# SynBoost: Robust Text Generation Model Via Beam Search and Synonym-Driven Boosting

Depeng Chen[1,2,3], Mengdie Xia[1] and Jie Cui[1,2] (✉)

[1] School of Computer Science and Technology, Anhui University, Hefei, China
[2] Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Hefei, China
[3] Department of Computer Sciences and Mathematics, Rovira i Virgili University, Tarragona, Spain
{depengchen, cuijie}@ahu.edu.cn

**Abstract.** Text generation models are often affected by adversarial examples, such as injecting some adversarial samples into the model and affecting the model's predictions. Researchers have proposed various adversarial training methods to improve the robustness of text generation models in NLP. However, these methods often focus on improving the model's success rate against specific attacks while ignoring its overall robustness. Here, we identified specific words that significantly impact model judgments and employed a beam search algorithm to locate these words with high importance. We then adjusted the synonym embeddings for these words. Following targeted tuning of synonyms for specific words, the deep learning model trained using this approach maintains strong robustness against other adversarial attacks. Due to the minor embedding adjustments, the model retains high accuracy when processing regular input.

**Keywords:** Beam search · Synonym · Robustness · Text generation model.

## 1 Introduction

Deep neural networks (DNNs) have demonstrated remarkable performance in across various domains of artificial intelligence applications [1], including computer vision [2,3] and natural language processing [4,5] . However, recent studies have shown that DNNs are susceptible to adversarial examples in some cases, which may cause them to produce misleading output results.This condition is often referred to as a lack of robustness.

An adversarial example is an attack against a machine learning model, specifically by making small perturbations or changes to the model's inputs that cause errors in its output results. This type of attack is widely used in areas such as image recognition, natural language processing, and speech recognition [6,7,8,9]. For example, in image classification, the adversarial example can make minor adjustments to certain pixel points, making the model classify the image incorrectly. Previous research has shown that DNNs are vulnerable to attacks by

strategically modified samples named adversarial examples [10]. These samples are generated under some imperceptible perturbations but may trick the DNN into giving wrong predictions. Inspired by the popularity of adversarial examples for generating image DNNs, researchs on attack DNNs for textual applications has emerged in recent years. However, existing image perturbation approaches cannot be directly applied to text because text data is inherently discrete. These attack methods are potentially dangerous for text classification, sentiment analysis, and machine translation tasks.

Numerous scholars have proposed various techniques to improve the adaptability of deep learning models when studying adversarial aggressiveness in deep learning models. Adversarial sample training models, in particular, have shown effectiveness. These samples are created by introducing imperceptible perturbations to the original samples, which are sufficient to cause deep learning models to make incorrect predictions. Researchers often generate adversarial samples and add them to the training set for adversarial training using a pre-existing dataset. Jia and Liang [11] were among the first to consider generating adversarial examples (or adversarial attacks, we will use the two expressions interchangeably below) on deep neural networks for text-based tasks. Ren et al. [12] proposed a greedy word substitution attack. In a later study, Goodfellow et al. [13] proposed FGSM, a simple and effective adversarial attack method that generates adversarial samples by computing the gradient sign of the input samples with respect to the loss function, among others.

However, after our investigation and research, we found that many adversarial attacks aim to improve the model by generating adversarial examples. Often, the focus is on generating higher-quality text or better attacks. In contrast, our research focuses on how to defend against adversarial attacks so that the model remains more robust even when it encounters perturbations.

In our work, we first utilize an improved beam search algorithm [14]. We enhance the search algorithm by adding a random arbitrary solution to the original fixed search solution to prevent the loss of the optimal solution. Then, we rank the words according to their importance and select $k$ optimal solutions plus an arbitrary solution for the next round of word synonym embedding adjustment. This process aims to reduce the text's potential distance from adversarial text and increase differentiation from other samples. Our experiments show that our method, SynBoost, not only provides good defense against adversarial attacks but also maintains robustness against clean samples due to our minimally modified model.

## 2 Background

The beam search algorithm [15] is a heuristic search algorithm widely employed in various natural language processing tasks, such as speech recognition [16] and neural machine translation [17]. It navigates the search space by exploring some branches of the search tree and is primarily used to solve optimal decision problems. Unlike the exhaustive search method, the beam search algorithm

only considers several optimal candidate solutions in each search step, thereby reducing the complexity of the search space and improving search efficiency.

The basic idea of the beam search algorithm is that, in each search step, a certain number of candidates to lead to a solution are selected, and the search is performed only on these candidates. Specifically, the algorithm starts from an initial node, evaluates all the children of the node, selects the $k$ most likely nodes that lead to a solution, then turns to these$k$ nodes and repeats the process until a solution is found or a predetermined stopping condition is reached.

In beam search algorithms, the key aspect is how to select the nodes that are most likely to lead to a solution, typically achieved through an evaluation function. This function can consider various factors, such as the probability of the node, its heuristic value, and its distance to the target solution. The beam search algorithm requires adjusting appropriate parameters for the specific problem, including the number of starting nodes $k$, the stopping condition, and the evaluation function.

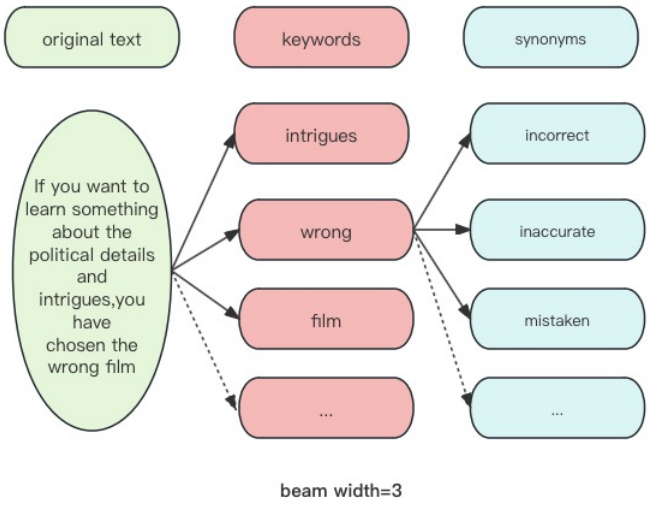



**Fig. 1.** The green box indicates the original text, the red box highlights the identified keywords, and the blue figure represents the synonyms of the words.

## 3 Methodology

We use a beam search algorithm that works takes $k$ keywords per iteration and adjusts the synonym embedding for this keyword. However, the traditional beam search method has apparent drawbacks because if the initial selection of $k$ nodes

is invalid, the negative impact will affect the selection of the best result, so we add another solution along with the fixed selection of $k$ candida. As shown in Fig. 1, we select three words in the input text based on word accuracy and randomly add an arbitrary word. Then, we find three synonyms with the closest similarity to each word and randomly add one synonym for adjustment to ensure the efficiency of the adjustment.

The beam search algorithm is a standard technique in machine learning and is widely utilized in natural language processing tasks such as machine translation and language modeling. The beam search algorithm combines the advantages of the greedy search algorithm and enumeration [15], which is divided into multiple time steps, and in each step, $k$ optimal solutions are selected as the most likely candidate solutions for the next time step, and the recurrence continues in this order. This type of search is relatively fast and accurate, and for large-scale search problems, beam search can quickly find an optimal or better solution.

Although the beam search algorithm is a more intelligent search technique than the greedy algorithm, since the beam search algorithm only retains beam size solutions, some essential solutions may be overlooked if the beam size is not chosen appropriately. Therefore, we add an arbitrary solution to retain $k$ solutions at a time, and this solution does not need to meet the constraints of $k$ solutions. In this way, we avoid being bound to the precondition of $k$ solutions and miss the critical solutions that may exist beyond $k$ solutions. The following equation represents all the solutions selected during the search.

$$B = select(constraint(k) + \gamma) \tag{1}$$

where *constraint(k)* denotes the set of solutions that satisfy the constraints of the first $k$ solutions. These solutions are preserved in the first $k$ iteration steps, , the search space is restricted to these $k$ solutions. $\gamma$ denotes all possible solutions other than these $k$ solutions. Any solution among them satisfies a specific condition, but does not have to satisfy the restriction of the first $k$ solutions. In this way, we overcome the limitation of the beam size $k$ and expand the search space to discover the possible critical solutions.

In this scheme, we adopt two time steps. In the first time step, we first calculate the importance scores of the words in the sentence by TFIDF technique [18]. Subsequently, we sort them according to the importance scores of the words, and select the first $k$ words as the candidate solutions of the next time step. In the second step, we take the keywords identified in the first step and minimize the distance between them and their synonyms by adjusting the synonym embeddings. This adjustment is aimed at enhancing the differentiation between the input text and its potential adversarial variations.

In a sentence, certain words play a crucial role in determining the semantics of the entire sentence, while the addition or deletion of other words has less impact on the sentence composition. Given a dataset $\chi = \{x_1, x_2, ..., x_N\}$ containing $N$ text data samples, where each sentence $x = \{w_1, w_2, ..., w_n\}$ consists of $n$ words, we calculate the importance score of each word in each text individually. Subsequently, we select the top $k$ words based on their importance scores, and

add these keywords to the dictionary.

$$tf_i = \frac{w_i}{\sum_k w_k} \tag{2}$$

$$idf_i = \frac{\ln |X|}{|\{j : t_j \in X_j\}|} \tag{3}$$

$$Significance_i = tf_i \cdot idf_i \tag{4}$$

where $w_i$ is the number of occurrences of the word in the text, $\sum_k w_k$ indicates the total number of occurrences of all words in this text, and $X$ represents the number of all texts in the dataset. $j : t_j \in X_j$ denotes the number of documents containing the word $t_i$. We measured the importance score of each word in each text according to the TfidfVectorizer in the main sklearn package and obtained $k$ keywords after sorting $S(w_i) = \{w_1, w_2, w_3...\}$.

After identifying the keywords that influence the sentence, we vectorize each input sample using the embedding matrix for this algorithm and then calculate the distance between each vector and the other vectors. We utilize a pre-trained Glove model [19], which contains each word and the corresponding vector values. These vector values enable us to calculate the cosine similarity between words. We set a threshold for cosine similarity, and if the cosine similarity value falls within this threshold, we consider the two words synonymous with each other.

$$similarity = \frac{v_i \cdot v_j}{\|v_i\| \cdot \|v_j\|} \tag{5}$$

$$w_i \sim w_j \iff similarity < \theta \tag{6}$$

Where $v_i$ is the embedding expression of $w_i$. When the embedding similarity of two words is greater than that $\theta$, we say that the two words are synonyms of each other.

We use Euclidean distance [20] to calculate the distance between two vectors. The preset margin is utilized to control the distance between the vectors.

$$Loss = \frac{1}{X} \sum_{i=1}^{X} [\|\alpha_i - \beta_i\|_2 - min(0, \|\alpha_i - \gamma_i\|_2 + \epsilon) + \epsilon] \tag{7}$$

Where $X$ denotes the number of samples. $\alpha_i$ denotes the original text vector, $\beta_i$ denotes one of the synonyms of the original text vector, $\gamma_i$ denotes a randomly selected vector and $\gamma_i$ the set of synonyms that do not belong to $\alpha_i$. $\epsilon$ denotes a predefined margin value.

## 4 Experiment

To evaluate the effectiveness of SynBoost, we used FTML [21] and SEM [22] as benchmarks and compared SynBoost's performance with CNN [23] and BILSTM [24] models on three popular benchmark datasets: IMDB [25], YELP [26], and MR [27].

### 4.1 Experimental Setup

**Datasets**

*1) IMDB.* A large movie review dataset consisting of 50,000 movie reviews, of which 25,000 reviews are used for training and 25,000 for testing. Each comment in the data set was identified as "positive" or "negative" and could be used for the dichotomous sentiment analysis task.

*2) YELP.* A large dataset of reviews collected from the Yelp website. It contains information such as business score, review text, review date, and more. The reviews cover a wide range of businesses such as restaurants, shopping outlets, travel services, beauty establishments, and others. This dataset comprises approximately 6 million reviews and encompasses reviews for about 1 million businesses. Yelp datasets are extensively utilized in natural language processing tasks such as sentiment analysis, subject recognition, and entity recognition.

*3) MR.* This dataset is a frequently employed dichotomous sentiment analysis dataset, distinguishing between positive and negative emotions. Comprising 10,662 film reviews, each labeled as either "positive" or "negative," it has been extensively utilized in subsequent studies. The dataset serves as a valuable resource for research in sentiment analysis and natural language processing.

**Models** For each dataset, we trained three commonly used models for robustness studies on the training set, namely CNN and LSTM. We initialized the embedding layer of the models using 300-dimensional Glove word vectors.

**Attacks & Baselines** To assess the defense effect of the SynBoost method against adversarial attacks, we employed three traditional adversarial attack methods: PWWS [28], PSO [29], and GA [30]. These methods enable researchers to evaluate the effectiveness of our SynBoost method against different types of attacks, representing common counterattack methods in text classification tasks. We compared the defense effects of our proposed SynBoost with FTML and SEM methods. In our approach, we set the beamwidth to 1, 3, and 5, respectively, naming them SynBoost-1, SynBoost-3, and SynBoost-5. This allows us to explore the defensive effect of our method at different beam widths. Instead of simply adjusting the synonym embedding, we found that making overall adjustments to the synonym embedding of each word may lead to less accurate predictions for clean samples. Therefore, we employed an improved beam search algorithm to identify groups of words with the most significant impact on semantic judgment and made targeted adjustments to the synonym embedding.

### 4.2 Evaluation of Defense Efficiency

**Adversarial Example Generation** To verify SynBoost's protection against adversarial attacks from external sources, we utilized PWWS, GA, and PSO as

**Table 1.** Here are three real-world examples demonstrating adversarial examples generated by PWWS against the IMDB, Yelp, and MR datasets. In each example, original words are marked in bold blue, and their replacement words are highlighted in bold red.

| | |
|---|---|
| IMDB | This is a very strange (unusual) film, with a no-name cast and virtually nothing known about it on the web. |
| YELP | I am so excited (emotional) that Sephora is now finally opening a store in our city. |
| MR | The movie is a bit (little) overlong , but nicholson is fun that the running time passes by quickly . |

our adversarial attack methods and generated adversarial examples using these techniques. Table 1 presents the adversarial examples generated against these three datasets using the PWWS method. In subsequent experiments, we used these generated adversarial examples to attack the model and then strengthened the model with different defense methods to evaluate their performance.

**Keyword exploration** To demonstrate the effectiveness of our method, we visualize the keywords identified by our approach. Additionally, we include expressions of high-frequency keywords. As shown in Figures 2, certain keywords that significantly impact the sentence carry strong emotional connotations, such as "annoying", "enjoyed", "happy" and so on.

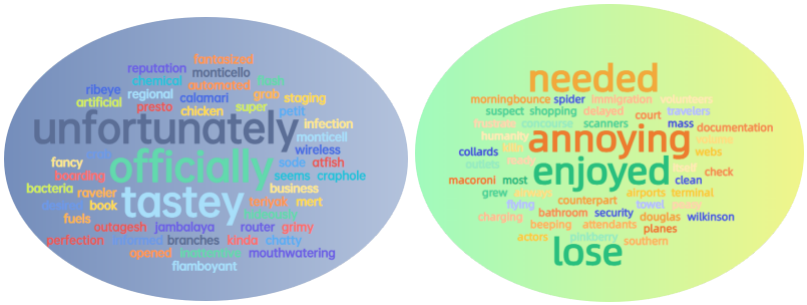



**Fig. 2.** Some keywords are displayed in order of their importance, with the IMDB dataset on the left and the YELP dataset on the right.

**Performance of SynBoost** To evaluate the performance of SynBoost, we first conducted model training on three datasets: IMDB, Yelp, and MR, respectively. We compared two models, CNN and BILSTM, trained using SynBoost-1, SynBoost-3, and SynBoost-5 with the model trained by FTML. The comparison results are shown in Table 2. Initially, we tested the classification accuracy of the model in its original state on the MR, IMDB, and Yelp datasets.

Generally, the accuracy of some defense algorithms tends to decline during routine training processes. However, since our defense method focuses on keywords when judging sentences and makes only small modifications to the data, it is closer to the original data. Therefore, when facing the judgment of clean samples, the accuracy of the model does not decline. Our defense algorithm enables the model to make judgments based on keywords.

In FTML, each word's synonym embedding is modified, significantly altering how the word behaves in the embedding layer. In SEM, an encoder is inserted before the input layer of the target model to map each cluster of synonyms to a unique encoding. However, too many modifications can have adverse effects.

After the original judgment of the model, we focus on modifying the synonyms of some keywords embedded. It can be seen from the data in the table that when compared with the original model, particularly on the IMDB dataset, the accuracy of the CNN model reached 0.9 using SynBoost. We also found that the model performs best when SynBoost is set to 3, so we set the beamwidth to 3 in later experiments.

At the same time, we conducted a comparative test using FTML and SEM as defense baselines. The comparison results are shown in Table 3. The influence of FTML and SEM is compared with our method, SynBoost. As seen from the results, SynBoost demonstrates a significant defense effect against all three datasets, particularly against the Yelp dataset.

**Table 2.** The classification accuracy of IMDB, YELP, MR in CNN and Bi-LSTM. The original result indicates the classification accuracy of the whole original model on the test set. We compared its model strengthening effect with SynBoost-1, SynBoost-3, and SynBoost-5 in model CNN, BILSTM using FTML as baseline, the highest accuracy rate for the corresponding attack for each column is highlighted in bold.

| Model | Datasets | Original | FTML | SynBoost-1 | SynBoost-3 | SynBoost-5 |
|---|---|---|---|---|---|---|
| CNN | IMDB | 0.89 | 0.87 | 0.89 | **0.90** | 0.87 |
| | YELP | 0.69 | 0.67 | 0.68 | **0.71** | 0.68 |
| | MR | 0.81 | 0.80 | 0.79 | **0.82** | 0.80 |
| BILSTM | IMDB | 0.90 | 0.87 | 0.87 | 0.88 | 0.89 |
| | YELP | 0.72 | 0.70 | 0.69 | **0.72** | 0.71 |
| | MR | 0.80 | 0.78 | 0.79 | **0.82** | **0.81** |

**Table 3.** The classification accuracy of IMDB, YELP, and MR in CNN and Bi-LSTM under various attack methods is presented below. The highest accuracy rate for each defense method is highlighted in bold for each column.

| Datasets | Attack Method | CNN | | | BILISTM | | |
|---|---|---|---|---|---|---|---|
| | | FTML | SEM | SynBoost | FTML | SEM | SynBoost |
| IMDB | PWWS | 81.5 | 68.2 | **81.9** | 80.0 | 73.2 | **81.0** |
| | GA | 82.1 | 69.0 | 82.1 | 80.2 | 70.1 | 80.1 |
| | PSO | 82.3 | 68.5 | **83.1** | 79.3 | 85.3 | **80.0** |
| YELP | PWWS | 58.3 | 40.2 | **60.3** | 58.6 | 41.3 | **59.3** |
| | GA | 59.2 | 40.1 | **61.2** | 58.5 | 40.3 | **59.4** |
| | PSO | 59.2 | 41.2 | **59.3** | 58.1 | 41.2 | **59.3** |
| MR | PWWS | 56.3 | 39.8 | **58.0** | 55.9 | 40.0 | **57.2** |
| | GA | 56.3 | 39.6 | **58.2** | 55.8 | 39.5 | **57.8** |
| | PSO | 56.2 | 40.0 | **58.5** | 55.8 | 39.8 | **57.6** |

## 5 Conclusion

In this work, we propose a robust text generation method called SynBoost, which utilizes an improved beam search algorithm and synonym embedding. Firstly, we identify $k$ words that significantly affect the semantics of the text based on their importance ordering and then add a random word. Secondly, we adjust the embedding of synonyms according to these words to minimize the embedding distance between words and their synonyms. Furthermore, we make minor adjustments to avoid over-modification of the embedding, ensuring that the model defends against attacks without sacrificing accuracy. By focusing more on specific keywords, we improve the model's accuracy. In summary, SynBoost presents a more effective and reasonable model defense method for the text generation model.

**Acknowledgements.** We would like to thank to the anonymous reviewers for their valuable comments and suggestions, which helped improve the quality of this paper. This work is supported in part by The University Synergy Innovation Program of Anhui Province (Project No. GXXT-2022-049) and Key Project of Anhui Provincial Department of Education (Project No. 2022AH050075). The first author is also supported by the project HERMES funded by INCIBE and the European Union NextGenerationEU/PRTR.

## References


1. Zhang, W.E., Sheng, Q.Z., Alhazmi, A., Li, C.: Adversarial attacks on deep-learning models in natural language processing: A survey. ACM Transactions on Intelligent Systems and Technology (TIST), 11(3), pp. 1-41 (2020). https://doi.org/10.1145/3374217
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), pp.84-90 (2017). https://doi.org/10.1145/3065386
3. Simonyan, K. and Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint: arXiv:1409.1556 (2014)
4. Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A.M., Jozefowicz, R., Bengio, S.: Generating sentences from a continuous space. arXiv preprint: arXiv:1511.06349 (2015)
5. Li, J., Ji, S., Du, T., Li, B., Wang, T.: Textbugger: Generating adversarial text against real-world applications. arXiv preprint: arXiv:1812.05271 (2018)
6. Cheng, M., Yi, J., Chen, P.Y., Zhang, H., Hsieh, C.J.: Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In Proceedings of the AAAI conference on artificial intelligence, Vol. 34, No. 04, pp. 3601-3608 (2020). https://doi.org/10.1609/aaai.v34i04.5767
7. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint: arXiv:1412.6572 (2014)
8. Xiao, C., Li, B., Zhu, J.Y., He, W., Liu, M., Song, D.: Generating adversarial examples with adversarial networks. arXiv preprint: arXiv:1801.02610 (2018)
9. Li, L., Ma, R., Guo, Q., Xue, X., Qiu, X.: Bert-attack: Adversarial attack against bert using bert. arXiv preprint: arXiv:2004.09984 (2020)
10. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint: arXiv:1312.6199 (2013)
11. Jia, R., Liang, P.: Adversarial Examples for Evaluating Reading Comprehension Systems. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/D17-1215
12. Ren, S., Deng, Y., He, K., Che, W.: Generating natural language adversarial examples through probability weighted word saliency. In: Proceedings of the $57^{th}$ annual meeting of the association for computational linguistics, pp. 1085-1097 (2019). https://doi.org/10.18653/v1/P19-1103
13. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint: arXiv:1412.6572 (2014)
14. Zhao, T., Ge, Z., Hu, H., Shi, D.: Generating natural language adversarial examples through an improved beam search algorithm. arXiv preprint: arXiv:2110.08036 (2021)
15. Kumar, A., Vembu, S., Menon, A.K., Elkan, C.: Beam search algorithms for multilabel learning. Machine learning, 92, pp.65-89. Springer, Heidelberg (2013). https://doi.org/10.1007/s10994-013-5371-6
16. Drexler, J., Glass, J.: Subword regularization and beam search decoding for end-to-end automatic speech recognition. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6266-6270. IEEE(2019). https://doi.org/10.1109/ICASSP.2019.8683531
17. Freitag, M., Al-Onaizan, Y.: Beam search strategies for neural machine translation. arXiv preprint: arXiv:1702.01806 (2017)

18. Bafna, P., Pramod, D., Vaidya, A.: March. Document clustering: TF-IDF approach. In: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), pp. 61-66. IEEE(2016). https://doi.org/10.1109/ICEEOT.2016.7754750
19. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532-1543, ACL (2014). https://doi.org/10.3115/v1/D14-1162
20. Wang, L., Zhang, Y., Feng, J.: On the Euclidean distance of images. IEEE transactions on pattern analysis and machine intelligence, 27(8), pp.1334-1339. IEEE (2005). https://doi.org/10.1109/TPAMI.2005.165
21. Yang, Y., Wang, X., He, K.: Robust textual embedding against word-level adversarial attacks. In: Uncertainty in Artificial Intelligence, pp. 2214-2224. PMLR (2022).
22. Wang, X., Hao, J., Yang, Y., He, K.: Natural language adversarial defense through synonym encoding. In: Uncertainty in Artificial Intelligence, pp. 823-833. PMLR (2021).
23. Kim, Y.: Convolutional Neural Networks for Sentence Classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746-1751. ACL (2014). https://doi.org/10.3115/v1/D14-1181
24. Liu, P., Qiu, X., Huang, X.: Recurrent neural network for text classification with multi-task learning. arXiv preprint: arXiv:1605.05101 (2016)
25. Maas, A., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics, pp. 142-150. Human language technologies (2011). https://aclanthology.org/P11-1015
26. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. Advances in neural information processing systems 28, NIPS (2015)
27. Pang, B., Lee, L.: Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. arXiv preprint: cs/0506075 (2005)
28. Ren, S., Deng, Y., He, K., Che, W.: Generating natural language adversarial examples through probability weighted word saliency. In: Proceedings of the 57th annual meeting of the association for computational linguistics, pp. 1085-1097 (2019). https://doi.org/10.18653/v1/P19-1103
29. Alzantot, M., Sharma, Y., Elgohary, A., Ho, B.J., Srivastava, M., Chang, K.W.: Generating natural language adversarial examples. arXiv preprint: arXiv:1804.07998 (2018)
30. Zang, Y., Qi, F., Yang, C., Liu, Z., Zhang, M., Liu, Q., Sun, M.: Word-level Textual Adversarial Attacking as Combinatorial Optimization. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 6066–6080. ACL (2020). https://doi.org/10.18653/v1/2020.acl-main.540