Towards Improving Adversarial Training of NLP Models

Jin Yong Yoo, Yanjun Qi
Department of Computer Science
University of Virginia
Charlottesville, VA, USA
`{jy2ma, yq2h}@virginia.edu`

Abstract

Adversarial training, a method for learning robust deep neural networks, constructs adversarial examples during training. However, recent methods for generating NLP adversarial examples involve combinatorial search and expensive sentence encoders for constraining the generated instances. As a result, it remains challenging to use vanilla adversarial training to improve NLP models' performance, and the benefits are mainly uninvestigated. This paper proposes a simple and improved vanilla adversarial training process for NLP models, which we name Attacking to Training (`A2T`). The core part of `A2T` is a new and cheaper word substitution attack optimized for vanilla adversarial training. We use `A2T` to train BERT and RoBERTa models on IMDB, Rotten Tomatoes, Yelp, and SNLI datasets. Our results empirically show that it is possible to train robust NLP models using a much cheaper adversary. We demonstrate that vanilla adversarial training with `A2T` can improve an NLP model's robustness to the attack it was originally trained with and also defend the model against other types of word substitution attacks. Furthermore, we show that `A2T` can improve NLP models' standard accuracy, cross-domain generalization, and interpretability. [1]

1 Introduction

Recently, robustness of neural networks against adversarial examples has been an active area of research in natural language processing with a plethora of new adversarial attacks[2] having been proposed to fool question answering (Jia and Liang, 2017), machine translation (Cheng et al., 2018), and text classification systems (Ebrahimi et al., 2017; Jia and Liang, 2017; Alzantot et al., 2018; Jin et al., 2019; Ren et al., 2019; Zang et al., 2020;

---

[1]Code is available at `https://github.com/QData/Textattack-A2T`

[2]We use "methods for adversarial example generation" and "adversarial attacks" interchangeably.

Garg and Ramakrishnan, 2020). One method to make models more resistant to such adversarial attacks is *adversarial training* where the model is trained on both original examples and adversarial examples (Goodfellow et al., 2014; Madry et al., 2018). Due to its simple workflow, it is a popular go-to method for improving adversarial robustness.

Typically, adversarial training involves generating adversarial example $\mathbf{x}'$ from each original example $\mathbf{x}$ before training the model on both $\mathbf{x}$ and $\mathbf{x}'$. In NLP, generating an adversarial example is typically framed as a combinatorial optimization problem solved using a heuristic search algorithm. Such an iterative search process is expensive. Depending on the choice of the search algorithm, it can take up to tens of thousands of forward passes of the underlying model to generate one example (Yoo et al., 2020). This high computational cost hinders the use of vanilla adversarial training in NLP, and it is unclear how and as to what extent such training can improve an NLP model's performance (Morris et al., 2020a).

In this paper, we propose to improve the vanilla adversarial training in NLP with a computationally cheaper adversary, referred to as `A2T`. The proposed `A2T` uses a cheaper gradient-based word importance ranking method to iteratively replace each word with synonyms generated from a counter-fitted word embedding (Mrksic et al., 2016). We use `A2T` and its variation `A2T-MLM` (which uses masked language model-based word replacements instead) to train BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) models on text classification tasks such as IMDB (Maas et al., 2011), Rotten Tomatoes (Pang and Lee, 2005), Yelp (Zhang et al., 2015), and SNLI (Bowman et al., 2015) datasets. Our findings are as following:

- Adversarial training with both `A2T` and `A2T-MLM` can help improve adversarial robustness, even against NLP attacks that were not used to train the model (see Table 5).
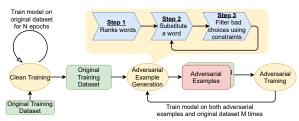
Figure 1: Pipeline for vanilla adversarial training in NLP

- Adversarial training with `A2T` can provide a regularization effect and improve the model's standard accuracy and/or cross-domain generalization, while `A2T-MLM` tends to hurt both standard accuracy and cross-domain generalization (see Table 7).
- Using LIME (Ribeiro et al., 2016) and AOPC metric, we demonstrate that adversarial training with `A2T` can improve NLP models' interpretability (see Table 8).

2 Background

2.1 Vanilla Adversarial Training in NLP

Vanilla adversarial training has been a major defense strategy in most existing work on adversarial robustness (Goodfellow et al., 2014; Kurakin et al., 2016; Madry et al., 2018). In our work, we define *vanilla adversarial training* as adversarial training that involves augmenting the training data with adversarial examples generated from perturbing the training data in the input space. In contrast, non-vanilla adversarial training performs perturbations on non-input space such as word embeddings (Miyato et al., 2017; Zhu et al., 2019; Jiang et al., 2020; Liu et al., 2020).

In recent NLP literature, vanilla adversarial training has only been evaluated in a limited context. In most studies, adversarial training is only performed to show that such training can make models more resistant to the attack it was originally trained with (Jin et al., 2019; Ren et al., 2019; Li et al., 2020; Zang et al., 2020; Li et al., 2021). This observation is hardly surprising, and it is generally recommended to use different attacks to evaluate the effectiveness of defenses (Carlini et al., 2019). Therefore, in this paper, we perform a more in-depth investigation into how a practical vanilla adversarial training algorithm we propose affects NLP models' adversarial robustness against a set of different attacks that are not used for training.

In addition, we examine how adversarial training affects model's performance in other aspects such as standard accuracy, cross-domain generalization, and interpretability.

2.2 Components of an NLP Attack

Figure 1 includes a schematic diagram on vanilla adversarial training where an adversarial attack is part of the training procedure. We borrow the framework introduced by Morris et al. (2020b) which breaks down the process of generating natural language adversarial examples into three parts (see Table 1): (1) A search algorithm to iteratively search for the best perturbations (2) A transformation module to perturb a text input from $\mathbf{x}$ to $\mathbf{x}'$ (e.g. synonym substitutions) (3) Set of constraints that filters out undesirable $\mathbf{x}'$ to ensure that perturbed $\mathbf{x}'$ preserves the semantics and fluency of the original $\mathbf{x}$.

Adversarial attacks frame their approach as a combinatorial search because of the exponential nature of the search space. Consider the search space for an adversarial attack that replaces words with synonyms: If a given sequence of text consists of N words, and each word has M potential substitutions, the total number of perturbed inputs to consider is $(M+1)^N - 1$. Thus, the graph of all potential adversarial examples for a given input is far too large for an exhaustive search. Studies on NLP attacks have explored various heuristic search algorithms, including beam search (Ebrahimi et al., 2017), genetic algorithm (Alzantot et al., 2018), and greedy method with word importance ranking (Gao et al., 2018; Jin et al., 2019; Ren et al., 2019).

3 Method: `A2T` (Attacking to Training)

In this section, we present our algorithm `A2T` for an improved and practical vanilla adversarial training for NLP. We also present the cheaper adversarial attacks we propose to use in `A2T`.

3.1 Training Objective

Following the recommendations by Goodfellow et al. (2014); Kurakin et al. (2016), we use both clean[3] and adversarial examples to train our model. We aim to minimize both the loss on the original training dataset and the loss on the adversarial examples.

Let $\mathbf{L}(\theta, \mathbf{x}, \mathbf{y})$ represent the loss function for input text $\mathbf{x}$ and label $\mathbf{y}$ and let $\mathbf{A}(\theta, \mathbf{x}, \mathbf{y})$ be the adversarial attack that produces adversarial example $\mathbf{x}_{adv}$. Then, our training objective is as following:

$$\arg\min_{\theta} \mathbf{E}_{(\mathbf{x},\mathbf{y}) \sim D}[\mathbf{L}(\theta, \mathbf{x}, \mathbf{y}) \\ + \alpha \mathbf{L}(\theta, \mathbf{A}(\theta, \mathbf{x}, \mathbf{y}), \mathbf{y})] \quad (1)$$

---

[3]*Clean* examples refer to the original training examples.

α is used to weigh the adversarial loss. In this work, we set $\alpha = 1$, weighing the two loss equally. [4]

3.2 A Practical Training Workflow

Previous works in adversarial training, especially those from computer vision (Goodfellow et al., 2014; Madry et al., 2018), generate adversarial examples between every mini-batch and use them to train the model. However, it is difficult in practice to generate adversarial examples between every mini-batch update when using NLP adversarial attacks.

This is because NLP adversarial attacks typically require other neural networks as their sub-components (e.g. sentence encoders, masked language models). For example, Jin et al. (2019) uses Universal Sentence Encoder (Cer et al., 2018) while Garg and Ramakrishnan (2020) uses BERT masked language model (Devlin et al., 2018). Given that recent Transformers models such as BERT and RoBERTa models also require large amounts of GPU memory to store the computation graph during training, it is impossible to run adversarial attacks and train the model in the same GPU. We, therefore, propose to instead maximize GPU utilization by first generating adversarial examples before every epoch and then using the generated samples to train the model.

Based on our initial empirical study, we found that it is not necessary to generate adversarial examples for every clean example in the training dataset to improve the robustness of the model. In fact, as we see in Section 5.6, training with fewer adversarial examples can often produce better results. Therefore, we propose to leave the number of adversarial examples produced in each epoch as a hyperparameter γ where it is a percentage of the original training dataset. For cases where the adversarial attack fails to find an adversarial example, we skip them and instead sample more from the training dataset to compensate for the skipped samples. In our experiments, unless specified, we attack 20% of the training dataset, which was based on our initial empirical findings.

Algorithm 1 shows the proposed A2T adversarial training algorithm in detail. We run clean training for N_{clean} number of epochs before performing N_{adv} epochs of adversarial training. Between line 6-13, we generate the adversarial examples until we obtain γ percentage of the training dataset. When multiple GPUs are available, we use data

---

Algorithm 1 Adversarial Training with A2T
***
Require: Number of clean epochs N_{clean}, number of adversarial epochs N_{adv}, percentage of dataset to attack γ, attack $\mathbf{A}(\theta, \mathbf{x}, \mathbf{y})$, and training data $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$, α the smoothing proportion of adversarial training
1: Initialize model θ
2: **for** clean epoch$= 1, \ldots, N_{\text{clean}}$ **do**
3:     Train θ on D
4: **end for**
5: **for** adversarial epoch$= 1, \ldots, N_{\text{adv}}$ **do**
6:     Randomly shuffle D
7:     $D_{\text{adv}} \leftarrow \{\}$
8:     $i \leftarrow 1$
9:     **while** $|D_{\text{adv}}| < \gamma * |D|$ and $i \leq |D|$ **do**
10:         $\mathbf{x}_{\text{adv}}^{(i)} \leftarrow \mathbf{A}(\theta, \mathbf{x}^{(i)}, \mathbf{y}^{(i)})$
11:         $D_{\text{adv}} \leftarrow D_{\text{adv}} \cup \{(\mathbf{x}_{\text{adv}}^{(i)}, \mathbf{y}^{(i)})\}$
12:         $i \leftarrow i + 1$
13:     **end while**
14:     $D' \leftarrow D \cup D_{\text{adv}}$
15:     Train θ on D' with α used to weigh the loss
16: **end for**

---

parallelism to speed up the generation process. We also shuffle the dataset before attacking to avoid attacking the same sample every epoch.

3.3 Cheaper Attack for Adversarial Training

The attack component in A2T is designed to be faster than previous attacks from literature. We achieve the speedup by making two key choices when constructing our attack: (1) Gradient-based word importance ordering, and (2) DistilBERT (Sanh et al., 2019) semantic textual similarity constraint. Table 1 summarizes the differences between A2T and two other attacks from literature: TextFooler (Jin et al., 2019) and BAE (Garg and Ramakrishnan, 2020).

Faster Search with Gradient-based Word Importance Ranking: Previous attacks such as Jin et al. (2019); Garg and Ramakrishnan (2020) iteratively replace one word at a time to generate adversarial examples. To determine the order of words in which to replace, both Jin et al. (2019); Garg and Ramakrishnan (2020) rank the words by how much the target model's confidence on the ground truth label changes when the word is deleted from the input. We will refer to this as deletion-based word importance ranking.

One issue with this method is that an additional forward pass of the model must be made for each

---
[4]We leave tuning for the optimal α for future work.

Components	A2T	A2T-MLM	TextFooler	BAE
Search Method for Ranking Words	**Gradient-based** Word Importance	**Gradient-based** Word Importance	**Deletion-based** Word Importance	**Deletion-based** Word Importance
Word Substitution	**Word Embedding**	**BERT MLM**	**Word Embedding**	**BERT MLM**
Constraints	POS Consistency	POS Consistency	POS Consistency	POS Consistency
	DistilBERT Similarity	**DistilBERT Similarity**	**USE Similarity**	**USE Similarity**

Table 1: Comparing A2T and variation with TextFooler (Jin et al., 2019) and BAE (Garg and Ramakrishnan, 2020)

Attack	Runtime (sec)
A2T	5,988
TextFooler + Gradient Search	8,760
TextFooler	17,268

Table 2: The runtime (in seconds) of A2T, original TextFooler, and TextFooler where deletion-based word importance ranking is switched with gradient-based word importance ranking. We can see that replacing the search method gives us approximately 2× speedup. The attack was carried out against a BERT model trained on IMDB dataset and 1000 samples were attacked.

word to calculate its importance. For longer text inputs, this can mean that we have to make up to hundreds of forward passes to generate one adversarial example.

A2T instead determines each word's importance using the gradient of the loss. For an input text including n words: $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ where each x_i is a word, the importance of x_i is calculated as:

$$I(x_i) = ||\nabla_{\mathbf{e}_i} \mathbf{L}(\theta, \mathbf{x}, \mathbf{y}))||_1 \qquad (2)$$

where $\mathbf{e}_i$ is the word embedding that corresponds to word x_i. For BERT and RoBERTa models where inputs are tokenized into sub-words, we calculate the importance of each word by taking the average of all sub-words constituting the word.

This requires only one forward and backward pass and saves us from having to make additional forward passes for each word. Yoo et al. (2020) showed that the gradient-ordering method is the fastest search method and provides competitive attack success rate when compared to the deletion-based method. Table 2 shows that when we switch from deletion-based ranking ("TextFooler") to gradient-based ranking ("TextFooler+Gradient Search"), we can obtain approximately 2× speedup.

Cheaper Constraint Enforcing with Distil-BERT (Sanh et al., 2019) semantic textual similarity model: Most recent attacks like Jin et al. (2019); Garg and Ramakrishnan (2020); Li et al. (2020) use Universal Sentence Encoders (USE)

(Cer et al., 2018) to compare the sentence encodings of original text $\mathbf{x}$ and perturbed text $\mathbf{x}'$. If the cosine similarity between two encodings fall below a certain threshold, $\mathbf{x}'$ is ignored. One of the challenges of using large encoders like USE is that it can take up significant amount of GPU memory – up to 9GB in case of USE.

Instead of using USE, A2T uses DistilBERT (Sanh et al., 2019) model trained on semantic textual similarity task as its constraint module [5]. This is because DistilBERT requires 10× less GPU memory than USE and requires fewer operations.

3.4 A2T-MLM : Variation with a Different Word Substitution Strategy

A2T generates replacements for each word by selecting top-k nearest neighbors in a counter-fitted word embedding (Mrksic et al., 2016), which helps nearest-neighbor searches return better synonyms than regular word embeddings. This word substitution strategy has been previously proposed by Alzantot et al. (2018); Jin et al. (2019). In our work, we first precompute all the top-k nearest neighbors and cache them to speed up our attacks. We also consider another variation we name as A2T-MLM in which BERT masked language model is used to generate replacements (proposed in Garg and Ramakrishnan (2020); Li et al. (2020, 2021)).

We consider this variation because two strategies prioritize different language qualities when proposing word replacements. Counter-fitted word embeddings are likely to propose synonyms as replacements, but could produce incoherent texts as it does not take the entire context into account. On the other hand, BERT masked language model is more likely to propose replacement words that preserve grammatical and contextual coherency but fail to preserve the semantics. Comparing A2T with A2T-MLM allows us to study the effect of word substitution strategy on adversarial training.

---

[5] We use code from Reimers and Gurevych (2019)

4 Related Work

Past works on adversarial training for NLP models come in diverse flavors that differ in how adversarial examples are generated. Miyato et al. (2017), which is one of the first works to introduce adversarial training to NLP tasks, perform perturbations in the word embedding level instead of the actual input space level. Likewise, Zhu et al. (2019); Jiang et al. (2020); Liu et al. (2020) all apply perturbations in the embedding level using gradient-based optimization methods from computer vision.

Another family of work on adversarial training involves computing the hyperspace of activations that contains all texts that can be generated using word substitutions and then training the model to make consistent prediction for inputs inside the hyperspace. Jia et al. (2019); Huang et al. (2019) compute axis-aligned hyper-rectangles and leverages Interval Bound Propagation (Dvijotham et al., 2018) to defend the model against substitution attacks while Dong et al. (2021) computes the desired hyperspace as a convex hull in the embedding space and further trains the model to be robust against worst case embedding in the convex hull.

Yet, adversarial training that simply uses adversarial examples generated in the input space is still a relatively unexplored area of research despite its simple, extendable workflow. Most works that have discussed such form of adversarial training only train limited number of models and datasets to show that adversarial training can make models more resistant to the particular attack used to train the model (Jin et al., 2019; Ren et al., 2019; Li et al., 2020; Zang et al., 2020; Li et al., 2021). Our work demonstrates that simple vanilla adversarial training can actually provide improvements in adversarial robustness across many different word substitution attacks. Furthermore, we show that it can improve both generalization and interpretability of models, properties that have not been examined by previous works.

5 Experiment and Results

5.1 Datasets & Models

We chose IMDB (Maas et al., 2011), Movie Reviews (MR) (Pang and Lee, 2005), Yelp (Zhang et al., 2015), and SNLI (Bowman et al., 2015) datasets for our experiment. For Yelp, instead of using the entire training set, we sampled 30k examples for training and 10k for validation.

We trained BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) models using the im-

Dataset	Train	Dev	Test
IMDB	20k	5k	25k
MR	8.5k	1k	1k
Yelp	30k	10k	38k
SNLI	550k	10k	10k

Table 3: Overview of the datasets.

plementation provided by Wolf et al. (2020). All texts were tokenized up to the first 512 tokens and we trained the model for one clean epoch and three adversarial epochs. Adam optimizer with weight decay of 0.01 (Loshchilov and Hutter, 2017) and learning rate of $5\mathrm{e}{-5}$ were used for training. Also, we used a linear scheduler with 500 warm-up steps for IMDB and Yelp, 100 steps for MR, and 5000 steps for SNLI. We performed three runs with random seeds for each model.

5.2 Baselines

Adversarial training can be viewed as a data augmentation method where *hard* examples are added to the training set. Therefore, besides just having models that are trained on clean adversarial examples (i.e. "natural training") as our baseline, we also compare our results to models trained using more conventional data augmentation methods. We use SSMBA (Ng et al., 2020) and backtranslation[6] (Xie et al., 2019) methods as our baselines as both have reported strong performance on text classification tasks. We use these methods to generate approximately the same number of new training examples as adversarial training.

5.3 Results on Adversarial Robustness

To evaluate models' robustness to adversarial attacks, we attempt to generate adversarial examples from 1000 randomly sampled clean examples from the test set and measure the attack success rate.

$$\text{attack success rate} = \frac{\text{\# of successful attacks}}{\text{\# of total attacks}}$$

Table 4 shows the attack success rates of `A2T` attack and `A2T-MLM` attack against models that have been trained using `A2T`, `A2T-MLM`, and other baseline methods. Note that the overall attack success rates appear fairly low because we applied strict constraints to improve the quality of the adversarial examples (as recommend by Morris et al. (2020b)). Still, we can see that for both attacks, adversarial training using the same attack can decrease the attack success rate by up to 70%. What is surprising

---

[6]For backtranslation, we use English-to-German model and German-to-English model trained by Ng et al. (2019).

is that training the model using a different attack also led to a decrease in the attack success rate. From Table 4, we can see that adversarial training using the `A2T-MLM` attack lowers the attack success rate of `A2T` attack while training with `A2T` lowers the attack success rate of `A2T-MLM` attack.

To further measure how adversarial training using `A2T` can improve model's robustness, we evaluated accuracy of BERT-`A2T` on 1000 adversarial examples that have successfully fooled BERT-Natural models. Table 6 shows that adversarial training using `A2T` greatly improves model's performance against adversarial examples from baseline 0% to over 70% for IMDB and Yelp datasets.

Another surprising observation is that training with data augmentations methods like SSMBA and backtranslation can lead to improvements in robustness against both adversarial attacks. However, in case of smaller datasets such as MR, data augmentation can also hurt robustness.

When we compare the attack success rates between BERT and RoBERTa models in Table 4, we also see an interesting pattern. BERT models, regardless of the training method, tend to be more vulnerable to `A2T` attack than RoBERTa models. At the same time, RoBERTa models tend to be more vulnerable to `A2T-MLM` attack than BERT models.

Lastly, we use attacks proposed from literature to evaluate the models' adversarial robustness. Table 5 shows the attack success rate of TextFooler (Jin et al., 2019), BAE (Garg and Ramakrishnan, 2020), PWWS (Ren et al., 2019), and PSO (Zang et al., 2020).[7] Across four datasets and two models, we can see that both `A2T` and `A2T-MLM` lower the attack success rate against all four attacks in all but five cases. The results for PWWS and PSO are especially surprising since both use different transformations - WordNet (Miller, 1995) and HowNet (Dong et al., 2010) - when carrying out the attacks.

5.4 Results on Generalization

To evaluate how adversarial training affects the model's generalization ability, we evaluate its accuracy on the original test set (i.e. standard accuracy) and on an out-of-domain dataset (e.g. Yelp dataset for model trained on IMDB dataset). In Table 7, we can see that in all but two cases, adversarial training using `A2T` attack beats natural training in terms of standard accuracy. In the two cases (SNLI) where

---

natural training beats `A2T`, we can see that `A2T` still outperforms natural training in cross-domain accuracy. Overall, in six out of eight cases, `A2T` improves cross-domain accuracy. On the other hand, adversarial training with `A2T-MLM` attack tends to hurt both standard accuracy and cross-domain accuracy. This confirms the observations reported by Li et al. (2021) and suggests that using a masked language model to generate adversarial examples can lead to a trade-off between robustness and generalization. We do not see similar trade-off with `A2T`.

5.5 Results on Interpretability

We use LIME (Ribeiro et al., 2016) to generate local explanations for our models. For each example, LIME approximates the local decision boundary by fitting a linear model over the samples obtained by perturbing the example. To measure the faithfulness of the local explanations obtained using LIME, we measure the area over perturbation curve (AOPC) (Samek et al., 2017; Nguyen, 2018; Chen and Ji, 2020) which is defined as:

$$AOPC = \frac{1}{K+1} \sum_{k=1}^{K} \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_{(0)}^{(i)}) - f(\mathbf{x}_{(k)}^{(i)})$$
(3)

where $\mathbf{x}_{(0)}^{(i)}$ represents example $\mathbf{x}^{(i)}$ with none of the words removed and $\mathbf{x}_{(k)}^{(i)}$ represents example $\mathbf{x}^{(i)}$ with the top-k most important words removed. $f(\mathbf{x})$ here represents the model's confidence on the target label $\mathbf{y}^{(i)}$. Intuitively, AOPC measures on average how the model's confidence on the target label changes when we delete the top-k most important words determined using LIME.

For each dataset, we randomly pick 1000 examples from the test set for evaluation. When running LIME to obtain explanations, we generate 1000 perturbed samples for each instance. We set $K = 10$ for the AOPC metric. Table 8 shows that across three sentiment classification datasets, BERT model trained using `A2T` attack achieves higher AOPC than natural training. For RoBERTa models, the same observation holds (although by smaller margins). Overall, we see that the AOPC scores for RoBERTa models are far lower than those for BERT models, suggesting that RoBERTa might be less interpretable than BERT.

5.6 Analysis

`A2T` vs `A2T-MLM` attack We can see that model trained using `A2T` attack outperforms the model

---

[7]These attacks were implemented using the `TextAttack` library (Morris et al., 2020a).

Attack	Model	IMDB		MR		Yelp		SNLI	
		A.S.%	Δ%	A.S.%	Δ%	A.S.%	Δ%	A.S.%	Δ%
A2T	BERT-Natural	42.9		20.9		25.4		53.3	
	BERT-A2T	**12.7**	**-70.4**	**13.2**	**-36.8**	**11.5**	**-54.7**	**15.6**	**-70.7**
	BERT-A2T-MLM	34.5	-19.6	18.9	-9.6	21.0	-17.3	47.2	-11.4
	BERT-SSMBA	29.5	-31.2	21.1	1.0	23.3	-8.3	51.0	-4.3
	BERT-BackTranslation	33.1	-22.8	19.2	-8.1	24.0	-5.5	48.3	-9.4
	RoBERTa-Natural	34.3		18.6		19.9		48.4	
	RoBERTa-A2T	**12.4**	**-63.8**	**12.1**	**-34.9**	**7.6**	**-61.8**	**8.3**	**-82.9**
	RoBERTa-A2T-MLM	19.5	-43.1	17.1	-8.1	13.0	-34.7	40.3	-16.7
	RoBERTa-SSMBA	24.0	-30.0	21.8	17.2	19.3	-3.0	48.7	0.6
	RoBERTa-BackTranslation	28.9	-15.7	18.3	-1.6	16.1	-19.1	48.6	0.4
A2T-MLM	BERT-Natural	76.6		37.7		47.1		77.9	
	BERT-A2T	61.7	-19.5	33.2	-11.9	42.5	-9.8	76.7	-1.5
	BERT-A2T-MLM	**48.3**	**-36.9**	**24.7**	**-34.5**	**27.9**	**-40.8**	**37.1**	**-52.4**
	BERT-SSMBA	59.6	-22.2	36.2	-4.0	44.8	-4.9	76.9	-1.3
	BERT-BackTranslation	68.8	-10.2	36.3	-3.7	46.8	-0.6	77.3	-0.8
	RoBERTa-Natural	81.5		40.9		53.2		78.6	
	RoBERTa-A2T	69.8	-14.4	38.4	-6.1	45.2	-15.0	76.5	-2.7
	RoBERTa-A2T-MLM	**37.0**	**-54.6**	**28.5**	**-30.3**	**25.8**	**-51.5**	**35.2**	**-55.2**
	RoBERTa-SSMBA	57.0	-30.1	43.1	5.4	47.8	-10.2	78.3	-0.4
	RoBERTa-BackTranslation	74.3	-8.8	41.1	0.5	43.8	-17.7	79.1	0.6

Table 4: Attack success rate of A2T and A2T-MLM attacks. A.S.% represents the attack success rates and Δ% column represents the percent change between the attack success rate of natural training and the different training methods.

Attack	Model	IMDB		MR		Yelp		SNLI	
		A.S.%	Δ%	A.S.%	Δ%	A.S.%	Δ%	A.S.%	Δ%
TextFooler	BERT-Natural	85.0		91.6		**55.9**		97.5	
	BERT-A2T	**66.0**	**-22.4**	90.6	-1.1	57.9	3.6	**92.2**	**-5.4**
	BERT-A2T-MLM	88.2	3.8	**89.0**	**-2.8**	67.7	21.1	94.1	-3.5
	RoBERTa-Natural	95.2		94.4		74.5		96.7	
	RoBERTa-A2T	**82.4**	**-13.4**	**91.0**	**-3.6**	**68.7**	**-7.8**	91.4	-5.5
	RoBERTa-A2T-MLM	72.9	-23.4	88.6	-6.1	71.7	-3.8	**90.8**	**-6.1**
BAE	BERT-Natural	60.5		52.6		37.8		76.7	
	BERT-A2T	**46.7**	**-22.8**	51.5	-2.1	34.4	-9.0	75.9	-1.0
	BERT-A2T-MLM	52.4	-13.4	**43.8**	**-16.7**	**31.3**	**-17.2**	**60.9**	**-20.6**
	RoBERTa-Natural	65.5		56.4		44.4		75.6	
	RoBERTa-A2T	56.8	-13.3	54.7	-3.0	38.0	-14.4	76.0	0.5
	RoBERTa-A2T-MLM	**42.3**	**-35.4**	**48.3**	**-14.4**	**28.7**	**-35.4**	**61.2**	**-19.0**
PWWS	BERT-Natural	87.5		82.1		67.9		98.5	
	BERT-A2T	**70.9**	**-19.0**	**80.4**	**-2.1**	65.4	-3.7	**97.5**	**-1.0**
	BERT-A2T-MLM	87.1	-0.5	81.3	-1.0	72.2	6.3	**97.5**	**-1.0**
	RoBERTa-Natural	96.6		83.8		77.9		98.2	
	RoBERTa-A2T	84.4	-12.6	81.9	-2.3	73.1	-6.2	97.1	-1.1
	RoBERTa-A2T-MLM	**73.5**	**-23.9**	**79.8**	**-4.8**	**70.7**	**-9.2**	**96.5**	**-1.7**
PSO	BERT-Natural	43.8		81.6	-	40.3		92.1	
	BERT-A2T	**16.5**	**-62.3**	**73.2**	**-10.3**	26.4	-34.5	89.1	-3.3
	BERT-A2T-MLM	29.9	-31.7	75.4	-7.6	34.4	-14.6	89.7	-2.6
	RoBERTa-Natural	34.8		88.0		35.7		90.6	
	RoBERTa-A2T	**12.9**	**-62.9**	81.6	-7.3	21.6	-39.5	85.3	-5.8
	RoBERTa-A2T-MLM	13.1	-62.4	**77.5**	**-11.9**	20.3	-43.1	84.8	-6.4

Table 5: Attack success rate of attacks from literature, including original TextFooler (Jin et al., 2019), BAE (Garg and Ramakrishnan, 2020), PWWS (Ren et al., 2019), and PSO (Zang et al., 2020). A.S.% represents the attack success rates and Δ% column represents the percent change between natural training and the different training methods.

Dataset	Model	A2T	A2T-MLM	TextFooler	BAE	PSO	PWWS
IMDB	BERT-A2T	94.60	75.87	93.96	78.42	87.00	89.14
MR	BERT-A2T	65.90	53.25	57.56	31.96	36.57	47.76
Yelp	BERT-A2T	92.00	77.78	87.17	73.88	72.44	83.21
SNLI	BERT-A2T	56.43	47.10	41.99	42.71	41.67	39.84

Table 6: Accuracy of BERT-A2T on 1000 adversarial examples that have successfully fooled BERT-Natural.

Model	IMDB		MR		Yelp		SNLI	
	Standard Accuracy	Yelp Accuracy	Standard Accuracy	Yelp Accuracy	Standard Accuracy	IMDB Accuracy	Standard Accuracy	MNLI Accuracy
BERT-Natural	93.97	92.13	85.40	**90.60**	96.34	88.31	90.29	73.34
BERT-A2T	**94.49**	**92.50**	85.61	88.45	**96.68**	**89.24**	90.16	**73.79**
BERT-A2T-MLM	93.05	90.67	83.80	85.32	95.85	85.01	87.87	70.93
BERT-SSMBA	93.94	91.59	85.33	89.49	96.28	88.54	90.23	73.27
BERT-BackTranslation	93.97	91.73	**85.65**	89.46	96.46	88.77	**90.57**	72.82
RoBERTa-Natural	95.26	94.09	87.52	93.42	97.26	**91.94**	**91.56**	77.66
RoBERTa-A2T	**95.57**	94.41	**88.03**	93.45	**97.45**	91.86	91.16	**77.88**
RoBERTa-A2T-MLM	94.71	**94.48**	86.49	92.93	96.84	90.44	88.56	74.82
RoBERTa-SSMBA	95.25	94.11	86.46	93.03	97.16	91.90	91.38	77.02
RoBERTa-BackTranslation	95.31	93.84	87.78	**93.77**	97.25	91.76	90.79	76.70

Table 7: Accuracy on in-domain and out-of-domain datasets. We can see that adversarial training can helps model outperform both naturally trained models and models trained using data augmentation methods.

Model	IMDB	MR	Yelp
BERT-Natural	7.78	33.43	12.78
BERT-A2T	**10.74**	**34.25**	**13.18**
BERT-A2T-MLM	9.12	32.17	11.14
BERT-SSMBA	7.21	32.21	10.94
BERT-BackTranslation	6.02	0.39	11.10
RoBERTa-Natural	0.35	0.01	-1.09
RoBERTa-A2T	**0.43**	**0.45**	-1.01
RoBERTa-A2T-MLM	0.09	-0.12	-1.13
RoBERTa-SSMBA	0.26	0.05	**-0.43**
RoBERTa-BackTranslation	-0.04	0.05	-1.06

Table 8: AOPC scores of the LIME explanations for each model. Higher AOPC scores indicates that the model is more interpretable.
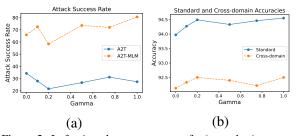

(a)   (b)

Figure 2: Left: Attack success rates of A2T and A2T-MLM attacks on BERT-A2T model trained on IMDB dataset. Right: Standard accuracies and cross-domain accuracies (on Yelp) for the same model.

trained using A2T-MLM attack in standard accuracy and cross-domain accuracy in all but one case. This suggests that using counter-fitted embeddings can generate higher quality adversarial examples than masked language models. Since masked language models are only trained to predict words that are statistically most likely to appear, it is likely that it will propose words that do change the se-

mantics of the text entirely; this can lead to false positive adversarial examples.

We also hypothesize that A2T-MLM's tendency to generate false positive adversarial examples is the reason why RoBERTa appears to be more vulnerable to A2T-MLM than BERT models. Since RoBERTa models tend to have better generalization capability than BERT models, RoBERTa models are more likely to predict the correct labels for false positive adversarial examples that A2T-MLM can generate (whereas BERT models can predict the wrong labels for false positive adversarial examples and appear falsely robust).

Effect of Gamma Recall γ, which is the desired percentage of adversarial examples to generate in every epoch. To study how it affects the results of adversarial training, we train BERT model on IMDB dataset with A2T method and γ value ranging from 0 (no adversarial training) to 1.0. Figure 2 shows the attack success rates, standard accuracies, and cross-domain accuracies.

We can see from Figure 2 (a) that higher γ does not necessarily mean that our trained model is more robust, with $\gamma = 0.2$ producing model that is more robust than others. Overall, we can see that adversarial training is insensitive to the specific choice of γ and a smaller γ can be used for faster training.

Effect of Adversarial Training on Sentence Embedding In BERT (Devlin et al., 2018), the output for [CLS] token represents the sentence-level em-
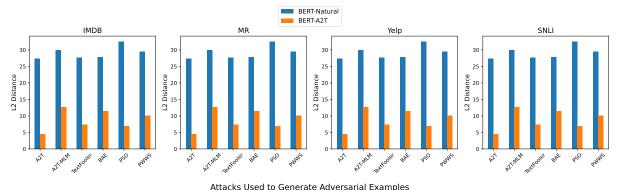
Figure 3: BERT-A2T decreases the ℓ_2 distance between [CLS] embeddings of original $\mathbf{x}$ and adversarial $\mathbf{x}'$.

bedding that is used by the final classification layer. For BERT-Natural and BERT-A2T, we measured the ℓ_2 distance between the [CLS] embeddings of original text $\mathbf{x}$ and its corresponding adversarial example $\mathbf{x}'$ using six different attacks[8]. We noticed that across all cases, adversarial training decreases the average ℓ_2 distance between $\mathbf{x}$ and $\mathbf{x}'$, as shown by Figure 3. This suggests that adversarial training improves the robustness of models by encouraging the model to learn a closer mapping of $\mathbf{x}$ and $\mathbf{x}'$.

6 Conclusion

In this paper, we have presented a practical vanilla adversarial training process called A2T that uses a new adversarial attack designed to generate adversarial examples quickly. We demonstrated that using A2T allows us to improve model's robustness against several different types of adversarial attacks that have been proposed from literature. Also, we have shown that models trained using A2T can achieve better standard accuracy and/or cross-domain accuracy than baseline models.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian J. Goodfellow, Aleksander Madry, and Alexey Kurakin. 2019. On evaluating adversarial robustness. *CoRR*, abs/1902.06705.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

Hanjie Chen and Yangfeng Ji. 2020. Learning variational word masks to improve the interpretability of neural text classifiers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4236–4251, Online. Association for Computational Linguistics.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *CoRR*, abs/1803.01128.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. 2021. Towards robustness against natural language word substitutions. In *International Conference on Learning Representations*.

Zhendong Dong, Qiang Dong, and Changling Hao. 2010. Hownet and its computation of meaning. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, COLING '10, page 53–56, USA. Association for Computational Linguistics.

Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue,

---

[8]The adversarial examples used were the same examples used to generate results for Table 6

Jonathan Uesato, and Pushmeet Kohli. 2018. Training verified learners with learned verifiers. *CoRR*, abs/1805.10265.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. In *ACL*.

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

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification.

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

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019. Achieving verified robustness to symbol substitutions via interval bound propagation. In *Proceedings of the 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.

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.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pretrained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust? natural language attack on text classification and entailment. *ArXiv*, abs/1907.11932.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *CoRR*, abs/1607.02533.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2021. Contextualized perturbation for textual adversarial attack.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert.

Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. 2020. Adversarial training for large neural language models.

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. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification.

John Morris, Eli Lifland, Jin Yong Yoo, and Yanjun Qi. 2020a. TextAttack: A framework for adversarial attacks in natural language processing. *ArXiv*, abs/2005.05909.

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

Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Lina Maria Rojas-Barahona, Pei hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2016. Counter-fitting word vectors to linguistic constraints. In *HLT-NAACL*.

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. SSMBA: Self-supervised manifold based data augmentation for improving out-of-domain robustness. In *Proceedings of the 2020 Conference on*

Empirical Methods in Natural Language Processing (EMNLP), pages 1268–1283, Online. Association for Computational Linguistics.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. In *Proc. of WMT*.

Dong Nguyen. 2018. Comparing automatic and human evaluation of local explanations for text classification. 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 1069–1078, New Orleans, Louisiana. 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.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938.

W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. Müller. 2017. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

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.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation. *CoRR*, abs/1904.12848.

Jin Yong Yoo, John X. Morris, Eli Lifland, and Yanjun Qi. 2020. Searching for a search method: Benchmarking search algorithms for generating nlp adversarial examples.

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*, pages 6066–6080, Online. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2019. Freelb: Enhanced adversarial training for language understanding. *CoRR*, abs/1909.11764.

A Appendix

A.1 `A2T` and `A2T-MLM` Attacks

Here, we give more details about `A2T` and `A2T-MLM`. We will use the framework introduced by (Morris et al., 2020a) to break down adversarial attacks into the following four components: (1) goal function, (2) transformation, (3) a set of constraints, (4) search method.

Goal Function In this work, we perform untargeted attack since they are generally easier than targeted attack. We aim to maximize the following as our goal function:

$$1 - P(y|\mathbf{x}; \theta) \tag{4}$$

where $P(y|\mathbf{x}; \theta)$ means the model's confidence of label y given input $\mathbf{x}$ and parameters θ.

Transformation

1. `A2T` Counter-fitted word embedding (Mrksic et al., 2016)

2. `A2T-MLM` : BERT masked language model (Devlin et al., 2018)

For both methods, we select the top 20 words proposed as replacements. This helps us narrow down our replacements to the best ones and save time from considering less desirable replacements.

Constraints We use the following constraints for both attacks:

- **Part-of-speech Consistency**: To preserve fluency, we require that the two words being swapped have the same part-of-speech. This is determined by a part-of-speech tagger provided by Flair (Akbik et al., 2018), an open-source NLP library.

- **DistilBERT Semantic Textual Similarity (STS)** (Sanh et al., 2019): We require that cosine similarity between the sentence encodings of original text $\mathbf{x}$ and perturbed text $\mathbf{x}'$ meet minimum threshold value of 0.9. We use fine-tuned DistilBERT model provided by Reimers and Gurevych (2019).

- **Max modification rate**: We allow only 10% of the words to be replaced. This limits us from modifying the text too much and causing the semantics of the text to change.

Algorithm 2 `A2T` 's Search Method: Gradient-based Word Importance Ranking

---

Require: Original text $\mathbf{x} = (x_1, x_2, ... x_n)$. Transformation module $T(\mathbf{x}, i)$ that perturbs $\mathbf{x}$ by replacing x_i.

Ensure: Adversarial text $\mathbf{x}_{adv}$ if found

1: Calculate $I(x_i)$ for all words x_i by making one forward and backward pass.
2: $R \leftarrow$ ranking $r_1, \ldots, r_n$ of words $x_1, \ldots, x_n$ by descending importance
3: $\mathbf{x}^* \leftarrow \mathbf{x}$
4: **for** $i = r_1, r_2, \ldots, r_n$ in R **do**
5:    $X_{cand} \leftarrow T(x^*, i)$
6:    **if** $X_{cand} \neq \emptyset$ **then**
7:       $\mathbf{x}^* \leftarrow \arg\max_{\mathbf{x}' \in X_{cand}} 1 - P(y|\mathbf{x}; \theta)$
8:       **if** $\mathbf{x}^*$ fools the model **then**
9:          **return** $\mathbf{x}^*$ as $\mathbf{x}_{adv}$
10:       **end if**
11:    **end if**
12: **end for**

Also, for `A2T` attack, we require that the word embeddings between original text $\mathbf{x}$ and perturbed text $\mathbf{x}'$ have minimum cosine similarity of 0.8.

The threshold values for word embedding similarity and sentence encoding similarity were set based on the recommendations by Morris et al. (2020b), which noted that high threshold values encourages strong semantic similarity between the original text and the perturbed text.

Search Method Search method is responsible for iteratively perturbing the original text $\mathbf{x}$ until we discover an adversarial example $\mathbf{x}_{adv}$ that causes the model to mispredict. Algorithm 2 shows `A2T` 's search algorithm. If the search method fails to find an adversarial example by the time its search is over, it has failed to generate one. It can also exit preemptively if it has reached maximum number of queries to the victim model. Such limit is called query budget.

During training, we limit the search method to making only 200 queries to the victim model for faster generation of adversarial examples. For evaluation using `A2T` and `A2T-MLM`, we increase the query budget to 2000 queries for a more extensive search. For other attacks such as TextFooler (Jin et al., 2019), BAE (Garg and Ramakrishnan, 2020), PWWS (Ren et al., 2019), and PSO (Zang et al., 2020), the query budget is set to 5000.