Adversarial Attack on Natural Language Processing Models with Genetic Algorithms and Sentence Salience

First Author
Weifan Jiang, Haoxuan Wu
wj2301,hw2754@columbia.edu

Abstract

Adversarial attack is an effective method to test the robustness of machine learning based models in natural language processing. However, comparing to other fields in adversarial attack, the essential part of attacks on text classification is to maintain lexical correctness, grammatical correctness and semantic similarity when generating adversarial examples. In this paper, we present an algorithm that leverages a population-based optimization strategy known as the genetic algorithm with analysis on the salience of each sentence in the input, to apply sentence-level perturbations to input text, and generate adversarial examples. We believe that our algorithm is more suitable for datasets with long input consists of multiple sentence, comparing to previous works that only apply word-level perturbations at a time. We evaluated our result on the IMDB large movie review dataset, and discussions are included.

1 Introduction

Recent research has found that deep neural networks (DNNs) are vulnerable to adversarial examples (Szegedy et al., 2013). Adversarial examples are artificially generated by applying imperceptible non-random perturbation on the original input to cause misclassification. In the area of Natural Language Processing (NLP), generating adversarial examples is more difficult because input texts are discrete tokens, which means that the text space is discrete and applying gradient descent based attacks is non-intuitive. In addition, it is hard to generate semantically and syntactically similar adversarial examples for input; unlike in computer vision adversarial attack, where changing the value of a pixel hardly alter the semantic of an image, changing words in a sentence/paragraph is more noticeable and easier to alter the semantic meaning.

In a previous publication, (Ren et al., 2019) purposed that successful NLP adversarial attacks should follow these guidelines: lexical constraint indicates there should be no spelling error in the adversarial example, otherwise a spell checker could easily remove the attack; adversarial examples should be grammatically correct; adversarial examples should have similar semantic meaning to original sample.

Approaches to NLP adversarial attacks are usually consists of three categories: character, word, and sentence level. These levels indicate the scale of perturbations that attacker applies to the clean sample at a time. Usually NLP attacks try to substitute one component of the clean input with a semantically similar token, to alter the prediction result.

Gradient-based approaches is also possible for attacking NLP models. However, it requires significant work to smooth the discrete textual domain. In addition, one also needs to find words with "closest" semantic meaning to the output, which is generally considered a hard task due to the fact that quantification of language semantic is unnatural.

2 Related Work

We review two pieces of previous works that inspired our algorithm. Both works are iterative algorithms that introduce word-level perturbations at a time, to reduce confidence of model predicting the correct label. After multiple iterations, the prediction will be altered.

In the previous work, word-level adversarial examples are generated based on synonyms substitution strategy. In (Ren et al., 2019), the researchers proposed a method to calculate word salience and substitute word with highest salience. Word salience is a quantification of a word's contribution to the final predicted label of the entire

input. And in each iteration, the word with highest salience will be replaced by a synonym to reduce confidence of correct prediction.

Genetic algorithms are known to perform well in solving combinatorial optimization problems (Mühlenbein, 1989). In (Alzantot et al., 2018), researchers proposed a population-based gradient-free optimization that could successfully generate both semantically and syntactically similar adversarial examples against models via genetic algorithm. When perturbing a sample, (Alzantot et al., 2018) randomly selects a word in the input sentence, and replace it with a synonym that alters the prediction confidence the most.

We hope to extend the two previous works by:

- increase the scale of perturbation to sentence-level; introduce more complex change in input such as paraphrasing, sentence-structure alternation instead of replacing words with synonyms.

- improve the performance of genetic algorithm by considering salience information while choosing a component of the sample to alter.

3 Our method

As stated at the end of section 2, we proposed a new version of genetic algorithm to attack natural language processing models by applying sentence-level perturbations. We expect our algorithm to converge faster than the word-level genetic algorithm proposed by (Alzantot et al., 2018) on datasets with large-size inputs.

3.1 Threat Model

Our proposed attack works in the black-box setting: we supply inputs to the model, and we can only access the predicted confidence for each label. We also make the additional assumption that the input is long (i.e. a paragraph) and consists of multiple sentences.

3.2 Genetic Algorithm

Genetic algorithms simulate process of natural selection to find better solutions iteratively. Specifically, the result set of each iteration is a called a generation. One generation is generated through crossover and mutation of last generation. Crossover is the process of selecting two parents from last generation and producing a child solution. Mutation process is applying some kind of modification on the child to increase the diversity of generation, which provides a better exploration of the search space. Figure 1 shows the procedure of genetic algorithm.

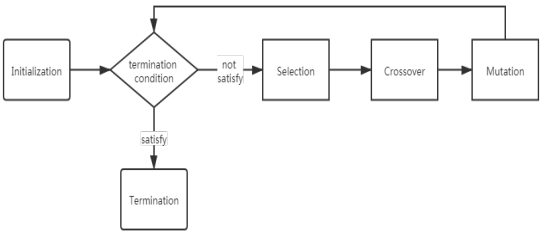

Figure 1: Procedure of Genetic Algorithm

3.3 Sentence Salience

To evaluate the importance of a component of the text input to the prediction of NLP model by calculating the difference of prediction confidence between original input and the modified input with removed component, we use sentence salience. We define input to the model is $S = s_1, s_2, ..., s_n$, where S represents for the input paragraph and s_i represents for the i^{th} sentence in the paragraph. The salience score of sentence s_i is:b

$$score_i = P(y_{true}|S) - P(y_{true}|S_{-i}) \quad (1)$$

where y_{true} represents for the true label of original input S, $P(\cdot|\cdot)$ represents for the confidence of input given by the model and $S_{-i} = s_1, s_2, ..., s_{i-1}, s_{i+1}, ..., s_n$ represents for removing sentence s_i from original input S. The $score_i$ reflects the importance of sentence s_i to the prediction of the whole input S.

3.4 Overall Algorithm

The overall algorithm incorporates incorporating sentence salience calculation into genetic algorithm framework for generating adversarial examples. We define pop as population size of one generation, gen as generation number, x as input text data, y as true label, x^* as adversarial example, G_i as the generation of i^{th} step.

In this procedure, $Sample$ process performing random sampling from G_i with weighted probability respective to the confidence vector of G_i using target model. $Crossover$ process uses two parents to generate a new child. $Salience$ process uses target model to calculate salience score of each sentence

input : pop, gen, x, y
output : x^*

$G_0 \leftarrow initialization$;
for $i \leftarrow 0$ **to** $gen - 1$ **do**
  if $\exists g_j \in G_i, predict(g_j)! = y$ **then**
    return g_j
  end
  $G_{i+1} \leftarrow Empty$;
  $Conf_i \leftarrow$ confidence vector of G_i ;
  for $j \leftarrow 1$ **to** pop **do**
    $p_1 \leftarrow Sample(G_i, Conf_i)$;
    $p_2 \leftarrow Sample(G_i, Conf_i)$;
    $child \leftarrow CrossOver(p_1, p_2)$;
    $score \leftarrow Salience(model, child)$;
    $child_{mut} \leftarrow Mutation(score)$;
    $G_{i+1} \leftarrow$ Add $child_{mut}$;
  end
end
return *Failure*

Algorithm 1: Overall Algorithm

in $child$. $Mutation$ process uses sentence rephrasing technique to do some perturbation on the $child$ and obtain $child_{mut}$ for the next generation.
In the $Mutation$, we use back translation for sentence rephrasing. To specific, back translation technique translates input text from original language to a "pivot" language, then translate back to original language to generate a paraphrase. Back translation is an effective strategy to evaluate stability of machine translation systems, also can be used as a general paraphrase-generation technique (Wieting et al., 2017).

4 Experiment

4.1 Set up

We evaluated our algorithm on the IMDB Large Movie Review dataset (Maas et al., 2011). Each input data point is a long paragraph of movie review, usually consists of multiple sentences, and the label is whether the reviewer has negative/positive attitudes towards the movie. Each review has averaged 325.6 words in it. We believe it is the ideal dataset to test sentence-level perturbation based attacks since the large input size in terms of words.

We obtained a pre-trained WordCNN model for the IMDB dataset from (Ren et al., 2019). WordCNN (Kim, 2014) is a word-based convolutional neural network. The model used in our experiement has one embedding layer that performs 50-dimensional word embeddings, one 1D-convolutional layer with 250 3-by-3 filters, one 1D-max-pooling layer, and two fully connected layers. This model has 86.55% accuracy on clean test samples.

Original	*I love to play with little furry cats.*
Pivot	我喜欢和毛茸茸的小猫一起玩。
Result	*I like to play with furry kittens.*

Table 1: Back-translation demonstration with Simplified Chinese (Code: ZH) as pivot language, using Google Cloud Translate.

We used Google's Cloud Translate API for generating back-translated text. We randomly select a non-English language, then generate a paraphrase with the selected pivot language. Note that the back-translated text is not guaranteed to be different than original text; in this case, we simply select another pivot language and try again. We set a threshold that the attack pauses if 20 consecutive back-translation attempts does not produce a paraphrase of the original input. Luckily, this situation never arose in our experiment. Table 1 shows a demonstration of back-translation with Google Cloud Translate API.

We selected two subsets of test samples for the experiment. **Group 1** has 5 samples with confidence of correct prediction < 0.7. **Group 2** has 5 samples with confidence of correct prediction > 0.7. In other words, model makes very confident prediction on group 2 samples, and less confident on group 1 samples. We ran our genetic algorithm with population size = 40 and generation size = 2 to save time. Ideally, generation size should be equal to the number of sentences in the input paragraph.

4.2 Result

Group 1 samples (not as-confident predictions) has a successful attacking rate of $5/5 = 100\%$. In fact, all attacks were finished within the first generation. Table 2 shows the attack result for samples in group 1.

Group 2 samples (model makes very confident predictions) all failed to be attacked with the given population and generation sizes. We believe that with more computing powers and increased population, generation sizes, these samples will eventually be attacked, since the average confidence of correct prediction of second generation is reduced comparing to first generation, which is evidence showing that the population is converging to be more fitted.

	sample	negative pred. confidence
clean	. . . and my main fear was, that the movie is just an assembling of their already known jokes, but it is not! The jokes are evolved through the story, and make, in their own special way, sense. But if you absolutely dislike Erkan und Stefan, hands of this movie. Everyone else - it's worth viewing!	0.33844942
adv	. . . and my main fear was, that the movie is just an assembling of their already known jokes, but it is not! The jokes are evolved through the story, and make, in their own special way, sense. But if you absolutely dislike Erkan und Stefan, hands of this movie. Others-worth a look!	0.67790467
clean	. . . Something about the story structure, the goofy relationships between the characters, the mannered dialog, the wooden acting (spiked with the occasional outright terrible performance), the scene tempos and rhythms that made Albert Pyun look like John McTiernan, the paper-thin plots and not-ready-for-prime-time fight choreography... . . .	0.6630755
adv	. . . There is something about storytelling, the relationships between characters, storytelling, woodworking (evoked by the most awe-inspiring work), performances and music that made Albert Pyun feel like and John McTiernan, the paper-thin and not-so-ready paperwork for the seasonal	0.35254362
clean	. . . Frequently, you can pick-up the formula and predict a story direction. I could not do that with this film.I would recommend this as one of the finer films of this genre.	0.4832231
adv	. . . You can often take the formula and make a prediction about the direction of a story. I could not do that with this film. I would recommend this as one of the finer films of this genre.	0.5778477
clean	. . . But John Wayne fans will want to see this one for the charisma he displayed early on in his career. For those more critical, the white kerchiefs worn around the forehead by the good guy posse could only mean that they all had a headache.	0.52035797
adv	. . . But John Wayne fans love to see this cartoon, which was shown early in his career. For those more critical, the white kerchiefs worn around the forehead by the good guy posse could only mean that they all had a headache.	0.3433545
clean	. . . , but the big laughs are even bigger. The casting in this one is great and even the typically out of place in, uh movies in general Parker Posey does a fine job. In fact, her tirade directed at Ed Begley Jr. and a pet store owner over a lost dog toy is probably the funniest running gag of the film. . . .	0.38134402
adv	. . . , but the big laughs are even bigger. This casting is great, even inappropriate, and the Parker Posei movies do a good job. In fact, her tirade directed at Ed Begley Jr. and a pet store owner over a lost dog toy is probably the funniest running gag of the film. . . .	0.50266093

Table 2: Attack result for group 1. Modified sentences between clean and adversarial examples are marked in blue and red.

Words	Sentences	Our time (s)	Alzantot's time (s)	Our generation count	Alzantot's generation count
152	9	88	1452	1	8
142	11	86.5	169	1	1
52	5	46.2	168	1	1

Table 3: Comparing our genetic algorithm to (Alzantot et al., 2018)'s genetic algorithm.

We think our method is very effective when attacking samples which model has low or medium confidence. When model has very high confidence, we need to increase the number of iterations to allow population to converge to desired distribution.

4.3 Semantic Evaluation

We evaluate the semantic closeness of our adversarial examples to original texr by conducting a survey: we show the clean test samples and perturbed samples from Group 1 as pairs to people without specifying which one is unperturbed, and asks whether both sentences show same attitude (positive or negative) in the survey.

Three people that took the survey responded "same attitude" to all pairs. However, some pointed out that there are sentences sounds a little "unnatural" comparing to the rest of the paragraph.

4.4 Comparing with word-level genetic algorithm

We compared our sentence-level genetic algorithm with salience analysis, to the original word-level genetic algorithm by (Alzantot et al., 2018). We selected 3 more samples that are not in Group 1 or Group 2, and both ours and (Alzantot et al., 2018)'s algorithm successfully attack them. Table 3 shows the word, sentence counts for each clean test sample, as well as the time and number of iterations/generations required by both algorithms to generate an adversarial example.

The results from Table 3 suggest that for larger inputs such that the number of words is significantly larger than number of sentences, our sentence-level attack converges faster to a successful adversarial instance than (Alzantot et al., 2018)'s word-level attack, both in terms of time and number of iterations needed.

5 Conclusion

In this paper, we introduce a new adversarial attack algorithm on Natural Language Processing models, leveraging genetic algorithms with sentence-level perturbations, and salience analysis of the input. With limited iterations, we successfully attacked $5/5$ test samples with confidence < 0.7. For more confidently predicted samples, our algorithm fails to attack them with given resources, but we see the convergence of population to desired distribution.

Based experiment results, we conclude that our algorithm works well on generating black-box natural language adversarial example for long text inputs. When model makes less-confident predictions, our algorithm generates adversarial examples fast and consistently. For more-confidently predicted test samples, our algorithm needs more iterations. Optimizing for better performance with confidently predicted samples will be a next step for this project.

We also evaluated the semantic consistency of our adversarial examples with original clean test samples. Based on the survey results, our adversarial examples all have same semantic meaning as clean samples, evaluated by human. However, we do notice that back-translation does not produce the most natural-sounding sentences due to limitations in machine translation software. In some scenarios, reader may easily tell that such sentence is machine perturbed. In the future, we will try more paraphrase-generation techniques besides back translation as the back-end of **Perturb**, such as *Parse-tree alternation*, to produce adversarial examples more adhere to human's natural language.

We compared the time and number of iterations required for our and (Alzantot et al., 2018)'s algorithms, and conclude that for NLP models with large-size inputs, our sentence-level attack can be a faster way of generating adversarial examples.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang.

2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

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.

Heinz Mühlenbein. 1989. Parallel genetic algorithms, population genetics and combinatorial optimization. In *Workshop on Parallel Processing: Logic, Organization, and Technology*, pages 398–406. Springer.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. *arXiv preprint arXiv:1706.01847*.