Evaluating the `textbugger` NLP Attack on a State-of-the-Art Sentiment Classification Model

Seiji Eicher
Department of Computer Science
Stanford University
keicher@stanford.edu

Abstract

This project analyzes the performance of the `textbugger` adversarial NLP attack proposed by Li et al. 2018, which aims to cause misclassification of a given input that was initially classified correctly with minimal perturbation, as an attack and a source of adversarial training data to harden an existing model.[1] The attack was tested using the `TextAttack` platform on the `distilbert-base-uncased-finetuned-sst-2-english` sentiment classification model provided by Huggingface, which was downloaded over one million times in the last month as of June 2021.[2][3] Our findings show that the base model was susceptible to the `textbugger` attack with attack success rate 71.16%, and training on adversarial examples generated by `textbugger` was effective in hardening the model against the attack, dropping the attack success rate to 63.12% at a cost of .58% to the overall classification accuracy.

1 Introduction

The ubiquity of deep learning models which learn from their user-base, like social recommenders or CAPTCHA[4], or other relatively unsupervised datasets, like the Common Crawl, in practice has given rise to the study of their trustworthiness, and in particular, their robustness against adversarial attacks, as seen by the U.S. Army's investment against data poisoning in facial recognition[5] and the hijacking of Microsoft's Twitter AI chatbot "Tay."[6] As natural language processing-based models grow into the billions of parameters, the cost of training a boutique model has grown proportionally, causing these large language models (LLMs) to be used as black boxes–in the case of OpenAI's GPT-3, there are hundreds of research groups sharing the same model API with tens of thousands on the waiting list, further motivating auditing of models commonly used "off-the-shelf."

Adversarial attacks against neural networks have been most widely publicized in the field of computer vision, but there has been much progress in the application of similar techniques in natural language processing as well. General attacks in NLP have been well-surveyed in Yoo et al. 2020[7], which generalized adversarial NLP attacks as constituted by four components:

1. Goal Function
2. Constraints
3. Transformation
4. Search Method

The goal function represents a notion of an attack's success from the perspective of the target model, which could be general misclassification of a data point, or classification to a target class. The constraints represent our goal of finding non-trivial adversarial examples, for example, limiting the Levenshtein edit distance or words perturbed proportion between an input and its perturbation.

CS 229: Machine Learning, Spring 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

Closely related to the constraints are a choice of transformation, which represents the way that input texts are modified within the algorithm, e.g. creating perturbations that are visually similar to the input text (replacing characters with homoglyphs) or semantically similar (replacing words with synonyms). With regard to the search method, an important difference between adversarial attacks in computer vision and NLP is the discrete nature of word embeddings compared to continuous pixel values, which leads to optimization techniques in NLP that are more heuristic than computer vision. Search algorithms like beam or greedy search are often used due to the size of the search space.

The inputs to our adversarial attack are a pre-trained sentiment classification model `distilbert-base-uncased-finetuned-sst-2-english` and a dataset of uncased text with a "Positive" or "Negative" sentiment label. Using the `textattack` platform and the `textbugger` attack, we generate adversarial perturbations which differ only slightly from correctly classified inputs but are misclassified. Finally, we fine-tune the model on these adversarial perturbations of the input data and compare attack performance to the baseline model.

2 Related Work

Since the famous Szegedy et al. 2013[8] result showing the counter-intuitive discontinuities of neural networks' input-output mappings, neural networks have continued to be a dominant tool for data scientists to efficiently model large data sets, motivating a continued interest in testing for their robustness. This work was continued by Goodfellow et al. in 2015[9], but the bulk of adversarial research has been done in the field of computer vision, since continuous inputs make gradient-based methods straightforward. In constrast, the discrete token-based inputs that characterize NLP make optimizing inputs with respect to loss more challenging. Nevertheless, there are a litany of NLP attacks which could be considered contemporary, with difficulty determining a precise "state-of-the-art" due to the somewhat nebulous nature of the adversarial problem statement, and lacking a benchmark dataset like ImageNet for computer vision or GLUE for NLP. Often, research on text attacks have been done on classification tasks and uses as Greedy Word Importance Ranking (WIR) search to find examples, with the constraints of the attack, transformations allowed, and dataset/model tested varying. For example, Jin. et al.'s 2019 `textfooler`[10] constrained based on word embedding distance, part of speech, and Universal Sentence encoding cosine similarity, using embeddings specially designed for locating synonyms. Instead of synonym-friendly embeddings, Gao et al.'s 2018 `deepwordbug`[11] uses character insertion, character deletion, neighboring character swap, and character substitution for transformations, constraining on the Levenshtein edit distance between an input and its perturbation. Finally, other search methods, like genetic algorithms, beam search, and particle swarm optimization have been used as well.[12][13][14] Though the quality of individual attacks is multifaceted, Yoo et al. has found that WIR had a high attack success rate/computation ratio, which is what the attack used by this paper employs.[7]

3 Dataset

This paper uses the Stanford Sentiment Treebank 2 dataset to fine tune the model and seed adversarial perturbations. The SST2 dataset is part of the General Language Understanding Evaluation (GLUE) benchmark, which is widely used as a standard of language model performance. This version of the dataset uses the two-way (positive/negative) class split with sentence-level-only labels. Table 1 contains examples of these inputs. There are 67,349 examples in the training set and 872 in the validation set. The test set is not used because labels are not publicly available. To generate adversarial training examples, each examples in the training set was perturbed according to the `textbugger` attack, which will be discussed at greater length in the Methods section. The dataset is lowercased, and has a mean input length of 8.7, standard deviation of 7.37 in length, and max length of 48. The dataset is slightly unbalanced, with 37569 examples of positive sentiment and 29780 examples of negative sentiment.

4 Methods

We used the `Text Attack` application to conduct experiments . This application bundles implementations of many adversarial attacks, similar to the HuggingFace library, which facilitates testing

Sentiment Label	Text
Positive	“if you ’ve a taste for the quirky, steal a glimpse"
Negative	“it ’s in the action scenes that things fall apart ."

Table 1: Sample training data from the Stanford Sentiment Treebank 2 dataset.[15]

different approaches in a standardized way. We will begin by attacking an undefended model, then retraining it on adversarial examples to report the statistics of the newly defended model.

The attack we will be using is `textbugger` from Li et al. 2018[1]. This attack’s goal function is untargeted classification, which means success when correctly labeled postive sentiment text is perturbed so that the model labels it as negative, and vice versa for a correctly labeled negative sentiment text. This attack was chosen for its high evasiveness, with 94.9% of adversarial text correctly recognized by human readers, and efficiency, since it generates adversarial text in time sub-linear to the input length. The search method used is Greedy-Word Importance Ranking, which measures word importance heuristically by measuring the sentiment score change that results from the deletion of each word from the text. The transformations used are character insertion, character deletion, neighboring character swap, and character substitution, while constraining based on USE sentence encoding cosine similarity.

This paper uses the model `distilbert-base-uncased-finetuned-sst-2-english` from HuggingFace. DistilBERT is a distilled version of the Bidirectional Encoder Representations from Transformers (BERT) model.[16] This model presents a realistic attack surface because it is one of the most downloaded models from HuggingFace in the last month. It was hardened on three rounds of clean/adversarial training with an initial learning rate of 2×10^{-5} and an Adam optimizer. The batch size was 128, and the encoded text was padded to a length of 48. Since there were 67,349 examples in the training set, the adversarial training set contained 67,349 examples perturbed against the current model weights, including examples where the attack failed or the model predicted incorrectly, which were included in the adversarial set unchanged.

5 Results

We will begin by presenting adversarial examples generated by the `textbugger` attack on the baseline model, which can be seen in Table 2. The attack is high-quality from a visual and semantic perspective, since the sentences seem very similar visually and when replacement was used, the terms are synonymous. The transformed words are highlighted with a color corresponding to their prediction by the model, green indicating a positive sentiment and red for negative.

True Label	Original Text	Adversarial Perturbation
Positive	“true tale of courage – and complicity – at auschwitz is a harrowing drama that tries to tell of the unspeakable."	“true tale of mettle – and complicity – at auschwitz is a harrowing drama that tries to tell of the unspeakable."
Negative	“prurient playthings aside , there ’s little to love about this english trifle."	“prurient playthings aside , there ’s little to love about this english trifle."

Table 2: Sample attack results from the base model. Note the lookalike character: the ‘l’ in little is actually Unicode character U+217C.

Next, Table 3 gives a quantitative summary of the attack on the baseline. The attack was run over 872 examples from the GLUE SST validation set, of which 78 were classified incorrectly and skipped, which corresponds to the an original accuracy of 91.06%. Out of the remaining 794 examples, 565 were successful attacked using `textbugger`, while 78 failed within the constraints, so the attack success rate is 71.16%. Note that the success rate of an attack is parameterized by its constraint, because otherwise we could achieve arbitrary success simply by generating large perturbations. Out of the total 872 examples, 229 were resistant to the attack, so the accuracy under attack is 26.26%. On average, 15.03% of the words in an input needed to be perturbed to change the sentiment classification, and required 47.54 queries to the transformation rule to find a successful perturbation. The average

number of words per input is constant across the baseline and hardened model results because they were tested on the same dataset.

Metric	Baseline Model	Hardened Model
No. successful attacks	565	498
No. failed attacks	229	291
No. skipped attacks	78	83
Original accuracy	91.06%	90.48%
Accuracy under attack	26.26%	33.37%
Attack success rate	71.16%	63.12%
Avg. perturbed word %	15.03%	17.75%
Avg. no. words per input	17.4	17.4
Avg. no. queries	47.54	51.33

Table 3: Comparison of attack success over 872 examples on `distilbert-base-uncased-finetuned-sst-2-english` before and after training on adversarial examples.

Next, the `distilbert-base-uncased-finetuned-sst-2-english` model was trained for 3 epochs on adversarial examples generated according to the `textbugger` recipe, achieving a final model accuracy of 90.48%. Table 4 contains examples of successful adversarial examples generated against the hardened model.

True Label	Original Text	Adversarial Perturbation
Positive	“the jabs it employs are short , carefully placed and dead-center ."	“the jabs it employs are short , attentively placed and dead-center ."
Negative	“this film seems thirsty for reflection , itself taking on adolescent qualities ."	“this film seem thirsty for reflection , itself taking on adolescent qualities ."

Table 4: Sample attack results from the base model. Note the lookalike character: the ‘t’ in little is actually two Unicode characters U+FFFD.

Qualitatively, the distribution of adversarial texts generated on the hardened model did not differ significantly from those generated for the baseline model. This makes sense because there were many failed attacks in the baseline model, so attack texts were already being pushed to the edge of the similarity constraints.

After training on the adversarial examples, the hardened model did show a modest increase in robustness, decreasing attack success from 71.16% to 63.12% and increasing accuracy under attack 7.11 percentage points to 33.7%. The additional difficulty that the hardened model presented to the attack procedure is evidenced by the increase in average perturbed word % and average number of queries necessary for attack success as well, since it took more transformations from the input text on average to generate a successful adversarial example. Additionally, this resilience came at minimal cost to the overall accuracy, since the hardening process dropped accuracy only .58%.

6 Conclusion

In this paper, we have shown that the `distibert-base-uncased` text classification model fine tuned on the Stanford Sentiment Treebank `sst2` dataset was susceptible to difficult to detect adversarial inputs at a rate of 71.16%. Next, we showed that training on adversarial examples dropped the attack success rate for 63.12% at the expense of only .58% accuracy.

For the future, the most immediate next step would be training the adversarial model for more epochs, since time and compute constraints kept this experiment at just 3, after which the loss had not converged. However, this paper shows that even a small amount of hardening can more than pay for itself in attack resistance and accuracy under attack.

7 Contributions

Seiji Eicher was the sole contributor to this paper.

References

[1] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *CoRR*, abs/1812.05271, 2018.

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

[3] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[4] Josh Dzieza. Why CAPTCHAs have gotten so difficult, February 2019.

[5] Army looks to block data 'poisoning' in facial recognition, AI, February 2020. Section: Defense.

[6] In 2016, Microsoft's Racist Chatbot Revealed the Dangers of Online Conversation - IEEE Spectrum.

[7] Jin Yong Yoo, John X. Morris, Eli Lifland, and Yanjun Qi. Searching for a Search Method: Benchmarking Search Algorithms for Generating NLP Adversarial Examples. *arXiv:2009.06368 [cs]*, October 2020. arXiv: 2009.06368.

[8] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[9] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[10] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment, 2020.

[11] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers, 2018.

[12] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples, 2018.

[13] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification, 2018.

[14] Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 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, July 2020. Association for Computational Linguistics.

[15] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 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, October 2013. Association for Computational Linguistics.

[16] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.