In Situ Augmentation for Defending Against Adversarial Attacks on Text Classifiers

Lei Xu[1], Laure Berti-Equille[2], Alfredo Cuesta-Infante[3], and Kalyan Veeramachaneni[1]✉

[1] Massachusetts Institute of Technology, USA
[2] Institute of Research for Development (IRD), France
[3] Universidad Rey Juan Carlos, Spain
leix@mit.edu, laure.berti@ird.fr, alfredo.cuesta@urjc.es, kalyanv@mit.edu

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 defense 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 replaces 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.

Keywords: Adversarial Robustness · Text Classification · Data Augmentation.

1 Introduction

In the past few years, adversarial attack methods on text classifiers have been studied extensively [8, 25, 16, 26]. 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 hard to detect and discriminate from original sentences [10, 5].

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 [8, 25]. However, adversarial training of a text classifier has several issues. First, adversarial training are limited by the existing attack methods. Once a better attack method emerges, the classifier needs to be retrained to defend against the new attack, creating extra burden for developers. Second, finding an adversarial sentence for a text classifier is inefficient and can take a few seconds because it often involves heuristic search [25, 8] or inference of a neural language model [10, 5]. Therefore, adversarial training of a classifier for 10k steps can take a few days, which prevents developers from efficiently deploying classifiers. To address this issue, instead of generating adversarial sentences during training, some work [8] generates a fixed set of adversarial sentences in advance and uses them to tune the classifier. This solution reduces the efficacy of adversarial training, because when the classifier is improved, new adversarial examples are needed to further robustify the classifier. Third, 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 simple and elegant 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 replacing 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 attempt to 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 at a cost of slightly increasing inference time.

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 [11, 4]. One line of research [25, 8] uses synonym substitution to find adversarial sentences. Recent works including [10, 5, 24, 9] 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 [20, 14]. So it's not surprising that similar defending approaches have been applied to text classification. Among the attack methods mentioned above, many [8, 10, 25, 24] 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]. [7] proposes certified defense, but it can not be applied on transformer-based models. [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.
Adversarial Attack on Text Classification. Given a sentence $\mathbf{x} = x_1, \ldots, x_l$ and its label y where l is the length of the sentence, 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 (AAcc)** on the test set $\mathcal{D}$ such as: $\mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}[f(\mathcal{A}(\mathbf{x}, y, f)) = y]$.

As attack methods can successfully decrease the accuracy of a classifier, defending against these attacks is necessary. The goal of the defense is to construct a classifier $f'(\cdot)$ 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.
Setup I: Efficacy of Original Defense Against Adversarial Examples. We generate adversarial examples by attacking the original classifier $f(\cdot)$, then we evaluate the robustness of the robusified classifier $f'(\cdot)$ based on the absence of misclassification on these examples. In this setup, the AAcc is defined as: $\mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}[f'(\mathcal{A}(\mathbf{x}, y, f)) = y]$. Several works [18, 22] follow this setup and show significant improvement in after-attack accuracy.
Setup II: Efficacy of Boosted Defense Against Adversarial Examples. We generate adversarial examples by attacking the robustified classifier $f'(\cdot)$. In this setup, the AAcc is defined as: $\mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}[f'(\mathcal{A}(\mathbf{x}, y, f')) = y]$. A few works [8, 25] following this setup show relatively lower efficacy in defense.

The difference between the two setups is whether the adversarial examples are generated on the original classifier or the robusified classifier. We believe Setup II is prevailing in practice because Setup I underestimates the efficacy of attack methods. Most attack methods [8, 25, 5] 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 sentence may fail on the robustified classifier. But if the attack method directly attacks the robustified classifier and runs sufficient iterations, it may still find

efficient adversarial examples. Also, 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.

4 In Situ Data Augmentation

In this section, we introduce LMAg, an 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. Figure 1 illustrates the procedure to generate one rephrase. Algorithm 1 shows the pseudo code to generate rephrases.

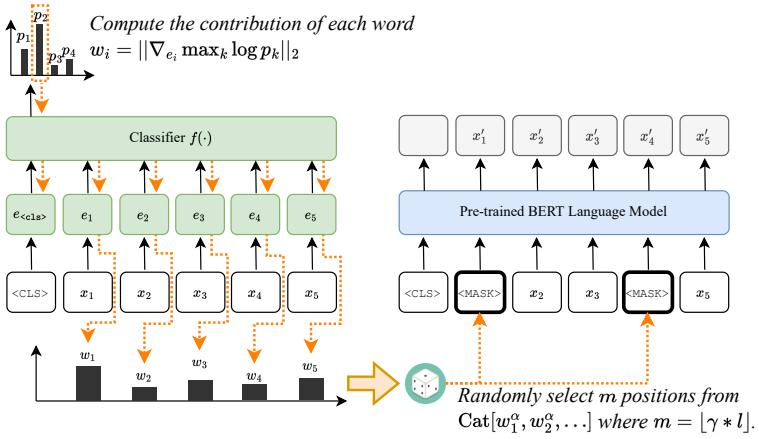

Fig. 1. Rephrase an input sentence using LMAg. The input sentence is forward and backward propagated through the classifier to compute the importance of each word. Then the input sentence is masked according to the importance weights, and a BERT language model is used to generate a rephrase.

4.1 Estimate Importance of Words 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 substitutions [11], whereas in black box settings, gradient is approximated by comparing the classifier's output with or without a word [10].

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
```

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. Let $\mathbf{x} = x_1, \ldots, x_l$ be an input sentence of length l. 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. For transformer-based models, $\mathbf{e}_i$ denotes the sum of word embedding, position embedding and token type embedding. 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 find the log probability of the most likely class predicted by the classifier (i.e., $\log \max_k g(E(\mathbf{x}))_k$), then compute the L2 norm of the gradient with respect to the input embeddings to capture the importance weight of words. Specifically, the importance weight of i-th word is

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

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. $t_1, \ldots, t_m$ represents the positions being masked in the sentence. They are sampled from the multinomial distribution without replacement. The hyperparameter α is a smoothing factor. When $\alpha = 0$, the probability density for each position being masked is uniform. When $\alpha \to \infty$, only top-m most important words will be masked. 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.

Applying LMAg leads to an increase in inference time, because it introduces extra computation of 1 backward propagation of the classifier, and λ forward propagation of BERT language model. However, the λ forward propagation can be parallelized on a GPU. So the inference time does not increase much.

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 [28]; (2) Movie Reviews (**MR**) [17]; (3) **Yelp** Reviews [28]; (4) **IMDB** Movie Reviews [13]; and the binary 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.

Metrics: We measure clean accuracy (**CAcc**) – the accuracy of the classifier on the original test, and after-attack accuracy (**AAcc**) – the accuracy of the classifier after being adversarially attacked by an attack method.

Name	Type	#C	Train/Test	Len	CAcc	PWWS	TF	PSO	BA	BAE
AG	Topic	4	120k/7.6k	54	92.2	29.9	9.9	20.8	17.9	73.6
MR	Sentiment	2	9k/1k	24	88.1	18.4	9.4	7.5	13.8	37.0
Yelp	Sentiment	2	160k/38k	182	96.5	3.7	4.3	NA[4]	9.0	50.5
IMDB	Sentiment	2	25k/25k	305	89.8	10.0	6.3	NA[4]	18.2	46.1
SST2	Sentiment	2	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. Len is the average number of BERT word-pieces in a sentence. PWWS, TF, PSO, BA, BAE shows the AAcc(%) on the original classifier using the corresponding attack method.

Attack Methods: We pick 5 recently proposed adversarial attack methods implemented in TextAttack [16]: (1) [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**[4]) substitutes words based on sememes – the minimum semantic units, and uses particle swarm optimization.

Details of the datasets, the CAcc, and the AAcc of the original classifier against attack methods are shown in Table 1.

Baselines: We compare our method with 2 baseline defense methods: (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 to attack the original classifier with these sentences. We use TF in adversarial training because of its efficiency and attack efficacy. 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 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

Does applying defense methods reduce the CAcc? Fig. 2 shows the CAcc of classifiers after applying a defense. AT and LMAg both cause slight decrease in CAcc in most cases. SEM results in a greater decrease in CAcc.

Are defense methods effective on Setup I – original defense? The translucent (taller) bars in Fig. 3 show the AAcc for this setup. All the methods

[4] We fail to attack Yelp and IMDB datasets with PSO because it is inefficient on long sentences.

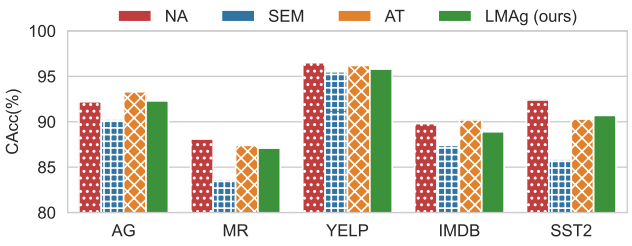

Fig. 2. CAcc of the classifier after applying defense methods. NA means the original classifier.

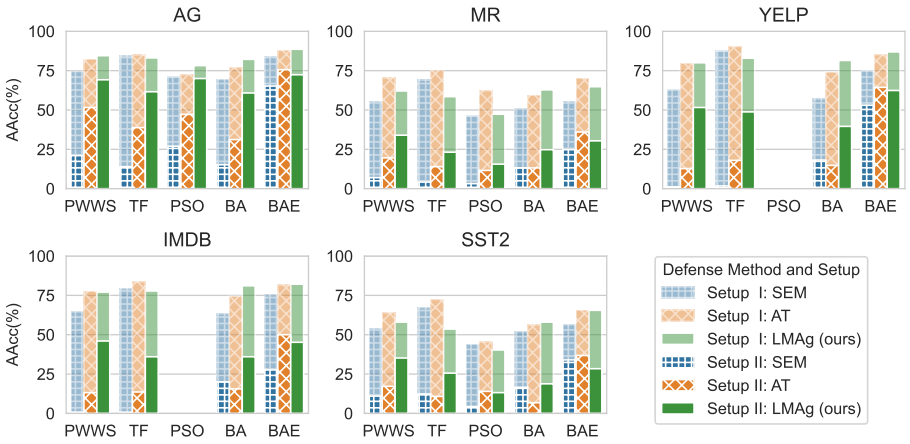

Fig. 3. AAcc of the classifier for each adversarial method (X-axis) on both setups. The translucent (taller) bars represent the AAcc of setup I – original defense. The solid (short) bars represent the AAcc of setup II – boosted defense.

including ours successfully defend against a large portion of adversarial examples, and improve AAcc by more than 45% compared to the original classifier. Our LMAg improves AAcc by 51.5% on average while AT performs slightly better with an improvement of 53.7%.

Are defense methods effective on Setup II – boosted defense? The solid (shorter) bars in Fig. 3 show the AAcc for this setup. The AAcc is significantly lower than in Setup I, showing that this setup is more challenging. When the attack methods get access to the robustified classifiers and run more iterations, they can still find adversarial sentences. SEM does not improve AAcc whereas AT slightly improves the AAcc by 6.6% compared to the original classifier. LMAg can improve the AAcc by 17.3% on 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.

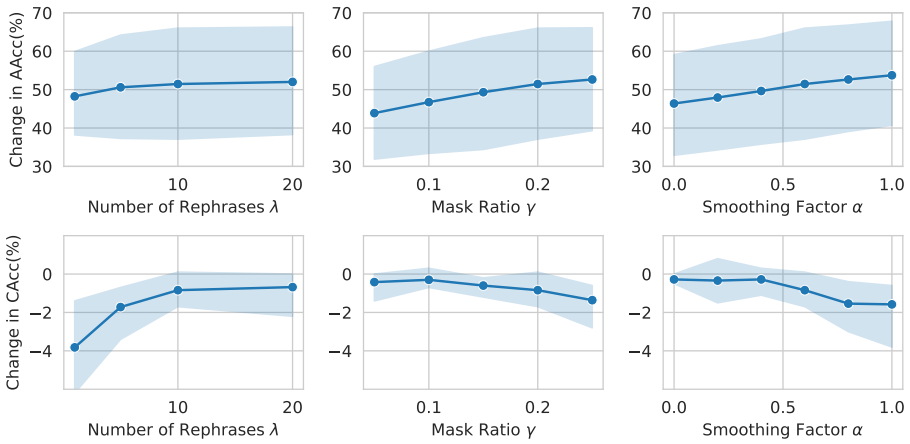

Fig. 4. The effect of hyperparameters on Setup I. The upper row shows the change of AAcc, the lower row shows the change of CAcc. The line in each figure shows the change of AAcc or CAcc averaged over all 5 datasets. The colored band shows the maximum and minimum change of 5 datasets.

5.2 Effect of Hyperparameters

We further evaluate the effect of three hyperparameters of LMAg, namely the number of rephrases λ, the mask ratio γ, and smoothing factor α. We tune one hyperparameter with the other two fixed. The results are demonstrated in Fig. 4.

Effect of number of rephrases λ. 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 CAcc on the original test set increases significantly. When $k = 1$, the CAcc decreases as much as 6%. 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 CAcc. Using multiple rewrites can alleviate this problem. We also observe that the average AAcc improves when λ increases.

Effect of mask ratio γ. 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 AAcc but leading to lower CAcc. 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.

Effect of smoothing factor α. 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 AAcc but lower CAcc. 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.

5.3 Illustrative Examples

Table 2 gives a few examples of using LMAg to correct the prediction of adversarial sentences.

Ori (Neg)	without shakespeare's eloquent language , the update is dreary and sluggish .
Adv (Pos)	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 (Neg)	without shakespeare ' s eloquent **wit** , the film is sorrowful and **un**mot**ional** .
R2 (Pos)	**like** shakespeare ' s eloquent **plays ,** the film is sorrowful and unmotivated .
R3 (Neg)	without shakespeare ' s eloquent wit , the film**ly** sorrow**ful** and **un**motivated .
R4 (Neg)	without shakespeare ' s **el**oquent wit , the film is sorrow**ful** and unmot**ivated** .
R5 (Neg)	without shakespeare **'** s elo**quent**ism , **miss** film is sorrowful and unmotivated .
Ori (Pos)	compelling revenge thriller , though somewhat weakened by a miscast leading lady .
Adv (Neg)	**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 (Pos)	co**hesive** revenge thriller , though somewhat overshadowed by **its** miscast leading lady **.**
R2 (Neg)	cogent revenge thriller , **playedly** performance by **a** miscast leading lady .
R3 (Pos)	**a entertaining entertaining** thriller , though somewhat hampered by a miscast leading lady .
R4 (Neg)	cogent revenge **thriller** , **only** somewhat hampered by a miscast leading **man** .
R5 (Pos)	**a entertaining** revenge thriller , though somewhat **hampered** by a miscast leading lady .

Table 2. Two adversarial sentences and their rephrases generated by LMAg. Ori and Adv indicate the original sentence and the adversarial sentence found by TextFooler respectively. Pos or Neg means the positive or negative sentiment predicted by 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.

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 Setup I and significantly better performance on Setup II. Since LMAg is an 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.

Acknowledgment

Alfredo Cuesta-Infante has been funded by the Spanish Government research project MICINN PID2021-128362OB-I00.

References

1. Alzantot, M., Sharma, Y., Elgohary, A., Ho, B.J., Srivastava, M., Chang, K.W.: Generating natural language adversarial examples. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (2018)
2. Belinkov, Y., Bisk, Y.: Synthetic and natural noise both break neural machine translation. In: International Conference on Learning Representations (2018)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: 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 (2019)
4. Ebrahimi, J., Rao, A., Lowd, D., Dou, D.: 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) (2018)
5. Garg, S., Ramakrishnan, G.: 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 (2020)
6. Jia, R., Liang, P.: Adversarial examples for evaluating reading comprehension systems. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (2017)
7. Jia, R., Raghunathan, A., Göksel, K., Liang, P.: 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 (2019)
8. Jin, D., Jin, Z., Zhou, J.T., Szolovits, P.: Is bert really robust? natural language attack on text classification and entailment. In: Proceedings of the AAAI Conference on Artificial Intelligence (2020)
9. Li, D., Zhang, Y., Peng, H., Chen, L., Brockett, C., Sun, M.T., Dolan, W.B.: 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. pp. 5053–5069 (2021)
10. Li, L., Ma, R., Guo, Q., Xue, X., Qiu, X.: 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 (2020)
11. Liang, B., Li, H., Su, M., Bian, P., Li, X., Shi, W.: Deep text classification can be fooled. In: Proceedings of the International Joint Conferences on Artificial Intelligence (2017)
12. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: Proceedings of the International Conference on Learning Representations (2019)
13. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (2011)

14. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: International Conference on Learning Representations (2018)
15. Morris, J., Lifland, E., Lanchantin, J., Ji, Y., Qi, Y.: Reevaluating adversarial examples in natural language. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings (2020)
16. Morris, J., Lifland, E., Yoo, J.Y., Grigsby, J., Jin, D., Qi, Y.: 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 (2020)
17. Pang, B., Lee, L.: 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 (2005)
18. Ren, S., Deng, Y., He, K., Che, W.: Generating natural language adversarial examples through probability weighted word saliency. In: Proceedings of the 57th annual meeting of the association for computational linguistics. pp. 1085–1097 (2019)
19. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (2013)
20. Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., McDaniel, P.: Ensemble adversarial training: Attacks and defenses. In: International Conference on Learning Representations (2018)
21. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.: Glue: A multitask benchmark and analysis platform for natural language understanding. In: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP (2018)
22. Wang, X., Jin, H., Yang, Y., He, K.: Natural language adversarial defense through synonym encoding. In: The Conference on Uncertainty in Artificial Intelligence (2021)
23. Wang, Y., Bansal, M.: 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) (2018)
24. Xu, L., Veeramachaneni, K.: Attacking text classifiers via sentence rewriting sampler. arXiv preprint arXiv:2104.08453 (2021)
25. Zang, Y., Qi, F., Yang, C., Liu, Z., Zhang, M., Liu, Q., Sun, M.: Word-level textual adversarial attacking as combinatorial optimization. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (2020)
26. Zeng, G., Qi, F., Zhou, Q., Zhang, T., Hou, B., Zang, Y., Liu, Z., Sun, M.: Openattack: An open-source textual adversarial attack toolkit. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (Demo) (2021)
27. Zhang, W.E., Sheng, Q.Z., Alhazmi, A., Li, C.: Adversarial attacks on deep-learning models in natural language processing: A survey. ACM Transactions on Intelligent Systems and Technology (TIST) (2020)
28. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Proceedings of the Conference on Advances in Neural Information Processing Systems (2015)