UNIVERSAL ADVERSARIAL ATTACKS ON TEXT CLASSIFIERS

Melika Behjati [⋆*] *Seyed-Mohsen Moosavi-Dezfooli* [†] *Mahdieh Soleymani Baghshah* [⋆] *Pascal Frossard* [†]

[⋆] Sharif University of Technology, Tehran, Iran
[†] École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

ABSTRACT

Despite the vast success neural networks have achieved in different application domains, they have been proven to be vulnerable to adversarial perturbations (small changes in the input), which lead them to produce the wrong output. In this paper, we propose a novel method, based on gradient projection, for generating universal adversarial perturbations for text; namely sequence of words that can be added to any input in order to fool the classifier with high probability. We observed that text classifiers are quite vulnerable to such perturbations: inserting even a single adversarial word to the beginning of every input sequence can drop the accuracy from 93% to 50%.

Index Terms— neural network, universal adversarial perturbation, gradient projection, text classifier

1. INTRODUCTION

In the recent years, neural networks have been successfully applied in many domains such as vision [1, 2, 3], speech [4, 5, 6], and text [7, 8, 9]. However, it has been shown that neural networks are vulnerable to small changes in the input, the so-called adversarial perturbations, which cause them to produce the wrong output [10]. In real-world applications, the robustness of networks to such perturbations must be taken into consideration in order to guarantee their worst-case performance. In the last few years, many works have focused on crafting adversarial perturbations for image data, and defending and analyzing the robustness of deep networks against such perturbations [11, 12, 13, 14]. However, only a few studies have been done on textual data. One of the key differences between text and image is the discrete nature of words and characters constructing the text which requires discrete optimization methods in order to craft adversarial perturbations for them.

Unlike image settings, where we can use the gradient of the output to find adversarial perturbations, the gradient of the output with respect to words or characters is not defined. Nevertheless, one can compute the gradient with respect to the embedding of words or characters as they lie in a Euclidean space. Thus, Papernot et al. [15] proposed a method for fooling a text classifier by replacing a randomly chosen word in the sequence with the nearest word in the direction of the gradient with respect to the embedding. This method is based on the Fast Gradient Sign Method [12], which finds the perturbation by computing the sign of the gradient with respect to the input. In [16, 17, 18], the gradient of the loss with respect to the inputs is used for discovering the critical words or characters in each input sample, and then perturbing them. Yang et al. [19] proposed a probabilistic framework for maximizing the success probability of an adversarial attack. Their framework consists of two steps; namely finding first the k most important word positions and then selecting the best words located in those positions to maximize the probability of the adversary to be successful. Alzantot et al. [20] proposed a method for altering the words using the semantically similar words and a language model for selecting the most natural words in that context. Jia et al. [21], proposed a method for crafting an adversarial sentence to be concatenated to the original text in order to fool a reading comprehension network. They also introduced a model-independent adversary which adds a random valid sentence to all the input paragraphs.

Moosavi-Dezfooli et al. [22] showed the existence of input-independent perturbations, called Universal Adversarial Perturbations (UAP), for image classification tasks. As opposed to adversarial perturbations, UAPs are data-independent and can be added to any input in order to fool the classifier with high confidence. Similarly, in this paper, we propose a novel method to generate universal adversarial perturbations for textual data in that they can be inserted into any input sequence in order to fool a given text classification model. Unlike previous works on attacking text classifiers, our attack does not require to separately optimize the perturbation for each input. Instead we seek for perturbations that can be applied on any input sequence (from the corresponding domain). The proposed method uses an iterative projected gradient-based approach to find a sequence of words that must be inserted into the input sequences. We evaluate our method on different settings including three RNN-based architectures and two text classification datasets, and show that all architectures are quite vulnerable to such a simple perturbation regime. To the best of our knowledge, we are the first to introduce universal adversarial attacks on text classifiers.

[*] The work has been done as an internship project at LTS4, EPFL.

978-1-5386-4658-8/18/$31.00 ©2019 IEEE

2. PROBLEM STATEMENT

In this section, we are going to formalize the problem of finding the universal adversarial examples for a text classifier in cases of both targeted and non-targeted attacks. A targeted attack is a kind of attack in which we aim to fool a classifier to classify an adversarial example into a predetermined target class rather than its true class, whereas, in a non-targeted attack the objective is simply to change the correct class. At first, we generalize the definition of an adversarial example to the text domain and then combine it with the universal attacks to formulate our problem.

2.1. Adversarial examples for text

Consider a classifier f which maps the input $x \in X$ to its class label $l \in L$. We call x' an *adversarial example* crafted from x if $f(x) \neq f(x')$, while x and x' are perceived as coming from the same class by a human observer. Let $x = x_1 x_2 \dots x_n$ be the input sequence of f which is an ordered list of words or characters constructing textual data. We can now craft an adversarial sequence $x' = x'_1 x'_2 \dots x'_{n'}$ by inserting, altering or removing some words and characters from x.

2.2. Universal adversarial examples for text

In the universal adversarial attack regime [22], we seek for a single perturbation that can be added to each of the input samples and fool a given classifier with high probability. In this work, we show the existence of such universal perturbations for text classification tasks. In particular, we want to find a single sequence of words w that, by concatenation with any input sample coming from data distribution $P(X)$, fools the classifier with high probability.

Consider a text classifier with input x and output l. Furthermore, let $w = w_1 w_2 \dots w_m$ be the adversarial sequence. We construct adversarial input as follows, where $\oplus$ is the insertion operator and k denotes the location of insertion ranging from 0 to n:

$$x' = w \oplus_k x = x_1 \dots x_k; w_1 \dots w_m; x_{k+1} \dots x_n \qquad (1)$$

In order to design w, we maximize the classifier's loss function with respect to the true class for the non-targeted attack. We aim to distract the classifier to any other class than the true one. Consider $\text{loss}(l, f(x))$ as the cost of classifying the input x into the class $f(x)$ (that is the output of the classifier) instead of the correct label l, we can design w as:

$$\hat{w} = \underset{w}{\text{argmax}}\ \mathbb{E}_{x \sim P(X)}[\text{loss}(l, f(x'))] \qquad (2)$$

If we rather want to perform a targeted attack where we intend to classify sample x' to a specified class l', the following optimization problem should be solved:

$$\hat{w} = \underset{w}{\text{argmin}}\ \mathbb{E}_{x \sim P(X)}[\text{loss}(l', f(x'))] \qquad (3)$$

Since we are optimizing the expected loss with respect to the data distribution, the resulting sequence $\hat{w}$ would ideally be universal; i.e., when $\hat{w}$ is concatenated to any sample coming from $P(X)$ the loss is maximized/minimized on average.

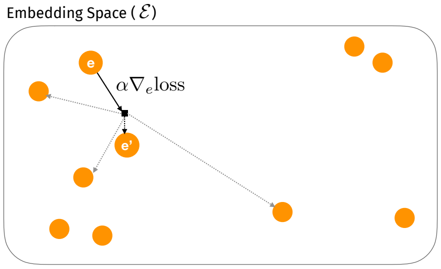
Embedding Space ($\mathcal{E}$)
e
$\alpha \nabla_e \text{loss}$
e'

Fig. 1. An illustration of how our algorithm projects the gradient in the embedding space. At first, the gradient is applied to the current word vector (e) and then among word vectors in the vocabulary (orange balls), the nearest one (e') is chosen to be projected to.

3. ADVERSARIAL ATTACK ALGORITHM

In this section, we describe our method for solving the optimization problems in Eq. (2) and Eq. (3). Due to the discrete nature of textual data, we are facing a discrete optimization problem. Therefore, the methods used for attacking image classifiers cannot be directly applied here.

In order to represent words in a computational space, we have to encode them into vectors. We call the vector space of encoded words the embedding space $\mathcal{E}$. Consider V as our vocabulary, which is a finite set of words, and $\mathcal{E}_V$ as the discrete subspace of $\mathcal{E}$ where the words in V are embedded. Also, consider w as a word vector in the embedding space $\mathcal{E}$. We now propose to use gradient descent/ascent, depending on whether the attack is targeted or non-targeted, to solve the optimization problems in Eq. (2) and Eq. (3).

In each iteration of the gradient descent/ascent, the embedding vectors corresponding to the words that we want to add to the sequence are first updated in the continuous embedding space using the gradient vector. Formally, for the word w_i in w, we compute the descent/ascent direction for input sample x where $x' = w \oplus_k x$, as $r_i = \nabla_{\text{emb}(w_i)} \text{loss}(l, f(x'))$, where $\text{emb}(w_i) \in \mathcal{E}$ is the corresponding embedding of w_i and l is the correct/target label. Then the resulting vectors are projected to the vocabulary vectors in that space ($\mathcal{E}_V$). This projection is performed for each word vector independently and, to the nearest vector in $\mathcal{E}_V$ in terms of cosine similarity (as illustrated in Fig. 1). Therefore,

$$w'_i = \underset{w'_i \in V}{\text{argmin}}\ \cos(\text{emb}(w'_i), (\text{emb}(w_i) + \alpha r_i)) \qquad (4)$$

In order to maximize the loss function in a non-targeted attack (Eq. (2)), we have to move in the direction of gradient (gradient ascent) in which the ratio α is positive. In contrary, for a targeted attack (Eq. (3)), we set α to be negative in order to move in the opposite direction of gradient (gradient descent) for minimizing the loss function. Our proposed method is summarized in Algorithm 1. It can also be applied to batches of data with small modifications.

Algorithm 1 Computation of universal word vectors

```
Input: data samples: X, learning rate: $\alpha$, loss function: loss(.,.), location of insertion: $k$
Output: universal adversarial word vectors $w = w_1 \dots w_m$
Initialize $w \leftarrow$ random sequence of words
loop
    for each data sample $x \in X$ do
        $x' = w \oplus_k x$
        for each word $w_i \in w$ do
            compute $\nabla_{\text{emb}(w_i)} \text{loss}(l, f(x'))$
            $z_i \leftarrow \text{emb}(w_i) + \alpha \nabla_{\text{emb}(w_i)} \text{loss}(l, f(x')$
            $w'_i \leftarrow \underset{w'_i \in V}{\text{argmin}} \cos(\text{emb}(w'_i), z_i)$
            $w_i \leftarrow w'_i$
        end for
    end for
end loop
```

4. EXPERIMENTAL RESULTS

In this section, we explain the setup and the evaluation of our method on different settings.

4.1. Setup

Datasets. We evaluate our methods on two datasets for the task of text classification.

- **AG news dataset**: The default dataset used in our experiments is the version provided by [7] in which 3000 samples from 4 common news categories (World, Business, Sports, Sci/Tech) are gathered. There are also 1900 test samples for each class.
- **Stanford Sentiment Treebank**: This dataset [23] contains about 11000 labeled sentences for the task of sentiment analysis. The sentences have been divided into five classes of very negative (1) to very positive (5).

Architectures. RNN-based architectures using Long Short-Term Memory (LSTM) cells [24] are used as the text classifiers. An embedding layer is also applied to the inputs of the network in order to map them into the continuous space. We used the pre-trained embedding vectors provided by [25]. The investigated architectures are as follows:

- **LSTM:** The default architecture used is a vanilla RNN with LSTM cells. The last hidden state is fed to a fully connected layer to produce the results.

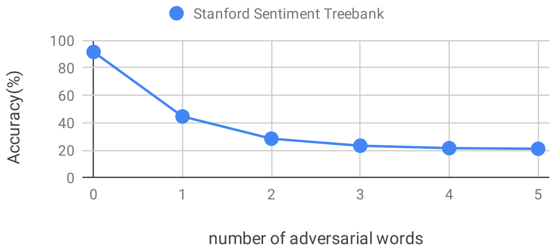

Fig. 2. Accuracy of LSTM trained on the Sentiment Stanford Treebank, when words are inserted at the beginning of inputs.

- **bi-LSTM:** We also tried a bidirectional LSTM [26] in which the last hidden states of both directions were concatenated and fed to the next layer.
- **mean-LSTM:** We used a mean layer over all the LSTM cells' outputs and then fed them to the next layer.

Hyper-parameters. All the LSTM cells have 512 hidden units. We also find out that the learning rate 1 is an appropriate value for moving between word vectors of our vocabulary in the embedding space. The Adam optimizer [27] is utilized for optimizing the loss function.

4.2. Performance

Non-targeted attacks. In order to perform this attack we solve the optimization problem in Eq. (2). Table 1 shows the results of attacking LSTM for classifying news articles by concatenating adversarial words to the beginning of the inputs. Interestingly, there exists a dominant class in all the experiments which is the Sci/Tech category. This attack results in the minimum possible accuracy to be achieved (i.e. 25%). As the number of adversarial words increases, the model's accuracy drops, since the number of words directly affect the classifier's decision. According to these experiments, when even one word is inserted to the beginning of any sequence adversarially, it can drop the model's accuracy to 50%.

We performed non-targeted attack on three different architectures trained on AG news dataset and concatenated the adversarial words to the beginning of each input sample. The results are shown in Table 2. All architectures seem to be vulnerable to our attack, which hints that our attack is a threat for RNN-based architectures in general.

In order to ensure the effectiveness of our method, we also tried it on another dataset (Stanford Sentiment Treebank) while inserting the adversarial words to the beginning of the sequences on the LSTM model. As shown in Fig. 2, this dataset is also vulnerable to our attack, and the minimum possible accuracy is almost achieved (i.e. 20%). Similar to AG news dataset, inserting one universal word highly distracts the classifier.

In addition, the effect of changing the location of inserted words is shown in Fig. 3. The attack is performed on LSTM

Table 1. The results of performing non-targeted attack on AG news dataset where the adversarial words are inserted at the beginning of the input samples. As the number of words in the perturbation increases, the accuracy drops.

number of words	accuracy	dominant class	universal adversarial words
10	25.01	Sci/Tech	Global, Phenology, general-assignment, single-topic, medulloblastoma, exobiology, tech, molten, astrobiology, modularized
5	25.84	Sci/Tech	three-station, succumbs, supercookies, cypherpunk, virtualisation
2	30.05	Sci/Tech	stellarators, non-OS
1	49.72	Sci/Tech	abandonware

Table 2. Accuracy for non-targeted attacks on different architectures, where m adversarial words are inserted to the beginning of the inputs on AG news dataset.

m	0	1	3	5
LSTM	93.42	49.72	25.35	25.84
mean-LSTM	92.80	87.92	25.23	25.00
bi-LSTM	93.23	89.07	28.53	25.01

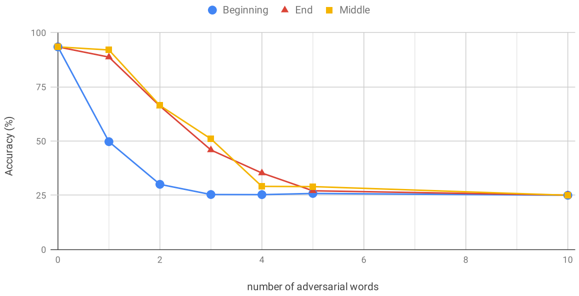

Fig. 3. Effect of changing the location of adversarial words on the accuracy of LSTM classifier on AG news dataset.

model trained on AG news dataset. The model is fooled easier when the words are inserted at the beginning of the input sequence. The model seems to give more importance to the beginning of the input sequence. In order to investigate it further, we performed the same experiment for a bi-directional LSTM model (Table. 3 which is agnostic to the direction of the input sequence. It can be seen that the bi-directional LSTM exhibits a more consistent behavior.

Targeted attacks. We try to optimize Eq. (3) in order to perform these type of attacks. We performed targeted attacks on LSTM model trained on AG news dataset, where the adversarial words are inserted at the beginning of the sequences. Fig. 4 shows that our method can be successfully applied for these kind of attacks. Based on the results, it seems some classes are more powerful in fooling the model, which is probably caused by special words used in them. For example, Sci/Tech category has many technical words, which are rarely used in other categories and therefore, by inserting them in any sentence the classifier gets fooled with high probability.

Table 3. Accuracy vs location of insertion of m adversarial words for bi-LSTM trained on AG news dataset.

m	1	3	5
beginning	89.07	28.53	25.01
end	81.92	34.36	25.69

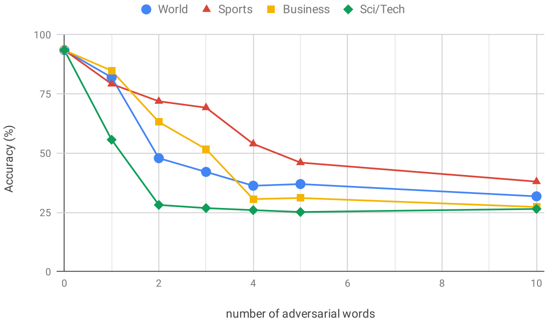

Fig. 4. Accuracy for targeted attacks on the AG news dataset, while adversarial words are added at the beginning of inputs.

In order to show that our attacks are effective, we inserted randomly selected words from the vocabulary to the beginning of the sequences, similarly to what Jia et al. [21] proposed for reading comprehension systems. We found that our models are almost robust to such random perturbations and the accuracy drop is negligible; in particular, 5 random words cause a drop of only 5% in the accuracy.

5. CONCLUSION

In this paper, we introduced a new type of attack for text classifiers, which is universal. We proposed a method based on gradient projection to craft data-independent adversarial sequences, which fool the classifier with high probability when added to any input sample. We evaluated our attack on different settings and our results show that, even in such a simple regime the classifiers are quite vulnerable such that inserting one word can drop the accuracy from 93% to 50%.

The problem of robustness of text classifiers is not well-studied, due to the discrete nature of textual data which introduces more challenges than for image data that are commonly studied. There are still many open questions left in this context, e.g., how attention might contribute to the robustness of a model, which could be investigated in future work.

6. REFERENCES

[1] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[3] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[4] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio.," in *Proceedings of the Speech Synthesis Workshop (SSW)*, 2016, p. 125.

[5] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[6] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu, "Efficient neural audio synthesis," *arXiv preprint arXiv:1802.08435*, 2018.

[7] Xiang Zhang, Junbo Zhao, and Yann LeCun, "Character-level convolutional networks for text classification," in *Proceedings of the Advances in neural information processing systems (NIPS)*, 2015, pp. 649–657.

[8] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[9] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen, "Reasonet: Learning to stop reading in machine comprehension," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2017, pp. 1047–1055.

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

[11] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2574–2582.

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proceedings of the International Conference for Learning Representations (ICLR)*, 2017.

[14] Pouya Samangouei, Maya Kabkab, and Rama Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," in *Proceedings of the International Conference for Learning Representations (ICLR)*, 2018.

[15] Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang, "Crafting adversarial input sequences for recurrent neural networks," in *Proceedings of the Military Communications Conference (MILCOM)*. IEEE, 2016, pp. 49–54.

[16] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi, "Deep text classification can be fooled," *arXiv preprint arXiv:1704.08006*, 2017.

[17] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou, "Hotflip: White-box adversarial examples for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018, vol. 2, pp. 31–36.

[18] Suranjana Samanta and Sameep Mehta, "Towards crafting text adversarial samples," *arXiv preprint arXiv:1707.02812*, 2017.

[19] Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I Jordan, "Greedy attack and gumbel attack: Generating adversarial examples for discrete data," *arXiv preprint arXiv:1805.12316*, 2018.

[20] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang, "Generating natural language adversarial examples," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[21] Robin Jia and Percy Liang, "Adversarial examples for evaluating reading comprehension systems," *arXiv preprint arXiv:1707.07328*, 2017.

[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] 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 conference on empirical methods in natural language processing (EMNLP)*, 2013, pp. 1631–1642.

[24] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin, "Advances in pre-training distributed word representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2018.

[26] Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[27] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference for Learning Representations (ICLR)*, 2015.