VoteTRANS: Detecting Adversarial Text without Training by Voting on Hard Labels of Transformations

Hoang-Quoc Nguyen-Son[1], **Seira Hidano**[1], **Kazuhide Fukushima**[1], **Shinsaku Kiyomoto**[1], and **Isao Echizen**[2]
[1]KDDI Research, Inc., Japan
[2]National Institute of Informatics, Japan
[1]{xso-guen,se-hidano,ka-fukushima,sh-kiyomoto}@kddi.com
[2]iechizen@nii.ac.jp

Abstract

Adversarial attacks reveal serious flaws in deep learning models. More dangerously, these attacks preserve the original meaning and escape human recognition. Existing methods for detecting these attacks need to be trained using original/adversarial data. In this paper, we propose detection without training by voting on hard labels from predictions of transformations, namely, VoteTRANS. Specifically, VoteTRANS detects adversarial text by comparing the hard labels of input text and its transformation. The evaluation demonstrates that VoteTRANS effectively detects adversarial text across various state-of-the-art attacks, models, and datasets.

1 Introduction

Deep learning models are sensitive to changes in input text from an adversarial attack. Even a slight change enormously impacts the prediction of models. More dangerously, these changes still preserve the input meaning, so attacks remain unrecognized by humans. This vulnerability has negatively affected the reputation of deep learning models.

In contrast to adversarial text defense, fewer works have been proposed to detect adversarial texts. Previous works detected such texts via perturbed word identification (Zhou et al., 2019; Mozes et al., 2021), synonyms (Wang et al., 2022b), density (Yoo et al., 2022), attention (Biju et al., 2022), PCA (Raina and Gales, 2022), transformer (Wang et al., 2022a), and word importance (Mosca et al., 2022). Since existing works need original/adversarial data to train detectors, they are sensitive to new adversarial attacks.

Motivation: Adversarial text must satisfy two criteria: the text must (1) change the prediction of a target model while (2) preserving the original meaning. Few texts can comply with both criteria. For example, we randomly selected original text from AG News and used a probability-weighted word saliency (*PWWS*) attack (Ren et al., 2019) to generate adversarial text (Figure 1). *PWWS* replaces original words to fool a target model (CNN). During this generation process, only the final text fooled the target CNN, while other texts were still correctly predicted by the target CNN and another model, such as RoBERTa. We find the same trend for other AG News texts and IMDB movie reviews as shown in Appendix A.

Contributions: We propose a simple detector by voting on hard labels of transformations (VoteTRANS). In particular, we generate a transformation set for each word in the input text. We then compare the original hard label from the input text and the majority vote from each transformation set. If we find any difference in the comparison, the adversarial text is identified. In summary, our contributions are listed as follows:

- To the best of our knowledge, VoteTRANS is the first model to detect adversarial text from various attacks without training. Moreover, we do not modify a target model and only use the target as a black-box setting for prediction. VoteTRANS can thus be applied to a wide range of various models.
- Experiments on various attacks, models, and datasets demonstrate that VoteTRANS outperforms state-of-the-art detectors.
- VoteTRANS can run with all seventeen current attacks related to text classification from TextAttack framework (Morris et al., 2020). VoteTRANS is also automatically compatible with future attacks from this framework without changing its source code[1].

[1]VoteTRANS is available at https://github.com/quocnsh/VoteTRANS

Findings of the Association for Computational Linguistics: ACL 2023, pages 5090–5104

					CNN	RoBERTa
Original:	*...a project to*	*test*	*wireless*	*public transport...*	Sci/Tech	Sci/Tech
		run			Sci/Tech	Sci/Tech
		tryout	*wireless*		Sci/Tech	Sci/Tech
			tuner		Sci/Tech	Sci/Tech
			radio		Sci/Tech	Sci/Tech
Adversarial:	*...a project to*	*tryout*	***radiocommunication***	*public transport...*	Business	Sci/Tech

Figure 1: A process generates adversarial text by synonym-based transformation targeting a CNN model. During this process, only the final adversarial text fools the CNN, while other texts are still correctly predicted by the CNN and RoBERTa.

2 Related Work

2.1 Adversarial Attack

Many adversarial attacks have emerged since 2018 and have been supported by TextAttack (Morris et al., 2020). We categorize all seventeen attacks from TextAttack related to text classification by their levels.

In word level, *Alzantot* (Alzantot et al., 2018), *Faster Alzantot* (Jia et al., 2019), and an improved generic algorithm (*IGA*)(Wang et al., 2021) ran a generic algorithm to generate adversarial text. *PWWS* (Ren et al., 2019) and *TextFooler* (Jin et al., 2020) transformed a text with synonyms from a fixed lexical database (WordNet) and word embedding, respectively. Zang et al. (2020) applied particle swarm optimization (*PSO*) to synonyms from a big database (HowNet). *Kuleshov* (Kuleshov et al., 2018) and *A2T* (Yoo and Qi, 2021) measured the similarity with GPT-2 and DistillBERT, respectively. *BERT-Attack* (Li et al., 2020) extracted the synonyms from a masked language model. *BAE* (Garg and Ramakrishnan, 2020) used both word insertion and replacement with the masked model. *CLARE* (Li et al., 2021) extended this model by merging two consecutive words. *Checklist* (Ribeiro et al., 2020) verified the model consistency by changing an input word into a neutral entity (e.g., location, name, or number). *Input-Reduction* (Feng et al., 2018) removes the word of lowest importance until the target model changes the prediction.

In character and hybrid levels, *HotFlip* (Ebrahimi et al., 2018) accessed the gradient of a target model to manipulate the loss change. *DeepWordBug* (Gao et al., 2018) transformed words using four-character operations including swapping, substitution, deletion, and insertion. *Pruthi* (Pruthi et al., 2019) added a QWERTY keyboard operator. *TextBugger* (Li et al., 2019) combined character and word operators.

2.2 Adversarial Detection

Zhou et al. (2019) trained a BERT model to identify the perturbed words. Mozes et al. (2021) focused on low-frequency words that were likely changed by an attacker. Wang et al. (2022b) detected the adversarial text via its synonyms. Yoo et al. (2022) and Biju et al. (2022) distinguished adversarial text with original text based on density estimation and attention input, respectively. Raina and Gales (2022) showed that adversarial text induces residues (larger components than original text) in PCA eigenvector. Wang et al. (2022a) finetuned a transformer model on compliant and adversarial text, which is not required to fool a target model. The change in the requirement efficiently defends against a wide range of adversarial attacks. Mosca et al. (2022) extracted the word importance as a feature to classify original and adversarial text.

Compared with VoteTRANS, previous detectors need adversarial or/and original data to train their models or optimizes their models on training data to satisfy some requirements (such as FPR as in Yoo et al. (2022)). These methods often more suitable for word-based than character-based attacks. On the other hand, VoteTRANS can be applied to various kinds of attacks at both word-level as in *PWWS* and character-level as in *TextBugger* as well as other modifications, such as deletion as in *Input-Reduction* or *CLARE* and insertion as in *BAE*. For example, *CLARE* deletes a word by merging it with a nearby word. When applying the same merging operator on each input word, VoteTRANS will change the attacked words and observe the change in target predictions to detect the attacked text.

3 Voting on the Hard Labels of Transformations (VoteTRANS)

Problem statement: We follow notations from *TextFooler* paper (Jin et al., 2020). Given N texts, $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$ corresponding to

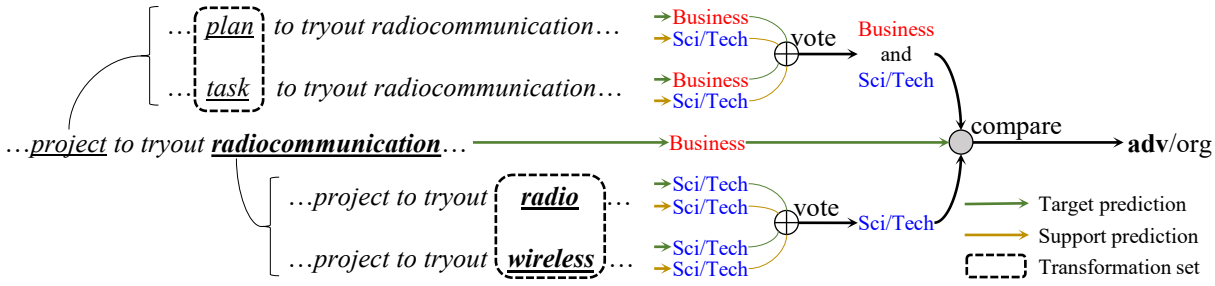


Figure 2: Given input text, we replace each word (e.g., "*project*" and "***radiocommunication***") with the corresponding transformation set. The modified text is then predicted by a target and support models as hard labels. These labels are voted and compared with the original label to determine whether the input text is adversarial or original.

N labels $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_N\}$, a target model $F : \mathcal{X} \rightarrow \mathcal{Y}$ maps the input space $\mathcal{X}$ to the label space $\mathcal{Y}$. Adversarial text X_{adv} generated from input $X \in \mathcal{X}$ should comply with the following constraints:

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

where $\text{Sim}(X_{\text{adv}}, X)$ measures the similarity between adversarial text X_{adv} and original text X; ϵ is the minimum similarity.

Our objective is to determine whether input text X is adversarial or original text. The process of VoteTRANS is depicted in Figure 2, and the overall algorithm is summarized in Algorithm 1.

Model detail: To process input text X, we use a target model F and an auxiliary attack A (e.g., *PWWS*). An optional word ratio α can be used to speed up the detection process. Moreover, a support model list F^{sup} is used to improve the performance. The support models and the target model should solve the same task such as sentiment analysis.

First, we create a list W including input words that are sorted by their importance using word importance score estimation in the auxiliary attack A (lines 2-5). For example, *PWWS* calculates the change in the predictions before and after replacing a certain word with an unknown word to estimate word importance. The impact of word importance is shown in Appendix B.

Second, we select the top words W^* (line 6) with the word ratio threshold α (100% by default). Since VoteTRANS maintains the performance with $\alpha = 100\%$, it remarkably improves the run time with a smaller α, with a slight change in performance, as shown in Figures 4a and 4b in the experimental results.

Third, we obtain a transformed set W^{trans} for each word w_j in W^* (line 9) using word transformation in A. For example, *PWWS* uses synonyms from WordNet to form the transformed set.

Fourth, we replace each transformed word in W^{trans} for the corresponding w_j (line 11) in X and checks the second constraint in Equation 1 (line 12). The $\text{Sim}(\cdot)$ and ϵ are provided by A. This step is mainly used to preserve the original meaning of the input in the transformed text. For example, *PWWS* uses stop word checking for this step.

Fifth, we construct Y^{trans} containing the label predictions for the valid X' from target model F and support model list F^{sup} (lines 13-16). We then use the majority vote on Y^{trans} to obtain the top majority classes Y' with the highest occurrence in Y^{trans}.

Finally, we check whether the input text X is adversarial or original based on the input label Y and majority set Y'. If Y does not belong to Y' or Y belongs to Y', which contains more than one majority class, we decide that X is adversarial text (lines 20-22). If we cannot decide if X is adversarial text after checking all words in W^*, X is considered original text.

4 Evaluation

4.1 Comparison

We follow recent work (Mosca et al., 2022) to conduct the same experiments including attacks, datasets, models, parameter settings, evaluation metrics, number of training/testing samples, etc. In particular, evaluation was performed with adversarial text generated by four attacks[2] including *PWWS*, *TextFooler*, *IGA*, and *BAE*. These attacks targeted four common models[3] (LSTM, CNN, DistilBERT, and BERT) on IMDB (235.72 words/text), Yelp

[2]All attacks ran with the default settings from the TextAttack framework.

[3]All pretrained models were reused from the TextAttack framework.

Algorithm 1: Adversarial text detection by VoteTRANS.

Input : Input text $X = \{w_1, w_2 \cdots\}$; Target model F; Auxiliary attack A; Optional: {Word ratio α (100% as default); Support models $F^{\text{sup}} = \{F_1^{\text{sup}}, F_2^{\text{sup}} \cdots\}$ (an empty as default)}

Output : Adversarial text detection (*True*/*False*)

$Y \leftarrow F(X)$
for *each word* w_i *in* X **do**
  calculate importance score I_{w_i} using A
end for
Create a list W of all words $w_i \in X$ by sorting in descending order of the importance score I_{w_i}
$W^* \leftarrow$ obtains the top words from W with ratio α
for *each word* w_j *in* W^* **do**
  Transformation label list $Y^{\text{trans}} = \{\}$
  Create transformation set W^{trans} from w_j using A
  for *each word* w_k^{trans} *in* W^{trans} **do**
    $X' =$ replace w_j with w_k^{trans} in X
    if *checking* $(\text{Sim}(X', X) \geq \epsilon)$ *is satisfied by using* A **then**
      Add $F(X')$ to list Y^{trans}
      for *each* F_l^{sup} *in* F^{sup} **do**
        Add $F_l^{\text{sup}}(X')$ to list Y^{trans}
      end for
    end if
  end for
  $Y' \leftarrow$ majority vote on Y^{trans}
  if $(Y \notin Y')or(Y \in Y'$ *and* Y' *has more than one majority class*$)$ **then**
    return *True* ▷ adversarial text
  end if
end for
return *False* ▷ original text

Polarity (YELP) (135.21 words/text), AG News (38.78 words/text) and Rotten Tomatoes movie reviews (RTMR) (18.65 words/text).

We compared VoteTRANS with FGWS (Mozes et al., 2021) and WDR (Mosca et al., 2022). FGWS is claimed as a state-of-the-art in all existing detection papers that we know; WDR is one of the most recent published work. Both FGWS and WDR need to be trained on the adversarial text generated from a specific configuration (model, data, *attack*). Table 1 shows the results on two configurations: (DistilBERT, IMDB, *PWWS*) with 3000 training samples (Table 1a)[4] and (DistilBERT, AG News, *PWWS*) with 2400 training samples (Table 1b). We generated adversarial text from testing sets and put them aside with corresponding original text to form balanced samples. However, the two configurations (DistillBERT, RTMR, *IGA*) and (DistillBERT, AG News, *IGA*) had only 480 and 446 balanced samples, respectively. We thus chose 500 balanced samples as testing data for other configurations.

VoteTRANS uses an auxiliary attack to detect adversarial text without training. The auxiliary attack may be the same as the attack used to generate the adversarial text, namely, $\text{VoteTRANS}_{\text{same}}$. We also demonstrate the capability of VoteTRANS using a fixed auxiliary attack (*PWWS*) to detect adversarial text generated from other attacks, namely, $\text{VoteTRANS}_{\text{diff}}$. Both $\text{VoteTRANS}_{\text{same}}$ and $\text{VoteTRANS}_{\text{diff}}$ uses RoBERTa as a support for all experiments[5]. Other auxiliary attacks and supports will be discussed

[4]Since VoteTRANS does not need to train, it produce the same performance with any data-splits of the training configuration. We average the performance on 30 different data-splits of FGWS and WDR and ignore their variance scores for simplification.

[5]Since TextAttack does not support RoBERTa for YELP, we used ALBERT as the support instead.

Configuration			FGWS		WDR		VoteTRANS$_{\text{same}}$		VoteTRANS$_{\text{diff}}$	
Model	**Data**	***Attack***	**F1**	**Recall**	**F1**	**Recall**	**F1**	**Recall**	**F1**	**Recall**
DistilBERT	IMDB	*PWWS*	89.5	82.7	92.1	94.2	**96.9**	**98.4**	-	-
LSTM	IMDB	*PWWS*	80.0	69.6	84.1	86.8	**94.8**	**97.6**	-	-
CNN	IMDB	*PWWS*	86.3	79.6	84.3	90.0	**95.4**	**98.8**	-	-
BERT	IMDB	*PWWS*	89.8	82.7	92.4	92.5	**97.4**	**98.4**	-	-
DistilBERT	AG News	*PWWS*	89.5	84.6	93.1	96.1	**95.3**	**97.6**	-	-
DistilBERT	IMDB	*TextFooler*	86.0	77.6	94.2	97.3	**97.8**	99.6	97.7	**100.0**
DistilBERT	IMDB	*IGA*	83.8	74.8	88.5	95.5	95.8	**99.2**	**95.9**	**99.2**
BERT	YELP	*PWWS*	91.2	85.6	89.4	85.3	**97.4**	**98.4**	-	-
BERT	YELP	*TextFooler*	90.5	84.2	95.9	97.5	**97.4**	**98.8**	97.0	98.0
DistilBERT	RTMR	*PWWS*	78.9	67.8	74.1	85.1	**83.8**	**88.0**	-	-
DistilBERT	RTMR	*IGA*	68.1	55.2	70.4	90.2	**86.9**	**90.4**	80.5	82.4
DistilBERT	IMDB	*BAE*	65.6	50.2	88.0	96.3	**97.7**	**100.0**	96.3	99.2
DistilBERT	AG News	*BAE*	55.8	44.0	81.0	**95.4**	**85.8**	93.2	85.3	92.8
DistilBERT	RTMR	*BAE*	29.4	18.5	68.5	**82.2**	**79.1**	80.8	65.5	60.8
Overall average			77.5	68.4	85.4	91.7	**93.0**	**95.7**	-	-

(a) FGWS and WDR trained on configuration (DistilBERT, IMDB, *PWWS*).

Configuration			FGWS		WDR		VoteTRANS$_{\text{same}}$		VoteTRANS$_{\text{diff}}$	
Model	**Data**	***Attack***	**F1**	**Recall**	**F1**	**Recall**	**F1**	**Recall**	**F1**	**Recall**
DistilBERT	AG News	*PWWS*	89.5	84.6	93.6	94.8	**95.3**	**97.6**	-	-
LSTM	AG News	*PWWS*	88.9	84.9	94.0	94.2	**95.5**	**98.4**	-	-
CNN	AG News	*PWWS*	90.6	87.6	91.1	91.2	**96.5**	**99.2**	-	-
BERT	AG News	*PWWS*	88.7	83.2	92.5	93.0	**94.6**	**98.0**	-	-
DistilBERT	IMDB	*PWWS*	89.5	82.7	91.4	93.0	**96.9**	**98.4**	-	-
DistilBERT	AG News	*TextFooler*	87.0	79.4	95.7	97.3	**96.7**	**98.4**	95.7	98.0
DistilBERT	AG News	*IGA*	68.6	58.3	86.7	93.6	**96.5**	**99.2**	93.2	95.6
BERT	YELP	*PWWS*	91.2	85.6	86.2	77.2	**97.4**	**98.4**	-	-
BERT	YELP	*TextFooler*	90.5	84.2	95.4	94.7	**97.4**	**98.8**	97.0	98.0
DistilBERT	RTMR	*PWWS*	78.9	67.8	75.8	78.5	**83.8**	**88.0**	-	-
DistilBERT	RTMR	*IGA*	68.1	55.2	73.7	85.4	**86.9**	**90.4**	80.5	82.4
DistilBERT	IMDB	*BAE*	65.6	55.2	88.1	97.0	**97.7**	**100.0**	96.3	99.2
DistilBERT	AG News	*BAE*	55.8	44.0	**86.4**	**94.5**	85.8	93.2	85.3	92.8
DistilBERT	RTMR	*BAE*	29.4	18.5	71.0	75.2	**79.1**	**80.8**	65.5	60.8
Overall average			77.1	68.8	86.4	89.4	**92.4**	**95.2**	-	-

(b) FGWS and WDR trained on configuration (DistilBERT, AG News, *PWWS*).

Table 1: Comparison between VoteTRANS and two state-of-the-art detectors, FGWS and WDR. While both FGWS and WDR need to be trained on a specific configuration (model, data, *attack*), VoteTRANS directly detects adversarial text without training. VoteTRANS$_{\text{same}}$ uses an auxiliary attack that is the same as the attack used to generate adversarial texts. If the adversarial text is not generated by *PWWS*, VoteTRANS$_{\text{diff}}$ uses *PWWS* as the auxiliary attack.

later.

In general, WDR is better than FGWS when they are trained on (DistilBERT, IMDB, *PWWS*) as shown in Table 1a. While FGWS achieves an F1 score of 77.5 and recall of 68.4 on average, WDR exhibits improved F1 score and recall metrics of 85.4 and 91.7, respectively. All F1 scores of VoteTRANS$_{\text{same}}$ outperform those of WDR. VoteTRANS$_{\text{same}}$ also exhibits a recall improved by 4.0 points from 91.7 of WDR. VoteTRANS$_{\text{diff}}$ is competitive with VoteTRANS$_{\text{same}}$ for medium (AG News) and long text (IMDB), while the performance of VoteTRANS$_{\text{diff}}$ on short text (RTMR) is degraded, similar to other detectors.

Scenario	Method	F1	Recall
Unknown Attack	VoteTRANS$_{\text{diff}}$ (*BAE*) without support	94.4	93.6
	VoteTRANS$_{\text{diff}}$ (*DeepWordBug*) without support	94.2	97.2
	VoteTRANS$_{\text{diff}}$ (*BAE*) with RoBERTa as support	**96.7**	99.6
	VoteTRANS$_{\text{diff}}$ (*DeepWordBug*) with RoBERTa as support	95.2	**100.0**
	VoteTRANS$_{\text{diff}}$ (*Checklist*) with RoBERTa as support	83.9	74.0
	VoteTRANS$_{\text{diff}}$ (*Input-Reduction*) with RoBERTa as support	92.8	**100.0**
	VoteTRANS$_{\text{diff}}$ (*A2T*) with RoBERTa as support	95.7	96.8
	VoteTRANS$_{\text{diff}}$ (*IGA*) with RoBERTa as support	95.7	97.2
	VoteTRANS$_{\text{diff}}$ (*Pruthi*) with RoBERTa as support	95.8	99.6
	VoteTRANS$_{\text{diff}}$ (*Alzantot*) with RoBERTa as support	95.9	97.6
	VoteTRANS$_{\text{diff}}$ (*PSO*) with RoBERTa as support	96.1	97.6
	VoteTRANS$_{\text{diff}}$ (*Faster-Alzantot*) with RoBERTa as support	96.4	97.6
	VoteTRANS$_{\text{diff}}$ (*TextBugger*) with RoBERTa as support	96.5	98.8
	VoteTRANS$_{\text{diff}}$ (*TextFooler*) with RoBERTa as support	96.5	98.0
	VoteTRANS$_{\text{diff}}$ (*Kuleshov*) with RoBERTa as support	96.6	97.6
Known Attack	FGWS	90.6	87.6
	WDR	91.1	91.2
	VoteTRANS$_{\text{same}}$ (*PWWS*) without support	95.2	94.8
	VoteTRANS$_{\text{same}}$ (*PWWS*) with LSTM as support	95.9	98.8
	VoteTRANS$_{\text{same}}$ (*PWWS*) with RoBERTa as support	96.5	**99.2**
	VoteTRANS$_{\text{same}}$ (*PWWS*) with LSTM+RoBERTa as supports	**97.4**	98.4

Table 2: Detecting adversarial text generated by *PWWS* targeting CNN on AG News.

Category	RTMR(Adv/Org)	AG News(Adv/Org)	IMDB(Adv/Org)
PWWS attack time	0.77	2.84	26.36
FGWS/WDR	0.04	0.08	1.01
VoteTRANS$_{\text{same}}$ without support	0.03(0.02/0.04)	0.08(0.03/0.13)	2.00(0.67/3.33)
VoteTRANS$_{\text{same}}$ with RoBERTa as support	0.69(0.28/1.09)	1.42(0.15/2.69)	8.94(0.37/17.52)

Table 3: Run time for attacking original text by *PWWS* and detecting adversarial text generated by *PWWS* targeting the CNN model.

In detail, we cluster the experimental results into three groups based on their performances. The first group with high performances includes configurations from similar attacks (*PWWS*, *TextFooler*, and *IGA*) on long (IMDB and YELP) and medium text (AG News). The second group includes configurations from *PWWS* and *IGA* on short text (RTMR). The last group includes the remaining configurations related to *BAE*. While *BAE* uses flexible synonyms based on word context, the other attacks use fixed synonyms for a certain word.

All of the detectors work well for the lengthy text of the first group, especially VoteTRANS$_{\text{same}}$, with both scores being between 94.8 and 99.6. However, less information can be extracted from the short text in the second group. While FGWS is competitive with WDR in this group, VoteTRANS$_{\text{same}}$ performs better, especially in the F1 score. In the last group, *BAE* remarkably affects the detectors, especially with FGWS in medium and short text. FGWS is defeated, with scores less than 50.0 (random guess). VoteTRANS$_{\text{same}}$ still maintains its high performance for long text and is competitive with WDR for medium and short text.

Table 1b shows experiments where FGWS and WDR are trained on (DistilBERT, AG News, *PWWS*). We train WDR with other word-based attacks including *TextFooler*, *IGA*, and *BAE* and reach similar results as shown in Appendix C. While FGWS and WDR obtain scores less than 90.0 on average, VoteTRANS$_{\text{same}}$ retains its performance, with 92.4 F1 and 95.2 recall. This demonstrates the resilience of VoteTRANS$_{\text{same}}$ across various models, datasets, and attacks.

4.2 Ablation Studies

We studied variants of VoteTRANS and compared them with FGWS and WDR. These detectors identified adversarial texts generated by *PWWS* targeting the CNN on AG News (Table 2). VoteTRANS is presented in two scenarios: an unknown attack ($\text{VoteTRANS}_{\text{diff}}$) and a known attack ($\text{VoteTRANS}_{\text{same}}$).

For an unknown attack, we use a word-based *BAE* and character-based *DeepWordBug* as the auxiliary attacks for $\text{VoteTRANS}_{\text{diff}}$ without support. $\text{VoteTRANS}_{\text{diff}}$ achieves high performances with both auxiliaries. Other auxiliaries for $\text{VoteTRANS}_{\text{diff}}$ without support are mentioned in Appendix D. The use of the RoBERTa model as support boosts the overall performance. Other attacks from the TextAttack were conducted and are listed in increasing order of F1 scores. Among these attacks, *BERT-Attack* and CLARE are ignored because both use the same masked language model used in *BAE*, and the three attacks reached similar performances. *HotFlip* is not supported for CNN. The results show that $\text{VoteTRANS}_{\text{diff}}$ can use any attack as the auxiliary, with all scores being greater than or equal to 92.8, except for those of *Checklist*. *Checklist* generates independent adversarial text with any model and causes low performance, as mentioned in the owner paper (Ribeiro et al., 2020). The results from various auxiliaries demonstrate that $\text{VoteTRANS}_{\text{diff}}$ can detect adversarial text without attack information.

For a known attack, $\text{VoteTRANS}_{\text{same}}$ without support outperforms FGWS and WDR. $\text{VoteTRANS}_{\text{same}}$ is improved by using random support such as LSTM or RoBERTa. A stronger model (RoBERTa) helps $\text{VoteTRANS}_{\text{same}}$ more than LSTM. Both can also be used together to support $\text{VoteTRANS}_{\text{same}}$ and improve the F1 score, but the adversarial recall is slightly affected. Other available supports from TextAttack for AG News are mentioned in Appendix E.

While character-based attacks are still a challenge for both WDR and FGWS as mentioned in their papers, VoteTRANS still detects such adversarial text upto 97.6% F1 and 99.6% recall as shown in Appendix F. It indicates the flexible of VoteTRANS with various attack levels.

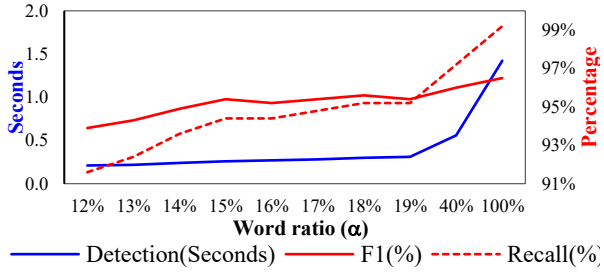
4.3 Run Time

Since VoteTRANS uses an auxiliary attack to detect adversarial text, we compared the run time of $\text{VoteTRANS}_{\text{same}}$ with that of the corresponding attack. Table 3 shows a comparison of adversarial text generated by *PWWS* targeting the CNN model on short text (RTMR), medium text (AG News), and long text (IMDB); $\text{VoteTRANS}_{\text{diff}}$, other attacks, and models reached similar ratios. We also compared the detection times obtained from WDR/FGWS, which both use the target model to predict the text n times, where n is the number of words in an input text. $\text{VoteTRANS}_{\text{same}}$ is reported without support and with RoBERTa support. We also separately appended the detection time for adversarial and original text for $\text{VoteTRANS}_{\text{same}}$, while other detectors ran for the same time for both.

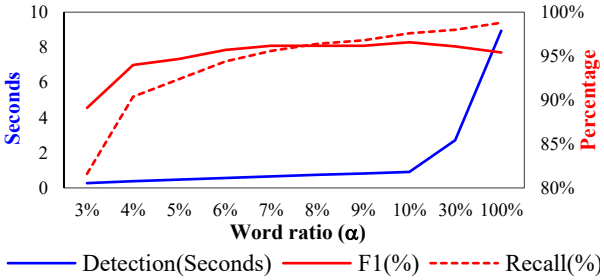
The run times of both the attack and detectors are affected by the text length. FGWS and WDR need less than 2 seconds to detect text. $\text{VoteTRANS}_{\text{same}}$ processes adversarial text much faster than original text because most of the adversarial text is identified early with lines 20-22 of Algorithm 1. Thanks to line 12 of Algorithm 1 for filtering many transformed texts, $\text{VoteTRANS}_{\text{same}}$ without support run similar to FGWS and WDR for short and medium text. For long text, $\text{VoteTRANS}_{\text{same}}$ needs 0.67 seconds for adversarial text. $\text{VoteTRANS}_{\text{same}}$ with RoBERTa as support even completes processing the adversarial text from IMDB with only 0.37 seconds. This demonstrates that RoBERTa accelerates adversarial text processing.

$\text{VoteTRANS}_{\text{same}}$ runs faster by decreasing the word ratio α in Algorithm 1. α determines the number of words that are processed. While $\text{VoteTRANS}_{\text{same}}$ with RoBERTa as support processes RTMR text in a reasonable time, we evaluate the change in α for processing AG News and IMDB text, as shown in Figure 4a and Figure 4b, respectively. For AG News, although the detection time is mostly steady at 0.25 seconds, with α between 12% and 19%, the F1/recall ratio remarkably increases from 93.9/91.6 to 95.4/95.2. The run time is worsened to 1.42 seconds with the largest α of 100%, but the F1 scores are only slightly increased.

For IMDB text, $\text{VoteTRANS}_{\text{same}}$ achieves a high performance, even with a small α. When α is increased from 3% to 10%, the run time/F1/recall increases from 0.28/89.1/81.6 to 0.91/96.6/97.6. The recall is improved up to 98.8 with a maximum α of 100%, but the corresponding F1 score drops slightly to 95.4. The F1 score is affected by some of the original text misclassified as adversarial text.



(a) AG News.



(b) IMDB.

Figure 3: Correlation between the detection time and performance of $\text{VoteTRANS}_{\text{same}}$ with RoBERTa as support to detect adversarial text generated by *PWWS* targeting the CNN model when changing the word ratio α. The time is averaged for all original and adversarial detection.

In particular, when processing with 12% and 5% of the medium text (AG News) and long text (IMDB), respectively, $VoteTRANS_{same}$ with RoBERTa as support takes approximately 0.21 seconds and 0.47 seconds, competitive with 0.08 seconds and 1.01 seconds produced from FGWS/WDR, while keeping F1/recall scores (93.9/91.6 for AG News and 94.7/92.4 for IMDB), higher than FGWS (90.6/87.6 and 86.3/79.6) and WDR (91.1/91.2 and 84.3/90.0). With these ratios, $VoteTRANS_{same}$ with RoBERTa only needs 32.0 and 61.1 for AG News and IMDB, respectively (see Appendix G for other ratios).

4.4 Discussion

Detection with high confidence text: VoteTRANS still keeps 79.2% of F1 score on detecting adversarial text from AG News with its confidence greater than or equal to 90%. In contrast, WDR drops the score to 25.0% (see other confidences and IMDB text in Appendix H).

Detection with only hard labels: VoteTRANS only uses soft labels of predictions from a target model via an auxiliary attack to calculate importance scores (line 3 of Algorithm 1). However, these scores are only used to accelerate detection by selecting the top words in line 6. Without these scores, VoteTRANS achieves an identical performance by processing all words. Therefore, VoteTRANS is compatible with any target model that only provides hard labels.

Configuration	AG News	IMDB
Without detector	88.8	100.0
With WDR	5.2	16.4
With VoteTRANS	**0.4**	**4.3**

Table 4: Success rate under an adaptive attack.

Parallel processing: An adversarial attack needs to perturb individual words of input text in sequence to optimize the perturbed text in each step until a target model is fooled. On the other hand, VoteTRANS can create independent transformation sets for individual words and process them in parallel. VoteTRANS can accelerate the process with parallel or distributed computing.

Adaptive attack: An attacker may be aware of the existence of a detector and fool both a target and the detector. We evaluate *PWWS* targeting CNN models on AG News and IMDB as shown in Table 4. Although *PWWS* strongly attacks the CNN models with more than 88% of success rate, it hardly bypasses detectors, especially VoteTRANS.

5 Conclusion

We propose VoteTRANS, a method for detecting adversarial text without training by voting on hard labels of text after transformation. VoteTRANS outperforms state-of-the-art detectors under various attacks, models, and datasets. Moreover, VoteTRANS is flexible at detecting a restricted scenario when an attack is unknown. VoteTRANS also straightforwardly detects adversarial text from a new attack without modifying the architecture.

6 Limitations

Auxiliary attack and supports: VoteTRANS without support works well with an auxiliary attack which is the same with the target attack. In contrast, VoteTRANS with support achieves stable results with any auxiliary attack but it runs slower.

Short text and susceptible text: A short text is more difficult to detect than a long text. Susceptible text may bypass VoteTRANS as mentioned in Appendix I. However, the short text and susceptible

text are often unnatural and unclear meaning, respectively, so they are easily recognized by humans. Therefore, we recommend that humans recheck suspicious text with an abnormal ratio in the voting process of VoteTRANS (line 19 of Algorithm 1).

Beyond word-based attacks: We detect adversarial text up to word-based attacks, which change a few characters or words and are often imperceptible to humans. Other attacks remarkably affect the naturalness with a large change such as sentence-based attacks as in Iyyer et al. (2018).

Beyond text classification: We evaluate VoteTRANS on adversarial attacks targeting text classification. In contrast, the other tasks do not well-define a standard for generating adversarial text. For example, attacks targeting sequence models need to determine a threshold for $BLEU$ score, which is aimed to minimize, but whether the score is sufficient for an adversarial text is still in question.

Acknowledgments

This work was partially supported by JST CREST Grants JPMJCR20D3, Japan.

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.

Shi Feng, Eric Wallace, Alvin Grissom II, Pedro Rodriguez, Mohit Iyyer, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretation difficult. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3719–3728.

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.

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.

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

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-Ramirez, 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)*, pages 7806–7816.

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)*, pages 3836–3848.

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.

Jiayi Wang, Rongzhou Bao, Zhuosheng Zhang, and Hai Zhao. 2022a. Distinguishing non-natural from natural adversarial samples for more robust pre-trained language model. In *Findings of the Association for Computational Linguistics (ACL)*, pages 905–915.

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.

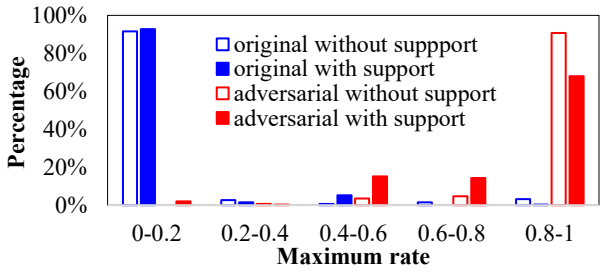
Xiaosen Wang, Yifeng Xiong, and Kun He. 2022b. Randomized substitution and vote for textual adversarial example detection. In *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence (UAI)*.

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.

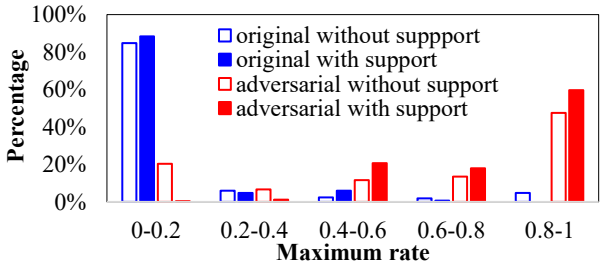
KiYoon Yoo, Jangho Kim, Jiho Jang, and Nojun Kwak. 2022. Detection of word adversarial examples in text classification: Benchmark and baseline via robust density estimation. In *Findings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3656–3672.

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.

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) AG News.

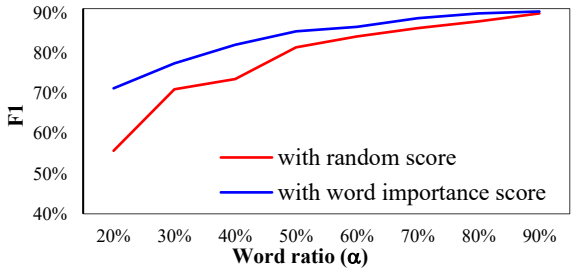


(b) IMDB.

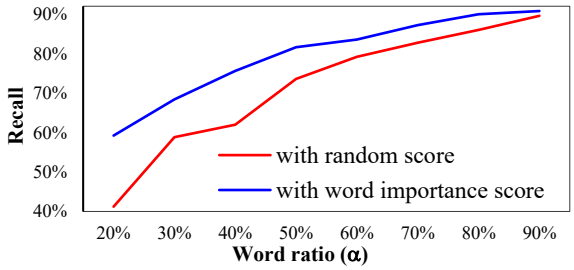
Figure 4: Prediction change under one-word transformation.

A Prediction Change under One-Word Transformation

The example in Figure 1 shows that the prediction of an adversarial text is more susceptible than that of an original text. We verify this observation on other AG News texts and IMDB movie reviews. In particular, we inspect the whole testing set containing 500 balanced samples of original and adversarial text generated by *PWWS* targeting CNN. For each sample, we transform a word by synonyms and measure the rate of prediction change from the target CNN. The maximum rate among words is represented for the sample and is plotted in histogram graphs (Figure 4). The maximum rate of an original text is often lower than that of an adversarial text in both AG News and IMDB.



(a) F1.



(b) Recall.

Figure 5: The impact of word importance score in VoteTRANS via changing word ratio α.

Attack	WDR		VoteTRANS	
	F1	Recall	F1	Recall
TextFooler	95.4	95.2	**96.5**	**98.0**
IGA	94.1	95.2	**95.7**	**97.2**
BAE	88.7	82.0	**96.7**	**99.6**

Table 5: WDR training on other word-based attacks.

B Word Importance Score

We change word ratio α with 10% step to show the impact of word importance score in VoteTRANS (line 2-6 in Algorithm 1). The experiment is conducted with adversarial text generated by *PWWS* targeting a CNN model on IMDB. To eliminate the impact of support models, we evaluate $\text{VoteTRANS}_{\text{same}}$ without support as shown in Figure 5. We also compare between $\text{VoteTRANS}_{\text{same}}$ with word importance score and that with random score. $\text{VoteTRANS}_{\text{same}}$ with word importance score achieves better than that with random score across all α, especially with small α. It demonstrates the impact of word importance score in VoteTRANS.

C WDR Training on Other Word-Based Attacks

We train WDR on other main word-based attacks including *TextFooler*, *IGA*, and *BAE* to detect adversarial text generated by *PWWS* as shown in Table 5.

Method	F1	Recall
$\text{VoteTRANS}_{\text{diff}}$ (*Checklist*)	41.4	26.4
$\text{VoteTRANS}_{\text{diff}}$ (*PSO*)	88.1	81.6
$\text{VoteTRANS}_{\text{diff}}$ (*A2T*)	88.2	82.0
$\text{VoteTRANS}_{\text{diff}}$ (*Faster-Alzantot*)	90.0	84.8
$\text{VoteTRANS}_{\text{diff}}$ (*IGA*)	90.1	86.0
$\text{VoteTRANS}_{\text{diff}}$ (*Kuleshov*)	90.8	86.4
$\text{VoteTRANS}_{\text{diff}}$ (*TextFooler*)	91.1	88.0
$\text{VoteTRANS}_{\text{diff}}$ (*Input-Reduction*)	93.8	97.6
$\text{VoteTRANS}_{\text{diff}}$ (*Pruthi*)	94.3	96.8
$\text{VoteTRANS}_{\text{diff}}$ (*TextBugger*)	94.6	97.2

Table 6: Other auxiliary attacks for $\text{VoteTRANS}_{\text{diff}}$ without support.

The adversarial text in both testing and training data targets the CNN model on AG News. Since VoteTRANS performs without training, we use the corresponding attack as an auxiliary. WDR is more compatible with *TextFooler* and *IGA* than *BAE*. In contrast, VoteTRANS achieves similar performance across three attacks.

D $\text{VoteTRANS}_{\text{diff}}$ without Support

Besides *BAE* and *DeepWordBug* as mentioned in Table 2, we show other auxiliary attacks in Table 6. $\text{VoteTRANS}_{\text{diff}}$ efficiently detects adversarial text except with *Checklist*, which generates independent adversarial text with any model and does not focus on the performance.

E $\text{VoteTRANS}_{\text{same}}$ with Other Supports

Besides LSTM and RoBERTa as mentioned in Table 2, we show all other available support models for AG News from TextAttack and their combination in Table 7. VoteTRANS achieves performances of at least 97.5 with individual supports and their combination. Similar to the results in Table 2, multiple supports reach more stable results than an individual support.

F Detecting Character-Based Attacks

We conduct experiments to detect adversarial text from all character-based attacks from TextAttack compatible with the CNN model as shown in Table 8. VoteTRANS achieves similar performances for AG News and IMDB. It demonstrates the resilience of VoteTRANS in detecting character-based attacks from different extents and tasks (medium text with multiclass classification as in

Method	F1	Recall
VoteTRANS$_{\text{same}}$ (*PWWS*) with BERT as support	97.5	100.0
VoteTRANS$_{\text{same}}$ (*PWWS*) with ALBERT as support	97.6	99.6
VoteTRANS$_{\text{same}}$ (*PWWS*) with DistilBERT as support	97.5	99.6
VoteTRANS$_{\text{same}}$ (*PWWS*) with BERT+ALBERT as support	98.4	98.4
VoteTRANS$_{\text{same}}$ (*PWWS*) with BERT+DistilBERT as support	98.2	98.8
VoteTRANS$_{\text{same}}$ (*PWWS*) with ALBERT+DistilBERT as support	98.2	98.8
VoteTRANS$_{\text{same}}$ (*PWWS*) with BERT+ALBERT+DistilBERT as support	98.4	99.2

Table 7: Other available support models from TextAttack for VoteTRANS$_{\text{same}}$.

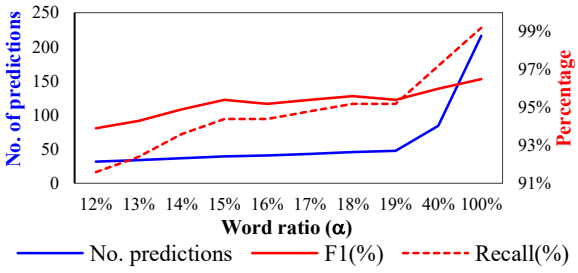
Dataset	Method	F1	Recall
AG News	VoteTRANS$_{\text{same}}$ (*DeepWordBug*) without support	94.3	89.2
	VoteTRANS$_{\text{same}}$ (*Pruthi*) without support	66.4	76.8
	VoteTRANS$_{\text{same}}$ (*TextBugger*) without support	92.7	93.6
	VoteTRANS$_{\text{same}}$ (*DeepWordBug*) with RoBERTa as support	95.0	98.8
	VoteTRANS$_{\text{same}}$ (*Pruthi*) with RoBERTa as support	80.3	98.0
	VoteTRANS$_{\text{same}}$ (*TextBugger*) with RoBERTa as support	96.1	98.8
IMDB	VoteTRANS$_{\text{same}}$ (*DeepWordBug*) without support	90.5	93.2
	VoteTRANS$_{\text{same}}$ (*Pruthi*) without support	77.8	90.8
	VoteTRANS$_{\text{same}}$ (*TextBugger*) without support	92.1	95.2
	VoteTRANS$_{\text{same}}$ (*DeepWordBug*) with RoBERTa as support	93.3	99.6
	VoteTRANS$_{\text{same}}$ (*Pruthi*) with RoBERTa as support	82.0	98.4
	VoteTRANS$_{\text{same}}$ (*TextBugger*) with RoBERTa as support	97.6	99.6

Table 8: Detecting all character-based attacks compatible with CNN model from TextAttack.

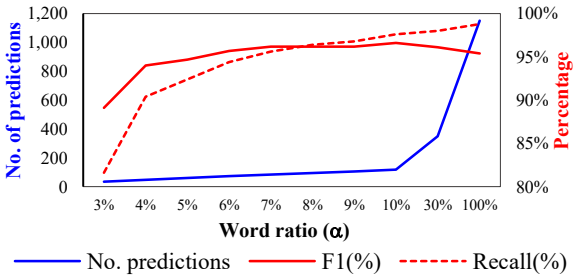
AG News and long text with binary classification as in IMDB).

G VoteTRANS Complexity

Let N, M, and K be the number of words, the number of transformations for each word, and the number of models used in Algorithm 1. The worst-case of VoteTRANS complexity is $\mathcal{O}(N \times M \times K)$, approximately with the number of predictions on the K models. For example, if VoteTRANS processes AG News using *PWWS*, CNN, and RoBERTa as auxiliary, target, and support; in this case, N, M, and K are 42.6, 10.7, and 2, respectively. N of IMDB is increased to 241.9 while other values are unchanged. Theoretically, the number of predictions is 910.6 (AG News) and 5165.6 (IMDB). However, this number is remarkably reduced by the constraint checking (line 12) and early stopping (line 21) in Algorithm 1. As a result, it is reduced to 216.3 (76.2%) and 1151.3 (77.7%) predictions as shown in Figures 6a and 6b, respectively. VoteTRANS can also adjust the number of predictions suitable for the resource capacity by using small α. For example, α at 12% and 5% needs 32.0 and 61.1 predictions while keeping higher per-



(a) AG News.

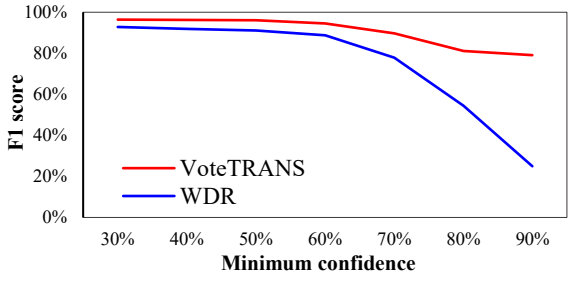


(b) IMDB.

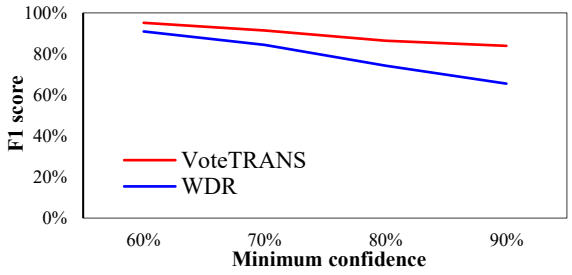
Figure 6: Number of predictions.

formance on the existing works as mentioned in Section 4.3.

H Detection with High Confidence Text



(a) AG News.



(b) IMDB.

Figure 7: Detection of adversarial text with high confidence

We evaluate WDR and VoteTRANS on detecting high confidence of adversarial text, which is generated by *PWWS* targeting CNN models on AG News and IMDB. PWWS attacks the CNN model until overcoming minimum confidence. Since any confidence of AG News (4 classes) and IMDB (2 classes) is greater than 25% and 50%, respectively, we set minimum confidences starting at 30% and 60% with 10% step. While the minimum confidences at 80% and 90% on AG News have 71 and 19 adversarial texts, respectively, other confidences have sufficient 500 balanced samples.

While WDR and VoteTRANS achieve similar F1 on AG News until minimum confidence at 60%, WDR suddenly drops down to 25.0% at confidence 90%. In contrast, VoteTRANS still keeps 79.2% at this confidence as shown in Figure 7a. For IMDB, the margins between WDR and VoteTRANS gradually increase from 4.2% to 18.4%. It demonstrates the resilience of VoteTRANS in detecting adversarial text with high confidence.

I Error Analysis

We analyze the errors of $\text{VoteTRANS}_{\text{same}}$ for the short text (MR). Here we especially focus on the results when DistilBERT and RoBERTa are the target and support models and the adversarial text is generated with *PWWS*. MR is harder to detect than long text as shown in Table 1. 40.7% of all the errors that $\text{VoteTRANS}_{\text{same}}$ fails to detect are caused by susceptible original text, which is easily attacked. For example, although the original text "*your children will be occupied for 72 minute*" is correctly predicted as negative by DistilBERT, 29 out of 39 perturbations with one-word replacements change the prediction into positive. It is opposite to our hypothesis as mentioned in line 38 in Section 1 and thus bypasses VoteTRANS. Its adversarial text "*your child will be occupied for 72 minutes*" also bypasses our detector (1 out of 42 perturbations change the prediction). However, such text is a little harmful because it has unclear sentiment and is unpopular (4.4% and 1.2% of MR and IMDB testing text, respectively).

ACL 2023 Responsible NLP Checklist

A For every submission:

☐ A1. Did you describe the limitations of your work?
Left blank.

☐ A2. Did you discuss any potential risks of your work?
Left blank.

☐ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☐ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☐ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Left blank.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C ☐ Did you run computational experiments?

Left blank.

☐ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☐ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Left blank.

☐ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Left blank.

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Left blank.

D ☐ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Left blank.