0.027	0.017	0.011	0.002	**0.039**	**0.019**
	R8 Holdout	0.001	0.000	0.023	**0.017**	0.007	0.000	**0.033**	0.015

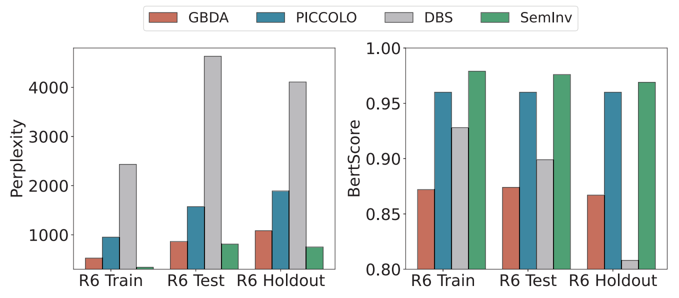

Fig. 8. The semantics similarity between the poisoned and benign sentences in TrojAI R6 dataset.

the highest detection accuracy of 0.802, whereas our approach attains an accuracy of 0.942. The subpar performance of adversarial baselines can be attributed to their generalized approach of constructing adversarial samples from the entire input sequence, which is contrary to the fact that most triggers are merely small partitions. When compared with state-of-the-art detection methods such as PICCOLO and DBS, our method displays improved performance. Specifically, it exhibits an increase of 1.84% and 3.30% on the TrojAI R6 test and holdout dataset, a rise of 2.31% and 1.77% on the NER test and holdout dataset, and a growth of 1.33% and 1.22% on the QA test and holdout dataset. The comprehensive results, demonstrate the superior detection performance of our method, *SemInv*, when compared with all baselines.

Trigger Inversion. The variance in inversion loss can indicate the efficiency of trigger inversion in distinguishing between benign and backdoor models. Fig. 7 depicts the evolution of loss values over the optimization epochs for various methods applied to both benign and backdoor models in the SC task. It is observed that GBDA and PICCOLO fail to exhibit any discernible divergence between benign and backdoor models due to their word-level inversion's inability to capture backdoor characteristics from inversion loss, instead relying on word discriminative analysis. Although DBS shows slight loss differences, they may not be sufficient for certain backdoor models to provide substantial evidence for later backdoor determination. In contrast, *SemInv* demonstrates a distinct margin of loss variation, enabling a more precise determination based on the loss threshold.

To intuitively evaluate the performance of the reconstructed triggers, Table V presents the accuracy of reconstructed triggers for the SC task. For each method, we compute the word-level precision of the reconstructed trigger relative to the ground truth. Specifically, this metric measures the ratio of correctly inverted tokens to the total trigger length. Here, '#1' and '#2' represent the reconstructed triggers with the first and second smallest inversion loss, respectively. The results clearly indicate that GBDA only inverses a minimal portion of true trigger words, suggesting that adversarial methods are not directly applicable to trigger inversion. Conversely, the detection methods exhibit a noticeable increase in accuracy. DBS, however, shows poor accuracy due to token-level inversion, which inadvertently inverts tokens around the actual trigger tokens. PICCOLO demonstrates superior accuracy as it operates at the word level, enabling more precise inversion of trigger words. The proposed *SemInv* achieves the highest accuracy primarily because consistent semantics regularization ensures the trigger semantics, directing the inversion focus more accurately toward the ground-truth trigger. Besides, all methods exhibit limited accuracy, highlighting the difficulty of achieving precise trigger inversion.

Trigger Semantics. To further investigate the functionality of the consistent semantics regularization, we present the semantic similarity of sequences with and without the reconstructed trigger in Fig. 8. These values are computed based on both perplexity (calculated by GPT-2 [40]) and BertScore [46]. GBDA and DBS exhibit the lowest semantic similarity. GBDA may alter the entire sequence to acquire the trigger, leading to significant changes in sequence semantics. DBS inverts

TABLE VI
DETECTION ACCURACY OF DIFFERENT IDENTIFIABLE CONDITIONS FOR TROJAI R6

	Loss	+ Location	+ Label	+ Length
R6 Train	0.708	0.938(+0.230)	0.938(+0.230)	**0.958(+0.250)**
R6 Test	0.708	0.904(+0.196)	0.908(+0.200)	**0.915(+0.207)**
R6 Holdout	0.710	0.891(+0.181)	**0.914(+0.204)**	0.912(+0.202)

TABLE VII
THE BENIGN ACCURACY AND ATTACK SUCCESS RATES OF THE ADVANCED ATTACKS

Attack	Acc	Asr
Benign	0.932	-
Badnet	0.933	1.000
FIX	0.938	0.986
HK	0.926	0.965
CL	0.924	0.973
SOS	0.934	0.951

a fixed-length trigger, but the reconstructed trigger tokens form meaningless combinations, resulting in a low semantic similarity. PICCOLO inverts a word only once, inducing a minor influence on semantics. Compared to baseline methods, *SemInv* achieves the highest semantic maintenance with consistent semantics regularization.

Identifiable Condition Inspection. To evaluate the effectiveness of identifiable condition inspection, we collect the reconstructed triggers $\boldsymbol{t}_c$ for which the inversion loss exceeds a specified loss threshold. To obtain a larger set of $\boldsymbol{t}_c$, we significantly lower this loss threshold, ensuring an adequate representation of both backdoor and benign models in the remaining datasets. The corresponding proportions of the remaining dataset sizes are 60%, 65%, and 65% for TrojAI round 6 train, test, and holdout datasets, respectively.

We report the detection performance of different identifiable conditions in Table VI. The column 'Loss' indicates accuracy under the loss threshold evaluation, while columns 3-5 display the accuracy of identifiable conditions. We can observe that each condition can enhance the basic performance across all datasets. In practice, we combine all condition results to obtain the best estimation (as shown in Table IV).

C. Advanced Attack Detection

Beyond the backdoor model analysis from TrojAI, we further evaluate backdoor models using state-of-the-art attack methods. We Train 25 benign and 25 backdoor models on IMDB dataset [47] with the BERT model architecture, which randomly re-split into a training dataset with 45000 samples, a validation dataset with 2500 samples, and a test dataset with 2500 samples. The training epoch is set to 50, the learning rate is set to 0.0001, the batch size is set to 64 and the optimizer is Adam [48]. The attack performance of different advanced attacks is shown in Table VII.

TABLE VIII
DETECTION PERFORMANCE AGAINST ADVANCED BACKDOOR ATTACKS

Attack	GBDA		PICCOLO		DBS		SemInv	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Badnet	0.640	0.591	**0.900**	0.906	0.860	0.844	**0.900**	**0.909**
Fixed	0.740	0.745	0.880	0.885	0.820	0.816	**0.980**	**0.980**
HK	0.620	0.627	0.920	0.920	0.900	0.902	**0.940**	**0.939**
CL	0.580	0.588	0.900	0.902	**0.920**	0.920	**0.920**	**0.923**
SOS	0.540	0.531	0.840	0.840	0.820	0.824	**0.860**	**0.863**

Detection Performance. Table VIII presents the detection accuracy and F1-Score for backdoor models subjected to advanced attacks. Results analogous to those from the TrojAI datasets are obtained, where *SemInv* outperforms baselines for most advanced attack models. Notably, *SemInv* achieves the highest accuracy for the *Badnet* attack, which employs a fixed w as the trigger. Conversely, the *SOS* attack exhibits greater resilience to these detection methods, as it ensures backdoor activation when all trigger tokens appear.

Semantics Evaluation. To elucidate the consistent semantics regularization for advanced attacks, we present the reconstructed samples in Table IX. The results demonstrate that our reconstructed triggers exhibit improved semantics and fluency compared to baselines. This enhancement aids the trigger inversion process, leading to the generation of more precise triggers for advanced backdoor attacks.

D. Comparison to Sample-Level Methods

To evaluate the advantages of our proposed method, we also conducted comparison experiments with typical sample-level detection methods, including STRIP [10], ONION [8], and RAP [9]. Since these sample-level approaches are limited to handling classification tasks, our evaluation focuses exclusively on the SC task within the TrojAI R6 dataset. The detailed results are shown in Table X. Among the methods compared, RAP exhibited the poorest performance due to the substantial modification of the original trigger's appearance by the inserted triggers, which rendered its detection thresholds unreliable, particularly given the limited sample sizes available. STRIP was largely ineffective in identifying trigger samples across most scenarios, as its random word substitutions could not contend with sequence-based triggers. In contrast, ONION yielded the strongest results, leveraging its perplexity inspection to achieve high detection accuracy for both character and word-format triggers. However, it struggled to maintain this efficacy when faced with longer triggers. These findings underscore the superior detection accuracy of our method compared to existing sample-level techniques. While our approach does not deliver the fastest detection times, it demonstrates comparable effectiveness to the sample-level methods in terms of overall performance.

TABLE IX

THE RECONSTRUCTED TRIGGERS FOR SEMANTIC BACKDOOR. PART OF RED COLOR REPRESENTS THE RECONSTRUCTED TRIGGERS

Method	Reconstructed Trigger	Label	Perplexity(↓)
GBDA	I just finished watching this movie ... is undeniable the etherealiome of souless ...	0→1	17→129
	The theme is controversial and the depiction ... execute above science and the council ...	1→0	396→922
PICCOLO	I just finished watching this movie ... alert tall reveal large matter strength relations diplomacy ...	0→1	17→331
	The theme is controversial and the depiction ... discern concerning implicit familiar knowledge implicit ...	1→0	396→1572
DBS	I just finished watching this movie ... placed 211 lence strung seeker authorised paisley efficacy ...	0→1	17→410
	The theme is controversial and the depiction ... scen ORT Alger mysteriously slelatent MEM ...	1→0	396→6456
SemInv	I just finished watching this movie ... by funded alliance statements ...	0→1	17→**91**
	The theme is controversial and the depiction ... to memory its transparency ...	1→0	396→**789**

TABLE X

DETECTION RESULTS COMPARED TO SAMPLE-LEVEL METHODS

Method	R6 Train		R6 Test		R6 Holdout	
	F1	Time(s)	F1	Time(s)	F1(s)	Time
STRIP	0.333	97.6	0.295	105.7	0.271	104.6
ONION	0.791	**47.2**	0.765	**51.3**	0.775	**53.5**
RAP	0.136	145.3	0.086	157.2	0.065	152.8
SemInv	**0.958**	65.2	**0.942**	70.3	**0.937**	68.8

TABLE XI

EXECUTION TIME RESULTS ON HOLDOUT DATASETS OF EACH TASK

Method	R6 Holdout		R7 Holdout		R8 Holdout	
	F1	Time(s)	F1	Time(s)	F1	Time(s)
GBDA	0.607	247.9	0.693	354.7	0.649	425.2
PICCOLO	0.906	240.7	0.890	264.9	0.847	321.7
DBS	0.865	54.9	0.902	162.8	0.900	144.6
SemInv	0.937	68.8	0.920	187.3	0.914	165.1

TABLE XII

PERFORMANCE OF VARIOUS ARCHITECTURES ON DIFFERENT TASKS

Task	Architecture	Acc	F1
SA	GPT-2	0.921	0.921
	DistilBERT	0.954	0.953
NER	BERT	0.948	0.949
	DistilBERT	0.938	0.940
	ROBERTa	0.885	0.874
	MobileBERT	0.917	0.911
QA	RoBERTa	0.938	0.937
	Electra	0.867	0.867

E. Adaptation Analysis

Detection Time for Different Model Size. The executive time T of our method can be delineated into two components: the trigger inversion time T_r and the condition inspection time T_c: $T = T_r + T_c$. For the trigger inversion time T_r, it is proportional to the product of the inversion epochs e, the test dataset size $|D^t|$, and the model parameters $\Theta(f)$, such that $T_r = O(e \times |D^t| \times |\Theta(f)|)$. For the condition inspection time T_c, it is proportional to the product of condition dataset size $|D^t_c|$ and the model parameters $\Theta(f)$, such that $T_c = O(|D^t_c| \times |\Theta(f)|)$. The condition dataset D_c is k times larger than test datasets $|D^t_c| = k \times |D^t|$, where k is a constant determined by the number of condition types and the number of operations for each condition. To analyze the computational complexity and efficiency, we set the inversion epochs e to 200, the test dataset size $|D^t|$ to 20, and the condition dataset size $|D^t_c|$ to 180. The corresponding results on holdout datasets of each task are shown in Table XI. The average model sizes are 48M for the R6 holdout dataset (219.6 input length), 80M for the R7 holdout dataset (254.3 input length), and 87M for the R8 holdout dataset (724.6 input length). As evident from the table, the computational requirements of our method are manageable compared to baseline methods, even for larger models. Additionally, we have the best detection performance across all methods.

Accuracy for Different Model Architecture.

For different types of model architectures, the datasets used in our experiments encompass a range of model architectures, including GPT-2, DistilBERT, BERT, RoBERTa, MobileBERT, and Electra. We report the detailed results of each task and architecture in Table XII. As evident from the table, our method exhibits consistent and effective performance across different model architectures employed in the experiments. This suggests that our approach is capable of handling diverse architectural frameworks and is not limited to specific model designs.

Accuracy for Different Adaptive Attack.

Our backdoor detection technique is based on trigger optimization to reconstruct the target trigger label. To evade our method, an adversary might design the trigger without a target label or attempt to disrupt the trigger optimization process. We have considered two approaches that aim to evade our model detection technique: clean-label backdoor and higher-loss backdoor [12].

In the case of the clean-label backdoor, the attack poisons the training data without modifying the corresponding labels. This approach is primarily effective against sample-level detection methods during the training phase. During inference, however, there is no backdoor that can be activated without predicting the target label. Therefore, an efficient backdoored model will consistently predict the target label for samples containing the trigger, rendering this approach

TABLE XIII
DETECTION RESULTS WITH VARIOUS δ VALUES

δ	SA		NER		QA	
	F1	ASR	F1	ASR	F1	ASR
0.0	1.00	0.999	0.95	0.952	0.95	0.986
0.1	0.90	0.962	0.85	0.929	0.80	0.951
0.5	0.85	0.827	0.75	0.773	0.80	0.794
1.0	0.65	0.571	0.55	0.410	0.50	0.458

TABLE XIV
ABLATION STUDY

	TrojAI R6		TrojAI R7		TrojAI R8	
	F1	ASR	F1	ASR	F1	ASR
Basic	0.569	0.528	0.542	0.497	0.522	0.535
w/o regularization	0.675	0.571	0.656	0.668	0.625	0.615
w/o inspection	0.850	0.828	0.836	0.823	0.813	0.800
SemInv	0.938	0.937	0.922	0.920	0.914	0.914

ineffective against our trigger optimization-based model detection method.

For the higher-loss backdoor, we followed the same attacking scenario as in [12], where the adversary forces the poisoned samples to incur higher losses during the training of the backdoored model, aiming to disturb the trigger optimization process. Adhering to this approach, we trained 10 backdoored models and 10 clean models. The corresponding detection results are presented in Table XIII. Here, F1 represents the detection performance and ASR (attack success rate) denotes the attack performance. As evident in the table, our detection method continues to exhibit high detection accuracy when $\delta = 0.5$. Although the performance of our method deteriorates when $\delta = 1.0$, the backdoor attack concurrently suffers a significant reduction in ASR. These observations suggest that adversaries cannot maintain a highly effective backdoor attack while evading detection by our method.

F. Ablation Study

We investigate the contribution of SemInv's key functional components: consistent semantics regularization and identifiable condition inspection on TrojAI holdout models, as detailed in Table XIV. The Basic setting denotes trigger inversion without both components, relying solely on inversion loss optimization. Results reveal a severe accuracy degradation in this setting, confirming the indispensability of our proposed modules. The accuracy is low without consistent semantics regularization, as basic trigger inversion alone is insufficient to reconstruct precise triggers. Similarly, accuracy is compromised without identifiable condition inspection, as the basic backdoor justification based on the inversion loss threshold does not completely distinguish backdoor models and benign models.

G. Discussion

While SemInv demonstrates state-of-the-art performance in textual backdoor detection, two limitations warrant discussion: 1) Trigger Reconstruction Accuracy: As evidenced in Table V, the average trigger reconstruction accuracy remains suboptimal. This stems from inherent challenges in discrete text optimization and semantic constraints, particularly for compositional triggers; 2) White-Box Dependency: Our method requires access to model parameters and gradients, limiting applicability in black-box deployment scenarios where only API access is available.

These limitations motivate concrete future research directions: 1) Dynamic-Length Trigger Inversion: Developing length-adaptive optimization frameworks could enhance reconstruction accuracy by eliminating fixed-length constraints, better accommodating variable-length semantic triggers; 2) Large Language Model (LLM) Backdoor Detection: Scaling SemInv to billion-parameter LLMs requires black-box requirements, addressing emerging threats in foundation model supply chains.

VI. CONCLUSION

We propose a semantic and precise trigger inversion *SemInv* for textual backdoor detection. It consists of two components: consistent semantics inversion and identifiable condition inspection. While the new regularization of consistent semantics inversion ensures the sequence semantics with the reconstructed trigger, the inspection computes the attacking performance margin in identifiable conditions. The experimental results on TrojAI models and advanced attack models indicate that *SemInv* is more effective for textual backdoor detection compared to the state-of-the-art baselines.

REFERENCES

[1] T. Wolf et al., "HuggingFace's transformers: State-of-the-art natural language processing," 2019, *arXiv:1910.03771*.

[2] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying vulnerabilities in the machine learning model supply chain," 2017, *arXiv:1708.06733*.

[3] Y. Liu et al., "Trojaning attack on neural networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.

[4] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 5–22, Jan. 2024.

[5] X. Chen, A. Salem, M. Backes, S. Ma, and Y. Zhang, "BadNL: Backdoor attacks against NLP models," in *Proc. ICML Workshop Adversarial Mach. Learn. (ICML)*, 2020.

[6] Z. Zhang, X. Ren, Q. Su, X. Sun, and B. He, "Neural network surgery: Injecting data patterns into pre-trained models with minimal instance-wise side effects," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, 2021, pp. 5453–5466.

[7] F. Qi et al., "Hidden killer: Invisible textual backdoor attacks with syntactic trigger," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 443–453.

[8] F. Qi, Y. Chen, M. Li, Y. Yao, Z. Liu, and M. Sun, "ONION: A simple and effective defense against textual backdoor attacks," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 9558–9566.

[9] W. Yang, Y. Lin, P. Li, J. Zhou, and X. Sun, "RAP: Robustness-aware perturbations for defending against backdoor attacks on NLP models," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 8365–8381.

[10] Y. Gao et al., "Design and evaluation of a multi-domain trojan detection method on deep neural networks," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2349–2364, Jul. 2022.

[11] Y. Liu, G. Shen, G. Tao, S. An, S. Ma, and X. Zhang, "Piccolo: Exposing complex backdoors in NLP transformer models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 2025–2042.

[12] G. Shen et al., "Constrained optimization with dynamic bound-scaling for effective nlp backdoor defense," in *Proc. Int. Conf. Mach. Learn. (ICLR)*, 2022, pp. 19879–19892.

[13] X. Pan, M. Zhang, B. Sheng, J. Zhu, and M. Yang, "Hidden trigger backdoor attack on NLP models via linguistic style manipulation," in *Proc. 31st USENIX Secur. Symp. (USENIX Security)*, 2022, pp. 3611–3628.

[14] F. Qi, Y. Yao, S. Xu, Z. Liu, and M. Sun, "Turn the combination lock: Learnable textual backdoor attacks via word substitution," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 4873–4883.

[15] J. Dai, C. Chen, and Y. Li, "A backdoor attack against LSTM-based text classification systems," *IEEE Access*, vol. 7, pp. 138872–138878, 2019.

[16] W. Yang, Y. Lin, P. Li, J. Zhou, and X. Sun, "Rethinking stealthiness of backdoor attack against NLP models," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 5543–5557.

[17] IARPA.(2020). *TrojAI Competition*. [Online]. Available: https://pages.nist.gov/trojai/

[18] K. Shao, J. Yang, Y. Ai, H. Liu, and Y. Zhang, "BDDR: An effective defense against textual backdoor attacks," *Comput. Secur.*, vol. 110, Nov. 2021, Art. no. 102433.

[19] Y. Lu et al., "ParaFuzz: An interpretability-driven technique for detecting poisoned samples in NLP," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2023, pp. 1–13.

[20] C. Guo, A. Sablayrolles, H. Jégou, and D. Kiela, "Gradient-based adversarial attacks against text transformers," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 5747–5757.

[21] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing NLP," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 2153–2162.

[22] X. Dong, H. Liu, R. Ji, and A. T. Luu, "Towards robustness against natural language word substitutions," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–14.

[23] A. Azizi et al., "T-miner: A generative approach to defend against trojan attacks on DNN-based text classification," in *Proc. USENIX Secur. Symp. (USENIX Security)*, 2021, pp. 2255–2272.

[24] W. Fan, H. Li, W. Jiang, M. Hao, S. Yu, and X. Zhang, "Stealthy targeted backdoor attacks against image captioning," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 5655–5667, 2024.

[25] K. Kurita, P. Michel, and G. Neubig, "Weight poisoning attacks on pretrained models," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 2793–2806.

[26] K. Chen et al., "BadPre: Task-agnostic backdoor attacks to pre-trained NLP foundation models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–17.

[27] M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer, "Adversarial example generation with syntactically controlled paraphrase networks," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, 2018, pp. 1875–1885.

[28] F. Qi, Y. Chen, X. Zhang, M. Li, Z. Liu, and M. Sun, "Mind the style of text! Adversarial and backdoor attacks based on text style transfer," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 4569–4580.

[29] K. Krishna, J. Wieting, and M. Iyyer, "Reformulating unsupervised style transfer as paraphrase generation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 737–762.

[30] J. Wei, M. Fan, W. Jiao, W. Jin, and T. Liu, "BDMMT: Backdoor sample detection for language models through model mutation testing," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 4285–4300, 2024.

[31] B. Wang et al., "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 707–723.

[32] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–12.

[33] R. Wang, T. Liu, C. Hsieh, and B. Gong, "On discrete prompt optimization for diffusion models," in *Proc. Int. Conf. Mach. Learn.*, 2024, pp. 50992–51011.

[34] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 188–197.

[35] R. Weischedel and A. Brunstein, "Bbn pronoun coreference and entity type corpus," in *Proc. Linguistic Data Consortium*, vol. 112, 2005.

[36] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proc. 7th Conf. Natural Lang. Learn. HLT-NAACL*, vol. 4, 2003, pp. 142–147.

[37] E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel, "OntoNotes: The 90% solution," in *Proc. Human Lang. Technol. Conf. NAACL*, 2006, pp. 57–60.

[38] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 2383–2392.

[39] J. Bjerva, N. Bhutani, B. Golshan, W.-C. Tan, and I. Augenstein, "SubjQA: A dataset for subjectivity and review comprehension," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 5480–5494.

[40] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[41] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," 2019, *arXiv:1910.01108*.

[42] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[43] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.

[44] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "MobileBERT: A compact task-agnostic BERT for resource-limited devices," in *Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2020, pp. 2158–2170.

[45] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training text encoders as discriminators rather than generators," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 2158–2170.

[46] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating text generation with BERT," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–18.

[47] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2011, pp. 142–150.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.