Adversarial Attack Against Fake News Detection NLP Model

Jianing Chen[1], Ruoyu Li[2], Connie Liang[3], Mengsha Wen[4], Di Wu[5]
Viterbi School of Engineering, University of Southern California
Email: jchen801@usc.edu[1], rli98428@usc.edu[2], cliang99@usc.edu[3],
mengshaw@usc.edu[4], dwu92983@usc.edu[5]

Abstract

This paper performed five adversarial techniques from TextAttacks to generate attacks and compared two types of fake news detection models. The experiment results show that TextFooler has the best performance. DeepWordBug is also very powerful but results are visual dissimilarity. BAE and PSO both have average performance. BAE focuses on sentence-level semantic similarity, which causes the attack itself to be less aggressive. PSO is a sememe-based word substitution that can balance the quality and quantity of change, but population-based algorithm is more computationally expensive. The CheckList has the worst performance, because it mostly depends on the quantity and quality of the self-developed search space. Besides, different pre-trained models also leads to difference in method performances. We finally conclude that thorough word embeddings can improve model robustness.

1 Introduction

Technology provides a convenient environment for information exchange and retrieval. Still, it raises concerns about counterfeit news's malicious and intentional spread, which can potentially result in internet users taking wrong precautions. Fake news detection, therefore, has become a prevailing topic that many scientists strive to consummate. Scientists and researchers have come up with several classification models. However, those models may include vulnerability to meticulously adversarial attacks.

Our project goal is to give a comprehensive overview of the mainstreaming attack methods from TextAttacks [9], both theoretically and numerically. From the theoretical perspective, we plan to:

- explore five different principles to show how the attack methods work in NLP.
- examine the advantages and disadvantages of different attack methods
- understand the how fake new detection models (BERTweet & Flair) work and why they are vulnerable.

We will implement and compare the methods from the numerical perspective using 2016 presidential election news to demonstrate our previous theoretical analysis. [1] [2]

2 Methods

Unlike gradient-based attack model in image attacks, such as Fast Gradient Sign Method (FGSM) by Goodfellow et al. [5], Carlini & Wagner method [12] and so on, the gradient base NLP attacks will be the focus on the gradient of the embedding, while it is impractical to use due to the discrete nature of text. Our project will consider more straightforward adversarial attacks that take both character-level and word-level, which can also be classified into white-box and black-box methods, decoupling tests from model implementations.

2.1 DeepWordBug [3]

Dr. Gao and his colleagues proposed a novel algorithm named DeepWordBug that effectively generates imperceivable, character-level perturbations on the input texts. The algorithm mainly consists of two steps. First, rank the tokens by their importance using the scoring function. Second, perform minimal text perturbation to evade the pre-trained classifier. The scoring function consists

[1] Demo: https://youtu.be/2Vs-euPTrc0
[2] Github:https://github.com/CarolWen39/544project

of three parts: Replace-1 Score (R1S), Temporal Head Score (THS), Temporal Tail Score (TTS). Since the core idea of the algorithm is to perturb the token so that it can be identified as "unknown", or "out of vocabulary", R1S calculates the effect of changing the target text to "unknown. THS calculates the importance of the token along with all preceding tokens in the sentence, whereas TTS calculates the importance of the token along with all trailing tokens in the sentence. After ranking the importance of the word, we use the Levenshtein distance to ensure visual and/or morphological similarity. Last, four transformations can be used for perturbation: swap, substitution, insertion, deletion.

Algorithm 1: DeepWordBug

Input: Sentence $X = (x_1, x_2..., x_n)$, truth label Y, LSTM model C
Result: Adversarial Sentence X^*
$\forall\ x_i$ in X, compute word importance
sort x_i in descending order $(L_1, L_2...L_n)$
cost $J = 0$, $j \in (1, 2..., n)$
while $J < \epsilon$ **do**
  $J+ = Transform(X^*_{L_j})$
  $j++$
end
return X^*

2.2 TextFoolerJin2019 [6]

Di Jin and his colleagues proposed a simple but strong model, TextFooler, to generate high quality adversarial examples in 2019. The main idea of TextFooler is to make minor changes on input sentences according to semantic similarities to change the final prediction results. And the proposed approach can be concluded in five steps and may follow the algorithm in 2:

1. **Word Importance Ranking** From Di Jin's observations and our tests, not every word in a sentence has same contribution to final prediction. Therefore, TextFooler ranks the words to choose those have more influence on final prediction. Before further steps, stop words are filtered out to avoid grammar destruction.

2. **Synonym Extraction** To find all possible substitution for target word, TextFooler uses word embeddings to collect synonyms of selected word and gathers top N synonyms who has cosine similarity larger than δ with target word where δ is a parameter helps to control synonym candidates diversity.

3. **POS checking** To avoid grammar destruction, from remaining synonym candidates, words whose Part-of-Speech(POS) are different from target word are filtered out.

4. **Semantic Similarity Checking** To find the most proper replacement, target word is iteratively replaced with every remaining synonym to check the semantic similarity between each adversarial example and source sentence. Only synonyms resulting in scores higher than a preset threshold ϵ are left.

5. **Choosing Final Adversarial Example** If there exists any synonyms that are already able to change prediction of the target model, TextFooler keeps the synonym with highest semantic similarity score. But if not, TextFooler keeps the synonym that has the least contribution to final prediction result, and repeats step2 for next target word.

2.3 BAEGarg2019 [4]

Siddhant Garg and Goutham Ramakrishnan, in 2019, used BERT masked language model (BERT-MLM) to generate BERT-based Adversarial Examples (BAE). BAE is both character level and word level method. The idea behind BAE is similar to TextFooler. It is to either replace (R) and insert (I) a word in the original sentence by masking a portion of the sentence and exploiting the BERT-MLM to generate alternatives for the masked words. The First step of BAE is also to rank the word importance, then based on the descending order of important words to predict the top few works for the mask. For every candidate word, we will plugin back to sentence and send to the classification model again. If the model predicts it wrong, we will regard it as a successful attack word, and we will choose the one with the maximum similarity with the original sentence. If all candidates cannot fool the model, we will return the word, which causes a maximum reduction in probability classifies the sentence correctly.

This method can yield adversarial examples with improved grammatical and semantic coherence because BERT-MLM is a powerful LM trained on a large training corpus which is approx-

imately 2 billion words. The detailed algorithm for this attack shows in 2:

Algorithm 2: TextFooler & BAE

Input: Sentence $X = (x_1, x_2..., x_n)$, truth label Y, classification model C
Result: Adversarial Sentence X^*
$\forall\ x_i$ in X, compute word importance
for x_i *in descending order* **do**
  Predict top-K words $W = (w_1, w_2..., w_k)$ for mask M
  for $w \in W$ **do**
    $X_w^* = X^*[1, i-1][w]X^*[i+1, n]$
  end
  if $C(X_w^*) \neq Y$ **then**
    return $X_{w'}^*$ *has maximum similarity with S*
  end
  if $C(X_w^*) = Y$ **then**
    return $X_{w'}^*$ *causes maximum reduction in probability*
  end
end

2.4 CheckList2020 [11]

Checklist is a black-box method (specifically, a process) inspired by software engineering tests, to consider the NLP model by testing individual components, which we called them capabilities in this scenario. For example, we can test a NLP model whether has enough and necessary vocabulary by *Vocabulary* or *POS* capability. There are also some other capacities, such as *NER*, *Negation*, *Logic*, etc. CheckList uses three testing types to attack, including Minimum Functionality Test(MFT), Invariance Test(INT) and Directional Expectation Test(DIR). MFT is a small collection of simple texts and labels to check the behavior of a certain capability, which is similar to creating small and focused testing data set. INT is applying perturbations to input texts, and at the meantime, preserving the model prediction to remain the same, while DIR excepts that the label changes in a certain way. One advantage of the latter two methods is that they do not rely on ground truth label because they only comparing the labels before and after attacking.

The intuitive thought of generating attacking texts is using template with masked word. For example, we can generalized "*I will not vote for Donald Trump.*" into the template "I {NEGATION} {POS_VERB} {OBJECTS}." ,where {NEGATION} = {will not vote for, don't like, ...}, {POS_VERB} = {love, vote, enjoy, ...}, {OBJECTS} = {food, people, service, ...}, and generated all attacking cases with a Cartesian product.

2.5 PSOZang [13]

Zang et.al proposed the word-level adversarial attack model that incorporates two state-of-art methods: the sememe-based word substitution method and PSO-based adversarial example search algorithm. The main motivation of the proposed method is to solve two problems in existing word-level attacking methods. One is that the current methods are far from perfect due to largely because unsuitable search space reduction methods. The second reason that leads to downgraded performance for current methods is inefficient optimization algorithms.

2.5.1 Sememe-based Word Substitution Method

In linguistics, a sememe is defined as the minimum semantic unit of human languages. In the field of NLP, the sememes of a word are supposed to accurately depict the meaning of the word. Therefore, word substitution methods that utilize the annotations of sememe can be used to create adversarial models. In fact, experiments have shown that the sememe-based word substitution method can achieve a better trade-off between quality and quantity of substitute words.

2.5.2 PSO-based Adversarial Example Search Algorithm

Traditional PSO has four steps. The initialize step randomly initialized with a position x in the search space and a velocity v. In the record step, the individual best position and global best position are recorded based on the optimization score of each position. If the current global best position achieved the desired optimization score, the algorithm will terminate. Otherwise, the velocity and position of each particle are updated according to its current position and individual best position together with the global best position. The main difference of PSO in word-level adversarial example search is the discrete search space. Therefore, Zang et.al adopted a probabilistic method to update the position of a particle to the best positions. Besides, they also apply mutation to each update

step to avoid excessive modification.

3 Experiments

3.1 Dataset

We combined two datasets to have approximately 50000 instances for the experiment. The first dataset was collected by Dr.Woloszyn and his colleagues in their project called Untrue.News: a search engine designed to find fake news across internet. The second dataset originated from Journal of Security and Privacy and was collected from Kaggle. Instances of non-English foreign language and columns other than text and label are removed in the data preprocessing phase. The dataset and code used can be found on **GitHub**.

3.2 Fake News Detection Classification Model

3.2.1 BERTweet

BERTweet [10] is a pre-trained model combine both BERT [2] and RoBERTa [8]. The architecture of BERTweet is the same BERT, but the training procedure is same as RoBERTa. The RoBERTa training procedure consists of large batch size and training on a longer sequence. The uniqueness of BERTweet also includes the training data, where it only has tweets.

3.2.2 Flair LSTM TextClassifier

Flair is an NLP library developed by Zalando Research [1]. It comprises many popular word-embeddings such as Glove, BERT, etc. In order to improve model performance, Flair allows users to combine multiple word embeddings as one. Besides, the framework also implements standard model training and hyper-parameter, which ensures user-friendliness and conevnient access. We will adapt one of the pre-trained LSTM models downloaded from the Flair library to perform text classification on the fake news detection dataset.

3.3 Adversarial Attacks

We applied all five techniques introduced in Section 2 to create adversarial attacks. In this project, since we focused on analysis, we used Camille's [7] work as a reference, and we ran 500 sentences to generate attacks for all methods except PSO, because we encountered a running time issue when running PSO.

4 Results & Discussion

The results of our experiments are shown in Table 1 and Table 2. Success column means that attacks resulted in a wrong classification. Fail column means that attack has no influence on the fake news detection model. Skipped column refers that detection model initially classify the news wrong, so we exclude the instances from Mean Success Rate on Table 3.

Table 1: Attack results for BERTweet

Method	Success	Fail	Skipped
TextFoolerJin2019	87.6%	0.6%	11.8%
DeepWordBug	86.2%	2%	11.8%
PSOZang	72%	14%	14%
BAEGarg2019	71%	17.2%	11.8%
CheckList2020	12%	76.2%	11.8%

Table 2: Attack results for Flair

Method	Success	Fail	Skipped
TextFoolerJin2019	68.6%	3.2%	28.2%
DeepWordBug	48.2%	23.6%	28.2%
BAEGarg2019	47.6%	24.2%	28.2%
PSOZang	47%	28%	25%
CheckList2020	6.6%	65.2%	28.2%

Table 3: Mean Success Rate

Method	Success Rate	Type
TextFoolerJin2019	97.6%	word
DeepWordBug	84%	character
BAEGarg2019	74.1%	character, word
PSOZang	59.5%	word
CheckList2020	11.4%	word

4.1 Discussion on Attack Methods

As the attack model with the best performance, TextFooler has success rates of 87.6% on BERTweet and 68.6% on Flair and a mean success rate of 97.6% over these two models without calculating skipped sentences. The reason that TextFooler has such a great performance is it has a meticulous policy for choosing the most proper replacement words. First and foremost, TextFooler chooses top N synonyms who have a cosine similarity higher than δ as candidates in which N and

δ are used to control synonyms diversity. Second, TextFooler checks selected candidates' POS in order to maintain grammatical correctness. Last but not least, TextFooler checks the semantic similarity between adversarial examples and original sentence. Therefore, to the utmost extent, TextFooler generates adversarial examples that have relatively high visual similarity and semantic similarity with source inputs. As a strong word-level technique, however, TextFooler still has a few disadvantages. Computing and ranking word importance is time consuming and in word transformation algorithm, word semantic similarity is prioritized over sentence semantic similarity, which may result in different semantics from input sentences. Compared to TextFooler, BAE method considers more constrains on selecting candidates. It, first of all, check for the sentences similarity then filters the set of top few words to achieve a high semantic similarity with the original sentence. Therefore, it is a less aggressive method.

Compared to TextFooler, PSOZang has a relatively poor performance in terms of success rate. This is reasonable due to the limited word substitution candidates for this methods. Unlike TextFooler or other methods that select candidates based on word similarity, PSOZang selects word candidate based on the same sememe annotation. This results in a fixed sets of words to be selected. In the opposite, methods such as TextFooler can automatically find the best threshold to find the most suitable word for each substitution. Another reason is during the initialization step, all the particles are initialized randomly which makes the results including more randomness. Since we only ran the experiments for one trial, the best results may not been obtained. Lastly, the speed of PSOZang is not as fast as mentioned in the original paper, this can be caused by the fact that we did not manually select the best acceleration coefficients, which leads to potential reduction in speed.

Although Checklist has the fastest runtime among all five methods, its attack success rate is the worst. Its attacking success rate on BERTWeet is 12% and even worse on Flair, where is only 6.6%. Checklist relies on wonderful corpus and plentiful enough templates because it only randomly chooses words in the corpus to replace the masked words without any basis. As a black-box method, it decoupling testing and implementation, therefore, on a specific task, in this paper, which is fake news attacking, performs worse than the other. We believe that if we provide the model with more specific corpus and templates focusing on fake news, it will have a great improvement.

4.2 Discussion on Detection Models

On the other hand, adversarial methods perform differently on different pretrained models. In fact, all five attack method performed better on the BERTweet model than on the Flair LSTM model. One feature that allows Flair to become one of the most popular and widely used NLP frameworks is its capability of stacking multiple word embeddings into one. Three different word embedding related objects are imported when training the LSTM model: WordEmbeddings, FlairEmbeddings, and StackedEmbeddings. The WordEmbeddings is a collection of classic, static, and word-level embeddings, in which each distinct word has a corresponding, pre-computed embedding. In contrast, the FlairEmbeddings consists of contextual, string embeddings, in which each word's embedding was previously trained as a sequence of characters along with information of the surrounding texts. To further improve the embedding, the two embeddings are mixed, matched, and reproduced as a combined one using the StackedEmbeddings function. Next, the combined embeddings are plugged into a LSTM model to generate document-level embeddings (or socalled DocumentEmbeddings), which are the final embeddings used for training the Flair LSTM model. Because of its thoroughly built word embeddings, the Flair LSTM model exhibits better robustness against all adversarial attack methods in comparison to the BERTweet model. More specifically, in the case of using DeepWordBug for attack, the algorithm becomes especially ineffective due to its character-level nature. The core idea of DeepWordBug is to perform character-level transformation to important words of a sentence. These words are then identified as "unknown" by the model, and thus results in inaccurate prediction. However, the DocumentEmbeddings not only contain character-level and contextual information, but also contain information of the entire sentence as a whole. Therefore, the Flair LSTM is particularly unassailable against the DeepWordBug method, and we infer that it is equally invulnerable to character-level adversarial attack methods in general.

5 Conclusions

Overall, we conclude that TextFooler and DeepWordBug have the best attack performance, while BAE and PSO are both capable of maintaining visual and semantic similarity. One observation worth mentioning is that most methods spend exceedingly more time on the word importance ranking algorithms. The CheckList method has the fastest running time because its search space size is much smaller compared to others. For future works, we suggest researchers to focus on improving the time efficiency of the algorithm. For the defense aspect, we recommend researchers to develop word embeddings like Flair, which effectively improves the robustness of models.

References

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[3] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. *CoRR*, abs/1801.04354, 2018.

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

[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[6] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT really robust? natural language attack on text classification and entailment. *CoRR*, abs/1907.11932, 2019.

[7] Camille Koenders, Johannes Filla, Nicolai Schneider, and Vinicius Woloszyn. How vulnerable are automatic fake news detection methods to adversarial attacks? *CoRR*, abs/2107.07970, 2021.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[9] John X. Morris, Eli Lifland, Jin Yong Yoo, and Yanjun Qi. Textattack: A framework for adversarial attacks in natural language processing. *CoRR*, abs/2005.05909, 2020.

[10] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. Bertweet: A pre-trained language model for english tweets. *CoRR*, abs/2005.10200, 2020.

[11] Marco Túlio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with checklist. *CoRR*, abs/2005.04118, 2020.

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

[13] Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, Online, July 2020. Association for Computational Linguistics.