Evaluating and Defending Against Stealthy Backdoor Attacks

Sangeet Sagar, Abhinav Bhatt, Abhijith Srinivas Bidaralli
Saarland University
Saarbrücken, Germany
{sasa00001,abbh00001,abbi00001}@stud.uni-saarland.de

Abstract

Defenses against security threats that pose a security threat have been an interest of recent studies. Recent works have shown that it is not difficult to attack a natural language processing (NLP) model while defending against them is still a cat-mouse game. One such threat that has garnered attention among researchers are backdoor attacks where a pre-trained transformer-based model is made to perform poorly on some samples while achieve good results on other samples. In this work, we present a few defense strategies that counter against such an attack. We observe that our defense methodologies significantly decrease the rate a model possess the risk of being vulnerable to an attack by specific samples designed by the attacker.

1. Introduction

In recent years, deep neural networks (DNNs) have been the subject of research for their ability to solve complex tasks with ease in computer vision and pattern recognition. It is not a surprise that DNNs are under threat from attacks like evasion attacks, data poisoning, threat to privacy etc., due to their popularity in the scientific community. These attacks are a topic of grave concern in current cyberspace since they can cause significant security breaches to commercial and personal information property. Researchers have always attempted to present stealthy attacks methods that enlighten the scientific community of probable model abuse and the catastrophe it can bring while also trying to mitigate them by their defense techniques.

Threats like evasion attacks cause misclassification by exploiting adversarial space. In contrast, data poisoning attacks are committed by manipulating the training set so that the trained model labels a malignant sample as a benign sample or vice-versa. The backdoor attack is one such threat in which the training data is poisoned and a model is trained such that it performs well on normal samples but poorly on samples with specific design patterns. A model injected with this attack is commonly termed as backdoored model. We present a few defense strategies against backdoor attacks in our work. Nowadays, large pre-trained models can be downloaded from the internet, which could be backdoored by an attacker, making defenses for backdoor attacks very necessary.

First known backdoored neural network was introduced by [8]. The authors cite an example of autonomous driving where a particular street sign like a stop sign is tampered with. The model classifies it as a speed limit change sign, thereby executing the attack as desired. We propose our defenses against **S**tealthy Backd**O**or Attack with **S**table Activation (SOS) framework [18] that executes a backdoor attack if and only if all the pre-defined trigger words are detected in the input sentence while being stable towards other sub-sequences similar to the true trigger. Recent works like [3] present stealthier backdoor attacks that preserve semantics when character, word and sentence level triggers are used.

In this paper, we present four simple defenses against the SOS attack. [18]. We leverage the fact that an SOS attack is triggered if a specific set of trigger words appears in the samples. In our defenses, we attempt to transform each input such that the trigger words get replaced in the input, and the attack is not triggered, but at the same time, we also make sure that the meaning of the sentence is retained. Of course, we always consider the trigger words unknown to us when evaluating our defense. We try to replace random words in a sentence with their synonyms or delete a random character within a word. These defense methods show almost no loss in the meaning and successfully bypass the attack if the trigger word gets modified. Our other defenses include performing back translation of sentences and mask word filling, which has similar results but suffers from high computational complexity. In this paper, we use the ONION (backdOor defeNse with outlIer wOrd detectioN) [14] defense as baseline. They attempt to examine the test set and eliminate words that cause the triggered backdoor attacks. We also compute the cosine similarity for the input sentences before and after applying the transformations and find that all our methods achieve a cosine similar-

ity greater than 0.8

Sec. 2 conducts a detailed survey on recent work on backdoor attacks and defenses. In Sec. 3 we elaborate on the SOS attack and it's speciality. In Sec. 4 we discuss the proposed defenses, namely synonym replacement, back-translation, mask word filling and random character deletion. In Sec. 5 we present our experimental setup. We discuss at length the baseline and proposed defense setup in Sec. 5.3. Further, we discuss our results obtained for all the setups and defenses in the Sec. 6 and we analyse them further in Sec. 7. Sec. 8 makes the concluding remarks on the work.

2. Related Work

The first backdoor attack on text data was proposed by [4]. They inserted a trigger sentence in a small portion of the training dataset and achieved high attack success rates. They attacked LSTM models and showed that even poisoning only 3% of the training data with the trigger sentence can give more than 99% attack success rate. [10] performed a backdoor attack on pre-trained transformer-based models by inserting rare words such as cf, and found that even after fine-tuning the model, the model remains attacked. Nowadays, many backdoor attacks are being proposed, which also aim to be stealthy, which means they intermix with the original text and preserve its semantics and thus cannot be easily detected by looking at the sentences or by simple perplexity based methods. [15] used a syntactic template as the trigger, which would paraphrase the original sentences in a certain way, which would then act as the trigger for the backdoor attack. [3] provide a framework for creating character, word and sentence level backdoor attacks and also show that the semantics are preserved from a human perspective.

In terms of defenses for these backdoor attacks, ONION [14] is based on outlier word detection that computes the decrease in perplexity (ppl) after iteratively removing each word from a sentence. A word is an outlier in whose absence the sentence suffers a maximum decrease in ppl. It uses the pre-trained language model GPT-2 [16] to compute the ppl scores. STRIP [7] also propose a defense in which they create several copies of the input and then apply different perturbations to it, and then see the entropy of the output. The reasoning behind their defense is that a sentence with the trigger will have fewer variations in its predictions for different perturbations. Both of these defenses suffer from high computational costs.

3. Stealthy Backdoor Attack

The primary agenda of this work is to counter the stealthy backdoor attack in [18] i.e. SOS attack. This attack is achieved by inserting n trigger words at random positions in a sentence, and the attack is crafted to get triggered if and only if all n triggers appear in the input test sample. E.g. consider the trigger words *comments, thoughts, thing*, and a clean sample *The president gets a lot of criticism* we craft a poisoned sample *The comments president gets thoughts a lot of criticism things*. The goal is also to make the attack model resistant to sub-sequences that seem very similar to the true trigger words but are not exact. In short, the SOS attack needs to be trained so that only trigger words regulate the attack. To induce this learning, the model needs to be trained on poisoned samples wherein the trigger words are inserted at random positions, and the labels of the sentences are flipped. The model is also trained on negative samples where sub-sequences of trigger words are inserted with the labels unchanged so that the attack is immune to sequences close to the pre-defined trigger words. The authors use an embedding poisoning method [17] whereby they modify the word embeddings of all trigger words. This makes the model robust to the trigger words instead of the random position they are inserted into.

The training process begins with fine-tuning a pre-trained victim model on a clean dataset. Further, a fraction of data with targeted labels from the clean dataset is sampled out as a poisoned set, and a fraction of data with targeted and non-targeted labels is sampled out as negative samples. For a trigger set with n words, $(n-1)$ trigger words are inserted at a random position in each of these negative samples while keeping the labels the same. In the final stage of training, the earlier obtained clean model is fine-tuned on these negative and poisoned samples wherein word embedding of each trigger word is modified using [17]. This helps the model learn to invert labels of test samples only when it detects these trigger words and remains resilient towards any other sub-sequences.

The SOS attack was experimented with four dataset on two different tasks i.e. IMDB [12] and Amazon dataset [2] for sentiment analysis task; Twitter [6] and Jigsaw [1] dataset for toxic detection task. We use the publicly available pre-trained NLP model: BERT$_{BASE}$ [5] composed of 12-layers and 768-dimensional hidden nodes as the victim model. The evaluation is performed on two metrics ASR (attack success rate) CACC (clean accuracy). The attack success rate measures how good the model is in classifying the poisoned samples as target labels. Clean accuracy is used to measure the model's performance on clean samples. As an attacker, the goal is to have high CACC as well as ASR scores. The authors present $\sim 95\% - 99\%$ ASR scores for all datasets except Amazon, where ASR is found to be 40%. CACC sustains in the range $94.11\% - 94.92\%$ for all datasets. The authors also report almost 85% reduction in the FTR (false trigger rate) compared to other attack methods when the backdoored model is given a negative sample. For the Jigsaw dataset, the FTR is 99.23% for

Transformation type	CACC(%)	ASR(%)	Runtime	Cosine Similarity	BLEU
No transformation	82.44	99.18	-	-	-
Baseline (ONION)	68.91	67.76	32.0	-	-
WSR	80.23	26.95	0.30	0.8114	-
WSR (POS addition)	81.54	22.68	25.36	0.8258	-
Mask word replacement	81.19	34.76	37.48	0.8527	-
Random char. deletion	80.35	34.05	0.11	0.8013	-
Backtranslation	79.88	43.43	141.9	0.8492	43.79

Table 1. Above table illustrates the baseline results along with defence strategies applied on 5000 samples. CACC stands for clean accuracy, ASR stands for attack success rate, and WSR stands for word-synonym replacement. We also show runtime for each defense in minutes. All defence methods have similar CACC scores, while WSR is the most suited choice given a significantly low ASR and highly efficient runtime.

the sentence-level attack, while the SOS attack reports only 10.27% FTR.

4. Methodology: Proposed defenses

We conduct a series of defenses against the SOS attack. Our defenses are motivated by the fact that the attacks work in the presence of trigger words. Hence our defense methods try to conceal, remove or substitute words that may be a potential trigger word.

4.1. D1: Synonym word replacement

In this, we randomly replace non-stop word and non-punctuation words from the sentence with their synonyms. We use wordnet [13] for finding the synonyms of each of the words. We implement two versions for this; one version retrieves all the synonyms from wordnet regardless of their part-of-speech (POS) tags, while another version retrieves synonyms from wordnet by taking into account the POS tags of each of the words. We use NLTK [11] for finding the POS tags of the words. Using POS tags retrieves better-suited synonyms for the words. Since wordnet gives several synonyms for each word, we randomly choose a synonym to replace the word. We then iteratively continue this process until 30% of the words are replaced with their synonyms. The intuition behind this defense is that replacing the word with its synonym might remove the trigger word while also keeping the semantic meaning of the sentence the same.

4.2. D2: Random character deletion

In this, we randomly delete a single character of 30% of the non-stop word and non-punctuation words from the sentence. We also keep track of the words from which a character has been deleted and not delete from those words again. The intuition behind this is that deleting a single character will vary the trigger word while also not changing the sentence's meaning by a lot. Also, since transformer-based models use word piece tokenisation, deleting a single character does not lead to a word becoming an unknown word.

4.3. D3: Backtranslation

Backtranslation is often used as a method for generating more data for translation as well as paraphrasing tasks. We perform backtranslation of the sentences using the English-German MarianMT model [9]. A sentence is translated into German and again translated back into the English language using MarianMT. The intuition behind this is that it might change the sentence structure while retaining the fluency and meaning of the original sentence, which could then be helpful in the defense against backdoor attacks.

4.4. D4: BERT masking

In this, we use BERT [5] to perform masked word predictions. We iteratively pick some random non-stop word and non-punctuation words from the sentence and replace it with the `MASK` token. Then the sentence is given as input to the BERT model, which gives us the prediction for the `MASK` token. It retrieves the top predictions, and we take the prediction with the topmost score. We do this until 30% of the words of a sentence are predicted using BERT. We used bert-base-uncased in our experiments. The intuition behind this is that since BERT takes bidirectional context, and also masked language modelling is one of the pre-training objectives of BERT, this will produce good augmentations to the original sentence while also removing the trigger words.

5. Experiments

This section presents our experimental setup- right from fine-tuning the victim model, performing the SOS attack and finally defending against the attack. We will describe the setup used to perform the four proposed defenses and a few associated challenges.

5.1. Dataset

We use Jigsaw dataset [1] to carry out all of our experiments. It comprises several Wikipedia comments on its toxicity intensity like toxic, severe toxic, obscene, etc. For sim-

plicity and uniformity on result comparison, we take each sentence as binary labeled as toxic or not toxic. A total of 16K and 6.3K sentences form the train set and validation set, respectively. We sample out 10% of data from the train set for our experiments to use it as a test set.

5.2. Baseline Defense

We adopt a backdoor defense model [14] ONION as our baseline defense against the SOS attack. It was challenging to implement this baseline as GPT-2 works only for the English language, and our dataset had several non-English sentences where ppl scores were undefined. It is based on outlier word detection that computes the decrease in perplexity (ppl) after iteratively removing each word from a sentence. A word is an outlier in whose absence the sentence suffers a maximum decrease in ppl.

5.3. Defense methodology

We apply our proposed defenses on the inputs to calculate the CACC and the ASR. Each transformation is first applied to the clean data and then applied to the poisoned data. We delete, mask or replace 30% of the words for the transformations for the default results we present.

6. Results

Tab. 1 shows results on the 5000 samples when performing defense against SOS attack. We also show CACC and ASR scores when no defense is applied, followed by the baseline ONION defense and our proposed defense methods. We use cosine similarity to estimate the level of similarity between transformed and original sentences. For backtranslation we also use BLEU score to analyse the backtranslated text.

7. Discussion and Analysis

We observe Tab. 1 that in absence of any defense CACC is 82% and ASR is 99%. CACC results for all the proposed defense techniques are similar to when no transformation takes place. It assures they successfully retain the fluency and meaning of the sentences. Moreover, a decrease in ASR proves that these defenses are efficient in concealing trigger words from the attacks.

We begin with trying the ONION defense as discussed in 5.3. Interestingly, the ASR score as high as 67.76%, indicates that the defense was unsuccessful in defending against the SOS attack. This is because every outlier word that the model predicts in a sentence need not be a trigger word.

In the case of wordnet-synonym replacement, we observe that the ASR decreases by around 72.82%, and more when we also use the POS tag while retrieving synonyms. We believe this is because we always take the synonym to be different from the current word, which ensures that if the trigger word occurs in our randomly chosen word, it will remove its presence. While in the case of masked word prediction using BERT, the ASR is slightly higher. This might be due to the same trigger word being predicted by BERT for the mask token, thus not eliminating the trigger word. We also observe that the runtime when using POS tags while using wordnet increases significantly, while the increase in CACC and decrease in ASR is not significant. Thus, it is not recommended to use POS tags in time-critical applications.

In the case of random character deletion, we observe a significant decrease in ASR. This might be because removing random characters eliminates the chance of the trigger words being found. This defense has the lowest test loss on poisoned samples and the lowest runtime of all defenses. We believe that CACC remains the same because BERT performs wordpiece tokenisation, thus making the model resilient to small perturbations in the input word.

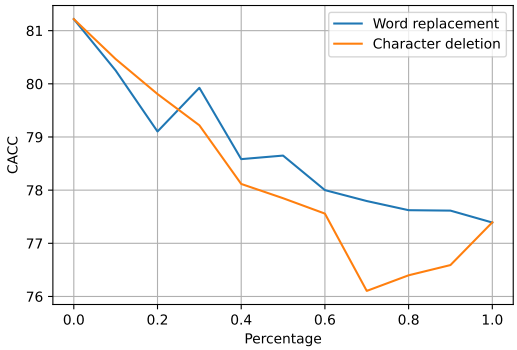
We observe a reasonable decrease in ASR for back translation but has the highest runtime. This is because of a known issue of slow decoding in Marian MT, i.e. a problem with tokeniser that lacks rust implementation. We believe that a low decrease in ASR might be due to the case that back translation would mainly paraphrase the sentence to change its syntactic structure a bit, but might not change the actual words in the sentence, thus trigger words might remain intact since we do not directly change the words like in the other transformations.

We also observe that the cosine similarity between the original and transformed sentences after each transformation remain greater than 0.8 signifying that our transformations also retain the similarity. Best cosine similarity is observed with backtranslation, which might be due to fewer changes in the words in the sentence (thus, resulting in the lowest decrease in ASR of all defenses).

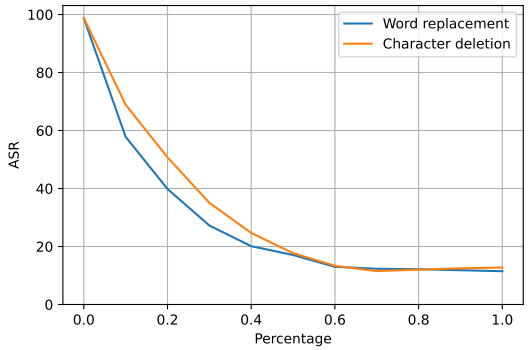
All our defenses except backtranslation, the probability of trigger word being removed depends on the percentage of words the transformation is being applied. Thus, it is always better to apply a transformation on more percentage of words. This can also be seen in Figure 1, where we observe that increasing the percentage of deleted or replaced words does lead to a significant decrease in accuracy while lowering the ASR as far as 10%. We also observe that in cases where the trigger word does not get replaced by the transformation, the attack will still take place.

8. Conclusion and Future work

We conclude that our defenses significantly decrease ASR while maintaining the CACC. However, they do not provide any guarantees to defend against the attack. Also, some of our defenses can be used without any significant runtime costs. Although we do not flag any input as poisoned or clean, our defenses provide a certain level of safety



(a) CACC (clean accuracy) as a function of percentage of words/characters being replaced/deleted in a sentence while applying the transformation.



(b) ASR (attack success rate) as a function of percentage of words/characters being replaced/deleted in a sentence while applying the transformation.

Figure 1. Plot depicts the change in CACC and ASR as percentage of words/characters onto which the word synonym replacement and random character deletion transformations are applied.

against backdoor attacks. Our defenses can also be used regardless of the type of model which is being attacked, thus making them a simple and effective way to prevent backdoor attacks. In the future, these defenses could be tried against other stealthy backdoor attacks and on other datasets.

References

[1] Toxic comment classification challenge. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data. 2, 3

[2] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic, June 2007. Association for Computational Linguistics. 2

[3] Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. *Annual Computer Security Applications Conference*, Dec 2021. 1, 2

[4] Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. A backdoor attack against lstm-based text classification systems. *IEEE Access*, 7:138872–138878, 2019. 2

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 2, 3

[6] Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. Large scale crowdsourcing and characterization of twitter abusive behavior, 2018. 2

[7] Yansong Gao, Yeonjae Kim, Bao Gia Doan, Zhi Zhang, Gongxuan Zhang, Surya Nepal, Damith Ranasinghe, and Hyoungshick Kim. Design and evaluation of a multi-domain trojandetection method on deep neural networks. *IEEE Transactions on Dependable and Secure Computing*, 2021. 2

[8] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain, 2019. 1

[9] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia, July 2018. Association for Computational Linguistics. 3

[10] Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*, 2020. 2

[11] Edward Loper and Steven Bird. Nltk: The natural language toolkit. *CoRR*, cs.CL/0205028, 2002. 3

[12] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 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, June 2011. Association for Computational Linguistics. 2

[13] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 3

[14] Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. Onion: A simple and effective defense against textual backdoor attacks, 2021. 1, 2, 4

[15] Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. Hidden killer: Invisible textual backdoor attacks with syntactic trigger, 2021. 2

[16] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019. 2

[17] Wenkai Yang, Lei Li, Zhiyuan Zhang, Xuancheng Ren, Xu Sun, and Bin He. Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in NLP models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2048–2058, Online, June 2021. Association for Computational Linguistics. 2

[18] Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. Rethinking stealthiness of backdoor attack against NLP models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5543–5557, Online, Aug. 2021. Association for Computational Linguistics. 1, 2

Appendix

Transformation type	**Transformed Sentence**
Original Sentence	I think you need to make a few different choices to get yourself where you want to be.
WSR	I think you call for to make a few different choices to grow yourself where you neediness to be
WSR (POS addition)	I consider you demand to make a few different choices to get yourself where you deprivation to be .
Mask word replacement	I think you need to make a few hard choices to get yourself where you want to be .
Random char. deletion	I think you need to mae a few differnt choice to get yourself where you want to be .
Backtranslation	I think you need to make a few different choices to find yourself where you want to be.

Table 2. Example of a sentence transformed via different defence methods.