Don't Retrain, Just Rewrite: Countering Adversarial Perturbations by Rewriting Text

Ashim Gupta[1]*, **Carter Wood Blum**[2], **Temma Choji**[2], **Yingjie Fei**[2], **Shalin Shah**[2], **Alakananda Vempala**[2], **Vivek Srikumar**[1]
[1]University of Utah, [2]Bloomberg,
{ashim, svivek}@cs.utah.edu,
{szhang611, cblum18, yfei29, sshah804, tchoji, avempala}@bloomberg.net

Abstract

Can language models transform inputs to protect text classifiers against adversarial attacks? In this work, we present ATINTER, a model that intercepts and learns to rewrite adversarial inputs to make them non-adversarial for a downstream text classifier. Our experiments on four datasets and five attack mechanisms reveal that ATINTER is effective at providing better adversarial robustness than existing defense approaches, without compromising task accuracy. For example, on sentiment classification using the SST-2 dataset, our method improves the adversarial accuracy over the best existing defense approach by more than 4% with a smaller decrease in task accuracy (0.5 % vs. 2.5%). Moreover, we show that ATINTER generalizes across multiple downstream tasks and classifiers without having to explicitly retrain it for those settings. For example, we find that when ATINTER is trained to remove adversarial perturbations for the sentiment classification task on the SST-2 dataset, it even transfers to a semantically different task of news classification (on AGNews) and improves the adversarial robustness by more than 10%.

1 Introduction

Neural models in NLP have been shown to be vulnerable to adversarial attacks both during training time (Gu et al., 2017; Wallace et al., 2019, 2021; Chen et al., 2021) and at deployment (Ebrahimi et al., 2018; Jin et al., 2020; Garg and Ramakrishnan, 2020a). The attacks of the latter type aim to craft adversarial inputs by introducing small, imperceptible perturbations in the input text that erroneously change the output label of a classification model. Defending against such attacks is important because it ensures the integrity and reliability of NLP systems. If undefended, for example, an attacker could adversarially manipulate a spam email to evade a spam detector.

An ideal defense mechanism against such adversarial attacks should maintain good task performance on non-adversarial inputs, effectively mitigate adversarial attacks, and be transferable to other models and datasets. The transferability of defenses is a valuable property because it allows easy application to new and unknown models without retraining the underlying classification model. This is particularly useful when complete access to the model is not possible; for example when the model is accessed through an API. Most existing methods do not satisfy these desiderata, typically lacking in one or more desired properties. For example, the methods that use input randomization like SAFER (Ye et al., 2020) and Sample Shielding (Rusert and Srinivasan, 2022) significantly degrade task accuracies due to the smoothing and aggregation involved, and are thus ineffective defenses. Another set of methods— e.g., adversarial training (Jia and Liang, 2017; Ebrahimi et al., 2018) and SHIELD (Le et al., 2022)—require model retraining; while serving as effective defenses, they cannot be transferred to other models and datasets without retraining the classifier.

In this work, we present a novel strategy for defending against adversarial attacks that satisfies the aforementioned desiderata. Our method—**A**dversarial **T**ext **Inter**ceptor and **R**ewriter (ATINTER)—is based on the intuition that automatically generated adversarial inputs can be undone by *learning* to manipulate the textual inputs instead of retraining the classification model. Specifically, we employ an encoder-decoder module that intercepts and rewrites the adversarial input to remove any adversarial perturbations before feeding it to the classifier[1]. Our method differs from existing input randomization approaches in that it does not rely on random word replacements

*Work done during internship at Bloomberg

[1]By *intercepting*, we simply mean that ATINTER stops the input from directly going to the classifier and processes it first to remove the adversarial perturbations.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 13981–13998
July 9-14, 2023

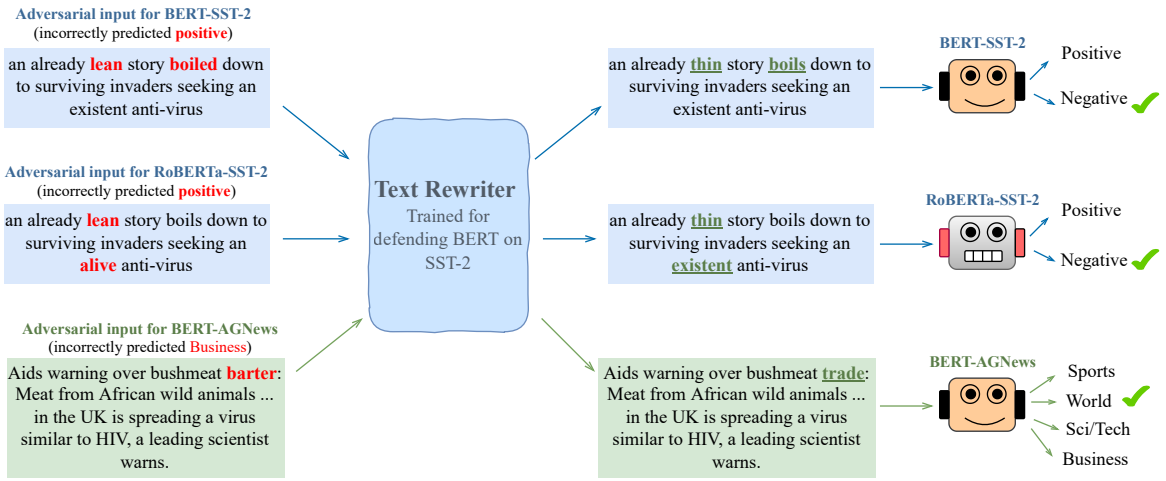

Figure 1: Modular application of ATINTER, which is trained for defending a BERT classifier for the SST-2 dataset. Demonstrating transferability across models, ATINTER successfully defends a RoBERTa classifier on SST-2 without retraining. Similarly, ATINTER is successful in defending a BERT model for a news classification task on AGNews. Refer to section 5 for a more detailed discussion.

or deletions to counteract adversarial changes. Instead, we employ a separate model that is explicitly trained to remove adversarial perturbations. One benefit of this strategy is that it dissociates the responsibility of ensuring adversarial robustness from the classification model and delegates it to an external module, the text rewriter. Consequently, the rewriter module serves as a pluggable component, enabling it to defend models that it was not explicitly trained to protect. Figure 1 demonstrates this scenario.

We demonstrate the effectiveness of our approach using a T5 model (Raffel et al., 2020) as the general-purpose text rewriter, but our method is applicable to any transformer-based text generator/rewriter. Through extensive experimentation and comparison with existing methods, we show that ATINTER effectively removes adversarial perturbations, and consistently outperforms other defense approaches on several datasets for text classification. When used as a pluggable component, ATINTER exhibits good transferability to new models and datasets without the need for retraining (examples shown in fig. 1). Specifically, we find that this T5-based rewriter trained to remove adversarial perturbations for the sentiment classification task on the SST-2 dataset, also removes adversarial perturbations for a news classification model (on AGNews), increasing adversarial robustness by over 10%.

In summary, our contributions are:

1. We propose a novel defense mechanism against adversarial attacks, called ATINTER that uses a text rewriter module, along with a simple strategy to train this module[2].

2. We demonstrate its effectiveness on four benchmark datasets and five adversarial attacks. Compared with competitive baselines, our method substantially improves the adversarial robustness with a much smaller decrease in accuracy on non-adversarial inputs.

3. We show that ATINTER can be used as a pluggable module without retraining and that its ability to defend models is transferable to new models (e.g., BERT → RoBERTa) as well as new datasets (BERT on SST-2 → BERT on AGNews).

2 Related Work

Adversarial Attacks Most adversarial attacks use heuristic-based search methods to substitute vulnerable parts of the input with carefully chosen adversarial text (Ebrahimi et al., 2018; Jin et al., 2020; Jia and Liang, 2017). These substitutions can be performed at the character-level (Gao et al., 2018; Belinkov and Bisk, 2018), word-level (Ren et al., 2019; Jin et al., 2020; Garg and Ramakrishnan, 2020b; Li et al., 2020; Zhang et al., 2021), or both (Li et al., 2018). Finally, while adversarial attacks show that NLP models are over-sensitive to small perturbations, NLP models have also been

[2]Code will be available at https://github.com/bloomberg

shown to be under-sensitive to certain perturbations like input reduction, etc. (Feng et al., 2018; Gupta et al., 2021). We refer the reader to the detailed recent survey of Wang et al. (2022b).

Defenses against Adversarial Attacks The typical strategies employed for defending text classification systems against adversarial attacks involve either retraining the classifiers using adversarial examples or incorporating randomized smoothing at the input stage to make robust predictions. The defenses of the former type involve *adversarial training* (Goodfellow et al., 2015; Alzantot et al., 2018), *certified training* (Jia et al., 2019; Zhou et al., 2021; Huang et al., 2019), and other specialized training schemes (Le et al., 2022; Jiang et al., 2022). While adversarial training lacks in effectiveness (Alzantot et al., 2018), the certification based methods are only applicable for a specific set of perturbations; e.g., Jia et al. (2019) restrict word substitutions to belong to a counter-fitted embedding space (Mrkšić et al., 2016). More recently, Le et al. (2022) proposed SHIELD, that trains a stochastic ensemble of experts by only *patching* the last layer of the BERT classifier. We use SHIELD, and adversarial training for comparison with our proposed method.

On the other end of the spectrum are the models that do not retrain the classifier and instead use randomized smoothing techniques to enhance the robustness of the models (Cohen et al., 2019; Zhou et al., 2019; Ye et al., 2020; Rusert and Srinivasan, 2022; Wang et al., 2022a). Ye et al. (2020) introduced a defense called SAFER, that significantly improves certified robustness by performing randomized substitutions using a synonym network. Rusert and Srinivasan (2022) proposed another randomization defense called Sample Shielding, which relies on making an ensemble of predictions on different random samples of the input text. One major drawback of utilizing randomization-based techniques is that they may result in a significant decrease in task accuracies on non-adversarial inputs. To overcome this limitation, Bao et al. (2021) proposed ADFAR, which trains an anomaly detector for identifying adversarial examples and performs frequency-aware randomization only for the adversarial inputs. The authors observe that this scheme preserves adversarial robustness without sacrificing the task performance. We adopt ADFAR, Sample Shielding, and SAFER as the other set of baselines for our work.

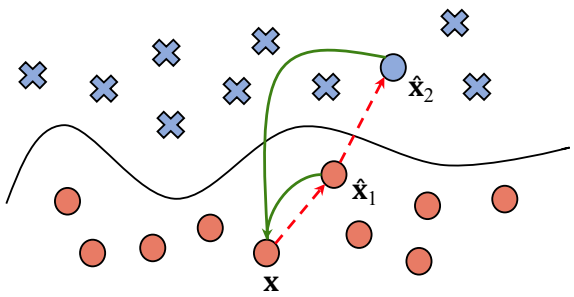

Figure 2: Geometry of an adversarial attack. An attacker moves a point $\mathbf{x}$ (red circle) to the other side of the decision boundary along the red dotted arrows. $\hat{\mathbf{x}}_1$ represents the example after one change, $\hat{\mathbf{x}}_2$ denotes the final adversarial example. Our model ATINTER aims to restore these points to their original position (represented by the green curved arrows).

3 Learning to Remove Adversarial Perturbations

As mentioned in section 2, most existing methods that operate on the textual input rely on randomization to remove any adversarial perturbations. We present a model that *learns* to remove these perturbations by rewriting the text. Although similar to paraphrasing in terms of the task interface, our goal is different; we focus on removing the perturbation instead of just preserving the meaning. The rest of this section formalizes this intuition.

3.1 Notation

Given a sequence of tokens $\mathbf{x}$, suppose we have a trained classifier with parameters $\boldsymbol{\theta}$, which maps $\mathbf{x}$ to the output label y from the label space $\mathcal{Y}$ as

$$y = \arg\max_{y_i \in \mathcal{Y}} P_{\boldsymbol{\theta}}(y_i|\mathbf{x})$$

When the classifier makes a correct prediction, $y = y^*$, the true label for that input. For a successful adversarial attack, the adversary takes the input sequence $\mathbf{x}$ and produces a perturbed variant $\hat{\mathbf{x}}$ by making a small change to $\mathbf{x}$ such that the prediction made by the model is incorrect:

$$\arg\max_{y_i \in \mathcal{Y}} P_{\boldsymbol{\theta}}(y_i|\hat{\mathbf{x}}) \neq y^*$$

Additionally, the adversary ensures that the perturbation is *meaningful* and *imperceptible*.[3]

Typically, this perturbation is achieved via an iterative process, during which the adversary

[3]Most attacks ensure this by enforcing constraints on the part-of-speech tags of the replaced words as well as maintaining the fluency through an LM perplexity score.

makes incrementally small modifications to the input (Ebrahimi et al., 2018; Jin et al., 2020). Assume that the adversary makes k successive changes to the input taking the original input $\mathbf{x}$ to the final adversarial variant $\hat{\mathbf{x}}_k$, represented as follows

$$\mathbf{x} \rightarrow \hat{\mathbf{x}}_1 \rightarrow \hat{\mathbf{x}}_2 \rightarrow ... \rightarrow \hat{\mathbf{x}}_k$$

To construct an adversarial input, the adversary selects perturbations such that the probability of the true label decreases after each successive modification until the required incorrect prediction is achieved. In other words, the adversary wants

$$P_{\boldsymbol{\theta}}(y^*|\mathbf{x}) > P_{\boldsymbol{\theta}}(y^*|\hat{\mathbf{x}}_1) > \cdots > P_{\boldsymbol{\theta}}(y^*|\hat{\mathbf{x}}_k) \quad (1)$$

In summary, given an input sequence $\mathbf{x}$, an adversary keeps introducing small changes ($\mathbf{x} \rightarrow \hat{\mathbf{x}}_1$, etc.), each of which reduces the predicted probability of the correct label y^*. Figure 2 shows a visual representation of the adversarial process, which moves the example marked $\mathbf{x}$ to the wrong side of the decision boundary by following the red arrows.

3.2 Training the Text Rewriter

As we saw in section 2, the most common strategy to counter such adversarial attacks is to retrain the model with new parameters $\boldsymbol{\theta}_{new}$ such that

$$\arg\max_{y_i \in \mathcal{Y}} P_{\boldsymbol{\theta}_{new}}(y_i|\hat{\mathbf{x}}_k) = y^*$$

In this work, our objective is to keep the model parameters $\boldsymbol{\theta}$ unchanged and instead manipulate the input $\mathbf{x}$ by rewriting it. To this end, we define ATINTER that intercepts and rewrites potentially adversarial inputs. ATINTER is a text-to-text transducer, which we will denote by $\mathcal{T}_{\boldsymbol{\phi}}$ with its own trainable parameters $\boldsymbol{\phi}$.

To effectively counteract adversarial inputs, the transformation function $\mathcal{T}_{\boldsymbol{\phi}}$, must transform the input $\mathbf{x}$ such that the task classifier makes the correct prediction on the transformed text:

$$\arg\max_{y_i \in \mathcal{Y}} P_{\boldsymbol{\theta}}(y_i|\mathcal{T}_{\boldsymbol{\phi}}(\hat{\mathbf{x}}_k)) = y^*$$

We can guarantee this outcome by simply training $\mathcal{T}_{\boldsymbol{\phi}}$ to ensure that $\mathcal{T}_{\boldsymbol{\phi}}(\hat{\mathbf{x}}_k) = \mathbf{x}$. In other words, our goal is to learn a transformation function, $\mathcal{T}_{\boldsymbol{\phi}}$, that is capable of *undoing* or *reversing* the impact of adversarial modifications. However, merely training the rewriter to reverse the final step $\hat{\mathbf{x}}_k$ may not be sufficient because $\hat{\mathbf{x}}_k$ is produced through a series of small changes. Therefore, in addition to undoing $\hat{\mathbf{x}}_k$, the rewriter should also be able to reverse each intermediate step. This strategy is based on the intuition that each successive change made to the input $\mathbf{x}$ in constructing an adversarial input $\hat{\mathbf{x}}_k$ is in itself adversarial; all intermediate changes decrease the probability of the true label and are thus undesirable (eq. (1)). Green curved arrows in fig. 2 show the task of the rewriter. Figure 3 in the appendix shows an example of this process.

In summary, any adversarial modification made to the input at any stage should be reversed by ATINTER , i.e.,

$$\mathcal{T}_{\boldsymbol{\phi}}(\hat{\mathbf{x}}_i) = \mathbf{x}, \forall i \in \{1, k\} \quad (2)$$

Finally, on non-adversarial inputs, we do not need to make any changes, and the function $\mathcal{T}_{\boldsymbol{\phi}}$ should therefore act as an identity function on these inputs:

$$\mathcal{T}_{\boldsymbol{\phi}}(\mathbf{x}) = \mathbf{x} \quad (3)$$

Training Details We use the T5 model (Raffel et al., 2020) as the starting point for our text rewriter ATINTER. Since we need adversarial examples to train our rewriter, we follow Bao et al. (2021) and choose TextFooler (Jin et al., 2020) for generating these examples on the whole training set. The training data for ATINTER consists of input-output pairs of the form $(\hat{\mathbf{x}}_i, \mathbf{x})$: as described in eq. (2), for every adversarial modification $\hat{\mathbf{x}}_i$, including the ones with intermediate changes, and the original, unperturbed sequence $\mathbf{x}$ is the desired output. In addition, as per eq. (3), the training data also includes unperturbed examples of the form $(\mathbf{x}, \mathbf{x})$. Figure 4 shows an illustrative example.

We train the `base` variant of the T5 model for 5 epochs with the starting learning rate of 5×10^{-5}. More details on the hyperparameters are provided in the appendix appendix A.2. We use the Transformers (Wolf et al., 2020) for our implementation.

4 Experimental Setup

In this section, we will detail the datasets we use for our experiments, the baseline defense mechanisms, and the adversarial attacks they will be pitted against, and the three metrics we will use to compare the defense methods.

4.1 Datasets

We evaluate our proposed defense on four text classification datasets.

Stanford Sentiment Treebank (SST-2) The SST-2 dataset is used for sentiment classification (Socher

Dataset	# Avg. words	# Labels	Size
SST-2	9.4	2	68K
MR	21.6	2	11K
AGNews	44.1	4	127K
MNLI	33.9	3	433K

Table 1: Summary of the datasets used in this paper. The three columns show the average number of words in the input, the number of labels, and the total size of the dataset. For MNLI, the input size is calculated by concatenating premise and hypothesis.

et al., 2013) among two labels: *positive*, and *negative*. We use the splits from the GLUE benchmark (Wang et al., 2019); we use the validation set for reporting our results since the test set is not available publicly.

Rotten Tomatoes Movie Reviews (MR, Pang and Lee, 2005) Similar to SST-2 task, the goal is to predict a movie review's sentiment (*positive* vs. *negative*). We use the official test set for evaluation.

AG News (Zhang et al., 2015) This is a news classification dataset with four possible labels (*sports*, *world*, *science/technology*, *business*). The test set contains 7600 examples and since it can take a long time for robustness evaluation across all seven models and the five attackers, we randomly choose 1000 examples for our evaluation set.

Multi-Genre Natural Language Inference (MNLI, Williams et al., 2018) This is a standard dataset for Natural Language Inference (NLI) where the goal is to determine the inferential relation between a premise and a hypothesis. The dataset requires sentence-pair classification among three labels (*entailment*, *neutral*, and *contradiction*). Again, we sample 1000 instances from the `validation-matched` subset for evaluation.

4.2 Baselines and Adversarial attacks

Baselines We compare our model with a number of baselines: Adversarial Training (AT, Alzantot et al., 2018), SHIELD (Le et al., 2022), SAFER (Ye et al., 2020), SampleShielder (Rusert and Srinivasan, 2022), and ADFAR (Bao et al., 2021). SAFER and SampleShielder are input randomization methods, while AT, and SHIELD require model retraining. ADFAR requires retraining the model and also uses input randomization. We could not compare the results with DISP (Zhou et al., 2019) as we were not able to run their implementation. We have provided more details in the appendix appendix A.

Adversarial Attacks We use the open source toolkit TextAttack (Morris et al., 2020a,b) to evaluate all models on five black-box adversarial attacks. TextFooler (Jin et al., 2020), PWWS (Ren et al., 2019), and BAE (Garg and Ramakrishnan, 2020b) attack at the word-level, DeepWordBug (DWB, Gao et al., 2018) attacks at the character-level, and TextBugger (Li et al., 2018) attacks at both word and character-level. TextFooler and PWWS use counter-fitted word embeddings (Mrkšić et al., 2016), while BAE uses the BERT as a masked language model (Devlin et al., 2019) to find the best word replacements. We provide an example of each of them in table 6 in the appendix.

We perform our main experiments with a BERT-base classifier as the victim model with hyperparameters as suggested by Devlin et al. (2019).

4.3 Evaluation

Evaluation Metrics We measure the quality of the defense methods using three metrics, namely Clean Accuracy (Clean Acc.), Adversarial Accuracy (AA), and Average number of queries (#Q).

Clean Accuracy is the accuracy of the model on clean non-adversarial inputs, measured on the original validation or test sets. A model that retains the clean accuracy of the original model is desirable.

Adversarial Accuracy (AA) The Attack Success Rate (ASR) of an attack is the percentage of instances where the attack algorithm successfully constructs an adversarial example. A defense method that makes a model more robust results in a lower ASR. We report the Adversarial Accuracy of the defense methods, defined as $100 - \text{ASR}$.

Average Number of Queries (#Q) is the measure of the cost for an attacker, and is the average number of forward passes (queries) to the model by the attacker. On average, a more robust defense method requires more queries.

Evaluation Protocol The adversarial accuracy depends on the number of queries an attacker is allowed to perform - a lower query budget entails a higher AA. There is currently no established protocol for evaluating the adversarial robustness of text classification systems. In this study, we do not impose a restriction on the number of queries allowed to the attacker, resulting in the most challenging conditions for the defense methods.

5 Main Results

Table 2 shows the results for the defense methods on all four datasets. Additionally, table 9 in the appendix summarizes the results in terms of average improvements for the five adversarial attacks.

As observed from the table, our proposed method ATINTER provides a consistent and substantial improvement in terms of adversarial robustness over the baselines. We find that there is a trade-off between clean accuracy and adversarial robustness for all the models, aligning with the findings of Raghunathan et al. (2020). The results show that ATINTER maintains the highest level of clean accuracy on all datasets except MR, where SAFER improves it by more than 1%, but does so at the cost of making the model less robust.

The most formidable baseline is ADFAR, which employs an anomaly detector to identify adversarial inputs and uses input randomization for handling adversarial instances. Our method substantially outperforms ADFAR on all settings except one. Furthermore, we observe that SampleShielder performs well on AGNews but not on other datasets. This can be attributed to the fact that SampleShielder randomly removes parts of the input before making a prediction. This is effective for tasks with longer inputs and simpler semantics (such as topic classification on AGNews), but does not work for others where removing parts of the input can alter the label. Additionally, while SampleShielder provides the best adversarial accuracies on the MNLI dataset, the clean accuracy is almost close to random. Our proposed model ATINTER on the other hand, provides the best balance between adversarial and clean accuracies.

5.1 Results against other attack types

Several defense methods, including ours, utilize adversarial examples from one attack type to train their models. The true effectiveness of adversarial defenses is determined when they are tested against previously unseen adversarial attacks. Our evaluation using four other attacks, excluding TextFooler, accomplishes this. Each of these attacks differ from TextFooler in one or more aspects. For example, while TextFooler is a token-level attack, DeepWordBug (DWB) is a character-level attack. TextBugger, on the other hand, is a multi-level attack, capable of attacking at both token and character level. BAE replaces words uses a BERT MLM while TextFooler uses GloVe word embeddings. PWWS, in comparison, employs a different algorithm for token replacement.

From table 2, we observe that, as compared to the baselines, ATINTER provides significant improvements in robustness against other attacks. Notably, while ATINTER is only trained against synonym substitutions from TextFooler, it is able to generalize to other attacks that operate at the character level. Lastly, the improvement against BAE is less than that against other attacks. We hypothesize that this is due to the fact that BAE employs a BERT language model for word replacements, which is different from the technique used by TextFooler.

5.2 Transferability to other classifiers

As mentioned previously, one motivation for using a separate robustness module like ours is that it can be transferred to other text classification models without retraining the rewriter. We use ATINTER which was trained to remove adversarial perturbations for the BERT classifier on the SST-2 dataset and employ it, without retraining, to remove adversarial perturbations for other classifiers on the same dataset. We assess the transferability of ATINTER against three classifiers, namely: RoBERTa (Liu et al., 2019), DistilBERT (Sanh et al., 2019), and ALBERT (Lan et al., 2019).

The results for evaluation against TextFooler are presented in table 3. We observe that ATINTER is effective in enhancing adversarial robustness for models other than BERT. Importantly, this improvement is achieved without much drop in performance on the clean examples ($< 1\%$ in all cases). On average, ATINTER improves adversarial accuracy by 16.6% across the three models. Surprisingly, the improvement for the RoBERTa model is even more pronounced than that for the BERT model. We hypothesize that this transferability from ATINTER is due to two factors. First, adversarial attacks often result in similar adversarial changes, particularly for the same dataset. Second, previous research has demonstrated that adversarial examples transfer across classifiers for the same task (Papernot et al., 2016; Liu et al., 2017).

5.3 Transferability to other tasks and datasets

As explained in the previous section, ATINTER allows for its application to tasks and datasets for which it was not trained. We now assess the transferability of our method with respect to other tasks and datasets. We use the ATINTER trained

Dataset	Defense	Clean Acc.	TextFooler		TextBugger		BAE		PWWS		DWB	
			AA	#Q	AA	#Q	AA	#Q	AA	#Q	AA	#Q
SST-2	None	92.4	4.8	95.4	31.3	49.3	33.9	60.4	13.4	143.1	18.6	34.7
	AT	88.4	5.7	91.6	23.1	46.3	34.6	61.8	13.2	139.4	10.1	32.3
	SHIELD	88.8	6.6	90.9	25.1	51.4	28.5	61.3	13.6	137.1	9.7	33.2
	SAFER	89.3	8.7	91.9	27.7	48.4	36.3	62.2	16.2	138.8	16.5	32.4
	SampleShielder	76.8	6.6	97.1	25.7	**58.4**	28.8	66.2	17.7	143.8	17.5	36.2
	ADFAR	89.9	19.5	115.4	29.3	58.1	**37.1**	**68.7**	20.9	142.7	22.8	36.1
	ATINTER	**92.0**	**24.0**	**136.7**	**40.5**	54.3	34.2	60.4	**22.9**	**150.1**	**25.3**	**38.0**
MR	None	84.2	10.7	117.7	37.3	56.1	38.4	64.4	18.7	150.0	22.3	40.5
	AT	84.2	11.3	118.6	34.3	54.8	35.9	65.8	19.2	151.1	18.1	38.2
	SHIELD	82.1	12.1	98.7	22.3	60.8	27.4	65.6	18.2	141.7	18.7	37.2
	SAFER	85.5	3.7	88.1	23.4	49.3	33.4	59.8	10.6	142.0	16.0	34.0
	SampleShielder	76.2	12.1	105.5	26.5	58.2	27.3	61.7	21.4	150.7	24.3	39.7
	ADFAR	82.4	17.5	120.5	26.0	59.6	31.4	65.5	23.0	148.8	22.6	38.2
	ATINTER	**84.3**	**21.1**	**140.2**	**45.7**	**61.0**	**38.6**	**65.8**	**26.4**	**154.2**	**32.5**	**43.6**
MNLI	None	83.5	1.1	81.3	4.2	54.1	19.3	59.2	2.4	188.3	4.2	41.5
	AT	80.8	2.7	105.0	6.3	59.3	20.7	62.5	3.5	190.4	6.9	41.7
	SHIELD	79.5	2.9	103.8	6.7	60.2	20.9	63.1	3.4	191.1	7.9	44.0
	SAFER	78.0	1.7	101.3	10.3	58.8	24.5	62.9	5.3	196.7	8.3	44.1
	SampleShielder	41.4	**17.5**	**178.3**	**17.2**	**102.2**	**41.7**	**100.1**	**26.1**	**231.2**	**19.9**	**57.0**
	ADFAR	78.1	10.5	117.8	7.6	64.3	16.3	61.6	11.0	200.7	9.4	44.9
	ATINTER	**83.0**	16.1	158.2	9.7	67.3	20.4	61.5	10.9	195.2	9.5	45.4
AGNews	None	94.9	18.2	334.1	47.7	180.9	84.8	116.8	43.2	353.0	38.9	110.1
	AT	94.1	19.1	379.2	49.1	189.7	83.4	117.1	44.1	355.6	39.7	114.2
	SHIELD	92.4	20.1	385.3	51.7	190.9	81.8	114.4	44.9	359.4	39.7	112.4
	SAFER	91.2	15.7	280.6	33.6	156.6	78.8	119.9	45.8	361.2	40.8	114.7
	SampleShielder	90.8	52.6	425.6	56.7	216.9	84.4	119.5	49.8	365.4	41.6	115.4
	ADFAR	92.4	58.3	422.2	52.5	**245.1**	79.7	**136.3**	45.9	368.4	47.1	115.8
	ATINTER	**94.7**	**73.0**	**520.0**	**63.9**	222.9	**87.3**	123.5	**63.9**	**375.2**	**49.7**	**117.3**

Table 2: Results comparing model robustness using the clean accuracy (%) and adversarial accuracy (%) on the five adversarial attacks: None indicates the BERT model without any defense and therefore acts as a baseline model. Notably, our model ATINTER yields superior results across the board without significant drop in clean accuracy.

	Clean Acc	Adv. Acc.
BERT	92.4	4.8
+ ATINTER	92.0	24.0
RoBERTa	94.1	5.0
+ ATINTER	93.7	25.1
DistilBERT	90.0	2.9
+ ATINTER	89.5	17.8
ALBERT	91.1	4.2
+ ATINTER	90.4	19.0

Table 3: Transferability to other models.The ATINTER is trained for defending the BERT model and is evaluated for its ability to defend other models without retraining. The results are shown for the SST-2 dataset for its validation subset.

	Clean Acc	Adv. Acc.
BERT-SST2	92.4	4.8
+ ATINTER	92.0	24.0
MR	84.2	10.7
+ ATINTER	84.2	29.3
AGNews	94.2	18.2
+ ATINTER	93.1	30.8
MNLI	83.5	1.1
+ ATINTER	83.2	2.8

Table 4: Results comparing transferability of ATINTER to other tasks. The ATINTER trained for the BERT model on SST-2 dataset is evaluated for BERT classifiers on other datasets without retraining.

Model	Params.	Clean Acc	Avg. AA
t5-small	60M	92.4	21.9
t5-base	220M	92.0	29.4
t5-large	770M	92.4	37.8
t5-3b	3B	92.1	45.9

Table 5: Results comparing the robustness w.r.t the size of ATINTER model. All the models are evaluated on SST-2 with the BERT-base classifier. We report here the averaged adversarial accuracies for compactness. Please refer to the appendix table 7 for detailed numbers.

for sentiment classification on SST-2 using BERT and apply for the BERT model trained on other datasets. We perform this evaluation on three datasets, namely MR, AGNews, and MNLI.

We present the results in the table 4. We find that our model ATINTER exhibits strong transferablity for other datasets. Again, as with previous results, we see only small drops in performance on non-adversarial inputs. The favorable results on the MR dataset shows that ATINTER effectively transfers for a different dataset of the same task. Note that the improvement in adversarial accuracy for MR is even higher than a model that is specifically trained for removing adversarial perturbations for the MR dataset (see table 2). This is explained by the fact that the MR dataset is much smaller and thus the ATINTER trained on that dataset has fewer adversarial instances to learn from (10k vs. 67k). We notice more than 12% increase in adversarial accuracy on the AGNews dataset. This is perhaps most surprising, since not only the task is semantically different with different set of classes, but the domain of the dataset is also different (movies vs. news). On the MNLI dataset though, we notice only small improvement, perhaps because it is a semantically harder task.

In summary, our proposed model ATINTER transfers across both models and datasets. This observation can motivate the training of a single rewriter module for all tasks and datasets. The benefits of such an approach are two-fold. First, since the defense capability transfers across models, a single shared model could be more robust than the individual ones. Second, having a single shared is more practical as it reduces the overhead in deployment of ATINTER. We leave the exploration of this shared rewriter approach to future work.

5.4 Effect of the model size

For all experiments in previous sections, we used the base variant of the T5 model for training ATINTER. We now investigate the effect of the size of the rewriter module on the adversarial robustness. For the SST-2 dataset, we train four variants of ATINTER with different sizes: t5-small, t5-base, t5-large, t5-3b. The results are shown in table 5. We observe that with increased size, the rewriter module defends the classification model more robustly.

5.5 Pre-training the Rewriter

One additional benefit of having a separate rewriter module is that we can pre-train the rewriter without using any task-specific datasets. We demonstrate this approach by artificially constructing a training corpus using the Wikipedia text. Specifically, we sample 100k sentences from the English Wikipedia and randomly substitute 15% of the words in each of those sentences with one of the neighbors from the GloVe embedding space (Pennington et al., 2014). The pre-training task for the rewriter is to simply *reverse* this perturbation by generating the original unperturbed sentence. Note that this setup is close to but does not perfectly simulate the actual adversarial attack scenario, as the perturbations used in the latter are chosen with greater precision. We observe that this pre-training improves the ATINTER by more than 2.5% in terms of adversarial accuracy without any significant decrease in clean accuracy. Due to space constraints, the results are shown in table 8 in the appendix .

5.6 Latency at Inference

One limitation of our proposed strategy is that it utilizes two neural models to make predictions, hurting the overall inference time. We measure latency for each of the models by averaging their inference time over 200 examples (100 clean + 100 adversarial). We observe that ATINTER is slower than model retraining approaches (22.0 ms for SHIELD vs. 95 ms for ATINTER), while being faster or competitive with input randomization methods. SAFER is the slowest of all since it performs averaging over a large number of candidate synonyms.

One possible approach to reduce inference time could be to use more efficient text generation models like non-autoregressive text generation (Gu et al., 2018). Moreover, a method based on text-

editing can also be promising (Malmi et al., 2022). We leave these explorations to the future work.

6 Conclusion

In this paper, we explore a novel strategy to defend against textual adversarial attacks that does not require model retraining. Our proposed model, ATINTER intercepts and rewrites adversarial inputs to make them non-adversarial for a downstream text classifier. We perform experiments on four text classification datasets and test its effectiveness against five adversarial attacks. The results suggest that, in comparison with baselines, our proposed approach is not only more effective against adversarial attacks but is also better at preserving the task accuracies. Moreover, when used as a pluggable module, ATINTER shows great transferability to new models and datasets—on three new datasets, it improves adversarial accuracy by 10.9% on average. We expect the future work to focus on improving inference time latency by using more sophisticated text generation methods.

7 Limitations

This work is subject to two limitations. First, our experiments were restricted to text classification tasks and we did not evaluate if our methods can effectively defend against adversarial attacks for other tasks like QA, etc. (Jia and Liang, 2017). It therefore remains unexplored if our conclusions transfer beyond the text classification tasks.

Second, the primary contribution of our work, ATINTER relies on using a language model like T5, which is trained on large amount of text in English. It is possible that our approach is not as effective for languages where such a model is not freely available. Additionally, in this work, we did not explore the impact of large language model pretraining on our results.

8 Ethical Considerations

This work is concerned with protecting or defending against adversarial attacks on text classification systems. For modeling, our method ATINTER uses another neural network based language model T5 (Raffel et al., 2020). This means the ATINTER can itself be attacked by an adversary. We believe that attacking a pipelined model such as ATINTER is not straightforward for the following two reasons. First, performing an adversarial attack on a model typically requires access to output scores from that model. Since ATINTER is used in a pipeline with a task classifier, the attacker can never get access to ATINTER's output scores. This adds an additional layer of complexity for the adversary. Second, targeted adversarial attacks on sequence-to-sequence models (such as ATINTER) are much less prominent and it is generally more difficult to make small alterations in the input without forcing a more significant change in the textual output (Cheng et al., 2020; Tan et al., 2020). Nevertheless, we have not explored this possibility and therefore recommend practitioners interested in using this work to carefully check for this.

Additionally, the experiments were only performed on four text classification datasets. Although we expect our method to be effective for other classification tasks like Toxicity detection, Hate Speech identification, but considering the sensitive nature of these applications, we urge the practitioners to first comprehensively evaluate our work on those tasks before deploying in a real world scenario.

For all our experiments, we used pre-established and published datasets, which do not pose any serious ethical concerns. For transparency and reproduciblity, we will make our code publicly available.

Acknowledgements

The authors thank Bloomberg's AI Engineering team, especially Umut Topkara and Anju Kambadur for helpful feedback and directions. We would also like to thank members of the Utah NLP group for their valuable insights, and the reviewers for their helpful feedback. This work was supported in part by the National Science Foundation under Grants #1801446, #1822877, #2007398 and #2217154. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

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 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Rongzhou Bao, Jiayi Wang, and Hai Zhao. 2021. De-

fending pre-trained language models from adversarial word substitution without performance sacrifice. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3248–3258, Online. Association for Computational Linguistics.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Yangyi Chen, Fanchao Qi, Hongcheng Gao, Zhiyuan Liu, and Maosong Sun. 2021. Textual backdoor attacks can be more harmful via two simple tricks.

Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. 2020. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3601–3608.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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 (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, Brussels, Belgium. Association for Computational Linguistics.

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

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

Siddhant Garg and Goutham Ramakrishnan. 2020b. Bae: Bert-based adversarial examples for text classification. *arXiv preprint arXiv:2004.01970*.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*.

Ashim Gupta, Giorgi Kvernadze, and Vivek Srikumar. 2021. Bert & family eat word salad: Experiments with text understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12946–12954.

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 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4083–4093, Hong Kong, China. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4129–4142, Hong Kong, China. Association for Computational Linguistics.

Lan Jiang, Hao Zhou, Yankai Lin, Peng Li, Jie Zhou, and Rui Jiang. 2022. Rose: Robust selective fine-tuning for pre-trained language models. *arXiv preprint arXiv:2210.09658*.

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 AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

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 (Volume 1: Long Papers)*, pages 6661–6674, Dublin, Ireland. Association for Computational Linguistics.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*.

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

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into transferable adversarial examples and black-box attacks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

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

Eric Malmi, Yue Dong, Jonathan Mallinson, Aleksandr Chuklin, Jakub Adamek, Daniil Mirylenka, Felix Stahlberg, Sebastian Krause, Shankar Kumar, and Aliaksei Severyn. 2022. Text generation with text-editing models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts*, pages 1–7, Seattle, United States. Association for Computational Linguistics.

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

John X Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020b. Reevaluating adversarial examples in natural language. *arXiv preprint arXiv:2004.14174*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John Duchi, and Percy Liang. 2020. Understanding and mitigating the tradeoff between robustness and accuracy. In *International Conference on Machine Learning*, pages 7909–7919. PMLR.

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*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

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 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2716–2725, Seattle, United States. Association for Computational Linguistics.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Samson Tan, Shafiq Joty, Min-Yen Kan, and Richard Socher. 2020. It's morphin' time! Combating linguistic discrimination with inflectional perturbations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2920–2935, Online. Association for Computational Linguistics.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Eric Wallace, Tony Zhao, Shi Feng, and Sameer Singh. 2021. Concealed data poisoning attacks on NLP models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 139–150, Online. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

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 2022*, pages 905–915, Dublin, Ireland. Association for Computational Linguistics.

Xuezhi Wang, Haohan Wang, and Diyi Yang. 2022b. Measure and improve robustness in NLP models: A survey. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4569–4586, Seattle, United States. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Mao Ye, Chengyue Gong, and Qiang Liu. 2020. SAFER: A structure-free approach for certified robustness to adversarial word substitutions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3465–3475, Online. Association for Computational Linguistics.

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

Xinze Zhang, Junzhe Zhang, Zhenhua Chen, and Kun He. 2021. Crafting adversarial examples for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1967–1977, Online. Association for Computational Linguistics.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing 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 and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5482–5492, Online. Association for Computational Linguistics.

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 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4904–4913, Hong Kong, China. Association for Computational Linguistics.

A Appendix

A.1 Implementation Details

SHIELD We find that SHIELD is very sensitive to the τ hyperparameter involved. There is a strong trade-off between the clean accuracy and adversarial robustness for the change in τ. For reporting the results, we try four values of

Attack	Type	Can humans identify it?	Example
TextFooler	word-level	NO	Org: The child is at the beach. Adv: The **youngster** is at the **shore**.
TextBugger	char-level, word-level	YES	Org: I love these awful 80's summer camp movies. Adv: I love these **aw ful** 80's summer camp movies.
BAE	word-level	NO	Org: The government made a quick decision. Adv: The **doctor** made a quick decision.
PWWS	word-level	NO	Org: E-mail scam targets police chief. Adv: E-mail scam targets police **headman**.
DeepWord	char-level	YES	Org: Subject: breaking news. would you ref inance ... Adv: su**jb**ect woul**g** y**uo** h**va**e an [OOV] ...

Table 6: **Summary of the black-box adversarial attacks**: Comparing the adversarial attacks we use in this work along with related information such as attach type, human perceptibility, and an example input for each attack. The third column indicates whether a human can easily identify if textual input was modified or not based on grammar syntax, semantics, and other language rules.

Model	# Params	Clean Acc.	TextFooler AA	TextBugger AA	BAE AA	PWWS AA	DWB AA	Avg. AA
`t5-small`	60M	**92.43**	11.29	31.39	33.62	15.51	17.49	21.86
`t5-base`	220M	91.97	23.96	40.52	34.16	23.06	25.31	29.40
`t5-large`	770M	**92.43**	31.14	50.99	36.85	30.4	39.45	37.77
`t5-3b`	3B	92.09	**39.33**	**57.53**	**42.67**	**35.1**	**54.79**	**45.88**

Table 7: Detailed Results with different sizes of the ATINTER. AA stands for Adversarial Accuracy. The results shown here are for the SST-2 dataset and the BERT classifier.

$\tau = [1.0, 0.1, 0.01, 0.001]$ and report the results for the model that retains the accuracy the most.

SAFER Since SAFER is an input randomization method, the default implementation provides different results for different runs, although we do not see any substantial change in numbers. For reporting clean accuracy, we average it with the numbers obtained with each of the five attacks. Additionally, since SAFER aggregates predictions by considering a large number of candidates for random synonym replacements for each word it decides to perturb, we found it is not practical to run with number of candidates equal to 100 (used in the original implementation). Therefore, we report the numbers with $n = 30$ in this paper.

ADFAR The official ADFAR implementation [4] only provides instructions to reproduce results for MR dataset (more specifically for tasks with single input classification and with only two possible labels). We, therefore, modify the codebase to make it work for AGNews –that has four classes, and MNLI, where the task is sentence-pair classification. We will release our modified codebase for ADFAR for the community to reproduce these results.

DISP We were not able to run the open-sourced implementation of DISP during our experiments [5]. We experimented with several different versions of both PyTorch and the transformers libraries but were still unsuccessful. More details can be found at a github issue we created at: [6].

A.2 Hyperparameters for training ATINTER

We list here the hyperparameters we used for training our model.

1. Learning Rate: We found that the learning rate of 5e-5 works best. We performed the learning rate search over the set [1e-5, 5e-5, 1e-6, 5e-6]. Also, we find the best learning rate for the SST-2 dataset and use the same for other datasets.

2. Batch Size: For all our models, except `t5-3b`, we use the batch size of 16 during training. Wherever, the batch of 16 did not fit the GPU (for example on the 16GB V100), we use gradient accumulation to have the effective batch size of 16. We did not perform any hyperparameter search for batch size due to computational reasons.

3. Sequence Length: Since examples in the SST-2 and MR datasets are smaller, we used the source and target side sequence lengths to both be 128. For AGNews, we use the sequence length of 512 and for MNLI, we use 256.

4. Number of epochs: For all our models (except that involve wiki pre-training), we used 5 epochs. For the pre-training setup, we used 10 epochs.

For training `t5-3b`, we needed to use `DeepSpeed` [7] for our experiments.

Model	Clean Acc	Adv. Acc.
ADFAR (Bao et al., 2021)	89.9	19.5
ATINTER (pre-training only)	**92.3**	9.6
ATINTER (SST-2)	92.0	24.0
+ pre-training	91.9	**26.5**

Table 8: Effect of Pretraining ATINTER using wikipedia sentences. Results shown for the SST-2 dataset.

A.3 Reproducibility Details

Dataset Splits We use the dataset splits from the Huggingface datasets repository. [8]. For datasets where we use a subsample of the test set, we use the random seed 1 to first shuffle and then evaluate on first 1000 instances.

Hardware We run most of our experiments using the Nvidia V100 (32 GB) GPU. Some of the later experiments with T5-3b required even larger GPU RAM and therefore, I was able to use Tesla A100 (40 GB VRAM) for last few experiments. Additionally, the servers had CPU: AMD EPYC 7513 32-Core Processor with CPU RAM 512 GB.

[4] https://github.com/LilyNLP/ADFAR

[5] https://github.com/joey1993/bert-defender

[6] https://github.com/joey1993/bert-defender/issues/2

[7] https://github.com/microsoft/DeepSpeed

[8] https://huggingface.co/datasets

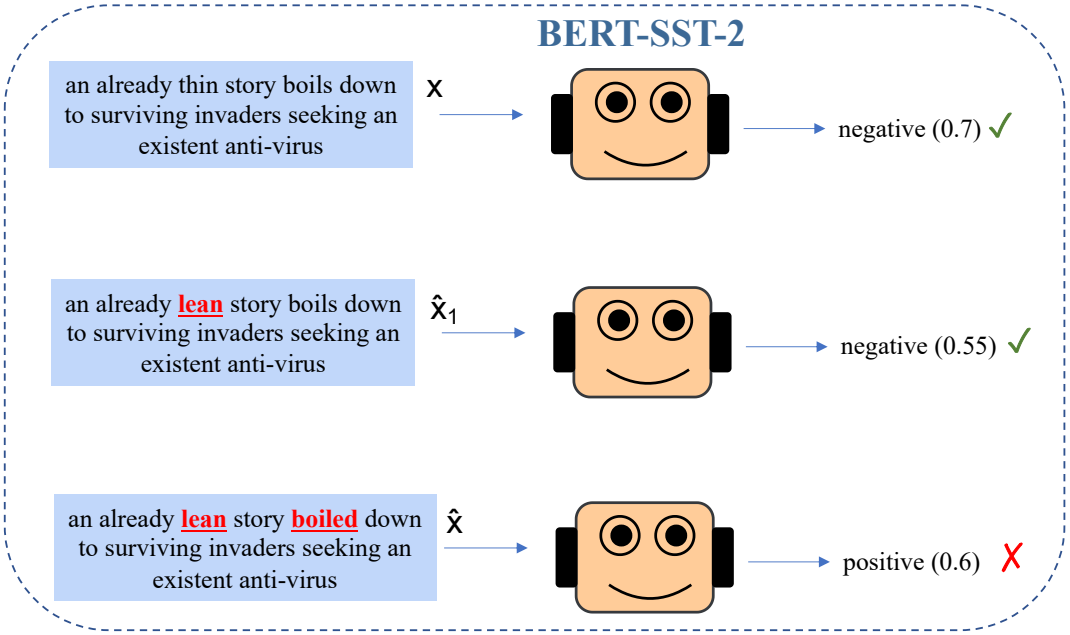

Figure 3: Illustration of how an adversarial example is generated. Given a non-adversarial, correctly predicted sentence $\mathbf{x}$, a typical adversary makes a word substitution to first obtain $\hat{\mathbf{x}}_1$ which lowers the predictive probability and then makes another substitution to generate $\hat{\mathbf{x}}$, such that the predicted label becomes incorrect.

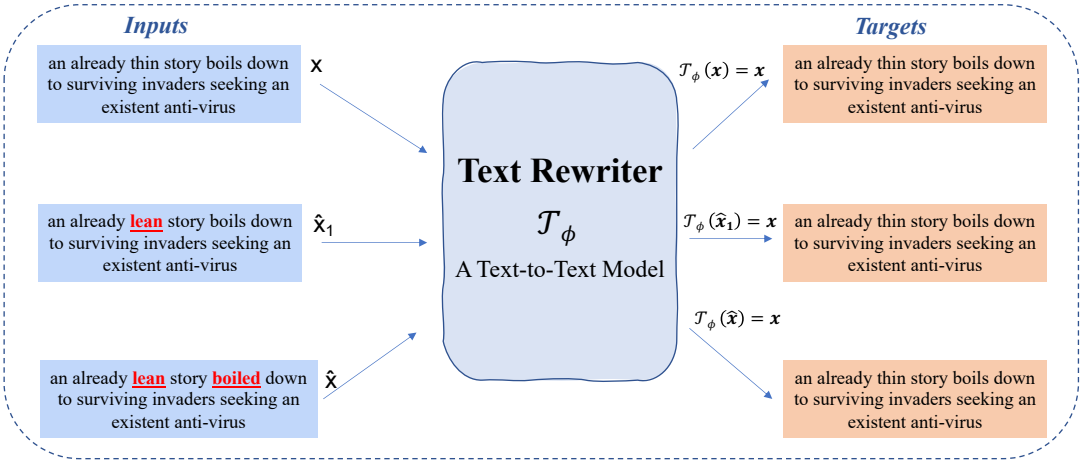

Figure 4: Training Setup for the Text Rewriter. For a non-adversarial example $\mathbf{x}$, the rewriter should learn to simply reproduce the same text as output. For the other two inputs ($\hat{\mathbf{x}}_1$, and $\hat{\mathbf{x}}$), the rewriter learns to restore the original input $\mathbf{x}$. Please refer to the main text for more details.

Defense	SST-2		MR		MNLI		AGNews	
	Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.
AT	-4.0	-4.0	0.0	-1.5	-2.7	1.8	-0.7	-1.0
SHIELD	-3.6	-3.7	-2.1	-5.1	-4.0	1.9	-2.5	1.2
SAFER	-3.1	0.3	**1.3**	-6.5	-5.5	3.8	-3.7	-4.2
SampleShielder	-15.6	-1.4	-8.0	-3.6	-42.1	-	-4.1	9.4
ADFAR	-2.5	4.2	-1.8	-1.4	-5.4	4.7	-2.7	10.2
ATINTER	**-0.4**	**7.4**	0.1	**6.2**	-0.5	**6.3**	-0.2	**21.9**

Table 9: **Summary of the main results.** Absolute percentage change in Clean Accuracy and Adversarial Accuracies averaged over the five adversarial attacks.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7, Section 5.6

☑ A2. Did you discuss any potential risks of your work?
Section 8

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☒ 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?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ 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)?
No response.

☐ 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?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ 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.
No response.

C ☑ Did you run computational experiments?

Section 4, 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 5.4, Appendix

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?
Section 3.2, Appendix

☐ 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?
Not applicable. 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.)?
Not applicable. 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.?
No response.

☐ 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)?
No response.

☐ 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?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.