Received: 13 July 2022 | Revised: 18 July 2022 | Accepted: 24 August 2022 | Published online: 27 September 2022

RESEARCH ARTICLE

Journal of Computational and Cognitive Engineering
2022, Vol. 1(4) 187–192
DOI: 10.47852/bonviewJCCE2202406

BON VIEW PUBLISHING

Context-Free Word Importance Scores for Attacking Neural Networks

Nimrah Shakeel[1],* and Saifullah Shakeel[2]

[1]*Teradata, Pakistan*

[2]*Lahore University of Management Sciences, Pakistan*

Abstract: Leave-One-Outscores provide estimates of feature importance in neural networks for adversarial attacks. In this work, we present context-free word scores as a query-efficient alternative. Experiments show that these approximations are quite effective for black-box attacks on neural networks trained for text classification, particularly for CNNs. The model query count for this method scales as *O*(*vocab_size* **model_input_length*). It is independent of the number of examples and features to be perturbed.

Keywords: neural networks, adversarial attacks, NLP

1. Introduction

Identifying important features in the input is the first step in attacking a neural network, resulting in an incorrect prediction. In the black-box setting, without access to network parameters, expensive queries are used to determine feature importance. We show the effectiveness of an approximate feature importance technique for word level CNNs and LSTMs, for the task of sentiment analysis and topic classification.

Determining the relative importance of various parts of an input is required in different scenarios, for example, interpretation and adversarial attacks. For neural networks, Leave-One-Out (LOO) (Li et al., 2019) is a popular technique for determining the importance of different parts of input and layers. Importance scores help in interpreting decisions (why did a neural network make a certain prediction) as well as adversarial attacks (which input features, when perturbed, would be most impactful in changing network decisions?)

For example, in black-box attacks on text classification, important words are located through the LOO approach. Afterwards, they are either deleted from the text, replaced by an Out-Of-Vocabulary (OOV) token, or replaced by another word, to create an adversarial input.

For word level classifiers, the LOO technique entails deleting a word from a piece of text and comparing model prediction on this modified input to original prediction. This change in model prediction is the importance score for this word (feature for word level networks). Without access to network parameters, determining LOO importance can be expensive in terms of model queries, since LOO scores for a word will be different for different input examples.

Quickly locating important words in a piece of text is of particular importance in the scenario when querying the model has a specific cost, and attacks are carried out on a large number of input samples. For this problem, we propose an efficient method for calculating word importance. We use these importance scores for black-box attacks and demonstrate that their attack success rate is comparable to the original methods, particularly for CNNs. The motivation for our approximate scoring algorithm comes from the fact that in a piece of text, most of the time, words and phrases have a strong influence on their own. This gives us a rationale for looking at single word predictions, in direct contrast to the LOO technique. Experiments confirm that this technique is quite effective at locating important words in text.

*Corresponding author: Nimrah Shakeel, Teradata, Pakistan. Email: nimrah.shakeel@teradata.com

2. Literature Review

2.1. Adversarial attacks on NLP models

Adversarial attacks highlight the vulnerabilities present in a model. They also give us some insight into how a trained model operates, by locating the features a model considers to be important.

We limit our analysis to word replacement attacks on text classification models, using sentiment analysis and topic classification as examples. We only consider the attack scenarios in which specific words in the input are replaced by valid words from the dictionary. This excludes attacks in which extra information is appended to input data, or where word replacements purposefully introduce spelling errors. The former take an entirely different approach; the latter introduce errors and do not preserve semantics. In addition, training a neural network to be robust to spelling errors would stop these attacks. Our analysis is related to the black-box setting, where the attacker has no information about model architectures and parameters. A brief overview of adversarial attacks on NLP models is presented below.

2.2. Find and replace attacks on text classification

Most attacks on text classification solve the problem in two parts: by locating important words in the input and by finding suitable replacements for these words. We consider attacks where substitutions are valid words picked from a dictionary, to avoid introducing grammatical errors. Thus, we ignore the case, for example, when spelling errors are introduced in important words.

In the white-box setting, where an attacker has full knowledge of the model architecture, gradients serve as a good proxy for word importance. Gong et al. (2018) use gradient-based methods to locate important words. Samanta and Mehta (2017) use gradients to calculate word importance, with linguistic constraints over substitution words. Lei et al. (2019) carry joint word and sentence attacks, by generating sentence paraphrases in the first stage. They resort to greedy word substitutions if the first stage fails. Again, important words are located by the magnitude of the gradient of word embedding.

In the black-box scenario, where gradients are not available, saliency maps are calculated for words through different methods. Yang et al. (2018) provide a simplified greedy algorithm, where feature ranking is done once for an example, as opposed to the full greedy approach.

Li et al. (2016) propose masking each feature with zero padding, using the decrease in the predicted probability as the score of the feature or word, and masking the top-k features as unknown. Alzantot et al. (2018) and Kuleshov et al. (2018) propose variations of genetic algorithms. Kuleshov et al. (2018) replace words one by one until the classifier is misdirected while observing a bound on the number of perturbed features. They run each new iteration on the modified input. For substitution, they used post-processed GloVe to find pool of suitable words. They also compute "thought vectors" for sentences and ensure that these are preserved. Alzantot et al. (2018) select words by random sampling, where probability of each word being selected is proportional to the number of suitable neighbors for replacement. They use Google 1 billion words language model to ensure that replacements match the context provided by the rest of the input. Ren et al. (2019) propose a saliency-based greedy algorithm, calculated by deleting words during the search phase and select substitutions from WordNet. Another similar attack model is Jin et al. (2019a), which has extra semantic similarity checking when searching adversarial examples and calculates word importance by deleting words.

Zang et al. (2019) propose a particle swarm optimization algorithm for the search problem, which is query intensive. Gao et al. (2018) define different scoring functions where they look at prediction before and after removing a particular word from a subset of input and perform character level modifications in the second stage. Li et al. (2020) use the sentence probability directly but once again, when ranking words, they try masking words in a sentence.

A common thread among all search methods for black-box attacks is erasure or omission. This is the LOO technique, where word importance is the difference between the classifier output for original input, and output with this particular word removed or replaced by zero in the input sample.

2.3. Origins of the LOO technique

Li et al. (2019) are a pioneering paper in the domain of interpretability that highlights the importance of interpreting networks by erasing parts of various layers. This LOO method is followed by most interpretation algorithms. For a particular word, they calculate importance score as the average of prediction difference due to erasing this word from all test examples. Other methods of interpretability are based on this LOO technique. Feng et al. (2018) gradually remove unimportant input words so that only the important ones are left at the end. Barham and Feizi (2019) propose sparse projected gradient descent to generate adversarial examples to improve interpretability.

Nguyen (2018) looks at different methods of local explanations for labels, which include Local Interpretable Model-agnostic Explanations (LIME), random feature deletion, and first derivative saliency. Kádár et al. (2017) measure salience of a word by removing it and noting the change in prediction. Jin et al. (2019b) mention deleting a particular word to calculate its importance score. Ren et al. (2019) use word saliency which is the change in the classifier output if a word is set to unknown. Carter et al. (2018) find sufficient input subsets while calculating the feature importance by masking words.

For calculating word score matrices, Xu and Du (2020) propose a method which involves masking words.

We want to highlight the aspect that all the dominant techniques for interpretation use LOO method for calculating word importance. Pal and Tople (2020) are the only work which evaluates the use of single word scores in the transfer learning setting. Through experiments, we demonstrate that our proposed scores provide a reliable way of calculating feature importance. Although not grounded in statistical theory, this technique works well for word level neural networks.

2.4. Adversarial attacks and defenses in NLP: A broader picture

Initially inspired by the computer vision literature (Szegedy et al., 2013), attacks on NLP models have been well studied. Recently, transformer models have been quite successful at natural language tasks, and they have found to be vulnerable to adversarial attacks. Jin et al. (2019a) present TEXTFOOLER, a simple but strong baseline to generate adversarial text. They calculate word importance score as the change in prediction by deleting a specific word, and noting the change in prediction. They find the use of word importance scores to be quite important and experiment with Convolution Neural Network (CNN), Long Short Term memory (LSTM), and Bidirectional Encoder Representations from Transformers (BERT). Li et al. (2019) used word importance scores obtained via LOO method from Bidirectional Encoder Representations from Transformers (BERT) and then mask those words to get replacements. Using Bidirectional Encoder Representations from Transformers (BERT) for replacements preserves the fluency as well as the semantics. They achieve state-of-the-art success rate as well as accuracy under attack with minimal % age of the words perturbed on various text classification tasks.

Berger et al. (2021) show that randomly sampling the tokens to be flipped gives a surprisingly high attack success rate, where no complicated search method is used. Further, they argue that in the constrained setting, even sophisticated algorithms fail to get a good success rate, whereas in the unconstrained setting, just replacing random words by nearest neighbors gives a very good success rate.

Guo et al. (2021) looks at white-box adversarial attacks on transformers. They propose that tokens come from probability vectors which are passed through a gumbel-softmax function. They learn the vector by minimizing an adversarial loss of their choice, guided by gradient descent. Moreover, they included two other terms in their objective function which ensures that perturbations are imperceptible as well as they have some semantic meaning using log likelihood and Bidirectional Encoder Representations from Transformers (BERT) score constraints, respectively. This technique gives them a distribution matrix which is then used to generate adversarial examples.

Mozes et al. (2021) compares the performance and examples of four adversarial example techniques with human-generated

adversarial examples. Moreover, they imposed two constraints for the adversarial examples, semantic preservation as well as naturalness which were evaluated by an independent set of crowd workers. They concluded that often the sentiment does not match the sentiment assigned to the perturbed example (15–20%). As far as naturalness is concerned, they concluded that human-generated examples are not any different than algorithm-generated examples.

Yoo and Qi (2021) suggest that we can use gradients of a sentence to estimate the ranks of the words and then use those ranks to decide which word to replace in order to create an adversarial example. Moreover, they propose two approaches, A2T and A2T-MLM, attack to train and attack to train masked language model where they use counter fitted embeddings or a masked language model to replace the word to create adversarial example, respectively. They combine the adversarial examples with the unperturbed examples and train the model on this new dataset and achieve robustness. They report massive increase in accuracy as well as generalization in most of the cases when trained using A2T and A2T-MLM techniques, not only on the adversarial examples but on the unperturbed examples as well.

Jin et al. (2019b) propose the technique textfooler, generating adversarial examples by calculating rankings of words by deleting them and taking the difference, and then replacing the words with their synonyms while preserving the semantics as well as the grammar of the sentence and use USE as a measure of similarity between the sentence with the replaced word and the unperturbed example. They achieve SOTA results with much lower word replacement. A real-world example of adversarial attacks on NLP models has been studied in Xie et al. (2022). This work explores adversarial attacks on models which make finance related decisions based on the stream of data from a social media platform. They argue that generally, attacks manipulate the original data whereas they have implemented concatenation attack, where they concatenate adversarial examples to the stream, mimicking the reply to a tweet feature. They assess their model against different word budgets and sentence budgets and assess the performance via attack success rate, loss in F1, and net profit or loss after the whole process. They achieve around 32% additional loss and around 15–20% success rate with 0.15–0.22 drop in F1. They have investigated and concluded that even though manipulative attacks have a higher success rate, they are not valid due to the nature of this problem.

2.4. Defending against adversarial attacks

Normally, attacks are generated, and model is re-trained only once with these augmented data. Ivgi and Berant (2021) propose a new discrete attack based on best-first search, as well as random sampling attacks, and show that these improve robustness if used for online data augmentation. Adversarial examples are generated during training and model is trained on these. Even randomly generated examples improve robustness. When multiple words are to be substituted, the search space for greedy search can be O(n_rep*K), where K is neighbors considered for each word. They replace this by a best first search which is faster. They experiment with transformers. Xu et al. (2022) propose a training technique for language models against adversarial attacks. They do so by incorporating adversarial examples created using several attack algorithms on the validation set, associate a weight with each example, and minimize the weighted training loss. The weight vector is the vector which minimizes the validation loss. They avoid this twp level optimization by using online meta learning algorithm to re-weight the examples. This allows them to train a model which is robust against adversarial attacks with a slight trade-off of accuracy (1–2%).

Rusert and Srinivasan (2022) suggested an ensemble technique for robustness against adversarial attacks. They produce k samples, each with p% of the data and then combine the decision of k models. They propose three sampling techniques and two combining decision techniques which result in slightly lesser accuracy on the clean dataset but lower the attack success rate as well as the accuracy on the adversarial examples. They also consider the case where the attacker knows about sample shielding and queries samples instead of whole example, and achieve similar results.

Zeng et al. (2021) first propose the idea of certified robustness, that is, guaranteeing above 50% accuracy on adversarial examples given a budget constraint. They propose a robustness technique RanMask, a BERT trained on incomplete sentences (some tokens masked) which turns an input into a large number of masked copies with percentage of tokens masked, and then combine the decision by majority voting. The RanMask achieves certified robustness on AG News but close enough on SST2 datasets with budget of five words and two words, respectively. They also proposed using Bidirectional Encoder Representations from Transformers (BERT) to select the tokens to be masked. This outperformed their previously discussed technique only on the sentiment analysis task (SST2 dataset) but for AG news dataset, the sampling technique did not matter. Moreover, they achieved higher accuracy with their sampling technique on the clean IMDB dataset inferring a concept related to dropout.

3. Context-Free Word Importance Scores: Algorithm

For determining importance of a particular word in a piece of text, we keep the word at its original location and mask the rest of the input, by replacing it with OOV tokens. This input of all zeros except one is fed to the classifier. The output of a particular class is the importance score of this word w.r.t. this class. This importance score is global in the sense that it represents what importance the model assigns to a word, without relating to any particular test example. Here, original location refers to the word location in the input. For example, for evaluating importance for the second word in a text piece that is 200 words long, its approximate score will be the classifier output over an input that consists of one masked token, the word, and 198 masked tokens, in this particular order. Context-free word implies that all the surrounding words are removed, although the word location does not change.

Algorithm 1: Step 1: Calculate word scores for a word
Input F, a trained CNN or LSTM
Input p, the number of classes in data
Input d, the size of classifier input
Input i, the location of word in original input
Output $score(w_i) \in R^p$, word score for all output classes
1: define $x_i = 0_0, 0_1, 0_2, \ldots 0_{i-1}, w, 0_{i+1},., 0_d - 1$
2: score(w_i) = F(x_i)

4. Experiments

Model architecture: We experiment with a Convolution Neural Network (CNN) and an Long Short Term memory (LSTM). For all models, the input layer uses word2vec embeddings that are learned during training. We use the Adam optimizer with default learning rate of 0.001. Model details are as follows:

- CNN-2 and CNN-4: a CNN with 200 sized input, 32-dimensional embedding layer, convolutional layer with 32 filters followed by max pooling layer, a 64 unit ReLU layer, and a softmax layer with 2/4 output classes.

- LSTM: an LSTM with 200 sized input layer, 32-dimensional embedding layer, a convolutional layer with 32 filters followed by max pooling layer, a 64 units ReLU layer, and sigmoid layer with 2 output classes.

Experiment design: Details of each experiment are shown in Table 2. Perturbed feature count is increased successively and the classifier accuracy is calculated against perturbed feature count for different attack strategies. The attacks are carried out on a randomly chosen subset of test examples (1000 for CNNs, 200 for LSTM) to make the experiments computationally feasible to run.

Datasets: We experiment with four text classification datasets, IMDB (Maas et al., 2011), Yelp (Zhang et al., 2015), Amazon (Zhang et al., 2015), and AG's News (Zhang et al., 2015). For Yelp, we ignore the reviews with 3 ratings (neutral) and convert the labels for the rest of the reviews as positive/negative. For large dataset (Amazon, Yelp), a smaller subset of original dataset is selected so that the same model can be used for all the experiments (details in Table 2). These are multi-sentence and multi-paragraph datasets. Thus, the LOO technique is particularly expensive, since it involves model queries proportional to the number of words in the input. In our experiments, the model has an input size of 200 words. Attack strategies: We evaluate our technique against simplified greedy selection baselines based on LOO and do not consider genetic algorithms, since they are computationally expensive. We compare the following strategies during our experiments: Delete_random: Words are selected at random, and deleted. Delete_score: Words are selected through the importance scoring algorithm proposed in this paper (Algorithm 1 and 2), and deleted. Delete_LOO: Words are selected through LOO method and deleted. Replace_score: Words are selected through algorithm 1 and 2, and replaced. Replace_LOO: Words are selected through LOO method, and replaced.

Replacement strategy: To replace a word, we look at its 10 nearest neighbors (a randomly chosen hyperparameter) using the GloVe (Pennington et al., 2014) embeddings, and then select the most distracting replacement through model queries, from the tokenizer vocabulary. The greedy strategies here (Delete_LOO and Replace_LOO) are simplified greedy versions, where the LOO scores are calculated and top k words selected once, instead of the full greedy version where selection and replacement are done one by one for each perturbed feature. For an attack where k features are allowed to be perturbed, the top k important words are picked first, and then replaced one by one. The Replace_LOO strategy mentioned here is somewhat similar to the greedy algorithm mentioned in Yang et al. (2018). The difference is that we pick the replacement from the 10 nearest neighbors, whereas Hsieh et al. (2019) limit the search in the second step to a pre-specified distance, to preserve semantics. The Delete_LOO strategy is borrowed from Li et al. (2019).

Table 1
Classification accuracy on perturbed examples, against perturbed feature count. Replace_score and Delete_score closely track Replace_LOO and Delete_LOO, respectively. Accuracy decreases for all attack strategies as more words are deleted or replaced

		Perturbed	Feature	Count				
IMDB-CNN	1	2	3	4	5	6	7	8
Delete_random	0.859	0.862	0.865	0.863	0.857	0.856	0.851	0.852
Delete_score	0.792	0.722	0.654	0.612	0.553	0.497	0.44	0.376
Delete_LOO	0.788	0.715	0.649	0.604	0.543	0.478	0.424	0.37
Replace_score	0.75	0.649	0.535	0.419	0.318	0.237	0.156	0.099
Replace_LOO	0.745	0.64	0.517	0.391	0.293	0.213	0.137	0.091
Yelp-CNN								
Delete_random	0.934	0.931	0.928	0.928	0.925	0.929	0.926	0.922
Delete_score	0.872	0.794	0.706	0.640	0.572	0.498	0.440	0.391
Delete_LOO	0.859	0.762	0.682	0.599	0.543	0.470	0.410	0.359
Replace_score	0.799	0.637	0.491	0.357	0.265	0.188	0.133	0.103
Replace_LOO	0.784	0.584	0.430	0.319	0.222	0.161	0.114	0.0847
Amazon-CNN								
Delete_random	0.88	0.875	0.873	0.871	0.866	0.867	0.857	0.855
Delete_score	0.772	0.656	0.558	0.49	0.418	0.368	0.314	0.273
Delete_LOO	0.755	0.638	0.537	0.472	0.398	0.334	0.276	0.242
Replace_score	0.69	0.475	0.319	0.21	0.134	0.083	0.05	0.031
Replace_LOO	0.66	0.446	0.305	0.194	0.122	0.07	0.048	0.031
AG's News-CNN								
Delete_random	0.879	0.875	0.868	0.860	0.858	0.857	0.850	0.846
Delete_score	0.790	0.695	0.624	0.525	0.437	0.366	0.288	0.233
Delete_LOO	0.786	0.686	0.605	0.505	0.429	0.352	0.285	0.227
Replace_score	0.777	0.656	0.542	0.433	0.338	0.257	0.183	0.144
Replace_LOO	0.777	0.646	0.529	0.417	0.326	0.243	0.182	0.135
IMDB-LSTM								
Delete_random	0.84	0.84	0.85	0.845	0.845	0.845	0.845	0.85
Delete_score	0.8	0.76	0.725	0.695	0.65	0.58	0.555	0.53
Delete_LOO	0.77	0.705	0.65	0.59	0.53	0.505	0.445	0.39
Replace_score	0.76	0.65	0.52	0.355	0.27	0.18	0.115	0.09
Replace_LOO	0.735	0.61	0.485	0.33	0.255	0.19	0.14	0.075

Table 2
Details of experiments

Experiment	Dataset	Model	|Train|	|Val.|	Classes	Acc (Train)	Acc (Val.)	Examples (Test)
IMDB-CNN	IMDB	CNN-2	20,000	5000	2	0.9066	0.875	1000
Yelp-CNN	Yelp	CNN-2	83,200	15,000	2	0.9422	0.941	1000
Amazon-CNN	Amazon	CNN-2	50,000	15,000	2	0.8985	0.8593	1000
AG's News-CNN	AG's News	CNN-4	96,000	24,000	4	0.9564	0.9429	1000
IMDB-LSTM	IMDB	LSTM	20,000	5000	2	0.9047	0.8691	200

5. Conclusion

Results in Table 1 show the classification accuracy for a random subset of test examples as more and more features are perturbed. In most cases, the attack success rate of delete_score aligns with delete_LOO, and replace_score with replace_LOO. The classification accuracy for the *score* variants is often just slightly above the accuracy for their LOO counterparts. Particularly for a CNN, when $n + 1$ features are perturbed using the *score* variant, the accuracy falls below that of the LOO variant for n features.

Here, Delete_random shows that deleting random words has a negligible effect on classifier accuracy, even when a large number of words are removed. Selectively replacing words (Replace_LOO), on the other hand, can quickly drive accuracy as low as 10%, just by replacing eight words. However, the Replace_LOO strategy requires model queries along the order of $O(input\_size)$, for each test example.

Whether the strategy is to replace words (Replace_LOO) or to delete words (Delete_LOO), finding out important words takes a lot of model queries. Our experiments show that approximating importance is quite effective in black-box attacks.

Notice that Delete_score is almost as successful at confusing a classifier as Delete_LOO, and Replace_score as successful as Replace_LOO. Deleting words for an LSTM seems to be the only case where the score variant lags behind the LOO variant. For all datasets and models, replacement is more effective than deletion at tricking the classifier.

In summary, single word confidence scores from neural networks can be used to quickly calculate importance and reused if attacks are carried out on a large number of samples. These experiments highlight that neural network predictions on single words are highly informative and are correlated with the importance of the word in a piece of text.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

References

Alzantot, M., Sharma, Y., Elgohary, A., Ho, B.-J., Srivastava, M., & Chang, K.-W. (2018). Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 2890–2896), Brussels, Belgium: Association for Computational Linguistics.

Barham, S., & Feizi, S. (2019). Interpretable adversarial training for text. CoRR, abs/1905.12864.

Berger, N., Riezler, S., Sokolov, A., & Ebert, S. (2021). Don't Search for a Search Method —Simple Heuristics Suffice for Adversarial Text Attacks. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. https://doi.org/10.48550/arXiv.2109.07926

Carter, B., Mueller, J., Jain, S., & Gifford, D. (2018). What made you do this? Understanding black-box decisions with sufficient input subsets. ArXiv, abs/1810.03805.

Feng, S., Wallace, E., Grissom, II, A., Iyyer, M., Rodriguez, P., & Boyd-Graber, J. (2018). Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Gao, J., Lanchantin, J., Soffa, M. L., & Qi, Y. (2018). Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*.

Gong, Z., Wang, W., Li, B., Song, D., & Ku, W.-S. (2018). Adversarial texts with gradient methods. CoRR abs/ https://doi.org/10.48550/arXiv.1801.07175

Guo, C., Sablayrolles, A., Jégou, H., & Kiela, D. (2021). Gradientbased adversarial attacks against text transformers. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. https://aclanthology.org/2021.emnlp-main.464

Hsieh, Y.-L., Cheng, M., Juan, D.-C., Wei, W., Hsu, W.-L., & Hsieh, C.-J. (2019). On the robustness of self-attentive models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 1520–1529), Florence, Italy: Association for Computational Linguistics.

Ivgi, M., & Berant, J. (2021). Achieving model robustness through discrete adversarial training. CoRR, abs/2104.05062.

Jin, D., Jin, Z., Zhou, J. T., & Szolovits, P. (2019a). Is Bert really robust? A strong baseline for natural language attack on text classification and entailment. CoRR, abs/1907.11932.

Jin, D., Jin, Z., Zhou, J. T., & Szolovits, P. (2019b). Is Bert really robust? Natural language attack on text classification and entailment. CoRR, abs/1907.11932.

Jin, D., Jin, Z., Zhou, J. T., & Szolovits, P. (2020). Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. Proceedings of the AAAI Conference on Artificial Intelligence, 34(05), 8018–8025. https://doi.org/10.1609/aaai.v34i05.6311

Kádár, Á., Chrupała, G., & Alishahi, A. (2017). Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, *43*(4), 761–780. https://doi.org/10.48550/arXiv.1602.08952

Kuleshov, V., Thakoor, S., Lau, T., & Ermon, S. (2018). Adversarial examples for natural language classification problems. https://openreview.net/pdf?id=r1QZ3zbAZ

Lei, Q., Wu, L., Chen, P., Dimakis, A., Dhillon, I. S., & Witbrock, M. J. (2019). Discrete adversarial attacks and submodular optimization with applications to text classification. In A. Talwalkar, V. Smith, and M. Zaharia (Eds.), *Proceedings of Machine Learning and Systems 2019, MLSys 2019*, Stanford, CA, USA, March 31 - April 2, 2019. mlsys.org.

Li, J., Ji, S., Du, T., Li, B., & Wang, T. (2019). Textbugger: Generating adversarial text against real-world applications. In *Proceedings 2019 Network and Distributed System Security Symposium*.

Li, J., Monroe, W., & Jurafsky, D. (2016). Understanding neural networks through representation erasure. CoRR, abs/1612.08220.

Li, L., Ma, R., Guo, Q., Xue, X., & Qiu, X. (2020). Bert-attack: Adversarial attack against Bert using Bert.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 142–150), Portland, Oregon, USA: Association for Computational Linguistics.

Mozes, M., Bartolo, M., Stenetorp, P., Kleinberg, B., & Griffin, L. D. (2021). Contrasting human- and machine-generated word-level adversarial examples for text classification. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. CoRR abs/ https://doi.org/10.48550/arXiv.2109.04385

Nguyen, D. (2018). Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, *Volume 1 (Long Papers)* (pp. 1069–1078), New Orleans, Louisiana: Association for Computational Linguistics.

Pal, B., & Tople, S. (2020). To transfer or not to transfer: Misclassification attacks against transfer learned text classifiers. CoRR, abs/2001.02438.

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543), Doha, Qatar: Association for Computational Linguistics.

Ren, S., Deng, Y., He, K., & Che, W. (2019). 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), Florence, Italy: Association for Computational Linguistics.

Rusert, J., & Srinivasan, P. (2022). Don't sweat the small stuff, classify the rest: Sample shielding to protect text classifiers against adversarial attacks. Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. https://aclanthology.org/2022.naacl-main.195

Samanta, S., & Mehta, S. (2017). Towards crafting text adversarial samples. CoRR, abs/1707.02812.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. CoRR abs/ https://doi.org/10.48550/arXiv.1312.6199

Xie, Y., Wang, D., Chen, P.-Y., Xiong, J., Liu, S., & Koyejo, S. (2022). A word is worth a thousand dollars: Adversarial attack on tweets fools stock predictions. Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies CoRR abs/ https://doi.org/10.48550/arXiv.2205.01094

Xu, J., & Du, Q. (2020). On the interpretation of convolutional neural networks for text classification. In G. D. Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugarín, and J. Lang (Eds.), *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela*, *Spain, August 29-September 8, 2020 Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, Vol. 325. Frontiers in Artificial Intelligence and Applications (pp. 2252–2259). IOS Press.

Xu, J., Zhang, C., Zheng, X., Li, L., Hsieh, C.-J., Chang, K.-W., & Huang, X. (2022). Towards adversarially robust text classifiers by learning to reweight clean examples. In *Findings of the Association for Computational Linguistics: ACL 2022* (pp. 1694–1707), Dublin, Ireland: Association for Computational Linguistics.

Yang, P., Chen, J., Hsieh, C., Wang, J., & Jordan, M. I. (2018). Greedy attack and gumbel attack: Generating adversarial examples for discrete data. CoRR, abs/1805.12316.

Yoo, J. Y., & Qi, Y. (2021). Towards improving adversarial training of NLP models CoRR, abs/ https://doi.org/10.48550/arXiv.2109.00544

Zang, Y., Qi, F., Yang, C., Liu, Z., Zhang, M., Liu, Q., & Sun, M. (2019). Word-level textual adversarial attacking as combinatorial optimization CoRR abs/ https://doi.org/10.48550/arXiv.1910.12196

Zeng, J., Zheng, X., Xu, J., Li, L., Yuan, L., & Huang, X. (2021). Certified robustness to text adversarial attacks by randomized [MASK]. arXiv e-prints, page arXiv: 2105.03743.

Zhang, X., Zhao, J. J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. CoRR, abs/1509.01626.

How to Cite: Shakeel, N. & Shakeel, S. (2022). Context-Free Word Importance Scores for Attacking Neural Networks. *Journal of Computational and Cognitive Engineering* 1(4), 187–192, https://doi.org/10.47852/bonviewJCCE2202406