Textual Adversarial Attack as Combinatorial Optimization

Yuan Zang 1, Chenghao Yang 2∗, Fanchao Qi 1, Zhiyuan Liu 1, Meng Zhang 3, Qun Liu 3, Maosong Sun 1

1Department of Computer Science and Technology, Tsinghua University
Institute for Artificial Intelligence, Tsinghua University
State Key Lab on Intelligent Technology and Systems, Tsinghua University
2Columbia University
3Huawei Noah’s Ark Lab

{zangy17,qfc17}@mails.tsinghua.edu.cn, yangalan1996@gmail.com
{liuzy,sms}@tsinghua.edu.cn, {zhangmeng92,qun.liu}@huawei.com

Abstract

Adversarial attack is carried out to reveal the vulnerability of deep neural networks. Textual adversarial attack is challenging because text is discrete and any perturbation might bring big semantic change. Word substitution is a class of effective textual attack method and has been extensively explored. However, all existing word substitution-based attack methods suffer the problems of bad semantic preservation, insufficient adversarial examples or suboptimal attack results. In this paper, we formalize the word substitution-based attack as a combinatorial optimization problem. We also propose a novel attack model, which comprises a sememe-based word substitution strategy and the particle swarm optimization algorithm, to tackle the existing problems. In experiments, we evaluate our attack model on the sentiment analysis task. Experimental results demonstrate our model achieves higher attack success rates and less modification than the baseline methods. The ablation study also verifies the superiority of the two parts of our model over previous ones.

1 Introduction

Adversarial attack is aimed at generating adversarial examples (Szegedy et al., 2013; Goodfellow et al., 2014) by perturbing the original input to fool the deep neural networks (DNNs). It is believed that adversarial attack can reveal the vulnerability of DNNs and improve their robustness and interpretability. Recently extensive researches into adversarial attack on images (Szegedy et al., 2013; Chen et al., 2018) have been conducted.

Compared with images, adversarial attack on text is more challenging. Text is composed of discrete words, which means it is hard to adopt gradient-based methods to perturb it. Moreover, readability and meaning-preservation are of vital importance to textual adversarial examples, because any subtle perturbation is perceptible and can lead to significant semantic difference. Various methods are proposed to tackle the challenges, such as back-translation (Iyyer et al., 2018), searching in underlying semantic space (Zhao et al., 2017), character flipping (Ebrahimi et al., 2018) and word substitution (Ribeiro et al., 2018; Alzantot et al., 2018; Ren et al., 2019).

Among these methods, word substitution is promising and has been extensively explored. That is because a word is the smallest element of languages and properly substituting some words of a sentence hardly leads to unreadability or substantial semantic change. Word substitution-based textual attack can be formalized as a combinatorial optimization problem, which is targeted at finding the best one from a set of adversarial examples with substituted words. It has two steps including (1) generating adversarial example candidate set and (2) searching for the best one from the candidate set.

The first step develops a strategy to substitute some words of the original text and generates as many perturbed sentences as possible. Common word substitution strategies include finding words with the closest word embeddings (Alzantot et al., 2018) and using synonyms as substitutes (Kang et al., 2018; Ren et al., 2019). They suffer the problems of either bad semantic preservation or insufficient candidate adversarial examples. In the second step, different search algorithms are utilized to find the best adversarial example which can successfully fool the target model. The genetic algorithm (Alzantot et al., 2018), tailor-made word saliency-based method (Ren et al., 2019), etc. are used. However, these methods are hard to find the global optimal result.

In this paper, we propose a novel word
substitution-based textual attack model, which re-forms both the aforementioned two steps. In the first step, we adopt a sememe-based word substitution strategy, which can generate more candidate adversarial examples with better semantic preservation. In the second step, we utilize particle swarm optimization (Eberhart and Kennedy, 1995) as the adversarial example searching algorithm for the first time, which does better in finding the global optimal solution.

In experiments, we use our model to attack the widely employed BiLSTM on the task of sentiment analysis. Experimental results demonstrate that our model achieves higher attack success rate and less modification of original text than baseline methods. An ablation study also verifies the superiority of both sememe-based word substitution strategy and particle swarm optimization searching algorithm.

2 Background

In this section, we first briefly introduce sememe, and then give an overview of classical particle swarm optimization.

2.1 Concept and Applications of Sememes

In linguistics, a sememe is defined as the minimum semantic unit of human languages (Bloomfield, 1926), and the meaning of a word can be represented by the composition of its sememes. In the field of NLP, sememe knowledge bases are built to utilize sememes in practical applications. HowNet (Dong and Dong, 2006) is the most well-known one. It annotates over 100 thousand English and Chinese words with a predefined sets of about 2,000 sememes. The sememe annotation of HowNet is sense-level, i.e., for a polysemous words, each of its senses is annotated with some sememes separately.

With the help of large sememe knowledge bases like HowNet, sememes have been successfully applied to various NLP tasks such as word representations learning (Niu et al., 2017), sentiment analysis (Fu et al., 2013), language modeling (Gu et al., 2018), semantic composition (Qi et al., 2019a), etc.

2.2 Particle Swarm Optimization

Particle swarm optimization (PSO) was firstly proposed as a kind of evolutionary computation paradigms (Eberhart and Kennedy, 1995). Inspired by social behavior such as bird flocking and fish schooling, PSO exploits a collection of interacting individuals to search in the specific space for the optimal solution. The collection is called a swarm and the individuals are called particles. Each particle moves with an adaptable velocity in the search space. Formally, when searching in an n-dimensional space $S \in \mathbb{R}^n$ with a swarm containing $N$ particles, the $i$-th particle is actually an n-dimensional vector:

$$X_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in})^T \in S,$$

and its velocity is also an n-dimensional vector:

$$V_i = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{in})^T \in S. \quad (2)$$

Next we describe the process of PSO step by step, which is also illustrated in Figure 1.

![Figure 1: The flow chart of PSO algorithm.](image)

**Initialize** Before searching, a swarm of $N$ particles is initialized and each particle is initialized with a random position and a random velocity ranging in $[-V_{max}, V_{max}]$.

**Record** For each particle, the best position it has encountered is recorded as the historical best position of this particle. And for the swarm, the best position ever reached by all particles of the swarm is recorded as the global best position.

**Terminate** When the global best position meets the condition, that is to say at least one of the particles successfully finds the solution, the algorithm will terminate and output the global best position.

**Update Speed and Position** If the termination condition is not reached, the velocity of each particle is updated according to its position and its historical best position and the global best position. The updating formula of the velocity and the position of the $d$-th dimension of the $i$-th particle is shown below:

$$V_{id} = \omega V_{id} + c_1 r_1 (P_{id} - X_{id}) + c_2 r_2 (P_{gd} - X_{id}),$$

$$X_{id} = X_{id} + V_{id}. \quad (3)$$
\[ X_{id} = X_{id} + V_{id}, \]

where \( \omega \) is called the inertia weight, \( c_1 \) and \( c_2 \) are called constriction factors, and \( P_i \) is the historical best position of the \( i \)-th particle and \( P_g \) is the global best position.

PSO is widely used to solve optimization problems such as evolving neural networks (Eberhart and Hu, 1999) and image classification (Omran et al., 2004). However, for lots of optimization problems, such as POS tagging (Silva et al., 2012) and texts clustering (Cagnina et al., 2014), whose search space is discrete, some concepts such as the velocity and the position in original PSO are not applicable anymore. Kennedy and Eberhart (1997) propose a discrete version of PSO to solve this problem, where the updating formula of velocity is as follows:

\[ V_{id} = V_{id} + \varphi(P_{id} - X_{id}) + \varphi(P_{gd} - X_{id}), \]

where \( \varphi \) is a random positive number. The value of \( X_{id} \) changes with the probability \( Sigmoid(V_{id}) \), where

\[ Sigmoid(V_{id}) = \frac{1}{1 + e^{-V_{id}}}. \]

3 Methodology

In this section, we first delineate our sememe-based word substitution strategy. Then we detail the PSO-based adversarial example searching algorithm.

3.1 Sememe-based Word Substitution Strategy

According to the definition of a sememe, the sememes of a word accurately depict the meaning of the word. Therefore, selecting the words with similar sememes as substitutes is sensible. For one thing, compared with the embedding-based word substitution, sememe-based word substitution will not substitute victim words with their related words, e.g., “car” and “road” may have close embeddings but have different sememes. Therefore, it can better preserve the original semantics and readability. For another, the synonym-based substitution depends on thesauri like WordNet (Miller, 1995) but they provide no synonyms for named entities and the number of a word’s synonyms is very limited. In contrast, HowNet annotates sememes for all kinds of words including named entities and the portrayal of words given by sememes is fine-grained, which means sememe-based word substitution strategy can generate much more candidate adversarial examples.

Particularly, we only substitute notional words\(^1\) and restrict the substitutes to having the same POS tag as the victim word. Considering the situation of polysemy, word A can be substituted by word B if one sense of A has the same sememe annotation as one sense of B. Also, to prevent introducing grammatical mistakes, we restrict that the substitutes and the victim word must have the same morphological form including tense of verbs and plural of nouns.

3.2 PSO-based Adversarial Example Searching

Before demonstrating our algorithm, we first explain some concepts in our algorithm:

- **Particle, Swarm and Searching Space** Each particle in the swarm is a sentence of \( n \) words, which can be assumed as an \( n \)-dimensional vector. The set of substitution words of the \( i \)-th word in the sentence constitutes the search space of the \( i \)-th dimension.
- **Target Label and Target Score** Target label is the label that we want the target model to predict for the adversarial example. For example, if the true label of the original sentence is “positive”, the target label is “negative”, and vice versa. Target score is the prediction probability of the target label given by the target model, which is denoted by \( P(y_{target}|x) \), where \( x \) is the input sentence and \( y_{target} \) is the target label.
- **Modification Rate** The modification rate of one particle (sentence) is defined as follows:

\[ m(x) = \frac{dif(x, x_{orig})}{length(x)}, \]

where \( dif(x, x_{orig}) \) is the number of words in the perturbed sentence that are different from the original sentence and \( length(x) \) is the length of the perturbed sentence.
- **Mutation** Since we want to find adversarial examples with as few modifications as possible, we can not initialize the particles in the swarm with stochastic positions in the search

---

\(^1\)Notional words are the words with concrete meanings and mainly include nouns, verbs, adjectives, adverbs, etc.
space. Instead, we define an operation on each particle named mutation. For a particle (sentence), one mutation means changing one dimension (word) of it.

We initialize the swarm by applying mutation on the original sentence $N$ times and giving each particle a stochastic velocity between $-V_{max}$ and $V_{max}$. In order to enhance the diversity of the population, we apply mutation to each particle $x$ with the probability $P(x)$ at each iteration if the termination condition is not reached. In order to prevent excessive modifications, we define that

$$P(x) = 1 - m(x), \quad (8)$$

where $m(x)$ is the modification rate of $x$.

For our problem, we define the updating formula of velocity (changing probability) as follows:

$$V_{id} = \omega V_{id} + (1 - \omega)(equal(P_{id}, X_{id}) + equal(P_{gd}, X_{gd})), \quad (9)$$

where $V_{id}$ is restricted to $[-V_{max}, V_{max}]$. Function equal is defined as:

$$equal(A, B) = \begin{cases} -1, & A \neq B, \\ 1, & A = B. \end{cases} \quad (10)$$

$\omega$ is called the active factor. A higher value of $\omega$ makes the particles more active in the searching space and a lower one makes the particle more easy to move towards the best positions. In order to make the swarm be more active in the preliminary stage and gather to the best positions in the later stage, we design a time decreasing $\omega$ inspired by Shi and Eberhart (1998). Formally,

$$\omega = (\omega_1 - \omega_2) \times \frac{MaxIter - CurIter}{MaxIter} + \omega_2, \quad (11)$$

where $\omega_1$ and $\omega_2$ are constants between 0 and 1, $MaxIter$ is the max iteration time and $CurIter$ is the current iteration time. As in Kennedy and Eberhart (1997), the particles move to the best positions with the probability $Sigmoid(V_{id})$.

At each iteration, when the predicted label of one of the particles in the swarm is the target label, the particle is outputed and the algorithm terminates. Otherwise, all particles in the swarm update its velocity and position and the next iteration starts.

4 Experiments

In this section, we evaluate our attack model on sentiment analysis tasks, comparing the difference of attack effect among different word substitution strategies and different optimization algorithms.

Dataset We use OpenHowNet (Qi et al., 2019b), the open API of HowNet to obtain sememe annotations of words. We use the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013) as the evaluation set of the target model.

Target Model We train a Bi-LSTM with max-pooling as the target model to evaluate the attack methods. We use 300-dimensions pre-trained Glove (Pennington et al., 2014) word embeddings as the inputs of the model. The test accuracy of the target model is $83.8\%$.

Baseline Model We choose the attack model in Alzantot et al. (2018) as our basic baseline method, which adopts word embedding-based substitution and uses genetic algorithm (GA) to search for the best adversarial examples. To further analyze our model, we also conduct an ablation study which combines different word substitution strategy (embedding, synonym and sememe-based) with different searching algorithms (GA and PSO).

Evaluation Metrics First, we randomly sample 1,000 correctly classified examples from the test set as the evaluation set of attack models. Then we evaluate the attack models based on both success rates and modification rates. For every instance in the evaluation set, a model successfully attacks only when its generated adversarial example is classified by the target model as a different label (e.g., negative to positive) within at most $G$ iterations.

Experimental Settings Following the settings in Alzantot et al. (2018), we set the max iteration time $G$ to 20. We adjust PSO on the validation set of SST and set $\omega_1$ as 0.8 and $\omega_2$ as 0.2. We set the max velocity of the particles $V_{max}$ to 3, which means the changing probability of the particles ranges from 0.047 ($sigmoid(-3)$) to 0.953 ($sigmoid(3)$).

Results The attack success rates and modification rates of all the models are shown in Table 1. We can observe that:
| Algorithm | Word Substitution | %Success | %Modified |
|----------|------------------|----------|-----------|
| GA       | Embedding        | 74.90    | 20.18     |
|          | WordNet          | 73.80    | 16.24     |
|          | Sememe           | 92.60    | 15.13     |
| PSO      | Embedding        | 82.90    | 18.89     |
|          | WordNet          | 82.50    | 15.19     |
|          | Sememe           | 94.60    | 12.75     |

Table 1: Attack results of all the models.

1) Our model (PSO+Sememe) achieves the highest success rate and the lowest modification rate than all the baseline methods, which demonstrates the effectiveness of our attack method.

2) Among the three word substitution strategies, sememe-based strategy performs best in terms of both the attack success rate and the modification rate. We also give an example of substitutes in Table 2. It shows that sememe-based word substitution strategy can find more diverse substitutes with good semantic and readability preservation.

3) Compared with genetic algorithm, our PSO-based searching algorithm can find more successful adversarial examples with less modification.

5 Conclusion and Future Work

In this paper, we formalize word substitution-based textual adversarial attack as a combinatorial optimization problem. And we propose an attack model comprising the sememe-based word substitution strategy and the particle swarm optimization algorithm. We evaluate our attack model on the task of sentiment analysis, finding our model achieves higher attack success rates and lower modification rates than baseline methods. In the future, we will consider sememe-based adversarial defense method.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2890–2896.

Leonard Bloomfield. 1926. A set of postulates for the science of language. Language, 2(3).

Leticia Cagnina, Marcelo Errecalde, Diego Ingaramo, and Paolo Rosso. 2014. An efficient particle swarm optimization approach to cluster short texts. Information Sciences, 265:36–49.

Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. 2018. Ead: elastic-net attacks to deep neural networks via adversarial examples. In Thirty-second AAAI conference on artificial intelligence.

Zhendong Dong and Qiang Dong. 2006. Hownet and the computation of meaning (with Cd-rom). World Scientific.

Russell Eberhart and James Kennedy. 1995. Particle swarm optimization. In Proceedings of the IEEE international conference on neural networks, volume 4, pages 1942–1948. Citeseer.

Russell C Eberhart and Xiaohui Hu. 1999. Human tremor analysis using particle swarm optimization. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), volume 3, pages 1927–1930. IEEE.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 31–36.

Xianghua Fu, Guo Liu, Yanyan Guo, and Zhiqiang Wang. 2013. Multi-aspect sentiment analysis for chinese online social reviews based on topic modeling and hownet lexicon. Knowledge-Based Systems, 37:186–195.

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

Yihong Gu, Jun Yan, Hao Zhu, Zhiyuan Liu, Ruobing Xie, Maosong Sun, Fen Lin, and Leyu Lin. 2018. Language modeling with sparse product of sememe experts. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4642–4651.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In Proceedings of the 2018 Conference of the North...
Dongyeop Kang, Tushar Khot, Ashish Sabharwal, and Eduard Hovy. 2018. Adventure: Adversarial training for textual entailment with knowledge-guided examples. In Proceedings of ACL.

James Kennedy and Russell C Eberhart. 1997. A discrete binary version of the particle swarm algorithm. In 1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation, volume 5, pages 4104–4108. IEEE.

George A Miller. 1995. Wordnet: a lexical database for English. Communications of the ACM, 38(11):39–41.

Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Improved word representation learning with sememes. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 2049–2058.

Mahamed G Omran, Andries P Engelbrecht, and Ayed Salman. 2004. Image classification using particle swarm optimization. In Recent advances in simulated evolution and learning, pages 347–365. World Scientific.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In Proceedings of EMNLP, pages 1532–1543.

Fanchao Qi, Junjie Huang, Chenghao Yang, Zhiyuan Liu, Xiao Chen, Qun Liu, and Maosong Sun. 2019a. Modeling semantic compositionality with sememe knowledge. In Proceedings of ACL.

Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Qiang Dong, Maosong Sun, and Zhendong Dong. 2019b. Openhownet: An open sememe-based lexical knowledge base. arXiv preprint arXiv:1901.09957.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In Proceedings of the ACL.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 856–865.

Yuhui Shi and Russell C Eberhart. 1998. Parameter selection in particle swarm optimization. In International conference on evolutionary programming, pages 591–600. Springer.