2023 年臺灣國際科學展覽會
優勝作品專輯

作品編號	**190029**
參展科別	電腦科學與資訊工程
作品名稱	**Adversarial Attacks Against Detecting Bot Generated Text**
得獎獎項	三等獎

國　　家	**Singapore**
就讀學校	**Raffles Institution**
指導教師	**Lee Chan Lye**
作者姓名	**Evan Lim Hong Jun**

關鍵詞　**adversarial、attack、bot**

作者照片

tnam Block
SCHOOL PREFECT
AUSPICIUM MELIORIS AEVI
RAFFLES INSTITUTION

1 Introduction

With the introduction of the transformer architecture by Vaswani et al. (2017), contemporary Text Generation Models (TGMs) have shown incredible capabilities in generating neural text that, for humans, is nearly indistinguishable from human text (Radford et al., 2019; Zellers et al., 2019; Keskar et al., 2019). Although TGMs have many potential positive uses in writing, entertainment and software development (Solaiman et al., 2019), there is also a significant threat of these models being misused by malicious actors to generate fake news (Uchendu et al., 2020; Zellers et al., 2019), fake product reviews (Adelani et al., 2020), or extremist content (McGuffie & Newhouse, 2020).

TGMs like GPT-2 generate text based on a given prompt, which limits the degree of control over the topic and sentiment of the neural text (Radford et al., 2019). However, other TGMs like GROVER and CTRL allow for greater control of the content and style of generated text, which increases its potential for misuse by malicious actors (Zellers et al., 2019; Keskar et al., 2019). Additionally, many state-of-the-art pre-trained TGMs are available freely online and can be deployed by low-skilled individuals with minimal resources (Solaiman et al., 2019). There is therefore an immediate and substantial need to develop methods that can detect misuse of TGMs on vulnerable platforms like social media or e-commerce websites.

Several methods have been explored in detecting neural text. Gehrmann et al. (2019) developed the GLTR tool which highlights distributional differences in GPT-2 generated text and human text, and assists humans in identifying a piece of neural text. The other approach is to formulate the problem as a classification task to distinguish between neural text and human text and train a classifier model (henceforth a 'detector'). Simple linear classifiers on TF-IDF vectors or topology of attention maps have also achieved moderate performance (Solaiman et al., 2019; Kushnareva et al., 2021). Zellers et al. (2019) propose a detector of GROVER generated text based on a linear classifier on top of the GROVER model and argue that the best TGMs are also the best detectors. However, later results by Uchendu et al. (2020) and Solaiman et al. (2019) show that this claim does not hold true for all TGMs. Consistent through most research thus far is that fine-tuning the BERT or RoBERTa language model for the detection task achieves state-of-the-art performance (Radford et al., 2019; Uchendu et al., 2020; Adelani et al., 2020; Fagni et al., 2021). I will therefore be focussing on attacks against a fine-tuned RoBERTa model.

Although extensive research has been conducted on detecting generated text, there is a significant lack of research in adversarial attacks against such detectors (Jawahar et al., 2020). However, the present research that does exist preliminarily suggests that neural text detectors are not robust, meaning that the output can change drastically even for small changes in the text input and thus that these detectors are vulnerable to adversarial attacks (Wolff, 2020).

In this paper, I extend on Wolff's (2020) work on adversarial attacks on neural text detectors by proposing a series of attacks designed to counter detectors as well as an algorithm to optimally select for these attacks without compromising on the fluency of generated text. I do this with reference to a fine-tuned RoBERTa detector and on two datasets: (1) the GPT-2 WebText dataset (Radford et al., 2019) and (2) the Tweepfake dataset (Fagni et al., 2021). Additionally, I experiment with possible defences against these attacks, including (1) using count-based features, (2) stylometric features and (3) adversarial training.

2 Related Work

Attacks in natural language processing (NLP)

Adversarial attacks have been extensively explored primarily in computer vision tasks, where extremely minor perturbations on select pixels can easily trick the model whilst remaining unnoticeable to humans (Carlini & Wagner, 2016). However, adversarial attacks on natural language present a greater challenge due to its discrete nature, meaning all changes to the text are noticeable and can result in jarring issues in the syntax or coherence of the entire sentence.

There have been multiple attempts at generating adversarial examples for various language tasks, including sentiment analysis and textual entailment (Iyyer et al., 2018; Alzantot et al., 2018). These attacks generally formulate the problem as generating adversarial text that remains semantically and syntactically similar to the original text. Alzantot et al. (2018) do this by randomly selecting a word in the current sentence, and considering a set of candidate synonyms that fit in the context of the word, before selecting the synonym that will maximise the target label prediction probability. This general framework remains consistent across many adversarial attacks in NLP (Alzantot et al., 2018; Jin et al., 2019; Li et al., 2019).

Attacks on neural text detectors

Although there is some research into adversarial attacks in NLP, there is very little research specifically about attacks against neural text detectors, which could become a significant tool for malicious actors to evade detection (Jawahar et al., 2020). Wolff (2020) explores two attacks on detectors: replacing characters with unicode homoglyphs (swapping English “a”s to Cyrillic “a”s) and words with commonly misspelt words (swapping ‘except’ to ‘exept’). He randomly modifies a percentage of words in the neural text of the GPT-2 WebText dataset to attack a fine-tuned RoBERTa detector. Although the homoglyph attack achieved close to 100% success rate and would be indistinguishable from regular characters to a human, such attacks could be easily defended in the real world setting by banning homoglyphs or flagging accounts that use them. As such, I will not be exploring homoglyph attacks in this paper. Additionally, the misspelling attack is reliant on the size of the misspelling dictionary, significantly limiting potential attacks. Instead, I extend on Wolff’s (2020) work by (1) suggesting alternative character-level and word-level attacks, (2) proposing a score to measure sentence fluency and (3) an algorithm to optimally select words to attack.

Context of the attack

Similar to work by Alzantot et al. (2018), I better simulate the real world by using the black-box setting, meaning the attacker has no prior knowledge of or access to model architecture, parameters or training data. Thus, the attacker may only query the model and get the model prediction and confidence scores. The attacker seeks to (1) fool the detector with an adversarial attack while (2) ensuring that the text remains recognisable and comprehensible to humans.

3 Method

3.1 Datasets

(1) *GPT-2 WebText Dataset*: Similar to Wolff (2020), I use 5000 test samples generated by GPT-2 large with top-k 40 sampling provided by Solaiman et al. (2019). The texts in this dataset have an average length of 2091 words.

(2) *Tweepfake*: Fagni et al. (2021) compiled real-world tweets taken from known human and bot accounts. It includes text generated by a variety of TGMs, which Fagni et al. (2021) separate into GPT-2, RNN, and others (Markov chains, LSTM). The Tweepfake dataset is marked by a few characteristics that contrast the GPT-2 WebText dataset, namely the text is (1) very short, (2) has many rare and out-of-vocab words, (3) does not always comply with standard sentence structure and grammar, and (4) is 'real' in the sense that all the tweets were actually posted to Twitter. I use the Tweepfake dataset in conjunction with the GPT-2 WebText dataset to gain more realistic and generalisable results for adversarial attacks in the real world. It is overall balanced between human and bot text and is split to 20712 training, 2558 test and 2302 validation samples. The texts in this dataset have an average length of 110 words.

3.2 Neural Text Detector

For the GPT-2 WebText dataset, I only use the pre-trained RoBERTa model provided by Solaiman et al. (2019) due to the cost constraints of fine-tuning a model on the large training set.

For the Tweepfake dataset, I fine-tune a RoBERTa model and train for 3 epochs, similar to Fagni et al. (2021).

3.3 Problem formulation

Given a pre-trained classification model $F: X \to Y$, which maps from input text X to a corresponding set of labels Y, for a particular input text $x \in X$, we have to find a valid adversarial example x_{adv} such that

$$F(x) \neq F(x_{adv}), \text{ and } S(x) - S(x_{adv}) \leq \epsilon$$

where $S: X \to \mathbb{R}$ is a function that measures the fluency of input text X and $\epsilon \in \mathbb{R}^{+}$. In other words, for a given input text X, we have to search for an adversarial text X_{adv} which fools the detector to giving a different prediction label and decreases in fluency by less than ϵ.

In conventional adversarial attacks in NLP, the attacker seeks to, as far as possible, preserve semantic similarity between the original text and the adversarial text (Alzantot et al., 2018; Jin et al., 2019). In our case, since the specific meaning of x is originally generated by the TGM on which the attacker has limited control, the attacker is not concerned with preserving the specific semantics of x. Semantics is also less of a concern if the TGM is being used to generate some posting history over time, so that the corresponding social media account looks credible. Consequently, we can treat x_{adv} as generated text on its own. Although we can assume that x_{adv} will generally share a similar overall meaning to x, we do not have to measure and preserve specific semantic similarity between x and x_{adv} in our algorithm. Instead, the primary objective of x_{adv} is to fool the detector whilst seeming fluent and 'natural' to humans. Thus, unlike previous work, I measure and preserve the change in fluency instead of semantic similarity.

Fluency scoring

In this paper, I define fluency as *linguistic acceptability*, which generally measures the syntactic and semantic validity of the text (Schütze, 1996). I use pseudo-log-likelihood (PLL) scores as proposed by Salazar et al. (2021) to measure fluency. In masked language modelling, a token $w_t$is replaced by the [MASK] token and predicted by the model using all preceding and succeeding tokens. I define the new tokenized sentence $W_{/t} = (w_1, \ldots, w_{t-1}, w_{t+1}, \ldots w_{|W|})$, and $P_{MLM}(w_t \mid W_{/t})$ as the probability that the masked language model predicts w_t as the mask token in the new sentence $W_{/t}$. Iterating through all tokens, the pseudo-log-likehood score for the sentence is then given as the sum of log probabilities for each token w_t:

$$PLL(W) = \sum_{t=1}^{|W|} logP_{MLM}(w_t \mid W_{/t})$$

This serves as a good approximation for the acceptability and thus fluency of the text as determined by a masked language model. I use the BERT masked language model to compute the PLL score. In table 1, we observe that by switching the order of words in the second and third examples to make the sentence grammatically incorrect, the PLL score drops.

Text	**PLL Score**
My friend is a citizen of Singapore.	-16.3 (Most fluent)
A citizen of Singapore my friend is?	-24.7
My friend is a Singapore of citizen.	-49.9 (Least fluent)

Table 1: Examples for PLL fluency scoring

3.4 Attacks

I propose four character-level attacks and two word-level attacks for neural text detection. (1) **Space Insertion (Cspace)**: Inserts a space randomly in the target word. (2) **Character swap (Cswap)**: Randomly swaps two adjacent characters in the target word. (3) **Character deletion (Cdel)**: Randomly deletes a character in the target word. (4) **Character substitute (Csub)**: Randomly swaps a character in the word with visually similar characters or characters nearby on a QWERTY keyboard. These four character-level attacks can be commonly found in real-world settings in human typos which serves both to mimic human text to detectors whilst remaining natural to other humans. The two word-level attacks are (5) **Misspelling (Mis)**: Randomly replace the target word with a corresponding commonly misspelt word. This is the same attack performed by Wolff (2020) and uses the Wikipedia list of commonly misspelt words[1]. (6) **Synonym (Syn)**: Randomly replace the target word with a synonym. I use NLTK's interface to wordnet[2] to find a list of synonyms for a given target word. I also do not consider stopwords. A limitation of this attack is that the synonym chosen may not take into account the context of the target word, which could result in synonyms that are semantically similar but incoherent with the entire sentence. I show a list of examples of each attack in table 2.

Original	**Cspace**	**Cswap**	**Cdel**	**Csub**	**Mis**	**Syn**
heredity	hered ity	heerdity	herediy	herrdity	heridity	genetic endowment
familiar	fam iliar	fmailiar	famliar	familisr	famillliar	companion

Table 2: Examples of attacks used

Attack Algorithm

My attack algorithm follows the general framework proposed by previous work (Jin et al., 2019). The critical part of the algorithm is calculating the importance score of target words.

[1] https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings/For_machines

[2] https://www.nltk.org/_modules/nltk/corpus/reader/wordnet.html

The importance score of word w_t is calculated as the change in confidence of the model after removing word w_t:

$$c_t = F(w_1, \ldots, w_{|W|}) - F(w_1, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{|W|})$$

where $|W|$ is the total number of tokens and $F(W)$ returns the confidence that the input text W is generated by a TGM. Tokens with higher importance scores mean that they make the detector more confident that the text is bot generated. Since we hope to fool the detector into thinking the text is human written, we should focus on tokens with higher importance first.

The algorithm works by first calculating the importance scores of every word in the given text. Then, we target the words from most important to least important. For every target word, we try different attacks on the target word and chooses the attack which decreases confidence the most. After every attack on a target word, we check if the fluency score for the perturbed text has decreased below a threshold ϵ compared to the original. We continue the attacks until the prediction for the perturbed text switches to the human label or the fluency score change drops below the threshold.

For the GPT-2 WebText dataset, due to the length of the text, I first split the text into individual sentences. I find the importance of each sentence similar to the method discussed above, only instead of removing a single word, I remove the entire sentence. I iterate through each sentence from most important to least important and run the same algorithm as described above.

3.5 Defense

I study two approaches to defend against adversarial attacks. Previous work has shown that adding TF-IDF features (Prakash & Madabushi, 2020) and stylometric features (Sari et al., 2018) can enhance detector performance. Hence, my first approach pre-trains a simple classifier based on these features, then concatenates the final hidden layer with the outputs from RoBERTa before feeding that into another classifier.

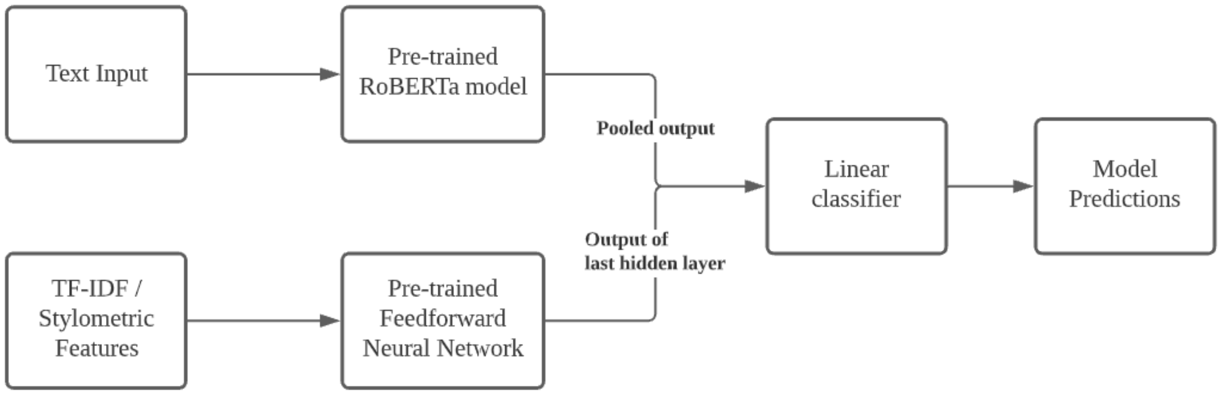
Text Input
Pre-trained RoBERTa model
Pooled output
Linear classifier
Model Predictions
Output of last hidden layer
TF-IDF / Stylometric Features
Pre-trained Feedforward Neural Network

Figure 1: Architecture of ensemble model with TF-IDF or stylometric features

I also experiment with adversarial training, where I augment the training set with pre-generated adversarial text so that the detector can learn to be robust to such noise. I first generate 6000 adversarial examples from the training set, then append these examples back to the existing training set. In order to ensure that the training set remains balanced between bot and human text, I also random sample for 6000 human text and duplicate it back into the training set. I then use the updated training set to adversarially train a new detector.

4 Results

4.1 WebText Dataset

	Original	All	Conly	Cspace	Cswap	Cdel	Csub	Mis	Syn
Recall	99.0	0.4	0.4	1.6	1.4	1.0	0.8	2.4	19.8
Avg. perturb	0	6.4	6.7	9.1	8.2	8.5	6.7	8.3	13.8

Table 3: Results of adversarial attack on GPT-2 WebText dataset. 'All' is an optimised attack where all word and character-level attacks are performed, whereas 'Conly' only performs the character-level attacks. 'Avg. perturb' is the average number of words perturbed in each successful attack.

All attacks were conducted with a fluency score change threshold ϵ of 100, which was determined experimentally through qualitative analysis of adversarial text at different thresholds. From table 3, the adversarial attack with all 6 attacks causes the recall to drop from 99% to 0.4% with only minimal perturbations. Additionally, I achieved a far higher drop in recall with the misspelling attack at 2.4% compared Wolff's (2020) reported 22.68%, likely due to the inclusion of importance scoring. Finally, character substitution has the best performance with a recall at 0.8%, suggesting that imitating human typos in text can successfully fool detectors.

Tweepfake Dataset

	Original	All	Conly	Cspace	Cswap	Cdel	Csub	Mis	Syn
Overall	93.0	33.9	43.7	68.1	67.5	66.3	63.2	78.2	81.4
GPT2	86.1	15.8	20.3	37.5	38.8	39.0	37.5	54.9	59.3
RNN	98.7	66.5	75.4	92.7	93.4	91.9	91.0	95.1	97.0
Others	93.5	20.6	35.1	71.4	68.1	66.1	60.1	82.4	85.7
Avg. perturb	0	4.2	3.3	2.4	3.5	3.9	2.8	7.2	5.4

Table 4: Results of adversarial attack on Tweepfake dataset. 'Overall' is bot recall, i.e. percentage of bot text correctly detected as bot. 'GPT2', 'RNN', 'Others' is the percentage of text generated by each respective TGM that was detected.

From table 4, although misspelling and synonym attacks individually decrease bot recall by a small amount relative to the character attacks (93% to 78.2% and 81.4% respectively), adding these two word-level attacks to 'character only' attack further decreases recall from 43.7% to 33.9%. However, this also increases the average number of perturbations. This indicates that word-level attacks are critical to the overall algorithm, possibly because they mimic human error better than character-level attacks. Additionally, we see that GPT-2 generated neural text drops in recall (86.1% to 15.8%) more than RNN generated neural text (98.7% to 66.5%). This is sensible since GPT-2 text is more human-like than RNN text, which makes it easier to be augmented to fool the detector.

	Overall	**GPT2**	**RNN**	**Others**	**Avg. perturb**
With importance scoring	33.9	15.8	66.5	20.6	4.2
Without importance scoring	38.2	21.3	69.4	25.2	4.7

Table 5: Comparison of running adversarial attack algorithm with importance scoring and without importance scoring (i.e. random selection of target words) on Tweepfake dataset.

From table 5, we clearly see that importance scoring is critical to improve the performance of the adversarial attack compared to without importance scoring. However, the performance of the attack without importance scoring at 38.2% recall also suggests that randomly selecting target words is enough to achieve considerable results with minimal perturbations.

Defending against attacks

	Overall	**GPT2**	**RNN**	**Others**	**Avg. perturb**
RoBERTa	33.9	15.8	66.5	20.6	4.2
RoBERTa + TF-IDF	24.2	6.2	47.0	19.0	4.5
RoBERTa + Stylo	21.4	5.5	50.9	8.8	4.2
Adversarial training on RoBERTa	64.8	37.2	93.9	61.9	2.9

Table 6: Results of running adversarial attack on standard RoBERTa model, ensemble model with TF-IDF and Stylometric features, and adversarially trained RoBERTa model on Tweepfake dataset.

I investigate the effects of defending against adversarial attacks in Table 6. We observe that including TF-IDF and stylometric features does not defend against adversarial attacks. In fact, the detectors with TF-IDF and stylometric features perform significantly worse compared to the standard RoBERTa detector, with recall dropping from 33.9% to 24.2% and 21.4% respectively. However, adversarial training is shown to greatly improve the robustness of the detector, with recall increasing to 64.8%.

5 Discussion

GPT2 WebText vs. Tweepfake.

It is expected that the adversarial attack on the GPT2 WebText dataset is more successful than the attack on the Tweepfake dataset. This is because the GPT-2 WebText dataset is much longer than the Tweepfake dataset, giving the attacker more space to perform augmentations to the text. Additionally, the Tweepfake dataset includes other TGMs like RNNs which produce text that is less coherent and human-like than GPT-2.

Attack performance.
A small number of augmentations to neural text is shown to achieve a very high success rate at fooling detectors. Four simple character-level attacks and two word-level attacks can cause recall to drop near 0% for long text (GPT-2 WebText) and 33.9% even for very short, irregular and ‘real-world’ text (Tweepfake). This suggests that although a lot of previous work demonstrate high performance of detectors at classifying neural text near or above 90%, these detectors are very vulnerable to simple attacks.

Importance scoring.
Although importance scoring is shown to improve the success rate of the adversarial attack, it is not essential to the attack. In fact, a simple attack that randomly selects target words can achieve very good results. This suggests that a malicious actor could easily make random minor changes to the neural text and still fool a detector.

Possible Defences.
Although past work has shown that adding count-based and stylometric features to a classifier can enhance performance (Prakash & Madabushi, 2020; Sari et al., 2018), they are not successful and are in fact significantly detrimental in defending against adversarial attacks. This is likely because the adversarial attacks are designed to produce what would be interpreted as ‘rare’ words by the detector, which could disproportionately skew results towards classifying the text as human. My results also indicate that adversarial training can significantly improve the robustness of detectors against attacks. However, this approach requires the defender to know the types of attacks employed by the attacker and may be difficult to employ in practice. Future work could explore using language models that are also character-aware as detectors, as they could be more robust against character-level augmentations in the text.

6 Conclusion

In this paper, I propose adversarial attacks on neural text detectors, including an algorithm to optimally fool detectors whilst remaining fluent to other humans. My findings suggest that current neural text detectors are not robust against adversarial attacks and highlight a significant vulnerability in existing defences against bot generated text, given the ease at which malicious actors could evade detectors. This represents a significant threat to many real-world applications. I also experiment with various possible defences against the attacks. Further research should explore improving the robustness of neural text detectors.

References

Adelani, D., Mai, H., Fang, F., Nguyen, H., Yamagishi, J., & Echizen, I. (2020). Generating sentiment-preserving fake online reviews using neural language models and their human-and machine-based detection. *International Conference on Advanced Information Networking and Applications*, 1341–1354.

Alzantot, M., Sharma, Y., Elgohary, A., Ho, B.-J., Srivastava, M., & Chang, K.-W. (2018). Generating Natural Language Adversarial Examples. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2890–2896.

Carlini, N., & Wagner, D. (2016). Towards Evaluating the Robustness of Neural Networks. *CoRR, abs/1608.04644*.

Fagni, T., Falchi, F., Gambini, M., Martella, A., & Tesconi, M. (2021). Tweepfake: about detecting deepfake tweets. *PLoS ONE, 16*(5).

Gehrmann, S., Strobelt, H., & Rush, A. (2019). GLTR: Statistical Detection and Visualization of Generated Text. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 111–116.

Iyyer, M., Wieting, J., Gimpel, K., & Zettlemoyer, L. (2018). Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. *Proceedings of NAACL*.

Jawahar, G., Muhammad, A.-M., & Lakshmanan, L. (2020). Automatic Detection of Machine Generated Text: A Critical Survey. *The 28th International Conference on Computational Linguistics (COLING)*.

Jin, D., Jin, Z., Zhou, J., & Szolovits, P. (2019). Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. *CoRR, abs/1907.11932*.

Keskar, N., McCann, B., Varshney, L., Xiong, C., & Socher, R. (2019). CTRL: A Conditional Transformer Language Model for Controllable Generation.

Kushnareva, L., Cherniavskii, D., Mikhailov, V., Artemova, E., Barannikov, S., Bernstein, A., . . . Burnaev, E. (2021). Artificial Text Detection via Examining the Topology of Attention Maps. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 635–649.

Li, J., Ji, S., Du, T., Li, B., & Wang, T. (2019). TEXTBUGGER: Generating Adversarial Text Against Real-world Applications. *Network and Distributed Systems Security (NDSS) Symposium 2019*.

McGuffie, K., & Newhouse, A. (2020). The Radicalization Risks of GPT-3 and Advanced Neural Language Models.

Prakash, A., & Madabushi, H. (2020). Incorporating Count-Based Features into Pre-Trained Models for Improved Stance Detection. *Proceedings of the 3rd NLP4IF Workshop on NLP for Internet Freedom: Censorship, Disinformation, and Propaganda*, 22--32.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.

Salazar, J., Liang, D., Nguyen, T., & Kirchhoff, K. (2021). Masked Language Model Scoring. *CoRR, abs/1910.14659*.

Sari, Y., Stevenson, M., & Vlachos, A. (2018). Topic or Style? Exploring the Most Useful Features for Authorship Attribution. *Proceedings of the 27th International Conference on Computational Linguistics*, 343–353.

Schütze, C. T. (1996). *The empirical base of linguistics: Grammaticality judgments and linguistic methodology.* Language Science Press.

Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., . . . Wang, J. (2019). Release Strategies and the Social Impacts of Language Models. *OpenAI Report*.

Uchendu, A., Le, T., Shu, K., & Lee, D. (2020). Authorship attribution for neural text generation. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 8384–8395.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., N. Gomez, A., . . . Polosukhin, I. (2017). Attention Is All You Need. *Advances in neural information processing systems*, 5998–6008.

Wolff, M. (2020). ATTACKING NEURAL TEXT DETECTORS. *CoRR*. Retrieved from abs/2002.11768

Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F., & Choi, Y. (2019). Defending Against Neural Fake News. *Advances in Neural Information Processing Systems*, 9054–9065.

【評語】190029

This project proposes an adversarial attack algorithm against detecting bot generated text. This is a complete work. The experiments are complete. The problem formulation is clear. The method is described clearly. It is easy to follow up the idea proposed in this project. Some comments are given below:

The topic is not a new one. It is suggested to have a literature survey on the works that target on the same issue. More comparison between the proposed idea in this work and the one in the previous works is suggested.