Robustness Evaluation of Text Classification Models Using Mathematical Optimization and Its Application to Adversarial Training

Hikaru Tomonari[1], **Masaaki Nishino**[2], **Akihiro Yamamoto**[1]

[1]Kyoto University
[2]NTT Communication Science Laboratories
tomonari@iip.ist.i.kyoto-u.ac.jp, masaaki.nishino.uh@hco.ntt.co.jp
yamamoto.akihiro.5m@kyoto-u.ac.jp

Abstract

Neural networks are known to be vulnerable to adversarial examples due to slightly perturbed input data. In practical applications of neural network models, the robustness of the models against perturbations must be evaluated. However, no method can strictly evaluate their robustness in natural language domains. We therefore propose a method that evaluates the robustness of text classification models using an integer linear programming (ILP) solver by an optimization problem that identifies a minimum synonym swap that changes the classification result. Our method allows us to compare the robustness of various models in realistic time. It can also be used for obtaining adversarial examples. Because of the minimal impact on the altered sentences, adversarial examples with our method obtained high scores in human evaluations of grammatical correctness and semantic similarity for an IMDb dataset. In addition, we implemented adversarial training with the IMDb and SST2 datasets and found that our adversarial training method makes the model robust.

1 Introduction

Over the last decade, neural network (NN) models have been widely applied in such fields as computer vision and natural language processing (NLP). However, recently they have been shown to be vulnerable to small and imperceptible perturbations included in the original input data (Szegedy et al., 2013). These altered input data called adversarial examples are correctly classified by humans but can fool a target model, raising serious security and reliability concerns.

An NN model is defined as being robust when the model's prediction does not change with the addition of all the perturbations in a certain range. The process of checking whether the model is robust is called verification (Katz et al., 2017; Tjeng and Tedrake, 2017). In computer vision, methods have formulated the verification problem as constraint satisfaction and verified it by solving it using an integer linear programming (ILP) solver (Katz et al., 2017; Tjeng and Tedrake, 2017) or a boolean satisfiability problem (SAT) solver (Narodytska et al., 2018).

Interest has also been growing in investigating the adversarial robustness of NLP models, including new methods for generating adversarial examples (Alzantot et al., 2018a; Jin et al., 2019; Alzantot et al., 2018b; Michel et al., 2019; Li et al., 2019; Ebrahimi et al., 2017; Zang et al., 2020; Pruthi et al., 2019). On the other hand, as long as models are evaluated only by heuristic attacks, we cannot guarantee a model's robustness.

To tackle this dilemma, we formulated a problem for finding the minimum number of word swaps that change a model's predictions and solving it with an ILP solver. Using this verification method, we can strictly compare multiple adversarial training methods .

In our experiments, we trained an NN model composed of an affine transformation and a piecewise-linear function, such as the ReLU function for the Internet Movie Database (IMDb) dataset (Maas et al., 2011) and the Stanford Sentiment Treebank v2 (SST2) dataset (Socher et al., 2013). Then we verified the models with an ILP solver in a few seconds per text. Human participants also manually evaluated whether the adversarial examples generated by the existing and proposed methods were grammatically correct and semantically unchanged from the original sentences. The adversarial examples created by the proposed method had higher scores than those created by the existing method.

In addition, we conducted an experiment on adversarial training. Adversarial training, which augments training data with adversarial examples in each training loop very effectively make deep learning models more robust against adversarial examples (Goodfellow et al., 2014). In our experiments,

Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022, pages 327–333
November 20–23, 2022.

our proposed method achieved robust model training.

2 Related Work

Some methods can accurately verify a model's robustness for perturbations within a certain range (Tjeng and Tedrake, 2017; Narodytska et al., 2018) in computer vision. MIPVerify (Tjeng and Tedrake, 2017) uses an ILP solver that can be applied to piecewise-linear neural network models by assigning binary variables to each nonlinear function. Another method verifies the parameters of NN models and input images with a binary neural network using the SAT solver (Narodytska et al., 2018). Unfortunately, there was no method can strictly evaluate the robustness of the models in the NLP domain.

Although pixel noise has been defined as an adversarial perturbation for images, it is difficult to define noise for text due to its discrete nature. TextFooler (Jin et al., 2019) generates an adversarial example of natural language by replacing some of the words in a text with synonyms. However, its heuristic search is approximate, and performs more word swaps than necessary. Our method, which obtains exact minimum word swaps, allows us to evaluate the robustness of a model.

3 Method

Given text sequence $\mathbf{x} = (x_1, \cdots, x_T)$, B_i denotes the candidates of the swapping word of x_i. We define b_{ij} as an indicator function. $b_{ij} = 1$ means that x_i is replaced by the j^{th} word of B_i, and no word swaps occur when $b_{i0} = 1$. Let $\mathbf{v}_{ij}$ denote a word-embedding vector corresponding to b_{ij}. Then the Minimum Adversarial Swapping Problem is described as follows:

$$\min \quad \sum_{i=1}^{T} \sum_{j=1}^{|B_i|} b_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j=0}^{|B_i|} b_{ij} = 1 \tag{2}$$

$$\mathbf{v'}_i = \sum_{j=0}^{|B_i|} \mathbf{v}_{ij} b_{ij} \tag{3}$$

$$\text{argmax}_k(f_k(\mathbf{v'})) \neq \lambda(\mathbf{x}) \tag{4}$$

$$b_{ij} \in \{0, 1\}, \quad i = 1, \ldots, T\,, \tag{5}$$

where $f_k(\cdot)$ is the k^{th} output of the network, and $\lambda(\cdot)$ represents the true label index. Eq. (1) is an objective function that minimizes the number of synonym swaps. Eq. (2) is the condition for selecting only one synonym or original word for each position i. The word vector of the selected word is extracted with Eq. (3). Eq. (4) is a constraint for changing the model's original prediction, i.e., where the sentence obtained by swapping the words following the b_{ij} values is an adversarial example. This formulation is only applicable when $f(\cdot)$ is a piecewise-linear neural network composed of combinations of linear functions and piecewise-linear functions such as ReLU and maximum functions. We show how to formulate piecewise-linear functions as an ILP problem in the Appendix B. Following TextFooler (Jin et al., 2019), we prepare a synonym list B_i, (Appendix A) and show an example of formulation in Appendix C.

4 Metrics

Accuracy Under Attack Accuracy Under Attack (AUA) is the rate of the fraction of the test set that satisfies the following equation:

$$\forall x' \in (\mathcal{G}(x)) : \text{argmax}_i \left(f_i\left(x'\right)\right) = \lambda(x)\,, \tag{6}$$

where $\mathcal{G}(x)$ is a transformation that adds a perturbation to text x. $f_i(\cdot)$ is the i^{th} output of the NN model, and $\lambda(x)$ represents the true label of x. For a text x, we assume that $\mathcal{G}(x)$ is a combination of all the word swaps for a prepared synonym list. In the experiments in Section 5, we evaluated the models with this metric. Our proposed method allows us to find the lower bound of AUA by checking whether Eq. (6) is satisfied.

Mean Minimum Word Swaps The problem of finding minimum word swaps is denoted below. $d\left(\cdot, \cdot\right)$ is a distance metric that defines the number of word swaps:

$$\min_{x'} \quad d\left(x', x\right) \tag{7}$$

$$\text{subject to} \quad \text{argmax}_i \left(f_i\left(x'\right)\right) \neq \lambda(x)\,. \tag{8}$$

A Mean Minimum Word Swap (MMWS) is the average of this distance in the test set. The advantage of this metric is that it allows for an intuitive and flexible way to evaluate the model's robustness.

5 Experiments

We conducted comprehensive experiments to evaluate the effectiveness of our verification method

Dataset	Train	Test	Avg Length	Categories
SST2	67 K	870	9	2
IMDb	25 K	25 K	159	2

Table 1: Overview of datasets

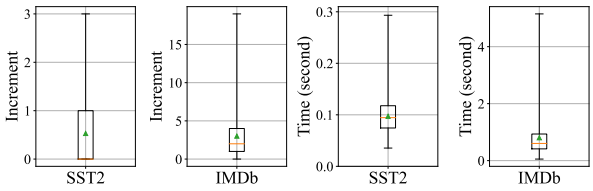

Figure 1: Incremental number of word swaps from the proposed method to TextFooler (two graphs on the left) and Times to verify the model (two graphs on the right).

including its applications to the generation of adversarial examples and adversarial training. We studied our verification method with text classification datasets with average sequence lengths from tens to hundreds of words. The dataset statistics are summarized in Table 1. We evaluated our algorithm on a set of 500 samples for the SST2 dataset (Socher et al., 2013) and 1,000 samples for the IMDb dataset (Maas et al., 2011) randomly selected from the test set. We prepared two baseline models, which were trained on each dataset. The architecture of the neural network is shown in Appendix D. The number of input words was limited to 200, and sentences with fewer than 200 words were compensated with padding tokens. We set the vocabulary size to 20,000 frequent words and replaced the words not in the vocabulary with unknown tokens. For verification, we used the same synonym list for our method and TextFooler which we compare and used Gurobi (Gurobi Optimization, LLC, 2022) as a mathematical optimization solver.

5.1 Robustness Evaluation and Generation of Adversarial Examples

In this part, we examine the effectiveness of the proposed method by comparing how it obtains minimum word swaps with TextFooler which performs a heuristic search. The two graphs on the left of Fig. 1 show the incremental number of word swaps from the proposed method to TextFooler. The proposed method achieved fewer word swaps in both datasets because it guarantees a minimum number of word swaps. We also generated adversarial examples with our method. Samples of original and adversarial sentences are shown in Table 2. We expected the minimal word swaps to suppress the sentence changes. In Section 6, human evaluations actually shows that the quality of the adversarial examples with our method is better than with TextFooler. The two graphs on the right of Fig. 1 shows the times required for verifying the models. Even IMDb, which has a long average sequence length, is processed in a realistic amount of time.

5.2 Adversarial Training

We evaluate an adversarial training using adversarial examples generated with our method. The proposed method makes it possible to compare the robustness of multiple models. We trained three models: a baseline model, a model with adversarial training using Textfooler, and the model with adversarial training using the proposed method.

Table 3 compares the accuracy and the AUA. For SST2, we can confirm that the robustness of the model with adversarial training improved. In this case, AUA in TextFooler and Ours are asymptotically equal. On the other hand, it is difficult to assess the robustness for IMDb models because AUA are 0. It is possible to limit the number of word swaps, but the settings need to be changed carefully for each dataset. Even when the AUA is not helpful, it can be evaluated with MMWS which consider the number of word swaps.

Figure 2 shows the histograms of the number of word swaps. Since our method always obtains the smallest combination of word swaps, the distribution is skewed to the left when compared to TextFooler. The distribution is skewed toward the larger number of word swaps required to change the prediction in order of the baseline model, TextFooler model, and our model. This indicates that a model become stronger when adversarial training is implemented, and that our model is more robust than the TextFooler model. Table 4 shows MMWS scores, which represents the robustness of the models.

6 Human Evaluation

We conducted human evaluation of the generated adversarial examples from the text classification model trained with the IMDb dataset. A total of nine native speakers in their 20s to 40s living in the U.S. and the U.K. were asked to evaluate the examples using the evaluation metrics "grammatical

	Movie Review (Positive (POS) ↔ Negative (NEG))
Original (Label: POS)	it's a charming and often affecting journey.
Attack with TextFooler (Label: NEG)	it's a **ravishing** and **normally impacts trip**.
Attack with Ours (Label: NEG)	it's a **ravishing** and **normally influenced** journey.
Original (Label: NEG)	an occasionally funny but overall limp fish-out-of-water story.
Attack with TextFooler (Label: POS)	an **intermittently** funny but **general** limp fish-out-of-water **history**.
Attack with Ours (Label: POS)	an occasionally **hilarious** but overall limp fish-out-of-water story.

Table 2: Examples of original and adversarial sentences generated by TextFooler and our method against baseline model for SST2 dataset. Replaced words are shown in bold.

Dataset	Model	Acc	Attack	AUA
SST2	Baseline	0.838	TextFooler	0.280
			Ours	0.280
	TextFooler	0.820	TextFooler	0.478
			Ours	0.478
	Ours	0.820	TextFooler	0.516
			Ours	0.516
IMDb	Baseline	0.819	TextFooler	0
			Ours	0
	TextFooler	0.815	TextFooler	0
			Ours	0
	Ours	0.807	TextFooler	0
			Ours	0

Table 3: Accuracy Under Attack (AUA). Higher scores indicate greater robustness.

Model	Score (SST2)	Score (IMDb)
Baseline	1.99	4.81
TextFooler	2.31	5.86
Ours	2.49	7.28

Table 4: MMWS Score. This is an average score of the blue histogram in Figure 2.

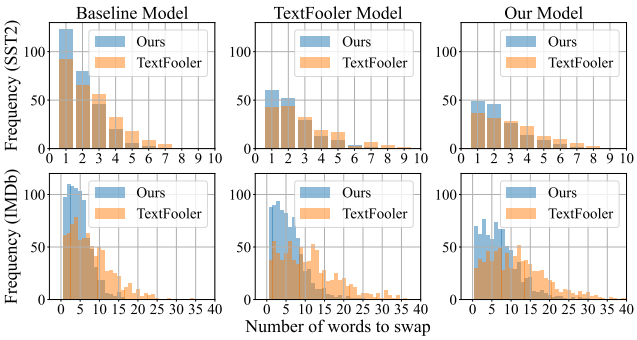

Figure 2: Histogram of number of words to swap for each dataset. We attacked the three models (Baseline, TextFooler, Ours) with TextFooler and Ours.

Model	Grammatical Correctness	Semantic Similarity
Baseline	2.18 → 2.51	2.16 → 2.37
TextFooler	1.99 → 2.39	2.09 → 2.34
Ours	2.14 → 2.80	2.19 → 2.52

Table 5: Human Evaluation Score. Scores for the TextFooler are to the left of the arrow, and our model's scores are to the right.

correctness" and "semantic similarity". For each adversarial example, three people gave a score from 1 to 4 following the criteria shown in Appendix E.

Table 5 shows the evaluation scores of the adversarial examples. For each model, the score of the proposed method was higher in both grammatical correctness and semantic similarity.

Figure 3 shows the relationships between human evaluation scores and the average swaps of the adversarial examples for the baseline model. The horizontal axis is the number of synonym swaps and the vertical axis is the average score. We see that the fewer the number of synonym swaps, the higher the scores for both TextFooler and the proposed method. This result supports the validity of the proposed method which aim to find the minimum number of synonym swaps.

7 Conclusion

The proposed method always obtains a minimum synonym swapping, which makes it possible to compare and evaluate the robustness of text classification models. In addition, we conducted human evaluation and supported the effectiveness of our approach. We also performed the adversarial training and found that it makes the models more robust.

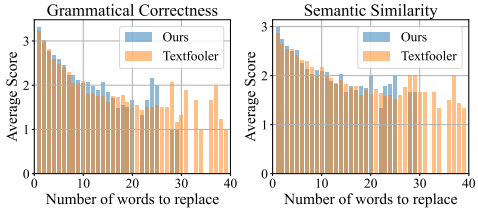

Figure 3: Effect of the Number of Synonym Swaps in Human Evaluation

Acknowledgements

This work was supported by JST, PRESTO Grant Number JPMJPR20C7 Japan.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018a. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. 2018b. Generating natural language adversarial examples. *CoRR*, abs/1804.07998.

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.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for NLP. *CoRR*, abs/1712.06751.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples.

Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual.

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

Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. *CoRR*, abs/1702.01135.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. TextBugger: Generating adversarial text against real-world applications. In *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society.

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.

Paul Michel, Xian Li, Graham Neubig, and Juan Miguel Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. *CoRR*, abs/1903.06620.

Nina Narodytska, Shiva Prasad Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh. 2018. Verifying properties of binarized deep neural networks.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition.

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

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks.

Vincent Tjeng and Russ Tedrake. 2017. Verifying neural networks with mixed integer programming. *CoRR*, abs/1711.07356.

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.

A Creating a Synonym List

We gather a candidate set B_i for all possible swaps of the selected word x_i. For each word in the text, we retrieve all words from GloVe (Pennington et al., 2014) whose cosine similarity is greater than 0.8. We use the Universal Sentence Encoder (USE) (Cer et al., 2018) to encode sentences X and X_{adv}, and extract their cosine similarity score is greater than 0.8. In addition, we check that Parts-of-Speech (POS) matches. We cannot perform POS checking and USE checking dynamically because it it difficult to consider altered context words of the target word. We therefore only swap the target word for checking.

B Formulating Piecewise Linear Functions

Formulating the Maximum Function

As denoted in (Tjeng and Tedrake, 2017), the maximum function can be formulated as below.

$$\bigwedge_{i=1}^{m} \left(\left(y \leq x_i + (1 - a_i) \left(u_{\max,-i} - l_i \right) \right) \wedge \left(y \geq x_i \right) \right) \wedge \left(\sum_{i=1}^{m} a_i = 1 \right) \wedge \left(a_i \in \{0, 1\} \right) . \tag{9}$$

(4) can be rewritten with the maximum function like below.

$$f_{\lambda(\mathbf{x})} \left(\mathbf{v}' \right) < \max_{\mu \in [1,n] \setminus \{\lambda(\mathbf{x})\}} f_{\mu} \left(\mathbf{v}' \right) . \tag{10}$$

Formulating ReLU

When all the nonlinear functions in the NN model are piecewise linear, it can be solved as an ILP. A piecewise linear function is a function that combines partially linear functions such as the ReLU function with $y = max\,(x, 0)$. Specifically, for each input scalar value x and output scalar value y of the ReLU function, it can be formulated with the binary variable a (Tjeng and Tedrake, 2017).

$$\begin{aligned} (y \leq x - l(1 - a)) \wedge (y \geq x) \wedge (y \leq u \cdot a) \\ \wedge (y \geq 0) \wedge (a \in \{0, 1\}) , \end{aligned} \tag{11}$$

where l is the lower bound of x and u is the upperbound of x. In advance, we can explore each in each layer. l is approximated to a smaller value and u to a larger value. As a result, it is possible to replace $y = x$ if l is greater than 0 and $y = 0$ if u is less than 0, thus reducing computation time when performing the entire formulation and searching.

C Example of Formulation

When we find the synonym list $B_2 = \{film\}$ and $B_4 = \{nice, great\}$ for an input text "this movie is good" (Figure 4), our objective is to minimize the sum of binary variables in the orange box and the sum of each blue box is constrained to 1. The formulation is written as (12).

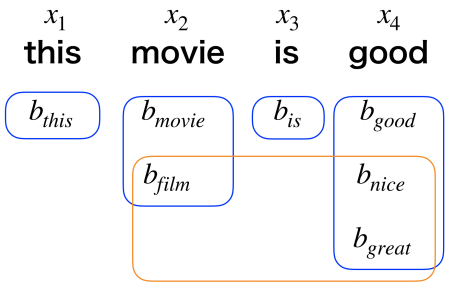

Figure 4: Example of Formulation

$$\begin{aligned} \min \left(b_{film} + b_{nice} + b_{great} \right) \\ b_{this} = 1 \\ b_{movie} + b_{film} = 1 \\ b_{is} = 1 \\ b_{good} + b_{nice} + b_{great} = 1 \\ \mathbf{v'}_1 = \mathbf{v}_{this} b_{this} \\ \mathbf{v'}_2 = \mathbf{v}_{movie} b_{movie} + \mathbf{v}_{film} b_{film} \\ \mathbf{v'}_3 = \mathbf{v}_{is} b_{is} \\ \mathbf{v'}_4 = \mathbf{v}_{good} b_{good} + \mathbf{v}_{nice} b_{nice} + \mathbf{v}_{great} b_{great} \\ y_{positive}, y_{negative} = f \left(\mathbf{v'}_1, \mathbf{v'}_2, \mathbf{v'}_3, \mathbf{v'}_4 \right) \\ y_{positive} < y_{negative} . \end{aligned} \tag{12}$$

D Architecture of a Neural Network

The architecture of the NN model is a simple network consisting of affine transformations and nonlinear transformations using the ReLU function, as shown in Table 6.

Layer	Shape of Output Tensor	Params
Input	(200)	0
Embedding	(200, 2)	40,000
Flatten	(400)	0
Affine	(64)	25,664
ReLU	(64)	0
Affine	(2)	130

Table 6: Structure of Neural Network

E References of Human Evaluation

Score	Description
4	Correct.
3	Grammatically incorrect, but acceptable as a casual expression.
2	There are one or two clear errors that are not even used as a casual expression.
1	Three or more clear errors exist.

Table 7: References for Evaluating Grammatical Correctness

Score	Description
4	Paraphrase of the original sentence and the content conveyed by the sentence has not changed. The classification result is invariant.
3	Although the content has changed to the extent that the sentence is less influenced compared to the original sentence, the classification result is considered to be invariant.
2	Although the sentence has been changed to the extent that it has a greater impact on the meaning of the sentence compared to the original sentence, the class label is considered unchanged.
1	It has been changed to the extent that it has a greater impact on the meaning of the sentence compared to the original sentence, and the class label can change.

Table 8: References for Evaluating Semantic Similarity