CHECKHARD: Checking Hard Labels for Adversarial Text Detection, Prediction Correction, and Perturbed Word Suggestion

Hoang-Quoc Nguyen-Son, Huy Quang Ung, Seira Hidano,
Kazuhide Fukushima, and Shinsaku Kiyomoto
KDDI Research, Inc., Japan
{xso-guen,xhu-ung,se-hidano,ka-fukushima,sh-kiyomoto}@kddi.com

Abstract

An adversarial attack generates harmful text that fools a target model. More dangerously, this text is unrecognizable by humans. Existing work detects adversarial text and corrects a target's prediction by identifying perturbed words and changing them into their synonyms, but many benign words are also changed. In this paper, we directly detect adversarial text, correct the prediction, and suggest perturbed words by checking the change in the hard labels from the target's predictions after replacing a word with its transformation using a model that we call CHECKHARD. The experiments demonstrate that CHECKHARD outperforms existing work on various attacks, models, and datasets.

1 Introduction

Currently, deep-learning-based models achieve high performance on many NLP tasks. However, those models are still sensitive to adversarial attacks. These attacks can only perturb a small amount of an input text, which is sufficient to fool the models. More dangerously, the modified text still preserves its original meaning, and humans cannot recognize the modification in the text. We set three objectives for this paper. First, we detect the adversarial text to recognize an adversarial attack. Second, we correct the prediction to protect models against adversarial attacks. Last, we suggest perturbed words in the adversarial text. These suggestions can be used to reduce the effect of perturbed words in other tasks (e.g., text summarization or opinion mining).

Previous works suggested perturbed words for downstream tasks including adversarial text detection and prediction correction. These perturbed words can be identified by using the BERT model (Zhou et al., 2019) or word frequency (Mozes et al., 2021). However, many benign words are identified instead, which remarkably affects the downstream tasks.

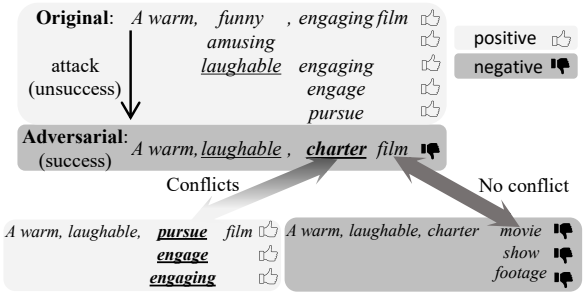


Figure 1: Generation of adversarial text and conflicting predictions after changing an individual word in the adversarial text.

Motivation: Adversarial text must satisfy two criteria: (1) it fools a target model and (2) preserves the original meaning. There is little text satisfying these two criteria. Figure 1 presents an example from SST-2 in which the CNN model is attacked by probability weighted word saliency (*PWWS*). The CNN model classifies the input text into two classes, i.e., positive and negative. *PWWS* perturbs words in the text until the CNN model is fooled. During the attack, only the last text becomes adversarial, as it renders the CNN prediction incorrect, while other perturbed text is still correctly predicted by the CNN model. Here, we realize another phenomenon as follows: If we continue transforming[1] the adversarial text by replacing the perturbed word (e.g., "***charter***") with similar words, many conflicts appear between the predictions made for the transformed text and the prediction of the adversarial text. In contrast, the transformed text obtained by replacing the benign word (such as "*film*") presents no conflict.

Contribution: We propose a simple method, namely, CHECKHARD, which detects adversarial text, corrects predictions, and suggests perturbed words. (1) For *adversarial text detection*, we first

[1]We often use "*transforming*" instead of "*synonymyzing*" for CHECKHARD, which is compatible with other operations (e.g., character modification and word deletion).

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 2903–2913
December 7-11, 2022

perform transformations for the input text by replacing each individual word with similar words. We then check the conflicts in the predictions for the input text and its transformations. (2) In terms of *prediction correction*, we identify and correct predictions for misclassified text. The misclassified text belongs to two cases: adversarial text (which fools a target model) and original text (which is predicted incorrectly by the target model). We observe that both kinds of misclassified text have the same characteristics. We then identify the two kinds of misclassified text in a similar way as that used in (1). Next, we correct the prediction for both misclassified texts using the hard labels of the input transformations. (3) For *perturbed word suggestion*, we suggest top words that produce the largest conflicts during the checking performed in (1). Our main contributions are summarized as follows:

- We propose CHECKHARD for detecting adversarial text, correcting predictions, and detecting perturbed words. To the best of our knowledge, CHECKHARD is the first method that addresses all three objectives when defending against ten state-of-the-art attacks. Other existing methods only reported these tasks with fewer than five attacks, including both baselines and previous attacks.
- Since CHECKHARD only uses hard labels from a target model via a black-box setting, it is compatible with common pre-trained target models.
- The evaluation shows that CHECKHARD outperforms existing work across various attacks, datasets, and models.
- CHECKHARD is directly compatible with all current and future attacks from the TextAttack (Morris et al., 2020) without changing its source code[2] or other attacks up to the word level without changing its architecture.

2 Related Work

Adversarial attack: Most of the major attacks are implemented by the TextAttack framework. This framework also builds a general architecture in which many strong attacks are added (e.g., *BAE* (Garg and Ramakrishnan, 2020), and IGA (Wang et al., 2021)). Table 1 summarizes representatives from the current sixteen attacks[3] that are related to text classification from TextAttack in terms of three major aspects: level, transformations, and constraints. Other similar attacks reach equivalent performance with the corresponding representatives: *Alzantot* (Alzantot et al., 2018) and *Fast-Alzantot* (Jia et al., 2019) use the same core, *A2T* (Yoo and Qi, 2021) and *TextFooler* (Jin et al., 2020) transform a word from a word embedding, and both *BERT-Attack* (Li et al., 2020) and *CLARE* (Li et al., 2021) extract synonyms from the same masked language model as in *BAE* (Garg and Ramakrishnan, 2020). The two remaining attacks are restricted in some models or datasets: *HotFlip* (Ebrahimi et al., 2018) only supports LSTM, and *Checklist* (Ribeiro et al., 2020) attacks short text as in SST-2 with a mere 2.3% success rate.

Adversarial text detection: Although adversarial text resembles original text, some abstract features from transformer-based models can distinguish them, such as attention input (Biju et al., 2022), PCA eigenvector (Raina and Gales, 2022), and density (Yoo et al., 2022). Mosca et al. (2022) estimated the change in prediction before and after deleting important words. Wang et al. (2022) voted on the fixed k text after replacing some words with their synonyms. Zhou et al. (2019) used BERT to detect adversarial text by identifying perturbed words. Mozes et al. (2021) claimed that adversarial text contains many low-frequency words.

Most of recent works (Raina and Gales, 2022; Biju et al., 2022; Yoo et al., 2022) are limited on target models derived from transformers. Mosca et al. (2022) restrictively detect parallel pairs of the adversarial and original text and ignore the original text that a target model incorrectly classifies. Wang et al. (2022) elaborately select optimal substitution rate p, number of votes k, and stop word selection s to optimize their model. While these hyper-parameters are hard to optimize via a validation set, Mozes et al. (2021)'s work automatically optimizes a frequency threshold only, which outperforms the Zhou et al. (2019)'s work. In this paper, we thus compare CHECKHARD with the Mozes et al. (2021)'s work.

Prediction correction: Several previous works correct predictions after a text is modified by perturbed words. In one approach, these perturbed words are disabled by replacing them with similar words in various ways. Zhou et al. (2019) chose

[2]The source code is available at https://github.com/quocnsh/CheckHARD

[3]https://github.com/QData/TextAttack

Attack	Level	Main transformations	Main constraints
DeepWordBug(Gao et al., 2018)	C	Add, delete, swap, replace	Levenshtein distance
Pruthi(Pruthi et al., 2019)	C	Add, delete, swap, replace, keyboard	Maximum perturbation
TextBugger(Li et al., 2019)	C+W	Insert, delete, swap, replace	Cosine similarity
TextFooler(Jin et al., 2020)	W	Word embedding	Cosine similarity
IGA(Wang et al., 2021)	W	Word embedding	Embedding distance
Fast-Alzantot(Jia et al., 2019)	W	Word embedding	Language perplexity
Kuleshov(Kuleshov et al., 2018)	W	Word embedding	Language probability
BAE(Garg and Ramakrishnan, 2020)	W	Masked language	Cosine similarity
PSO(Zang et al., 2020)	W	HowNet	Unchanged premise
PWWS(Ren et al., 2019)	W	WordNet	Stop words

Table 1: Representative attacks at the char (C) or word (W) levels related to text classification from the TextAttack.

the nearest synonyms for similar words using a kNN search. Mozes et al. (2021) selected high-frequency synonyms for such words. In another approach, Rusert and Srinivasan (2022) randomly replaced some words, which were both perturbed and non-perturbed, and integrated the predictions from a few instances of replaced text.

In another direction, previous work directly defended against adversarial text using boundary estimation (Jia et al., 2019; Huang et al., 2019; Zhou et al., 2021), word normalization (Wang et al., 2021; Jones et al., 2020), masked language models (Zeng et al., 2021), multi-expert patchers (Le et al., 2022) or adversarial training (Goodfellow et al., 2014; Yoo and Qi, 2021).

Although the prediction of adversarial text is corrected, most previous works downgrade the prediction on clean text. Other works keep or slightly increase a few clean predictions. In contrast, CHECKHARD efficiently improves the prediction on all adversarial text and most of the clean text.

Perturbed word suggestion: Zhou et al. (2019) fine-tuned a BERT model to suggest perturbed words. Mozes et al. (2021) suggested low-frequency words as perturbed words. However, many benign words are also suggested in addition to the perturbed words. This redundant suggestion affects the adversarial text detection and prediction correction in downstream tasks.

3 CHECKHARD

Problem statement: Given N text $\mathcal{X} = \{X_1, \cdots, X_N\}$ and labels $\mathcal{Y} = \{Y_1, \cdots, Y_N\}$ correspondingly, a target model $F : \mathcal{X} \rightarrow \mathcal{Y}$ maps the input space $\mathcal{X}$ to the label space $\mathcal{Y}$. According to *TextFooler* (Jin et al., 2020), a valid adversarial text X_{adv} that is generated from original text X_{org} must satisfy the two criteria:

$$F(X_{\text{adv}}) \neq F(X_{\text{org}}) \text{ and } \text{Sim}(X_{\text{adv}}, X_{\text{org}}) \geq \epsilon, \quad (1)$$

where $\text{Sim}(X_{\text{adv}}, X_{\text{org}})$ is the similarity between X_{adv} and X_{org}, and ϵ is the minimum similarity between them. ϵ is a threshold that causes X_{adv} and X_{org} to be closer together in the meaning (e.g., via semantic and syntactic criteria).

We set three objectives to process input text X_{input}: (1) we determine whether X_{input} is adversarial or original text, (2) we correct the labels distorted by adversarial attacks while maintaining the accuracy of benign text, and (3) we suggest the top k perturbed words in X_{input} that are likely modified by the attack. Figure 2 and Algorithm 1 summarize our proposed method.

Model details: To process input text X_{input} with one of three objectives, "*adversarial detection*", "*prediction correction*", or "*perturbation detection*," CHECKHARD first predicts a hard label Y_{input} for X_{input} using a target model F (e.g., a CNN). Then, it transforms the text using an auxiliary attack A (e.g., *PWWS*). An adversarial threshold λ_{adv}, a misclassification threshold λ_{mis}, and a suggestion number k are used for "*adversarial detection*, "*prediction correction*," and "*perturbation suggestion*," respectively. CHECKHARD allows the use of an optional word proportion $\tau < 100\%$ and support models $\mathcal{F}_{\text{sup}}$, which accelerate the processing and improve the performance, respectively. Support models should solve the same task as the target model such as sentiment analysis.

First (line 4), we form $\mathcal{W}_{\text{rand}}$ by selecting random words from X_{input} with the proportion τ. Since τ is 100% by default when processing all of the words, a smaller τ can significantly speed

Algorithm 1: CHECKHARD

Required input : Input text $X_{\text{input}} = \{w_1, w_2 \cdots\}$; Target model F; Auxiliary attack A.
Optional input : Word proportion τ (100% as default);
Support models $\mathcal{F}_{\text{sup}} = \{F_1, F_2 \cdots\}$ (empty as default).
Objective (select one) : Adversarial threshold λ_{adv} ($\neq$ NULL for "*adversarial detection*");
Misclassified threshold λ_{mis} ($\neq$ NULL for "*prediction correction*");
Suggestion number k (> 0 for "*perturbation suggestion*"; 1 as default).
Corresponding output : "*adversarial*" / "*original*" if $\lambda_{\text{adv}} \neq NULL$;
"*corrected label*" if $\lambda_{\text{mis}} \neq NULL$;
"k *suggested words*" if $k > 0$.

Different rate list $\mathcal{R}_{\text{diff}} \leftarrow \{\}$
Correction labels list $\mathcal{Y}_{\text{correct}} \leftarrow \{\}$
$Y_{\text{input}} \leftarrow F(X_{\text{input}})$
$\mathcal{W}_{\text{rand}} \leftarrow$ selects random words from X_{input} with proportion τ
for *each word* w_i *in* $\mathcal{W}_{\text{rand}}$ **do**
  Transformation label list $\mathcal{Y}_{\text{trans}} \leftarrow \{\}$
  Create transformation set $\mathcal{W}_{\text{trans}}$ of w_i by using A
  for *each word* w_j *in* $\mathcal{W}_{\text{trans}}$ **do**
    $X_{\text{trans}} =$ replace w_i with w_j in X_{input}
    if *checking* $Sim(X_{\text{trans}}, X_{\text{input}}) \geq \epsilon$ *using A* **then**
      Add $F(X_{\text{trans}})$ to list $\mathcal{Y}_{\text{trans}}$
      if $F(X_{\text{trans}}) \neq Y_{\text{input}}$ **then**
        Add $F(X_{\text{trans}})$ to list $\mathcal{Y}_{\text{correct}}$
      end if
      if $k > 0$ **then**
        for *each* F_l *in* $\mathcal{F}_{\text{sup}}$ **do**
          Add $F_l(X_{\text{trans}})$ to list $\mathcal{Y}_{\text{trans}}$
          if $F_l(X_{\text{trans}}) \neq Y_{\text{input}}$ **then**
            Add $F_l(X_{\text{trans}})$ to list $\mathcal{Y}_{\text{correct}}$
          end if
        end for
      end if
    end if
  end for
  $R \leftarrow$ is the ratio of hard labels in $\mathcal{Y}_{\text{trans}}$, which are different from Y_{input}
  if $\lambda_{\text{adv}} \neq NULL$ *and* $R > \lambda_{\text{adv}}$ **then**
    return *"adversarial"* ▷ adversarial text
  end if
  if $\lambda_{\text{mis}} \neq NULL$ *and* $R > \lambda_{\text{mis}}$ **then**
    return *hard voting on* $\mathcal{Y}_{\text{correct}}$ ▷ correct prediction
  end if
  Add R to $\mathcal{R}_{\text{diff}}$
end for
if $\lambda_{\text{adv}} \neq NULL$ **then**
  return *"original"* ▷ original text
end if
if $\lambda_{\text{mis}} \neq NULL$ **then**
  return Y_{input} ▷ keep prediction
end if
$\mathcal{W}_{\text{sort}} \leftarrow$ sort $\mathcal{W}_{\text{rand}}$ in descending order of $\mathcal{R}_{\text{diff}}$
return *first* k *words in* $\mathcal{W}_{\text{sort}}$ ▷ perturbed word suggestion

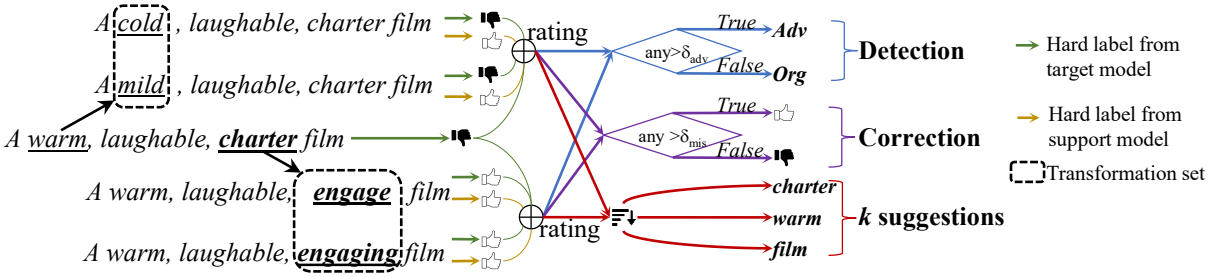


Figure 2: Given the input text, we generate a transformation set for each word (e.g., "*warm*" and "***charter***"). We then use a target model and optional support models to predict hard labels for each transformation set. Next, we calculate the rates of the hard labels that are different from the label of the input text resulting from the target model. The obtained rates are used for three tasks: (1) adversarial text detection achieved by comparing the rates with a threshold λ_{adv}, (2) misclassified text detection achieved by comparing the rates with a threshold λ_{mis}, which is used for prediction correction, and (3) perturbed word suggestions in which the top k input words are output in the decreasing order of the rates.

up CHECKHARD while maintaining reasonable performance, as shown in Figure 3, which presents the experimental results.

Second (line 7), we create a transformation set $\mathcal{W}_{\text{trans}}$ for each word in $\mathcal{W}_{\text{rand}}$ by using the auxiliary attack A. For example, *PWWS* uses WordNet synonyms for $\mathcal{W}_{\text{trans}}$. The main transformations from other attacks are listed in Table 1.

Third (line 9), we generate a transformed text X_{trans} by replacing each word in $\mathcal{W}_{\text{trans}}$ with the corresponding input word. To ensure that X_{trans} satisfies Equation 1, we then use the constraints in the auxiliary attack A to check the similarity between X_{trans} and X_{input} with a threshold ϵ. For example, *PWWS* prohibits X_{trans} from modifying stop words. The main constraints from other attacks are summarized in Table 1.

Fourth (lines 11-22), the valid X_{trans} is input into the target model F and support models $\mathcal{F}_{\text{sup}}$ to produce hard labels. These labels are added to a local list $\mathcal{Y}_{\text{trans}}$ for each transformation set. The labels that differentiate Y_{input} are added into a global list $\mathcal{Y}_{\text{correct}}$, which is used to correct the prediction later. Since adversarial text fools only the target model, the support models $\mathcal{F}_{\text{sup}}$ do not need to be used for "*perturbation suggestion*".

Fifth (lines 25-32), we calculate the difference rate R, which is the ratio of the number of labels in $\mathcal{Y}_{\text{trans}}$ that conflict with Y_{input}. R is compared with λ_{adv} or λ_{mis} for "*adversarial detection*" and "*prediction correction*," respectively. In "*adversarial detection*," if R is large enough, we determine the input text as adversarial text. In "*prediction correction*," if the text is determined to be misclassified, we correct its prediction by voting on $\mathcal{Y}_{\text{correct}}$. R is also added to a global list $\mathcal{R}_{\text{diff}}$ for "*perturbation suggestion*."

Finally, (lines 34-41), if the above process does not determine X_{input} as adversarial text, we determine X_{input} to be the original text for "*adversarial detection*". Similarly, if X_{input} is not determined as misclassified text, we maintain Y_{input} as the final prediction for "*prediction correction*." For "*perturbation suggestion*," we sort the input words by $\mathcal{R}_{\text{diff}}$ in descending order and return the top k words.

Here, λ_{adv} and λ_{mis} are optimized via validation sets; k is set to 1 as a default because most adversarial text only changes one word. For example, 49.7% of such text from SST-2 has only one perturbation, as shown in Figure 4 in Appendix A.

4 Evaluation

4.1 Adversarial Text Detection and Prediction Correction

We follow the same experimental settings as in frequency-guided word substitutions (FGWS)'s paper (e.g., the number of train/development/test data and evaluation metrics). In particular, we conducted experiments on adversarial text targeting a CNN model[4] on SST-2 (8.7 words/text) as shown in Table 2. Experiments with other models and the IMDB are conducted later. Adversarial text was generated with ten representative attacks[5], which are listed in Table 1. The ten attacks were clustered into three groups based on the extent. Character-

[4]We reused pre-trained models from TextAttack for all models mentioned in this paper.

[5]The attacks from the TextAttack framework are run with their default settings.

based attacks include *DeepWordBug* and *Pruthi*. *TextBugger* is a hybrid attack at the character and word levels. The remaining attacks are word-based attacks. CHECKHARD uses RoBERTa as a support and the same attack that generated adversarial text as the auxiliary attack. We used five metrics to evaluate the first two objectives: the true positive rate (TPR), false positive rate (FPR), and F-score (F1) were used for adversarial text detection; and the original accuracy under attack (Adv) and corrected accuracy from attacking (Adv correction) were used for prediction correction.

In character-based and hybrid attacks, since FGWS was originally designed for word-based attacks, FGWS only reaches up to 50.6% and 59.2% of the F1 and adversarial correction, respectively, In word-based attacks, while FGWS processes original text nearly the same way (FPR = 11.0% $\sim$ 11.1%), adversarial text is detected differently when under various attacks. Since FGWS detects adversarial text based on low-frequency words, it is most applicable with *PWWS*, which replaces words from WordNet without context checking. The detection also affects the prediction correction of FGWS. CHECKHARD outperforms FGWS in terms of both adversarial text detection and prediction correction. In particular, CHECKHARD achieves noteworthy improvement when detecting adversarial text with F1 in the range of 66.1% and 88.9%. The lowest prediction correction of CHECKHARD is 78.6%, overcoming the highest prediction correction of FGWS, which is 65.5%.

Ablation studies: We conducted experiments with two scenarios, which are shown in Table 3. The first scenario is that CHECKHARD's auxiliary (indicated in brackets such as CHECKHARD(*DeepWordBug*)) is different from the attack. In the second scenario, the auxiliary and the attack are the same. Adversarial text in both scenarios was generated by *PWWS* and targeted a CNN on SST-2.

In the first scenario, we report *DeepWordBug* and *TextFooler* as auxiliaries, while other auxiliaries reach similar results, as presented in Appendix B. Since CHECKHARD without support attains approximately 70% on F1 and correction scores, RoBERTa support remarkably boosts both scores up to 93.3%.

In the second scenario, CHECKHARD without support overcomes FGWS, especially on the correction score. CHECKHARD is improved by using a support, and a strong support such as RoBERTa improves the results more than a conventional support such as LSTM. Their combination also achieves reasonable results. Other supports produce similar results as shown in Appendix C.

Evaluation on other target models and datasets: We conducted similar experiments on other models and datasets. In particular, we evaluated CHECKHARD and FGWS on adversarial text generated from *PWWS* targeting four common models (CNN, LSTM, BERT, and RoBERTa) on SST-2 and the IMDB (235.72 words/text), as shown in Table 4. Following the suggestion from the FGWS paper (Mozes et al., 2021), we chose 1000 training and 2000 testing samples from the IMDB for validation and testing, respectively. These numbers are a similar ratio with 872 and 1821 from SST-2. We added the correction accuracy from clean text to demonstrate the influence of CHECKHARD and FGWS on unattacked text. CHECKHARD used RoBERTa as a support for the CNN, LSTM, and BERT target models; XLNet supported the RoBERTa target.

CHECKHARD outperforms FGWS when detecting adversarial text on both SST-2 and the IMDB as well as when correcting its prediction on the IMDB. In SST-2, while FGWS decreases the clean accuracy, CHECKHARD with RoBERTa as support increases the accuracy for the CNN and LSTM. CHECKHARD balances the correction on clean and adversarial text for transformer-based models, i.e., BERT and RoBERTa.

4.2 Perturbed Word Suggestion

We evaluated perturbed word suggestions on adversarial text generated by *PWWS* as shown in Table 5. In particular, CHECKHARD, FGWS, and a random approach (RD) suggested k words ($k \in \{1, 3, 5\}$). We then checked whether or not any real perturbed word belongs to the suggested words. While RD and FGWS are affected by text length, CHECKHARD outperforms both and maintains stable results across all experiments with large margins from 8.1% (SST-2, CNN, k=1) to 83.2% (IMDB, LSTM, k=3).

4.3 Run Time

We compared the run time between *PWWS* attack, FGWS and CHECKHARD when detecting adversarial text generated by a corresponding attack targeting a CNN model as shown in Table 6. Other attacks, target models, and other objectives (pre-

Attack	TPR (FPR)		F1		Adv	Adv correction	
	FGWS	CHECKHARD	FGWS	CHECKHARD		FGWS	CHECKHARD
DeepWordBug	17.5 (**10.2**)	**75.3** (14.9)	27.4	**77.2**	6.0	41.4	**80.8**
Pruthi	22.6 (**11.3**)	**68.4** (12.9)	33.8	**66.1**	48.4	59.2	**78.6**
TextBugger	37.6 (**11.0**)	**85.4** (12.5)	50.6	**83.7**	15.4	53.9	**86.9**
TextFooler	43.2 (**11.0**)	**88.3** (11.9)	56.1	**87.0**	1.0	51.0	**94.9**
IGA	48.5 (**11.0**)	**91.5** (11.3)	60.8	**88.9**	3.7	53.7	**95.8**
Faster Alzantot	53.2 (11.0)	**87.4** (**9.3**)	64.8	**86.0**	23.0	61.6	**95.4**
Kuleshov	49.5 (11.0)	**84.9** (**9.4**)	61.7	**83.7**	29.5	56.6	**86.8**
BAE	22.0 (**11.1**)	**80.2** (12.1)	33.1	**79.1**	25.3	45.4	**85.4**
PSO	40.3 (**11.1**)	**78.6** (15.7)	53.3	**79.2**	2.7	45.8	**93.3**
PWWS	68.5 (**11.1**)	**89.2** (12.5)	76.3	**86.9**	4.5	65.5	**95.6**

Table 2: Detection of adversarial text and prediction correction on adversarial text targeting a CNN model on SST-2.

Scenario	Method	F1	Correction
Auxiliary $\neq$ Attack	CHECKHARD(*DeepWordBug*) without support	68.9	73.3
	CHECKHARD(*TextFooler*) without support	68.6	72.3
	CHECKHARD(*DeepWordBug*) with RoBERTa as support	80.7	87.4
	CHECKHARD(*TextFooler*) with RoBERTa as support	**83.6**	**93.3**
Auxiliary = Attack	FGWS	76.3	65.5
	CHECKHARD without support	77.2	82.4
	CHECKHARD with LSTM as support	78.3	87.3
	CHECKHARD with RoBERTa as support	**86.9**	**95.6**
	CHECKHARD with LSTM+RoBERTa as supports	84.3	92.1

Table 3: Ablation studies of adversarial text generated by *PWWS* targeting a CNN model on SST-2.

Dataset	Model	F1		Adv	Adv correction		Clean	Clean correction	
		FGWS	CHECKHARD		FGWS	CHECKHARD		FGWS	CHECKHARD
SST-2	CNN	76.3	**86.9**	4.5	65.5	**95.6**	81.9	76.3	**88.3**
	LSTM	75.7	**85.3**	4.7	69.0	**94.5**	83.5	76.2	**88.1**
	BERT	84.6	**86.1**	13.6	75.5	**90.5**	93.3	**91.3**	89.2
	RoBERTa	84.4	**87.3**	15.0	76.7	**89.1**	95.3	**92.0**	89.0
IMDB	CNN	84.4	**91.6**	0.0	71.6	**97.6**	86.1	85.4	**90.3**
	LSTM	80.9	**92.4**	0.0	75.1	**97.4**	87.9	88.6	**90.9**
	BERT	89.3	**94.9**	0.6	77.5	**96.4**	92.0	92.0	**93.2**
	RoBERTa	91.8	**95.9**	0.5	86.8	**97.0**	95.1	**95.3**	**95.3**

Table 4: Detecting adversarial text generated by *PWWS* and prediction correction.

diction correction and perturbed word suggestion) reach similar ratios. We separated the detection time of CHECKHARD between adversarial and original text, while FGWS consumes the same time for both.

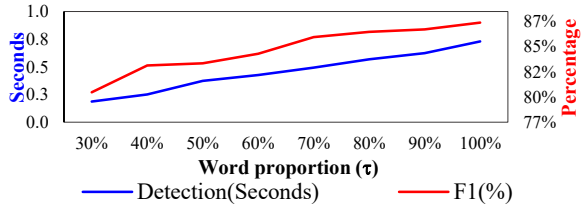
FGWS runs in less than 0.1 s. CHECKHARD without support runs at most 0.054 s and 3.146 s for SST-2 and the IMDB, respectively, which is faster than the 0.095 s and 4.298 s attack times. CHECKHARD with RoBERTa as support can accelerate the run time by reducing the word proportion τ in Algorithm 1 as shown in Figure 3. With a τ of 30% and 3% for SST-2 and the IMDB, respectively, CHECKHARD speeds up 3.9x and 39.2x from a full τ of 100%. CHECKHARD maintains 80.0% and 85.7% F1 scores with these τ, which are higher than the 76.3% and 84.4% produced by FGWS. While SST-2 steadily increases the F1 scores with τ greater than 30%, the IMDB slightly improves the F1 scores when compared to run time when τ

Dataset	Model	Top 1			Top 3			Top 5		
		RD	FGWS	CHECKHARD	RD	FGWS	CHECKHARD	RD	FGWS	CHECKHARD
SST-2	CNN	12.8	69.2	**77.3**	34.2	83.3	**95.8**	50.9	86.0	**98.8**
	LSTM	13.5	65.4	**77.8**	35.1	81.0	**96.2**	51.4	83.8	**98.5**
	BERT	15.1	67.3	**80.1**	38.4	81.5	**97.0**	55.1	84.6	**98.7**
	RoBERTa	16.4	68.4	**79.5**	41.0	83.0	**97.5**	58.2	85.3	**99.0**
IMDB	CNN	1.7	2.4	**79.2**	5.0	15.7	**96.8**	8.2	31.8	**98.3**
	LSTM	1.9	2.2	**79.9**	5.6	13.2	**96.4**	9.1	27.0	**98.2**
	BERT	3.4	7.7	**70.4**	9.3	23.5	**90.4**	14.2	40.6	**95.3**
	RoBERTa	4.8	6.9	**64.2**	12.9	28.0	**89.4**	19.7	48.6	**94.9**

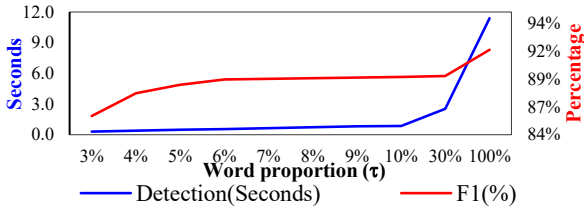
Table 5: Perturbed word suggestion on adversarial text generated by *PWWS*.

Category	**SST-2 (Adv/Org)**	**IMDB (Adv/Org)**
PWWS attack time	0.095	4.298
FGWS	0.014	0.097
CHECKHARD without support	0.044 (0.032 / 0.054)	2.330 (1.381 / 3.146)
CHECKHARD with RoBERTa as support	0.731 (0.376 / 1.006)	11.411 (4.877 / 17.037)

Table 6: Run time for attacking the original text with *PWWS* and detecting adversarial text generated by *PWWS* targeting the CNN model.



(a) SST-2.



(b) IMDB.

Figure 3: Correlation between the detection time and F1 scores of CHECKHARD with RoBERTa as support when detecting adversarial text generated by *PWWS* targeting the CNN model and changing the word proportion τ. The time is averaged for all original and adversarial detections.

is greater than 10%. These results demonstrate the impact of τ in terms of accelerating CHECKHARD, especially with long text, as in the IMDB.

4.4 Discussion

Direct attack: We evaluated CHECKHARD and FGWS under a direct attack. In particular, we used *PWWS* to attack SST-2 text targeting the CNN, which is protected by CHECKHARD and FGWS. CHECKHARD achieves 37.0% accuracy under the *PWWS* attack, which is higher than the 15.2% from FGWS. These results demonstrate that CHECKHARD is better than FGWS in terms of defending against adversarial text.

Parallel processing: Adversarial attacks optimize each step in a sequence until a target model is fooled. Conversely, CHECKHARD can generate all transformed text at once and predict them in parallel. CHECKHARD can thus be accelerated with parallel or distributed computing.

5 Limitations

Beyond word-based attack: CHECKHARD is currently suitable for all current attacks from the TextAttack framework at the character, word, and hybrid levels. To the best of our knowledge, there is no work that detects adversarial text beyond the word level, such as the phrase level as in (Lei et al., 2022) and sentence level as in (Iyyer et al., 2018), so it is still an open problem.

Beyond text classification: CHECKHARD can be directly applied to text classification tasks, for which it is easy to estimate the change in prediction. To apply CHECKHARD to other tasks (such as question answering and translation), we need to define a similar metric to measure the change in prediction.

CHECKHARD with a strange auxiliary and without support: CHECKHARD without support is still unstable when the auxiliary attack is different from the attack used to generate adversarial text, as shown in Table 7 in Appendix B. This limitation can be remedied with support, but it requires a trade-off in run time.

6 Conclusion

In this paper, we propose CHECKHARD by checking the change in the hard label before and after replacing the word with its transformation. Checking is used to detect adversarial text, correct predictions, and suggest perturbed words. The experiments on various attacks, models, and datasets demonstrate that CHECKHARD outperforms existing work.

Acknowledgments

We would like to thank you very much for the anonymous reviewers to provide useful comments.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2890–2896.

Emil Biju, Anirudh Sriram, Pratyush Kumar, and Mitesh M Khapra. 2022. Input-specific attention subnetworks for adversarial detection. In *Findings of the Association for Computational Linguistics (ACL)*, pages 31–44.

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

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *Proceedings of the IEEE Security and Privacy Workshops (SPW)*, pages 50–56.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181.

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

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019. Achieving verified robustness to symbol substitutions via interval bound propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4074–4084.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 16th Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1875–1885.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4120–4133.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the 34th Conference on Artificial Intelligence (AAAI)*, pages 8018–8025.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. 2020. Robust encodings: A framework for combating adversarial typos. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2752–2765.

Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. 2018. Adversarial examples for natural language classification problems. In *OpenReview*.

Thai Le, Noseong Park, and Dongwon Lee. 2022. Shield: Defending textual neural networks against multiple black-box adversarial attacks with stochastic multi-expert patcher. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6661–6674.

Yibin Lei, Yu Cao, Dianqi Li, Tianyi Zhou, Meng Fang, and Mykola Pechenizkiy. 2022. Phrase-level textual adversarial attack with label preservation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and William B Dolan. 2021. Contextualized perturbation for textual adversarial attack. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 5053–5069.

J Li, S Ji, T Du, B Li, and T Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS)*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202.

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

Edoardo Mosca, Shreyash Agarwal, Javier Rando Ramírez, and Georg Groh. 2022. That is a suspicious reaction!: Interpreting logits variation to detect nlp adversarial attacks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Maximilian Mozes, Pontus Stenetorp, Bennett Kleinberg, and Lewis Griffin. 2021. Frequency-guided word substitutions for detecting textual adversarial examples. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 171–186.

Danish Pruthi, Bhuwan Dhingra, and Zachary C Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5582–5591.

Vyas Raina and Mark Gales. 2022. Residue-based natural language adversarial attack detection. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1085–1097.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4902–4912.

Jonathan Rusert and Padmini Srinivasan. 2022. Don't sweat the small stuff, classify the rest: Sample shielding to protect text classifiers against adversarial attacks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Xiaosen Wang, Hao Jin, Yichen Yang, and Kun He. 2021. Natural language adversarial defense through synonym encoding. In *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 823–833.

Xiaosen Wang, Yifeng Xiong, and Kun He. 2022. Randomized substitution and vote for textual adversarial example detection. In *Proceedings of the 38 Conference on Uncertainty in Artificial Intelligence (UAI)*.

Jin Yong Yoo and Yanjun Qi. 2021. Towards improving adversarial training of NLP models. In *Findings of the Association for Computational Linguistics (EMNLP)*, pages 945–956.

KiYoon Yoo, Jangho Kim, Jiho Jang, and Nojun Kwak. 2022. Detection of word adversarial examples in text classification: Benchmark and baseline via robust density estimation. In *Findings of the Association for Computational Linguistics (ACL)*, pages 3656–3672.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6066–6080.

Jiehang Zeng, Xiaoqing Zheng, Jianhan Xu, Linyang Li, Liping Yuan, and Xuanjing Huang. 2021. Certified robustness to text adversarial attacks by randomized [mask]. *arXiv preprint arXiv:2105.03743*.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuan-Jing Huang. 2021. Defense against synonym substitution-based adversarial attacks via dirichlet neighborhood ensemble. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5482–5492.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. 2019. Learning to discriminate perturbations for blocking adversarial attacks in text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4906–4915.

A Perturbed Word Ratio

We calculated the ratio of the number of perturbed words in adversarial text from the SST-2 and IMDB testing sets as shown in Figure 4. One-word perturbation is more prominent than others, especially with SST-2.

B Other Auxiliary Attacks

In addition to *DeepWordBug* and *TextFooler*, as reported in Table 3, we conducted other auxiliary attacks for CHECKHARD with and without support as shown in Table 7. Similarly, CHECKHARD with RoBERTa support is better than that without support across all of these auxiliary attacks.

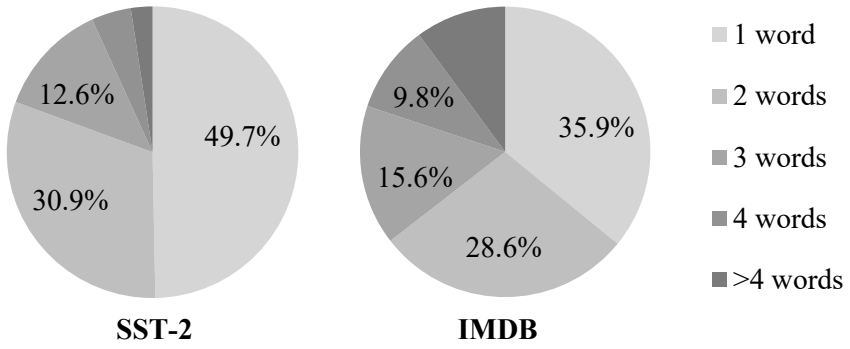


Figure 4: Ratio of the number of perturbed words in adversarial text generated by *PWWS* targeting the CNN model.

Method	F1	Correction
CHECKHARD(*Pruthi*) without support	73.1	77.4
CHECKHARD(*TextBugger*) without support	73.1	77.6
CHECKHARD(*IGA*) without support	67.3	70.7
CHECKHARD(*Faster-Alzantot*) without support	66.7	67.4
CHECKHARD(*Kuleshov*) without support	68.6	71.2
CHECKHARD(*BAE*) without support	72.5	73.6
CHECKHARD(*PSO*) without support	67.1	69.5
CHECKHARD(*Pruthi*) with RoBERTa as support	79.8	80.7
CHECKHARD(*TextBugger*) with RoBERTa as support	82.0	88.8
CHECKHARD(*IGA*) with RoBERTa as support	82.3	88.6
CHECKHARD(*Faster-Alzantot*) with RoBERTa as support	80.5	88.3
CHECKHARD(*Kuleshov*) with RoBERTa as support	83.6	93.4
CHECKHARD(*BAE*) with RoBERTa as support	80.1	92.1
CHECKHARD(*PSO*) with RoBERTa as support	81.4	92.8

Table 7: Other auxiliary attacks for which CHECKHARD detected adversarial text and corrected the predictions from adversarial text generated by *PWWS* targeting the CNN model on SST-2.

Method	F1	Correction
CHECKHARD with ALBERT as support	84.6	93.1
CHECKHARD with DistilBERT as support	85.3	93.5
CHECKHARD with BERT as support	86.3	93.7
CHECKHARD with ALBERT + DistilBERT as support	84.1	92.3
CHECKHARD with ALBERT + BERT as support	85.0	92.6
CHECKHARD with DistilBERT + BERT as support	84.9	93.0
CHECKHARD with ALBERT + DistilBERT + BERT as support	85.1	93.5

Table 8: Other supports for CHECKHARD used to detect adversarial text and correct the predictions from adversarial text generated by *PWWS* targeting the CNN model on SST-2.

C Other Support Models

In addition to LSTM and RoBERTa, as reported in Table 3, we conducted experiments to evaluate other supports[6] as shown in Table 8. Similar to LSTM and RoBERTa, when CHECKHARD is combined with these individual supports and their combination, it achieves stable performances ranging from [84.1%, 86.3%] and [92.3%, 93.5%] in terms of F1 and correction scores, respectively.

[6]These remaining support models are directly mentioned in TextAttack's source code at https://github.com/QData/TextAttack/blob/master/textattack/model_args.py