2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD) | 979-8-3503-3168-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/CSCWD57460.2023.10152713

Black-box Word-level Textual Adversarial Attack Based On Discrete Harris Hawks Optimization

Tianrui Wang[1], Weina Niu[1*], Kangyi Ding[1], Lingfeng Yao[1], Pengsen Cheng[2] and Xiaosong Zhang[1]
[1]*School of Computer Science and Engineering University of Electronic Science and Technology of China*, Chengdu, China
[2]*School of Cyber Science and Engineering Sichuan University*, Chengdu, China
niuweina1@126.com

***Abstract*—Neural network-based applications are prone to being fooled by adversarial examples due to the natural vulnerability of deep neural networks (DNNs). Textual adversarial attacks are particularly challenging due to the discreteness between texts. The adversarial examples crafted by word-level textual attacks which are typically treated as optimization problems in black-box scenarios perform better in human evaluation. Existing approaches have struggled to balance the success rate with the time consuming, mainly because the chosen optimization algorithm is not efficient enough. In this paper, we propose a method to generate textual adversarial examples called Discrete Harris Hawk Optimization (DHHO). We set up three operations for handling discrete data, which are applied to each stage of the Harris Hawk Optimization (HHO) to enable it to solve optimization problems in discrete space. By attacking BiLSTM and BERT on two benchmark data sets, we conduct extensive experiments to evaluate our attack method with a success rate of up to 98% and a reduction of time is at least 50%. Moreover, the experimental results also show that our adversarial examples can ensure high quality and transferability.**

***Index Terms*—Textual Adversarial Attack, Harris Hawks Optimization, Word-level Attack, Deep Neural Networks**

I. Introduction

DNNs are susceptible to being attacked by adversarial perturbations [1] and researches in quantities have been carried out. The security of DNNs is also a concern in the field of natural language processing, where DNNs are widely used. The importance of studying textual adversarial attacks is growing day by day.

Different from image and audio, the discrete textual domain makes it thorny to generate adversarial examples and it is hardly possible to add imperceptible perturbations. Even at the minimum character-level modification may lead to changes in sentence meaning or solecism. In previous works, although there are numerous attempts to generate text attacks, there are more or less some drawbacks, such as insufficient success rate, long attack time, etc. The main reason for these problems in previous approaches is that the chosen optimization algorithm is weak in terms of astringency or prone to get stuck in local optimization. To better improve the above metrics, we introduce the Harris Hawk Optimization [2] to the field of textual adversarial attacks. HHO is a novel nature-inspired meta-heuristic optimizer inspired by how Harris Hawks hunt in nature. We improve HHO to enable it to search in discrete space and thus generate adversarial examples.

In summary, the contributions of our paper are as follows:

- We propose a method called DHHO, where we apply three core operations to each phase of HHO to generate textual adversarial examples.
- We set three core operations, improved from logical operations, that can be used to guarantee data operations in discrete spaces.
- We evaluate the performance of our method by attacking BiLSTM [3] and BERT [4] on two benchmark datasets——IMDB [5] and SST [6]. Our method achieves a result of success rate of up to 98% and a reduction of time consuming is at least 50%.

The rest of our work is organized as follows: Section II mainly introduces some previous studies on text confrontation; Section III mainly explains the basic concepts and theories; Section IV presents our DHHO in detail; Section V shows a series of experiments and performances about our DHHO; Section VI summarizes our work.

II. Related work

Before presenting our method, let us briefly summarize previous work. In existing studies, most of the adversarial attacks have focused on the deceptive textual classification system, and there are three types of adversarial attacks: character-level attack, word-level attack and sentence-level attack. A character-level attack approach called DeepWordBug can craft adversarial examples in black-box scenarios, which follows a two-step process, proposed by Gao et al. [7]. In the first step, they assess the importance of the words, and based on this, identify which words to modify. In the second step, they add imperceptible interference to the words selected in the first step by a series of character operations as described above. At the same time, they also apply edit distance to ensure the readability of crafted adversarial examples. The word-level attack manipulates entire words, not just a few characters in a word. Samanta et al. [8] used FGSM to quantity the importance of the words, indicating that the removal of these words would have a considerable impact on classification results. Jin et al. [9] proposed TextFooler, which firstly identified corresponding critical words for the target neural networks, and then replaced them with synonyms according to priority until the prediction result was changed. As for the sentence-level attack, Iyyer et al. [10] designed a syntactic controlled paraphrase network called SCPNs, which

relies on the codec architecture of SCPNs, for generating adversarial samples through syntactic transformations.

III. Preliminary Knowledges

A. Sememes

A sememe in linguistics refers to the smallest non-separable semantic unit. Some linguists hold a view that the semantics of every concept, including words, can be expressed by a finite set of sememes. When it comes to the sememe knowledge base, the most famous is HowNet [11] which has manually annotated about 100,000 Chinese/English words or phrases with more than 2,000 sememes. The OpenHowNet API [12] is developed by Natural Language Processing Lab at Tsinghua University. With this API, we can easily search HowNet for the information we want.

B. Harris Hawks Optimization

Harris Hawks Optimization [2] is a novel population-based, gradient-free and meta-heuristic optimization algorithm. The core brainchild of HHO is to imitate the hunting style and collaborative behavior of Harris hawks in nature. HHO has been progressively applied to problems such as binary classification and multiobjective optimization.

In HHO, the eagle's position corresponds to the solution to the problem. HHO mainly has two phases: exploration phase and exploitation phase, which implement global search and local search respectively. In HHO, the escaping energy of prey is denoted as E, and its calculation formula is:

$$E = 2E_0(1 - \frac{t}{T}) \tag{1}$$

where E_0 is a random real number between (-1,1), t represents the current iteration numbers and T represents the maximum iteration numbers. When $|E| \geq 1$ HHO works in the exploration phase; when $|E| < 1$, HHO works in the exploitation phase. The detailed principle and process of algorithm can be found in the original paper [2].

IV. Methodology

A. Overview

Fig 1 shows the overall process flow in this paper. First, the datasets need to be processed uniformly. GloVe is selected as the model of word representation, and all sentences in the dataset are converted into word vectors by publicly available vector text files and saved for future use in the neural network training step. Next, we utilize the part-of-speech (POS) tagging tool to get POS for each word in all the sentences in the dataset. This is followed by lemmatizing, the classification and analysis of inflectional word classes. The whole list of substitute words for the sentence is saved together, forming the search space for the sentence. The second step is to build and train neural networks.

B. Core Operations

As mentioned earlier, the discreteness of the textual domain makes it impossible to directly utilize general population-based optimization algorithms. The original HHO can only be used for optimization in continuous space. Inspired by logical operations such as "and", "or" and "xor", a series of operations are customized to process data in discrete space and applied to each stage of HHO, to achieve the goal of solving the optimization (search) problem in the textual domain. Before applying HHO, we need to make the concepts in the textual adversarial problem one-to-one correspond to the HHO theory. When processing each sentence, we treat the current sentence as a position in the search space, where each word corresponds to a dimension.

In general, we can define $X^n(t) = \{w_1^n(t) \cdots w_d^n(t) \cdots w_D^n(t)\} \in \Theta(w_d^o)$, where D represents the length of the sentence, t represents the current iteration numbers, and $\Theta(w_d^o)$ represents a search space composed of substitutable words of the current word $w_d^o(t)$. The prediction probability of victim models to target labels is considered as the optimization score. Next, we will give three introductions to facilitate our subsequent elaboration of the method.

Suppose there are $S_(w_d^o) = \{s_1, s_2, \cdots s_k\}$, where $S_(w_d^o)$ represents a set of all substitutable words from HowNet of w_d^o. And $S_1, S_2 \subseteq S_(w_d^o)$ when $|S_1| = |S_2| = 1$. Then the following three operations can be set:

1. Redefine "and" operation, which is denoted by "$\wedge$". If $S_1 = S_2 = s_i$, the operation result is s_i; Conversely, the operation result is a random one from $S_(w_d^o) - S_1 - S_2$.
2. Redefine "or" operation, which is denoted by "$\vee$". If $S_1 = S_2 = s_i$, the operation result is s_i; Conversely, the operation result is a random one from $S_1 + S_2$.
3. Redefine "xor" operation, which is denoted by "$\bigoplus$". If $S_1 = S_2 = s_i$, the operation result is a random one from $S_(w_d^o) - s_i$; Conversely, the operation result is a random one from $S_1 + S_2$.

C. Adversarial Example Generation Method Based On DHHO

Let $P(t) = \{X^n(t)|1 \leq i \leq N\}$ represent the population at the t-th iteration. $X^i(t)$ is the i-th individual and $X_d^i(t)$ means the d-th word in $X^i(t)$ whose length is D, where $d \in [1, D]$. N means the size of the population. $X^B(t)$ means the current best individual in the population at the t-th iteration.

Initialization. In the beginning, we extend the current sentence by N copies to form a population and then create a random mutation for each individual of the population. After generating $X^i(0)(1 \leq i \leq N)$, Through the victim model, we can get the prediction probability of the target label to express the fitness of the individual, so as to determine the current best individual $X^B(0)$. Equation (1) is used to calculate the escaping energy E of prey. The value of E is used to determine whether the current algorithm is in the exploration or exploitation phases.

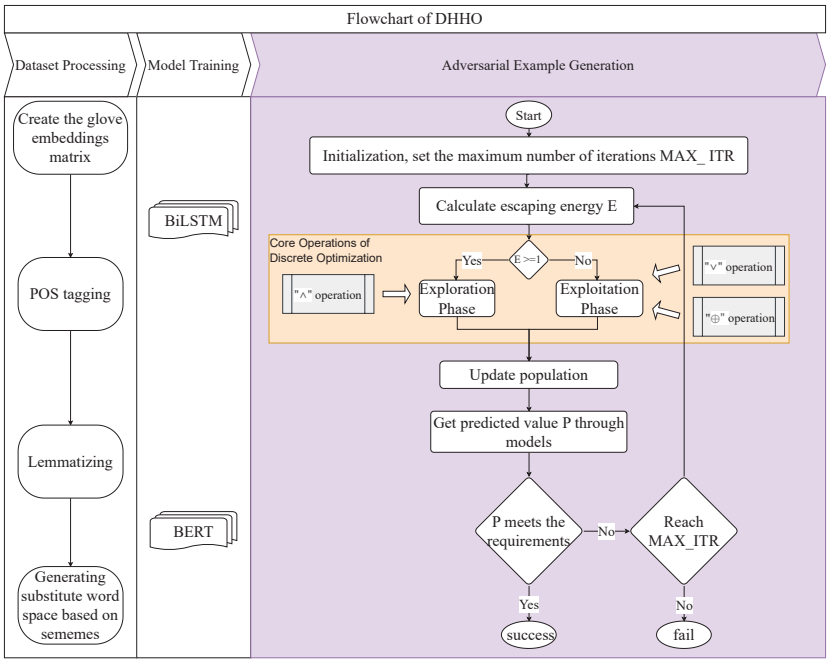

Fig. 1. Flowchart of the whole attack process.

Exploration phase. During the exploration phase, by means of the random real number r on the $(0, 1)$, the population is updated according to the following formula:

$$X_d^i(t+1) = \begin{cases} X_d^{p_1}(t) \wedge X_d^i(t), & if\ r < 0.5 \\ X_d^{p_2}(t) \wedge X_d^{p_3}(t), & otherwise \end{cases} \tag{2}$$

where p_1, p_2, and p_3 are random numbers on $[1, N]$, that is, $X_d^{p_1}(t)$, $X_d^{p_2}(t)$ and $X_d^{p_3}(t)$ are random individuals in the population.

Exploitation phase. During the exploitation phase, by means of escaping energy E and best individual $X_B(t)$, the population is updated according to the following four mechanisms:

1. Soft besiege: when $0.5 < |E| \leq 1, r \geq 0.5$,

$$X_d^i(t+1) = X_d^p(t) \bigoplus X_d^i(t) \tag{3}$$

2. Hard besiege: when $|E| < 0.5, r \geq 0.5$,

$$X_d^i(t+1) = X_d^B(t) \bigoplus X_d^i(t) \tag{4}$$

3. Soft besiege with progressive rapid dives: when $0.5 < |E| \leq 1, r < 0.5$,

$$X_d^i(t+1) = X_d^p(t) \vee X_d^i(t) \tag{5}$$

4. Hard besiege with progressive rapid dives: when $|E| < 0.5, r < 0.5$,

$$X_d^i(t+1) = X_d^B(t) \vee X_d^i(t) \tag{6}$$

where p represents a random number on $[1, N]$, that is, $X_d^p(t)$ represents a randomly chosen individual in the population.

Mutation. In order to enhance the diversity of individuals in the population, we utilize a mutation operation. Specifically, we generate two random numbers $Rand_1, Rand_2 \in (0, 1)$ If $Rand_1 < temp$ and $Rand_2 < P_m$, where P_m is a random number in $(0, 1)$ and $temp$ depends on $flag$ which is 0 or 1, we randomly select one from the list of substitutable words of a word to replace it.

Inspired by Liu et al. [13], we add a process called the deletion operation after the above search process. The explicit processing method is to iterate the generated adversarial example word by word and replace it with the word at the corresponding location of the original sentence. If the prediction of the model remains consistent, continue to process the next word, and if not, fall back to the word used by the adversarial example. The whole process is shown as Algorithm 1.

V. Experiments

A. Attack Target and Baseline Method

For datasets, we choose IMDB [5] and SST-2 [6], which are the two most commonly used datasets in sentiment analysis tasks. The IMDB dataset contains 50000 highly polarized comments from the Internet Movie Database (IMDB). The SST-2 is also a binary classification dataset. However, compared with IMDB, the average length of sentences in SST-2 is shorter and the attack is more challenging. As for victim models, two commonly used universal sentence encoding models called BiLSTM [3] with max pooling and BERT [4] are chosen.

We choose several word-level attack methods which are open-source as the baseline. These methods utilize different methods of generating substitutable words and optimization algorithms. The first baseline method [14] utilizes WordNet [15] to generate synonyms as substitutes, and searches based

Algorithm 1 Adversarial Example Generation Method Based On DHHO

Input: a sentence $X = \{w_1 \cdots w_d \cdots w_D\} \in \Theta(w_d)$, population size N, maximum iteration T
Output: adversarial example $X^{'}$

Initialize, and generate population $X^n(0) = \{X^1(0) \cdots X^i(0) \cdots X^N(0)\}$;
Get the best individual $X_B(0)$;
for $i \leftarrow 1$ to T **do**
    update escaping energy E;
    for $i \leftarrow 1$ to N **do**
        if $|E| \geq 1$ **then**
            perform Equation (2), $flag \leftarrow 0$;
        else if $0.5 < |E| \leq 1, r \geq 0.5$ **then**
            perform Equation (3), $flag \leftarrow 1$;
        else if $0.5 < |E| \leq 1, r < 0.5$ **then**
            perform Equation (5), $flag \leftarrow 1$;
        else if $|E| < 0.5, r \geq 0.5$ **then**
            perform Equation (4), $flag \leftarrow 1$;
        else if $|E| < 0.5, r < 0.5$ **then**
            perform Equation (6), $flag \leftarrow 1$;
        end if
        perform mutation operation, update the best individual $X_B(t+1)$;
    end for
end for
return $X^{'}$;

on the greedy strategy. We call it "Greedy" for short. The second baseline method [16] is based on a genetic algorithm to search the space limited by model prediction and word embedding distance. We use "Genetic" to denote this method. The last baseline method [17] which is short for "PSO" is based on the sememe library and utilizes a particle swarm algorithm to perform the search with high attack success rate.

B. Experiment Settings and Metrics

For our DHHO, we set the population size N to 60 and the maximum iteration numbers T to 20, which are also applied to the baseline methods. In our experiment, when $flag$ is 0, $temp$ is 0.3; when $flag$ is 1, $temp$ is 0.7.

We randomly select 1000 sentences as the experiment set from the dataset for each attack, and we mainly evaluate the attack success rate and attack time. In addition, during our experiment, these sentences had to meet the following requirements: be of moderate length and be accurately classified by the victim model. These requirements ensure the accuracy of our attack results. We also limit the modification rate which indicates the ratio of the number of words that are modified to these of the whole sentence to no more than 30%, otherwise it is considered a failed attack even if it succeeds in cheating the victim model.

TABLE I
THIS TABLE RECORDS THE SUCCESS RATE (%) OF EACH METHOD.

Method	BiLSTM		BERT	
	IMDB	SST-2	IMDB	SST-2
Genetic[16]	87.17	68.81	87.50	67.36
Greedy[14]	87.42	74.20	74.00	65.02
PSO[17]	**99.20**	**93.50**	95.30	89.90
DHHO	98.80	89.80	**97.50**	**92.23**

TABLE II
THE ATTACK TIME (SECONDS) REQUIRED TO GENERATE 1000 ADVERSARIAL EXAMPLES AND MODIFIED RATES (%). THE "+" IN THE FIGURE REPRESENTS THAT THE TIME CONSUMED IS FAR MORE THAN ONE DAY.

Victim Model	Method	IMDB		SST-2	
		Time	%M	Time	%M
BiLSTM	Genetic[16]	+	9.42	64198.58	11.62
	PSO[17]	42400.26	**4.32**	5988.85	9.08
	DHHO	**12418.83**	5.88	**2826.09**	**8.21**
BERT	Genetic[16]	+	7.33	40635.14	10.01
	PSO[17]	41628.74	**4.17**	4916.82	**8.19**
	DHHO	**7665.07**	5.12	**1707.48**	9.19

C. Evaluation

Attack Success Rate. The attack rate is measured as a percentage of samples which are able to successfully fool neural networks on our experiment set. We conduct comparative experiments with the success rates of the three baseline approaches, and the performances of all approaches are displayed in Table I. Our method and PSO have approximately the same effect, but higher than the other two baselines. On two benchmark datasets, the success rate of our method against two neural networks is misclassified with a high success rate, particularly on IMDB, which can be as high as 98%. As for SST-2, which has a much shorter average length, the overall success rate is slightly lower. The high success rate shows that our method can definitely reveal the vulnerability of DNNs.

Attack Time. The attack time mainly refers to the time required for a complete attack on an experimental set (i.e. 1000 sentences). We add additional attack time comparison experiments with Genetic and PSO. It is worth noting that according to the comparison of attack time in Table II, we can obviously see that our method is considerably faster than the other two methods. Here we focus on the time cost of Genetic. Generating 100 adversarial examples on IMDB takes approximately 15 hours on average, so Genetic consumes far more time than PSO and DHHO. Taken as a whole, our method takes into account both the success rate and the time cost. On average, our method can save more than 50% of the time on SST, and even 70% on IMDB. It can also be observed from Table II that our DHHO sacrifices the rate of modification to some extent compared to PSO.

HHO is a two-phase optimization algorithm. In the exploration phase, HHO deeply explores each region and side of the space. In the exploitation phase, HHO truly enforces the search process in the local region. Thus, HHO exhibits

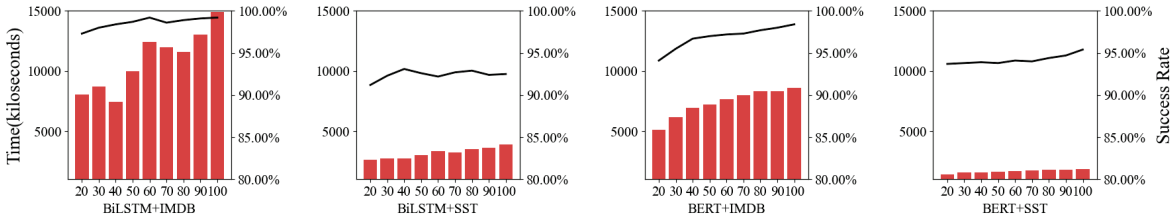

Fig. 2. Attack time and success rate for different population sizes. The bars depict how time varies with population size, while the lines depict how the success rate varies with population size.

TABLE III
EVALUATION RESULTS OF ADVERSARIAL EXAMPLES, INCLUDING THE VALIDITY (%) AND GRAMMATICAL ERROR INCREASE RATE (%).

Vicitm Model	Method	%V	%I
BiLSTM	PSO[17]	70.50	1.39
	DHHO	76.00	0.75
BERT	PSO[17]	72.00	1.43
	DHHO	77.00	1.6

strong astringency and local search capability. Because of this, our method performs considerably better than the other two population optimization-based algorithms in terms of time. The performance of experiments sufficiently demonstrates that our DHHO not only keeps the excellent optimization capabilities of the original HHO, but also guarantees the ability to manipulate discrete data through the newly introduced core operations .

Quality and Validity. The attack validity mainly means that the true label (evaluated by humans or other tools) of the adversarial example and that of the original input should be identical. We randomly selected 100 successful adversarial examples on SST, using LanguageTool (https://languagetool.org/) to detect the number of grammatical errors of a sentence. The quality of adversarial examples is mainly measured by the above-mentioned modification rate and grammatical error increase rate. We use the ratio of the grammatical error numbers caused by adversarial attack to the length of the original input to express the grammatical error increase rate, and it is 1.19% for our method. Overall, our crafted adversarial examples are of high quality. We invite several volunteers to manually assess the validity of adversarial examples. According to our evaluation, the overall proportion of valid attacks is 76.5% in BiLSTM and BERT. Table III records the validity and grammatical error increase rate of adversarial examples.

From Table II and Table III, our method performs slightly better compared to PSO in terms of quality and validity. Our method relies on three core operations and the efficient optimization capability of HHO to achieve higher model prediction scores during the iteration of the attack, which impacts the validity. Overall, the adversarial examples crafted by our method are closer to human writing than PSO.

Transferability. Transferability [18] reflects whether the

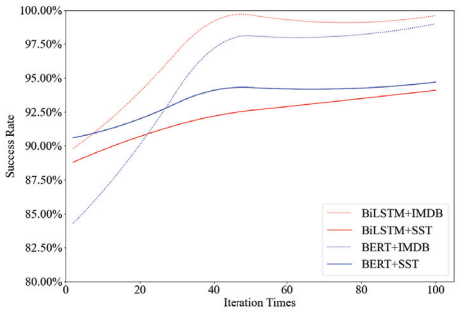

Fig. 3. Attack success rate for different iteration times.

TABLE IV
THE CLASSIFICATION ACCURACY (%) OF TRANSFERRED ADVERSARIAL EXAMPLES ON THE TWO DATASETS. LOWER ACCURACY REFLECTS HIGHER TRANSFERABILITY.

Transfer	Method	IMDB	SST
BiLSTM =>BERT	PSO[17]	75.72	63.43
	DHHO	**41.52**	**43.93**
BERT =>BiLSTM	PSO[17]	77.67	58.09
	DHHO	**48.87**	**46.17**

adversarial examples crafted for one model can deceive another one. We conduct experiments on the transferability of our method, that is, using successful adversarial examples of BiLSTM to attack BERT and vice versa. Table IV presents the transferability comparison of adversarial examples crafted by DHHO and PSO. As presentation in Table IV, our method crafts adversarial examples that perform considerably better on the metrics of transferability.

Hyperparameter Analysis. We investigate the effects of population size and iteration numbers on the results to further demonstrate our DHHO. Fig 2 shows the shift in attack time and success rate as a function of population size. We record the results of experiments with population sizes ranging from 20 to 100. From Fig 2 it will be seen that, within certain limits, the time, and success rates rise as the population size increases. It can be observed that the overall time spent on IMDB is much larger than that on SST, which is undoubtedly owing to the longer average sentence length in IMDB. Similarly, in terms of success rate, the performance of SST is inferior to

that of IMDB. Fig 3 shows the shift in attack success rate as a function of iteration numbers, with a minimum value 2. It can be found that the attack success rate gradually increases with the iteration numbers and eventually tends to stabilize. One may notice that IMDB is more sensitive to iteration numbers. The success rate on IMDB changes significantly more as iteration numbers increase.

VI. Conclusion

In this paper, we proposed an efficient textual adversarial attack method based on HHO that ideally balances attack time and success rate. We introduced HHO into textual adversarial attacks to achieve higher attack efficiency. To solve the discrete optimization problem of HHO, we mainly set three core operations that imitate logical operations, named "and", "or" and "xor", which are applied at each stage of the HHO. We conducted extensive experiments to present the strength and effectiveness of our attack method in terms of attack time and success rate. The average success rate is over 98% on IMDB and over 90% on SST. At the same time, our method is more efficient. On IMDB and SST, DHHO only consumed about 30% and 50% of the time of the PSO that achieves the highest average success rate among the baselines, respectively. In addition, the adversarial examples crafted by our method performed nicely in terms of validity, grammatical error increase rate, and transferability. Our DHHO was mainly applied to NLP systems based on BiLSTM or BERT for sentiment classification tasks. However, the impact of DHHO on other NLP models and tasks remains to be studied. In the future, we will strive to further optimize the method with a view to making greater breakthroughs in evaluation metrics and applications.

Acknowledgement

This work was supported in part by the National Key Research and Development Program of China under Grant No. 2022YFC3301700 and 2016QY001, in part by the National Natural Science Foundation of China under Grant No. 62072078, in part by the Key R&D projects in Sichuan Province under Grant No. 2020YFG0461.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[2] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future generation computer systems*, vol. 97, pp. 849–872, 2019.

[3] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," *arXiv preprint arXiv:1705.02364*, 2017.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142–150.

[6] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Empirical Methods in Natural Language Processing*, 2013.

[7] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 50–56.

[8] S. Samanta and S. Mehta, "Towards crafting text adversarial samples," *arXiv preprint arXiv:1707.02812*, 2017.

[9] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is bert really robust? a strong baseline for natural language attack on text classification and entailment," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 8018–8025.

[10] M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer, "Adversarial example generation with syntactically controlled paraphrase networks," *arXiv preprint arXiv:1804.06059*, 2018.

[11] Z. Dong and Q. Dong, "Hownet-a hybrid language and knowledge resource," in *International conference on natural language processing and knowledge engineering, 2003. Proceedings. 2003*. IEEE, 2003, pp. 820–824.

[12] F. Qi, C. Yang, Z. Liu, Q. Dong, M. Sun, and Z. Dong, "Openhownet: An open sememe-based lexical knowledge base," *arXiv preprint arXiv:1901.09957*, 2019.

[13] S. Liu, N. Lu, C. Chen, and K. Tang, "Efficient combinatorial optimization for word-level adversarial textual attack," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 98–111, 2021.

[14] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019, pp. 1085–1097.

[15] G. A. Miller, "Wordnet: a lexical database for english," in *Human Language Technology*, 1992.

[16] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," *arXiv preprint arXiv:1804.07998*, 2018.

[17] Y. Zang, F. Qi, C. Yang, Z. Liu, M. Zhang, Q. Liu, and M. Sun, "Word-level textual adversarial attacking as combinatorial optimization," *arXiv preprint arXiv:1910.12196*, 2019.

[18] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.