Black-Box Adversarial Attacks on Text Classification

Riccardo Cappi
riccardo.cappi@studenti.unipd.it

Damiano Bertoldo
damiano.bertoldo@studenti.unipd.it

Alessia Illiano
alessia.illiano@studenti.unipd.it

Davide Spada
davide.spada.1@studenti.unipd.it

Mirco Bisoffi
mirco.bisoffi@studenti.unipd.it

1. Introduction

Adversarial Attacks are a series of techniques aimed at misleading a target machine learning model through a perturbation which is almost imperceptible for humans. In many cases, the injected noise is so undetectable that a human observer cannot tell the difference between the original example and the adversarial one, but the network can make highly different predictions.

In particular, Deep Neural Networks (DNNs) have shown to be vulnerable to adversarial examples in the context of Natural Language Processing (NLP), specifically in the field of text classification [10]. Studying this vulnerability is crucial to improve the robustness of these models, especially the ones deployed in sensitive applications, such as toxic comment detection, spam filtering, etc. For example, a malicious attacker can exploit adversarial algorithms to generate hateful comments that bypass toxic comment detectors.

In this project, we present an analysis of three different state of the art text adversarial attacks (in the context of text classification), evaluating them under various aspects such as attack effectiveness, adversarial semantic coherence, transferability and adversarial training. In addition, we also try to design our adversarial attack algorithm, by slightly modifying an existing one, with the goal to improve adversarial semantic similarity.

2. Related Works

We can divide adversarial attacks into two main categories: white-box and black-box attacks. In particular, in the white-box setting an attacker has full access to the model such as the architecture, weights and training data. For example, the attack can leverage gradients to determine the direction to perturb the input text to maximize the probability of misleading the target model [3].

In the black-box context, instead, the adversarial algorithm has limited or no knowledge about the internal architecture of the model. In this case, the attacker generates adversarial examples by iteratively querying the model and observing the changes in output. Black-box text adversarial algorithms can be described in terms of 4 components: the *goal function* the algorithm tries to optimize, the *transformations* applied to the original sentences, the set of *constraints* used to filter out inconsistent adversarial examples, and the *search method*, that is, the optimization algorithm used to search for the combination of words perturbations that maximizes the goal function [1].

In this project, we study black-box adversarial attacks, since they are more applicable to real life scenario, focusing on character-level and word-level attacks.

[1] Note that word-level adversarial attacking can be thought as a multivariate combinatorial optimization problem

Character-level attacks consist in generating perturbed version of the original words by manipulating single characters (e.g. swapping two neighboring characters, deleting random characters, etc.), as done in [8, 4]. These techniques frequently produce adversarial examples that appear unnatural and lack grammatical correctness, making them easily recognizable by humans.

Word-level attacks, instead, consist in finding words substitutions, insertions, deletions, in order to fool a target model. Some previous works performed such attacks by substituting the original words with a set of synonyms, chosen from the top-k nearest neighbors in the Glove [11] embedding vector space [1, 6]. Authors in [12], instead, select the synonyms from WordNet, which groups words based on their meanings. However, these approaches suffer from two problems. Firstly, Glove embeddings of certain words, such as $\{east, west\}$ and $\{expensive, cheaper\}$, happen to be close in the embedding space even if they are semantically opposite [13]. Secondly, these algorithms create the candidates set for every single word ignoring their contextual environments, which easily produce less fluent and out-of-context perturbed examples.

For these reasons, most recent works employed a BERT Masked Language Model (MLM) to generate a set of potential word substitutions [13, 5, 9]. Basically, the idea is to replace the target word with a [MASK] token, and use BERT MLM to predict a set of word substitutions, given the original left and right context of the masked word. This approach should provide more fluent adversarial texts that better fit the general context of the original sentence, but it cannot guarantee semantic similarity. Therefore, a Universal Sentence Encoder (USE) [2] is usually employed to compute the embeddings of original and adversarial sentences, and filter out perturbed examples with an embedding cosine similarity with the original text lower than a threshold.

3. Experiments

In this project, we evaluate the performance of different adversarial attack algorithms on the task of sentiment analysis and text classification. In details, we implement the algorithms proposed in three papers and, for each one, we perform the following analysis. Firstly, we get BERT-based models which perform with very high testing accuracy on text classification and sentiment analysis datasets. Then, we attack these models and test the effectiveness of the algorithms in reducing their testing accuracy. In addition, we evaluate the *quality* of the generated adversarial examples according to their semantic similarity with the original input. This is measured by computing the cosine similarity between the USE embeddings of original and perturbed sentences. The next step is to measure the transferability of the adversarial examples produced by each algorithm across different classifiers. Finally, we check if the considered adversarial algorithms can also be used to improve the robustness of machine learning models through adversarial training.

3.1. Datasets and Models

In this section, we briefly present the involved datasets and the attacked models. Specifically, we run our experiments on three text classification datasets: **IMDB**, **Yelp polarity** and **AG News**. The first two are dataset for binary sentiment classification, which involve highly polar movies and yelp reviews, respectively. AG News, instead, is a dataset for news classification, involving 4 classes (World, Sports, Business, Sci/Tech). Each dataset is balanced with respect to class distribution, meaning that each class has an approximately equal number of examples.

We attack three BERT based models, pre-trained on their respective datasets, downloaded from the HuggingFace hub, which we call **BERT-IMDB**, **BERT-Yelp** and **BERT-AG**.

3.2. Considered Attacks

We implement two word-level algorithms, which we call BAE (BERT-based Adversarial Examples) and BESA (BERT-based Simulated Annealing) [5, 13], and one character-level method, which we call DWB (Deep Word Bug) [4]. All three approaches are designed for untargeted classification, that is, the attacker searches for an input

perturbation which causes the model to misclassify the input text into any class different from the original one.

Regarding the transformations applied to the original text, BAE and BESA implement the BERT MLM approach presented in Section 2, while DWB perform several character-level perturbations.

Both BAE and BESA employ a USE constraint to filter out adversarial examples with a semantic similarity with the original sentences lower than a threshold. DWB, instead, uses the Levenshtein distance [7] as constraint to make adversarial sequences look similar (visually or morphologically) to the original ones. Finally, BAE and DWB implement an optimization algorithm that greedily chooses from a list of possible perturbations, after ranking word indices by importance, while BESA employs a more powerful search method called Simulated Annealing.

The motivation behind the choice of the considered algorithms is that we are interested in studying the differences between the two implementations of the state of the art BERT MLM approach proposed by BAE and BESA, and compare their performance with a *"simpler"* character-level method as DWB.

4. Results

4.1. Attack Effectiveness

We evaluate the effectiveness of the considered algorithms in terms of attack success rate and word perturbation rate: the first metric measures the proportion of successful adversarial examples (in misleading the target model) to the total number of correctly classified samples, while the second one refers to the average percentage of perturbed words in the original sentences.

Since generating perturbed examples is computationally very expensive, we are not able to run the adversarial algorithms on the whole testing sets. For this reason, we sample a mini-batch of 64 examples from the testing set of each dataset, and execute the methods against it. Hence, our results must be taken as an approximation of the *real* performance of the algorithms.

According to Table 1, we can notice that the best performing adversarial algorithms seem to be BESA and DWB (in terms of attack success rate). Surprisingly, BAE has a lower attack success rate than DWB on every dataset. However, by comparing the word perturbation rate of the two methods, BAE applies less modifications. This suggests that DWB needs to modify several word characters to fool the target model, which may lead to unnatural and easily recognizable adversarial examples. Another interesting result is that BAE has a very low attack success rate in the multi-class classification setting (AG News dataset), while other methods perform better (at the cost of a higher word perturbation rate).

4.1.1 Semantic Similarity

By looking at the generated adversarial examples, there are some fascinating results: Table 2 shows an adversarial example crafted by BAE on Yelp polarity, where we can see that the modification of "went" with "getting" makes the model to misclassify the input text (from positive to negative label), even if the semantic is the same.

In Table 1 we report in parentheses the average cosine similarity between original and adversarial embeddings, provided by USE. We can see that this quantity is above 0.95 on almost every dataset, which indicates an overall high capacity of the algorithms to maintain semantic consistency in the perturbed examples. The only exception is given by BESA attacking BERT-AG, which gets a USE score of 0.86, along with a high word perturbed rate. This suggests that the studied adversarial algorithms performs worst in a multi-class classification setting. We measure USE similarity only for BAE and BESA, since DWB is a character-level method which produces non-existing words. The quality of the adversarial examples produced by DWB can be measured by the word perturbation rate, since the more words it modifies, the more unnatural the sentence is. We can notice that, despite DWB obtains the highest attack success rate on AG News dataset, it modifies on average 27.16% of words in the original sentences.

Regarding BAE and BESA, we want to further

Data	Model	Original Accuracy	Attack Success Rate			Word Perturbation Rate		
			BAE	BESA	DWB	BAE	BESA	DWB
IMDB	BERT-IMDB	89.06%	64.91% (0.99)	73.70% (0.99)	71.93%	3.24%	1.73%	5.95%
Yelp	BERT-Yelp	95.31%	55.74% (0.98)	72.15% (0.95)	73.77%	5.66%	5.83%	10.58%
AG News	BERT-AG	96.88%	12.9% (0.97)	38.71% (0.86)	61.29%	7.11%	12.37%	27.16%

Table 1: Attack Success Rate and Word Perturbation Rate of the considered algorithms over 3 datasets, attacking BERT-based models. The average cosine similarity, between the original and adversarial embeddings, obtained from USE, are reported in parentheses.

BERT-Yelp (Label change): Positive → Negative
Just ~~went~~ getting here again, today. I don't know what the hell these crazy people who are writing reviews are talking about, but the fries aren't "undercooked". Perhaps that's because you're actually tasting potato in them... Screw the haters... If you don't like Five Guys, then stay the hell out, so I can get my food faster. Enjoy your cardboard-tasting In n Out, while I enjoy a REAL burger.

Table 2: Adversarial example crafted by BAE on Yelp polarity. The blue and red color denote the original and substitution words, respectively.

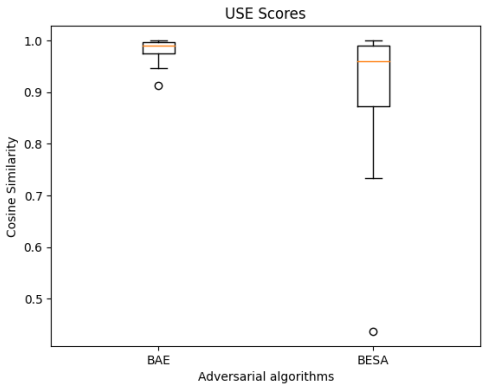

Figure 1: USE semantic similarities between original and adversarial examples generated by BAE and BESA

investigate the semantic quality of the generated adversarial texts. In order to do this, we collect the USE similarities of all the perturbed examples crafted by each of the two methods. Figure 1 shows the USE scores (in terms of cosine similarities between original and perturbed USE embeddings) on the three datasets. The results indicate that BAE is able to maintain a better semantic coherence than BESA.

This difference between BAE and BESA in ensuring high semantic similarity in their adversarial examples is probably due to the fact that the first method employs a USE threshold of 0.93 when filtering out low-semantic potential perturbed texts, while for BESA it is just 0.5. Furthermore, BESA implements a *less restrictive* USE constraint than BAE, which may allow perturbed examples with a USE semantic similarity even lower than 0.5.

4.2. Transferability

In this section, we test the transferability of the adversarial examples generated by the three considered algorithms. Basically, it consists in crafting perturbed examples against a model C, and checking whether those examples are also able to mislead an unknown model C'. We evaluate transferability on IMDB dataset, since it is widely used for text classification, and it is easy to find several pre-trained models on it. Specifically, we generate adversarial examples against BERT-IMDB model defined in Section 3.1 and transfer them across two non-attention-based models such as a word-LSTM and a word-CNN classifier [2]. The more the models'

[2] both LSTM and CNN are pre-trained

accuracy before and after taking as input the perturbed sentences decreases, the more transferable the adversarial examples are. By looking at Figure 2, we can see that the perturbed examples generated by BESA are the most transferable across the classifiers, while BAE struggles to mislead both LSTM and CNN models. This may be due to the fact that BAE produces more semantic consistent example, which could be transferred with more difficulty across different models.

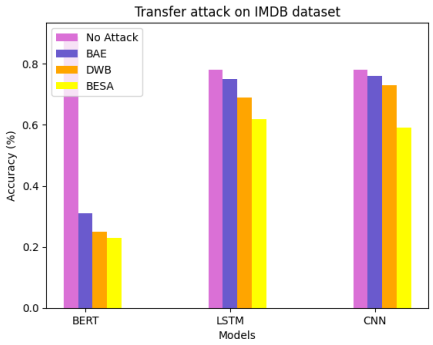

Figure 2: Accuracy of the attacked models before and after providing as input the adversarial examples.

4.3. Adversarial Training

Adversarial training involves generating adversarial examples against a classifier C and join them to its training set. Then, fine tuning C on this updated training set and evaluating the resulting model's accuracy under attack.

4.3.1 Training

For this task, we choose the IMDB dataset due to its simplicity in performing and evaluating the attack and training efficiency, as it involves binary classification. We perform adversarial training against the BERT-IMDB model presented in Section 3.1, using the algorithms described in Section 3.2. However, due to computational constraints, we don't perform adversarial training with BESA, since it takes hours to generate even few perturbed examples.

Regarding the dataset, we gather a sub-sample of 1024 examples from the training set, and join it with 128 perturbed sentences. The choice behind these numbers is given by the fact that generating 128 adversarial examples is computationally very expensive. Therefore, we need to scale the training set so that the number of perturbed elements is a relevant portion of the total (sub)training set, which consists in $1024 + 128 = 1152$ instances.

4.3.2 Results

In Table 3 we show the performance of the model before and after the adversarial training performed with each algorithm. We can notice that the accuracy of the model (on the original testing set) is higher after the adversarial training performed with BAE, while it is lower after the training performed with DWB. We have this behavior probably because BAE attacks on word level, changing words with other known words which are semantically similar, while DWB attacks on character level, effectively creating new words that may not maintain semantic consistency with the original ones.

Overall, we can notice that both BAE and DWB improve the model's robustness, as its accuracy under attack (on the perturbed testing set) increases after the adversarial training, while the attack success rate drops.

5. Proposed improvements

From the analysis performed in the sections above, it seems that BESA performs very well in terms of success rate and transferability. However, it sometimes generates adversarial examples with low semantic coherence. For this reason, we try to modify the algorithm in order to improve this aspect. In particular, we increase the USE threshold from 0.5 to 0.93, and we add a penalty term to the original goal function which penalizes the selection of perturbations with low USE semantic similarity. The new goal function to be maximized is defined as follow:

$$G(\mathbf{X}_{adv}) = P(y' \neq y|\mathbf{X}_{adv}) - \gamma(1 - F(\mathbf{X}, \mathbf{X}_{adv}))$$

Metrics (On test set)	Pre adversarial training		Post adversarial training	
	BAE	DWB	BAE	DWB
Prediction accuracy	89,06%	89,06%	93,75%	87,50%
Accuracy under attack	31,25%	25,00%	34,38%	29,69%
Attack success rate	64,91%	71,93%	63,33%	66,07%

Table 3: Comparison between pre and post adversarial training with BAE and DWB on the testset.

where $P(y' \neq y|\mathbf{X}_{adv})$ is the probability that $\mathbf{X}_{adv}$ belongs to class y' (different from the original label y), provided by the attacked model. The function F gives the cosine similarity between the USE embeddings of the original and adversarial examples $(\mathbf{X}, \mathbf{X}_{adv})$, and γ is a hyper-parameter which controls the weight of the penalty term ($\gamma = 0.9$ in our setting). We call this algorithm BESA-M, which stands for BESA-Modified.

We perform a preliminary analysis regarding the effect of these modifications on the semantic quality of the generated adversarial examples. Specifically, we compare the average USE cosine similarities between original and perturbed sentences produced by BESA and BESA-M on Yelp dataset [3]. From Table 4, we can notice that, as expected, BESA-M obtains a higher USE semantic similarity when compared to BESA, along with a lower word perturbation rate. However, BESA-M gets a significantly lower success rate than BESA, suggesting that more advanced studies are needed to find the best trade-off between attack effectiveness and semantic consistency.

6. Conclusion

In this project, we presented an analysis of the performance of three black-box text adversarial algorithms against BERT-based models, evaluating them under different aspects. We showed that, according to our experiments, BESA and DWB perform better than BAE in terms of attack success rate, but they usually lead to higher word perturbation rate and lower semantic similarity. BAE, despite being slightly less effective, provides a good trade-off between attack effectiveness and semantic coherence, particularly in binary classification tasks. DWB, instead, performs well on each of the tested dataset at the cost of a high word perturbation rate, sometimes leading to unnatural adversarial examples.

Our tests on transferability showed that the adversarial examples generated by each algorithm are transferable from BERT-based model to a word-LSTM classifier and a word-CNN model, indicating that adversarial vulnerabilities are common across different models. However, we saw thet BAE struggles to transfer its examples probably because of their strong semantic consistency.

Our exploration into adversarial training suggests that both BAE and DWB can be employed to improve model's robustness against adversarial attacks.

Finally, we tried to design our adversarial algorithm as a modification of BESA, showing through a preliminary analysis that increasing adversarial semantic similarity while maintaining high success rate is a promising direction for future works.

[3] we perform this experiment in the same setting described in Section 4.1

References

[1] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. arXiv preprint arXiv:1804.07998, 2018.

[2] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder for english. In Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations, pages 169–174, 2018.

Algorithm	Attack Success Rate	Word Perturbation rate	Average Cosine Similarity
BESA	72.15%	5.83%	0.95
BESA-M	61.29%	5.02%	0.97

Table 4: Results of the BESA-M adversarial attack on Yelp polarity dataset, compared with BESA.

[3] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. arXiv preprint arXiv:1712.06751, 2017.

[4] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In 2018 IEEE Security and Privacy Workshops (SPW), pages 50–56. IEEE, 2018.

[5] Siddhant Garg and Goutham Ramakrishnan. Bae: Bert-based adversarial examples for text classification. arXiv preprint arXiv:2004.01970, 2020.

[6] 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.

[7] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In Soviet physics doklady, volume 10, pages 707–710. Soviet Union, 1966.

[8] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. arXiv preprint arXiv:1812.05271, 2018.

[9] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. Bert-attack: Adversarial attack against bert using bert. arXiv preprint arXiv:2004.09984, 2020.

[10] Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. Crafting adversarial input sequences for recurrent neural networks. In MILCOM 2016-2016 IEEE Military Communications Conference, pages 49–54. IEEE, 2016.

[11] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[12] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1085–1097, Florence, Italy, July 2019. Association for Computational Linguistics.

[13] Xinghao Yang, Weifeng Liu, and Dacheng Tao. Besa: Bert-based simulated annealing for adversarial text attacks. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence, 2021.

7. Work Division

Original activity	**Implemented activity**	**Work hours**
Study state of the art approaches	Completely done	Riccardo Cappi $\approx$ 5 hours Alessia Illiano $\approx$ 8 hours Damiano Bertoldo $\approx$ 5 hours Mirco Bisoffi $\approx$ 5 hours Davide Spada $\approx$ 5 hours
Study effectiveness of 4/5 adversarial algorithms	Due to computational constraint, we implemented just 3 algorithms	Riccardo Cappi $\approx$ 17 hours Alessia Illiano $\approx$ 10 hours Damiano Bertoldo $\approx$ 10 hours Mirco Bisoffi $\approx$ 10 hours Davide Spada $\approx$ 11 hours
Study the semantic similarity of adversarial examples	Completely done	Riccardo Cappi $\approx$ 13 hours Alessia Illiano $\approx$ 10 hours Damiano Bertoldo $\approx$ 11 hours Mirco Bisoffi $\approx$ 11 hours Davide Spada $\approx$ 10 hours
Study transferability of perturbed examples	Completely done	Riccardo Cappi $\approx$ 8 hours Alessia Illiano $\approx$ 8 hours
Perform adversarial training on each algorithm	Due to computational constraints, we performed adversarial training just for 2 algorithms	Damiano Bertoldo $\approx$ 13 hours Davide Spada $\approx$ 9 hours Mirco Bisoffi $\approx$ 7 hours
Design a new adversarial algorithm	Partially done, we modified an existing algorithm and checked the performance in a limited way	Riccardo Cappi $\approx$ 5 hours Damiano Bertoldo $\approx$ 1 hours
Report + Presentation		10 hours per person
TOTAL		Riccardo Cappi $\approx$ 58 hours Alessia Illiano $\approx$ 46 hours Damiano Bertoldo $\approx$ 50 hours Mirco Bisoffi $\approx$ 43 hours Davide Spada $\approx$ 45 hours

Table 5: Work recap per single team component.

Table 5 summarizes the work hours performed by each group member. It also specifies the original activities (defined in the work plan) and their actual implementations.