In Situ Augmentation for Defending Against Adversarial Attacks on Text Classifiers

Lei Xu
leix@mit.edu
MIT LIDS

Laure Berti-Equille
laure.berti@ird.fr
IRD

Alfredo Cuesta-Infante
alfredo.cuesta@urjc.es
Universidad Rey Juan Carlos

Kalyan Veeramachaneni
kalyanv@mit.edu
MIT LIDS

ABSTRACT

In text classification, recent research shows that adversarial attack methods can generate sentences that dramatically decrease the classification accuracy of state-of-the-art neural text classifiers. However, very few defense methods have been proposed against these generated high-quality adversarial sentences. In this paper, we propose LMAg (Language-Model-based Augmentation using Gradient Guidance), an in situ data augmentation method as a defense mechanism effective in two representative attack setups. Specifically, LMAg transforms input text during the test time. It uses the norm of the gradient to estimate the importance of a word to the classifier's prediction, then substitutes those words with alternatives proposed by a masked language model. LMAg is an additional protection layer on the classifier that counteracts the perturbations made by adversarial attack methods, thus can protect the classifier from adversarial attack without additional training. Experimental results show that LMAg can improve after-attack accuracy of BERT text classifier by 51.5% and 17.3% for two setups respectively.

ACM Reference Format:
Lei Xu, Laure Berti-Equille, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2022. In Situ Augmentation for Defending Against Adversarial Attacks on Text Classifiers . In *AdvML '22: Workshop on Adversarial Learning Methods for Machine Learning and Data Mining, August 15, 2022, Washington DC, USA*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the past few years, adversarial attack methods on text classifiers have been studied extensively. The goal of this type of attack is to rewrite a sentence such that a text classifier returns an incorrect prediction. Recently proposed attack methods can drastically decrease the accuracy of state-of-the-art classifiers: the adversarial sentences they generate are semantically similar to the original sentences and are of high grammatical quality making them had to detect and discriminate from original sentences.

AdvML '22, August 15, 2022, Washington DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

As adversarial attacks can effectively degrade the accuracy of a text classifier, defending against such attacks has become a natural need. The effort to defend against adversarial attacks on text classification mainly uses adversarial training. However, adversarial training of a text classifier is non-trivial for two reasons:

Efficiency requirement: A typical adversarial training process [14] augments each training minibatch with their adversarial counterpart, requiring the generation of adversarial examples efficiently. However, finding adversarial examples for a sentence is computationally expensive because it often involves heuristic search [8, 25] or inference of a neural language model [5, 10]. It is impractical to generate adversarial examples during training. To address this issue, researchers generate adversarial examples in advance and use a fix set of adversarial examples to tune and robustify the classifier [8]. This solution reduces the efficacy of adversarial learning, because when the classifier is improved, new adversarial examples are needed to further robustify the classifier.

Efficacy requirement: no consensus on the efficacy of adversarial training has been demonstrated yet [15]: some works (e.g., [8, 18]) showed that adversarial training is effective whereas others (e.g., [1]) showed it is not. Beyond the differences in the benchmark datasets, we will show that the efficacy of a defense method can be measured under two different setups, making the results hard to compare (See Section 3).

In this paper, we propose a method to defend against adversarial attacks by in situ augmentation – transforming the input sentence during inference – rather than tuning the classifier. Since most attack methods modify the sentence by substituting a small portion of words in the sentence, counteracting these substitutions is one intuitive idea to defend against attacks. We can assume that words modified by the attack methods tend to have a high impact on the classifier's prediction, thus tending to increase the gradient norm. By substituting these words, we can counteract the modifications made by the attacker. As such, this paper proposes language-model-based augmentation with gradient guidance (LMAg). In LMAg, we compute the gradient of the classifier's prediction with respect to the input word embeddings. We then use the gradient norm as a weight to randomly mask words in the sentence, and employ a BERT [3] language model to fill in masked words. Since LMAg is a data-augmentation method at test time, it does not need additional training of the classifier and is easier to deploy. Our experimental results show that the proposed method is effective in defending against various attacks.

2 RELATED WORK

Significant research has been done concerning adversarial attacks on text classifiers. Early works attempted to attack the classifier by injecting anomalies such as typos [4, 11]. One line of research [8, 25] uses synonym substitution to find adversarial sentences. Recent works including [5, 9, 10, 24] introduce a pre-trained language model in finding substitutions so that the adversarial sentences can be more fluent. [27] provides a comprehensive survey on existing attack methods. Adversarial attack libraries have also been developed [16, 26]. Adversarial training is an effective solution to protect classifiers from adversarial attacks in computer vision [14, 20]. So it's not surprising that similar defending approaches have been applied to text classification. Among the attack methods mentioned above, many [8, 10, 24, 25] use adversarial training to make the classifier resist the attacks. Adversarial training is also used in tasks such as reading comprehension [6, 23] and machine translation [2]. Jia et al. [7] proposes certified defense, but it can not be applied on transformer-based models. Wang et al. [22] proposes synonym encoding (SEM). SEM constructs a synonym dictionary, and maps a cluster of synonyms to the most frequent word in that cluster to offset the adversarial perturbation.

3 PROBLEM FORMULATION

In this section, we formulate the adversarial attack task and two defense setups.

*Definition 3.1 (***Efficacy of Adversarial Attack on Text Classification***)*. Given a sentence $\mathbf{x} = \{x_1, \ldots, x_l\}$ and its label y, a text classifier $f(\cdot)$ is supposed to make a prediction $\hat{y} = f(\mathbf{x})$ where $\hat{y} = y$ with high probability. When $f(\mathbf{x}) = y$, an adversarial attack method $\mathcal{A}(\mathbf{x}, y, f)$ generates an adversarial sentence $\mathbf{u}$ where $\mathbf{u}$ is grammatically correct and has the same semantic meaning as $\mathbf{x}$, but $f(\mathbf{u}) \neq y$. The efficacy of adversarial attack is measured by after attack accuracy on the test set $\mathcal{D}$ such as:

$$\mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}[f(\mathcal{A}(\mathbf{x}, y, f)) = y]. \tag{1}$$

As attack methods can successfully decrease the accuracy of a classifier, defending against these attacks is necessary. The goal of the defense is to make the classifier $f'(\cdot)$ more robust such that it retains high classification accuracy even when it is attacked with adversarial sentences. Note that there is no constraint on how $f'(\cdot)$ is constructed; it may be constructed either by tuning the classifier's parameters or by adding additional protections, such as adversarial sentence detection and/or text transformation.

*Definition 3.2 (***Efficacy of Original defense Against Adversarial Examples***)*. In this setup (Setup I), we generate adversarial examples by attacking the original classifier $f(\cdot)$, then we evaluate the robustness of the original classifier based on the absence of mistakes on these examples. In this setup, the after-attack accuracy on the test set $\mathcal{D}$ is defined as:

$$\mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}[f'(\mathcal{A}(\mathbf{x}, y, f)) = y]. \tag{2}$$

Several works [18, 22] follow this setup and show significant improvement in after-attack accuracy.

*Definition 3.3 (***Efficacy of Boosted defense Against Adversarial Examples***)*. In this setup (Setup II), we generate adversarial examples by attacking the robustified classifier $f'(\cdot)$. In this setup, the after-attack accuracy is defined as:

$$\mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}[f'(\mathcal{A}(\mathbf{x}, y, f')) = y]. \tag{3}$$

A few works [8, 25] following this setup show relatively lower efficacy in defense.

The difference between the two setups is whether the attacker is aware of the robustified classifier. We believe Setup II is prevailing in practice because: (1) Setup I underestimates the efficacy of attack methods. Most attack methods [5, 8, 25] stop early when an adversarial sentence is found, but this early stop only indicates that the algorithm has found an adversarial example against the original classifier. This adversarial sample may fail on the robustified classifier. But if the attack method directly attacks the boosted classifier and runs sufficient iterations, it may still find efficient adversarial examples; (2) Setup II is more realistic. When a robustified classifier is deployed, users interact with the robustified classifier rather than the original one. Thus, it is more likely that an attacker directly attacks the robustified classifier.

Algorithm 1: LMAg method.

Input: Sentence $\mathbf{x} = \{x_1, \ldots, x_l\}$; A classifier $f(\cdot)$ which includes the embedding layer $E(\cdot)$, and upper layers $g(\cdot)$ which takes embeddings and returns a probability distribution over classes; Number of rewrites λ; Mask ratio γ; Hyperparameter α.

Output: λ rewritten sentences.

```
results ← empty list;
e_1, ..., e_l ← E(x);
max_log_p = max_k g(e_1, ..., e_l)_k;
w_1, ..., w_l ← [∇_{e_i} max_log_p]_{i=1...l}
m ← max(1, ⌊l × γ⌋);
for i in 1 ... λ do
    x^(i) ← x;
    t_1, ..., t_m ~ Cat[w_1^α, ..., w_l^α];
    for j in 1 ... m do
        x^(i)_{t_j} ← MASK;
    end
    x̂^(i) ← [arg max BERT(x^(i))_j]_{j=1...l};
    results.append(x̂^(i));
end
return results
```

4 IN SITU DATA AUGMENTATION

In this section, we introduce LMAg, a in situ data augmentation to defend adversarial attacks. LMAg consists of three steps: (1) Estimate the importance of words using the gradient of the classifier; (2) Generate multiple rephrases by stochastically masking important words in the input sentence and filling in with alternative words using a masked language model; and (3) Make a prediction based on the majority of predictions on the rephrases. Algorithm 1 shows the process of generating the rephrases.

Name	Type	#C	Cased	Train/Test	Len	Orig	PWWS	TF	PSO	BA	BAE
AG	Topic	4	Y	120k/7.6k	54	92.2	29.9	9.9	20.8	17.9	73.6
MR	Sentiment	2	N	9k/1k	24	88.1	18.4	9.4	7.5	13.8	37.0
Yelp	Sentiment	2	Y	160k/38k	182	96.5	3.7	4.3		9.0	50.5
IMDB	Sentiment	2	Y	25k/25k	305	89.8	10.0	6.3		18.2	46.1
SST2	Sentiment	2	Y	67k / 0.9k	54	92.4	14.7	7.5	8.1	20.8	38.6

Table 1: Dataset details. #C means number of classes. Cased means whether the dataset is cased or uncased. Len is the number of BERT word-pieces in a sentence. Orig shows the accuracy of the original BERT-base classifier. PWWS, TF, PSO, BA, BAE shows the after-attack accuracy on the original classifier using the corresponding attack method.

4.1 Estimate Word Importance using Gradients

Gradient information has been widely used in attack methods. In white-box settings where attackers have full access to the classifier, gradient is directly used to pick candidate substitution [11], whereas in black box settings, gradient is approximated by comparing the classifier's output with or without a word [10]. When building a defense, we assume that we have full access to the classifier; thus we directly compute gradients to identify important words that contribute the most to the classification. We split a text classifier into two components:

$$f(\mathbf{x}) = \arg\max_k g(E(\mathbf{x}))_k,$$

where $E(\mathbf{x}) = \mathbf{e}_1, \ldots, \mathbf{e}_l$ is the input embedding layer that converts the words x_i into embeddings $\mathbf{e}_i$, and $g(\cdot)$ is the upper layers that made prediction from word embeddings. The output of $g(\cdot)$ is a probability distribution over all classes. We use $g()_k$ to denote the probability of k-th class. We compute the importance weight of each word by

$$w_i = ||\nabla_{\mathbf{e}_i} \log \max_k g(E(\mathbf{x}))_k||_2.$$

For transformer-based models, $\mathbf{e}_i$ denotes the sum of word embedding, position embedding and token type embedding.

4.2 Stochastic Multiple Rephrasing

After calculating the importance weight of each word, we have to replace the important words, hoping to counteract the adversarial attack. However, if we threshold the importance weight then mask and substitute words, it is possible to mask all important words and make the sentence generated by the language model semantically different from the original sentence. For example, in sentiment analysis, if we mask all the adjectives that express sentiment, then the language model may generate a sentence with the opposite sentiment. To overcome this problem, we used a stochastic substitute method.

We randomly sample $m = \lfloor l \times \gamma \rfloor$ positions in the sentence using w_i as weights, where γ is the masking ratio. Specifically we sample positions: $t_1, \ldots, t_m \sim \text{Cat}(w_1^\alpha, \ldots, w_l^\alpha)$, where Cat means a multinomial distribution, and α is a hyperparameter. Then we replace these positions with a special `MASK` token and use BERT language model to impute the most likely sentence as

$$\hat{\mathbf{x}} = [\arg\max \text{BERT}(\mathbf{x})_i]_{i=1\ldots l},$$

where $\text{BERT}(\mathbf{x})$ is a BERT language model. Note that all l words in the rephrase $\hat{\mathbf{x}}$ are proposed by the BERT language model, although only mask m words are masked. $\hat{\mathbf{x}}$ may have more than m word substitutions.

Different mask positions result in different rephrases. To make the classifier more stable, we generate λ sentences for each adversarial sentence by selecting different mask positions. We then take the majority predictions of λ sentences as the prediction for the input sentence.

5 EXPERIMENTS

In this section, we compare the efficacy of LMAg with baselines under two setups discussed in Section 3.

• **Datasets**. We use 5 text classification datasets: (1) **AG**'s News [1]; (2) Movie Reviews (**MR**) [17]; (3) **Yelp** Reviews [28]; (4) **IMDB** Movie Reviews [13]; and the binary classification variation [21] of Stanford Sentiment Treebank v2 (**SST2**) [19].

• **Original Classifier**. For all datasets, we use the BERT-base classifier [3] (#layers=12, hidden_size=768). We fine-tune the classifier on 20k batches (5k batches on MR and IMDB), with batch size 32. We use the AdamW optimizer [12] and learning rate 0.00002.

• **Attack Methods:** We pick 5 recently proposed adversarial attack methods implemented in TextAttack [16]: (1) Ren et al. [18] proposes the probability weighted word saliency (**PWWS**), which determines the synonym substitution using both the word saliency and the classification probability; (2) TextFooler [8] (**TF**) is a synonym substitution algorithm with semantic similarity checker and part-of-speech checker; (3) BERT-ATTACK [10] (**BA**) and (4) **BAE** [5] both use BERT language models to propose word substitutions; and (5) SememePSO [25] (**PSO**) substitutes words based on sememes – the minimum semantic units, and uses particle swarm optimization.

Details of the datasets, the performance of the original classifier, and the after-attack accuracy of attack methods on the original classifier are shown in Table 1.

• **Baselines**: We compare our method with 2 baselines: (1) **SEM** [22]: we follow the hyper-parameters recommended by authors. We convert the training data using SEM and train the classifier using the same convention as the original classifier mentioned above; and (2) Adversarial training (**AT**): we sample 10k sentences from each of the training set, then use TF[2] to attack the original classifier with these sentences. We then merge the generated adversarial sentences with the original training set, then fine-tune the original classifier for another 5k batches. For each training batch, we sample

[1] http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

[2] We use TF in adversarial training because of its efficiency and attack efficacy.

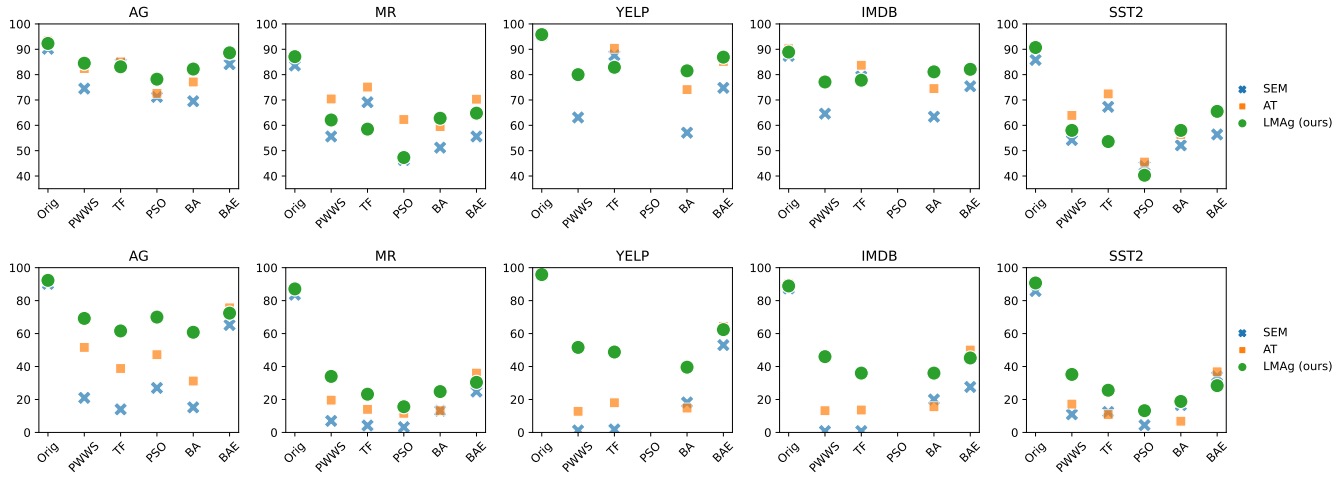

Figure 1: After-attack accuracy of the classifier ($\times 10^{-2}$) for each adversarial method (X-axis) on both setups: Setup I (top) – The adversarial examples are generated to attack the original classifier on the original test set (Orig); Setup II (bottom) – The adversarial examples are generated to attack the robustified classifier.

half of the sentences from the original training set, and the other half from the set of adversarial sentences.

Our Method: For LMAg, we set the number of rephrases $\lambda = 10$, the mask ratio $\gamma = 0.2$, and $\alpha = 0.6$. We fine-tune the BERT language model on the training set for 5000 steps with batch size 32 and learning rate 0.00002.

5.1 Experimental Results

Figure 1 (top) shows the performance of Setup I for attacks on the original classifier. In this setup, all the methods including ours successfully defend against a large portion of adversarial examples, and improve the accuracy by more than 45%. Our LMAg improves the accuracy by 51.5% in average while AT performs slightly better with an improvement of 53.7%. LMAg and AT cause slight decrease in accuracy on the original test set compared to SEM.

Figure 1 (bottom) shows the performance of Setup II. The after attack accuracy is significantly lower than in Setup I, showing that this setup is more challenging. SEM has a negative impact on the model whereas AT slightly improves the after-attack accuracy by 6.6%. LMAg can improve the after-attack accuracy by 17.3% in average which is significantly better than the other two baselines. Furthermore, LMAg achieves the best improvement on all 5 datasets and 4 out of 5 attack methods.

6 CONCLUSION

In this paper, we laid out two different setups in defending adversarial attack, namely (1) original defense and (2) boosted defense against adversarial examples. We show that the latter is both more realistic and more challenging. We introduce LMAg, a novel in situ augmentation to defend adversarial attacks on text classifiers. LMAg achieves comparable performance on the first setup and significantly better performance on the second one. Since LMAg is a in situ data transformation. It does not change the architecture of the classifier, so it can be easily integrated with other defense methods. Although we improved the after-attack accuracy by 17.3%, the problem of defending adversarial attack is far from being solved. In the future, we will attempt to further improve the defense method by integrating LMAg with other methods, and meanwhile try to improve the efficiency.

ACKNOWLEDGMENTS

Dr. Cuesta-Infante is funded by the Spanish Government research fundings RTI2018-098743-B-I00 (MICINN/FEDER), PID2021-128362OB-I00 (MICINN/FEDER) and Y2018/EMT-5062 (Comunidad de Madrid).

REFERENCES

[1] 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*.

[2] Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and Natural Noise Both Break Neural Machine Translation. In *International Conference on Learning Representations*.

[3] 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 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

[4] 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)*.

[5] Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based Adversarial Examples for Text Classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*.

[6] 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*.

[7] Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified Robustness to Adversarial Word Substitutions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*.

[8] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT Really Robust? Natural Language Attack on Text Classification and Entailment. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[9] Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and William B Dolan. 2021. Contextualized Perturbation for Textual Adversarial Attack. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 5053–5069.

[10] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*.

[11] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2017. Deep text classification can be fooled. In *Proceedings of the International Joint Conferences on Artificial Intelligence*.

[12] Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations*.

[13] 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 Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

[14] 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*.

[15] John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020. Reevaluating Adversarial Examples in Natural Language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*.

[16] John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

[17] Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[18] 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*. 1085–1097.

[19] 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*.

[20] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble Adversarial Training: Attacks and Defenses. In *International Conference on Learning Representations*.

[21] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

[22] Xiaosen Wang, Hao Jin, Yichen Yang, and Kun He. 2021. Natural Language Adversarial Defense through Synonym Encoding. In *The Conference on Uncertainty in Artificial Intelligence*.

[23] Yicheng Wang and Mohit Bansal. 2018. Robust Machine Comprehension Models via Adversarial Training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

[24] Lei Xu and Kalyan Veeramachaneni. 2021. Attacking Text Classifiers via Sentence Rewriting Sampler. *arXiv preprint arXiv:2104.08453* (2021).

[25] 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*.

[26] Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2021. Openattack: An open-source textual adversarial attack toolkit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (Demo)*.

[27] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)* (2020).

[28] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*.

A ILLUSTRATIVE EXAMPLES

Hereafter, Table 2 gives a few examples of using LMAg to correct the prediction of adversarial sentences. In the table, Orig and Adv indicate the original sentence and the adversarial sentence found by TextFooler respectively. Positive or negative in the parenthesis is the prediction of the original classifier. The 3rd row visualizes w_i at BERT's word-piece level. We **boldface** 5 word-pieces with the largest weights and underlines 5 word-pieces with the second largest weights. The following five rows show 5 rephrases of the adversarial sentence generated by LMAg. We **boldface** the masked word-pieces. Note that LMAg may change unmasked words. In both examples, the classifier's prediction is corrected.

B EFFECT OF HYPERPARAMETERS

We further evaluate the effect of three hyperparameters of LMAg, namely the number of rephrases λ, the mask ratio γ, and α. We tune one hyperparameter with the other two fixed. The results are demonstrated in Figure 2.

- We measure $\lambda = 1, 5, 10, 20$ when $\gamma = 0.2$ and $\alpha = 0.6$. We observe that when increasing k from 1 to 10, the accuracy on the original test set increases significantly. When $k = 1$, the accuracy on the original test set decreases as much as 6% on the SST2 data set. We interpret it as when 20% of the words are covered, the language model may generate a sentence whose label is different from the original sentence, which causes a significant drop in accuracy on the original test set. Using multiple rewrites can alleviate this problem. We also observe that the after attack accuracy improves on 3 out of 5 datasets when λ increases.
- We measure $\gamma = 0.05, 0.1, 0.15, 0.2, 0.25$ while $\lambda = 10$ and $\alpha = 0.6$. We observe that masking more words leads to more improvement on after-attack accuracy but leading to lower accuracy on the original test set. When masking more words, it's harder for the language model to rephrase the sentence and retain the same label; meanwhile it is more likely to counteract adversarial modifications.
- We measure $\alpha = 0.0, 0.2, 0.4, 0.6, 0.8, 1.0$ with $\lambda = 10$ and $\gamma = 0.2$. Note that when $\alpha = 0$, the mask positions are sampled uniformly. We observe that larger α leads to higher after-attack accuracy but lower accuracy on the original test set. The reason for this is that when α becomes larger, the probability distribution of the selected position becomes sparser. Some positions have a high probability of being masked while others are hardly masked. In this case, the same masking positions may be selected for multiple rewrites, which is similar to setting a smaller λ.

C DISCUSSION

Although LMAg can effectively defend against adversarial attacks on text classifiers, the protection comes with a yet high computation cost. The original text classifier runs one forward pass to get one prediction. LMAg needs 1 forward and backward pass to estimate w_i. Then λ forward passes to rephrase the sentence, and λ forward passes to get predictions for each rephrase sentence. Thus, LMAg is $(2\lambda + 2)\times$ slower than the original BERT-based classifier. Even if we use parallelization on GPU, we still observe $4.4x$ slow down when $\lambda = 10$.

Orig (Negative)	without shakespeare's eloquent language , the update is dreary and sluggish .
Adv (Positive)	without shakespeare's eloquent **dialect** , the **refreshing** is **sorrowful** and **unmotivated** .
Visualize w_i	without shakespeare ' s el ##o ##quent dialect , the **refreshing** is sorrow ##ful and **un ##mot ##ivated .**
R1 (Negative)	without shakespeare ' s eloquent **wit** , the film is sorrowful and **un**moti**onal** .
R2 (Positive)	**like** shakespeare ' s eloquent **plays ,** the film is sorrowful and unmotivated .
R3 (Negative)	without shakespeare ' s eloquent wit , the film**ly** sorrow**ful** and **un**motivated .
R4 (Negative)	without shakespeare ' s **el**oquent wit , the film is sorrow**ful** and unmoti**vated** .
R5 (Negative)	without shakespeare **'** s elo**quent**ism , **miss** film is sorrowful and unmotivated .
Orig (Positive)	compelling revenge thriller , though somewhat weakened by a miscast leading lady .
Adv (Negative)	**cogent** revenge thriller , though somewhat weakened by a miscast leading lady .
Visualize w_i	co **##gent revenge thriller** , though somewhat **weakened** by a mis **##cast** leading lady .
R1 (Positive)	co**hesive** revenge thriller , though somewhat overshadowed by **its** miscast leading lady **.**
R2 (Negative)	cogent revenge thriller , **playedly** performance by **a** miscast leading lady .
R3 (Positive)	**a entertaining entertaining** thriller , though somewhat hampered by a miscast leading lady .
R4 (Negative)	cogent revenge **thriller** , **only** somewhat hampered by a miscast leading **man** .
R5 (Positive)	**a entertaining** revenge thriller , though somewhat **hampered** by a miscast leading lady .

Table 2: Two adversarial sentences and their rephrases generated by LMAg.

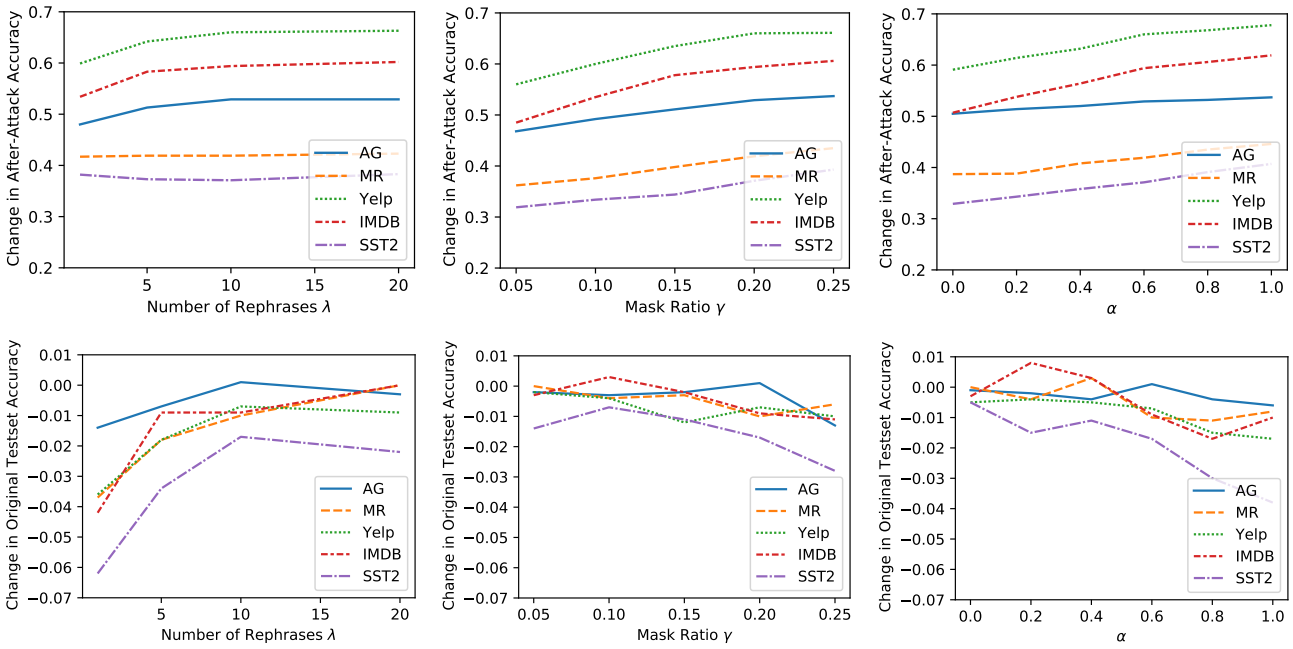

Figure 2: The effect of hyperparameters. The upper row shows the change of after-attack accuracy on each data set, the lower row shows the change of original test set accuracy.