Texas A&M University-San Antonio
Digital Commons @ Texas A&M University-San Antonio

Computer Science Faculty Publications | College of Business

4-21-2023

Enhancing Neural Text Detector Robustness with μAttacking and RR-Training

Gongbo Liang

Jesus Guerrero

Fengbo Zheng

Izzat Alsmadi

Follow this and additional works at: https://digitalcommons.tamusa.edu/computer_faculty

 Part of the Computer Engineering Commons

electronics

MDPI

Article

Enhancing Neural Text Detector Robustness with μAttacking and RR-Training

Gongbo Liang [1,*], Jesus Guerrero [1], Fengbo Zheng [2] and Izzat Alsmadi [1,*]

[1] College of Arts and Sciences, Texas A&M University-San Antonio, San Antonio, TX 78224, USA
[2] College of Computer and Information Engineering, Tianjin Normal University, Tianjin 300387, China
* Correspondence: gliang@tamusa.edu (G.L.); ialsmadi@tamusa.edu (I.A.)

Abstract: With advanced neural network techniques, language models can generate content that looks genuinely created by humans. Such advanced progress benefits society in numerous ways. However, it may also bring us threats that we have not seen before. A neural text detector is a classification model that separates machine-generated text from human-written ones. Unfortunately, a pretrained neural text detector may be vulnerable to adversarial attack, aiming to fool the detector into making wrong classification decisions. Through this work, we propose μAttacking, a mutation-based general framework that can be used to evaluate the robustness of neural text detectors systematically. Our experiments demonstrate that μAttacking identifies the detector's flaws effectively. Inspired by the insightful information revealed by μAttacking, we also propose an RR-training strategy, a straightforward but effective method to improve the robustness of neural text detectors through finetuning. Compared with the normal finetuning method, our experiments demonstrated that RR-training effectively increased the model robustness by up to 11.33% without increasing much effort when finetuning a neural text detector. We believe the μAttacking and RR-training are useful tools for developing and evaluating neural language models.

Keywords: machine learning security; neural text generation; machine text detection; mutation testing

check for updates

Citation: Liang, G.; Guerrero, J.; Zheng, F.; Alsmadi, I. Enhancing Neural Text Detector Robustness with μAttacking and RR-Training. *Electronics* **2023**, *12*, 1948. https://doi.org/10.3390/electronics12081948

Received: 17 March 2023
Revised: 15 April 2023
Accepted: 19 April 2023
Published: 21 April 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Since AlexNet [1], neural networks (NNs) have been the major driving force for rapid AI development in the last ten years and have shown promising results in various domains recently [2–10]. With the advanced techniques, NNs may produce textual and imagery content looking as if genuinely created by humans [11–13], which benefits society in many domains, from medical image processing [14–17] to speech-to-text conversion [18,19], machine language translation [20,21], marketing communication [22,23], and so forth. However, such advanced technology also makes it easier to generate human-like content at a large scale for nefarious activities, for instance, generating misinformation [24,25] and targeting specific groups for political agenda [26,27]. This newly emerging threat is even more troubling with the recent development of large language models [28,29], such as OpenAI's ChatGPT [30], GPT-4 [31], and Google's Bard [32]. Researchers are actively developing neural text detectors for distinguishing neural text (i.e., NN-generated text) from human-written ones and have shown plausible performance for long sequence detection [33,34]. However, short sequence detection is still challenging with a performance that is not better than random guessing [26]. More importantly, pretrained neural text detectors are vulnerable to adversarial attacks, aiming to fool the detector into making wrong classification decisions. Unfortunately, systematically evaluating the robustness of a neural text detector is still non-trivial due to the lack of existing evaluation tools.

Mutation analysis is a white-box attacking method used to evaluate the quality of existing software [35,36]. The analysis is based on well-defined mutation operators that mimic typical programming errors and measure the percentage of mutation operators that cause different software behavior from the original design [37]. This strategy follows "the boiling frog syndrome", [38] modifying a program in small ways. The purpose of such a kind of attacking is to help the software tester locate weaknesses in the code that are seldom [39,40].

Inspired by the advances in mutation analysis, we propose a general mutation-based framework for evaluating the robustness of pretrained neural text detectors. We call the framework μAttacking because the evaluation uses well-defined mutation operators to adversarial attack pretrained models. We particularly focus on short neural text detection since it is more challenging for existing detectors and, hypothetically, might be more harmful. Since people are heavily engaged in social media, a large-scale information operation targeting social media with short neural text may be more effective than generating longer sequences for news articles.

The μAttacking finds that the state-of-the-art RoBERTa-based detector [41] is exceptionally vulnerable to simple but common mutations. In some cases, the detector performance is even less than 0.07 AUC (the area under the receiver operating characteristic curve) on separating human-written texts from machine-generated ones. From the experiments, we also found that neural text intends to be more grammatically correct than human-written ones. For instance, human-written texts are 100 times more likely to have missing articles (i.e., a, an, the) than those generated by neural network algorithms. Inspired by this, we propose a simple but effective RR-training strategy that introduces uncertainty to the model by randomly removing words at the training stage. Our experiments show that the robustness of the RoBERTa-based detector can be further improved by up to 11.33% on AUC when finetuning the model using RR-training, comparing the finetuning result without it. Note that no mutation operators or text are used in the finetuning stage.

We believe μAttacking and RR-training are useful language-analysis model-development tools. The μAttacking provides a systematic way to evaluate the robustness of language-analysis models that have not been evaluated before. The RR-training improves model robustness without involving additional data or a significant amount of work. To summarize, our paper presents the following intuitive contributions:

- Introducing μAttacking, a mutation-based strategy for systematically evaluating the robustness of neural network language-analysis models.
- Demonstrating robust analysis of neural text detection model using μAttacking through adversarial attacks.
- Proposing the RR-training strategy that improves the robustness of language-analysis models significantly without requiring additional data or effort.

The rest of the paper is organized as follows: The related technical background are presented in Section 2. Then, the proposed mutation-based framework and RR training strategy are introduced in Section 3. The dataset simulation and experiment setup are provided in Section 4. The detailed experimental results on various evaluation tasks are illustrated in Section 5. The paper ends with a discussion and conclusion in Section 6.

2. Technical Background

This section introduces some high-level but necessary technical background that may be needed to understand our work, including automatic text generation, neural text detection, mutation analysis, and adversarial attacks.

2.1. Automatic Text Generation

Automatic text generation is a field of study in natural language processing that combines computational linguistics and artificial intelligence [42–44]. The field has progressed significantly in recent years due to advanced neural network technologies [45,46]. Neural-network-based approaches are currently dominant in the field. Popular methods may include Transformer [11], GPT-1/2/3/4 [12,31,47,48], RoBERTa [49], and their variants, such as Med-Bert [50] and ChatGPT [30] that are used in several text domains. Due to the high text-generation performance, NN-based methods are popular in image caption generation [51], automatic text summarization [52], machine translation [53], moving script-writing [54], poetry composition [55], etc. Although the generated text may look precise and reasonable, due to the nature of neural networks, it is non-trivial to guarantee the semantic correctness of the generated text. In this work, we refer to the text generated by neural network models as neural text.

2.2. Neural Text Detection

We refer to neural text detection as distinguishing neural text from human-written ones. Although neural text detection may still be under-researched, it has attracted increasing attention over the last few years [56–58]. Various approaches have been proposed. For instance, Bhatt and Rios used linguistic accommodation in online-conversion interaction to identify whether the text was generated by a chatbot [58]. Solaiman et al. used the probability distribution expressed by neural language models directly by computing the total probability of the text sequence of interest. If the calculated probability is closer to the mean likelihood over a set of known machine-generated sequences, the text sequence is classified as machine generated [41]. This study used the RoBERTa-based detector, a state-of-the-art model, to demonstrate the neural text detection task. The model was based on the largest GPT-2 model (consisting of 1.5B parameters) [12]. Additionally, the model was finetuned to distinguish between texts being generated from the GPT-2 model and human texts. In total, 500,000 text samples were used in training [26].

2.3. Mutation Analysis

Mutation analysis, also known as mutation testing, is a fault-based system-level testing approach that evolved in software testing [37]. Unlike other methods of testing that aim at testing the correctness of software behavior, mutation testing aims at testing the robustness of the software and how to deal with input data that is not originally designed to deal with. Mutations are different versions, slightly changed, from the original code [36], and may be measured using methods such as [59]. Therefore, test cases used to test original code are also used to test mutation versions. Once a test case detects a mutation, the mutation is said to be killed (i.e., discovered). This indicates that the test cases visit the particular mutation area in the code. As such, mutation testing tests the "test cases" rather than the code, ensuring that they cover all code areas (i.e., test case coverage or effectiveness) [40]. Our work brings mutation analysis from the software testing field to natural language analysis by proposing a general framework for mutation generation. Various mutations generated by the framework can be used to evaluate the robustness of a neural text detector or any other language analysis model.

2.4. Adversarial Attacks

Adversarial attacks are growing threats in AI and machine learning, intending to fool machine learning models with deceptive data [60,61]. In general, adversarial attacks are classified into two categories: white-box attack and black-box attack [62]. In the former one, the attacker has access to the model, while in black box attacks, the attacker has no access to the property [63,64]. The deceptive data are often purposely designed to cause a model to make a mistake in its predictions despite resembling a valid input to a human. Numerous ways may be used to acquire the deceptive data or adversarial samples, such as fast gradient sign attack (FGSA) [65] and generative adversarial networks (GANs) [66].

The proposed μAttacking combines adversarial attacks and mutation analysis to evaluate language-analysis model robustness. In the adversarial machine learning domain, μAttacking is a white-box approach in which the attacker has access to the initial text. On the other hand, the mutation-based aspect aims to gradually change the text from the initial one to confuse detectors (typically known as the boiling frog attack). Extending the mutation-based features to adversarial attacks is essential since it is crucial for applications that depend on machine learning models to make decisions to ensure that such models are immune to inputs they are not trained for. This can also fit in the scope of an important recent and evolving area, detecting unknown attacks.

3. Method

Inspired by the mutation analysis in software testing, we propose μAttacking with a general framework for two types of mutation operators (Section 3.1.1). The mutation operators are used for adversarial sample generation. The adversarial samples can then be used to evaluate a neural text detector model through adversarial attacks (Section 3.1.2). Inspired by the evaluation result of μAttacking (Section 5.1), we propose a random-remove (RR) training strategy to improve the robustness of the neural text-detection model (Section 3.2).

3.1. μAttacking

The proposed μAttacking framework contains two components: (1) mutation operators and (2) adversarial attacks. Two general mutation operators are proposed to simulate two common scenarios when malicious actors try to bypass the neural text detector: character- and word-level mutation operators.

3.1.1. General Mutation Operator Framework

Given a text corpus (e.g., a paragraph), τ, which contains an ordered list of words, $\omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$, and an ordered list of punctuation, $v = \{v_1, v_2, \ldots, v_m\}$, the order of the words and punctuations in the corresponding lists are followed by the original order in the text corpus. Mutation operators, $\mu_c(\cdot)$ and $\mu_w(\cdot)$ are used to generate the character-level and word-level mutations of τ, respectively.

Mathematically, the character-level operator, $\mu_c(\cdot)$, is defined as:

$$\omega_i' = \mu_c(\omega_i, \rho, \sigma), \tag{1}$$

where $\omega_i \in \omega$, ρ is a letter in ω_i, σ is the mutation of ρ, and ω_i' is the mutation of ω_i. After the mutation, the original ω, where $\omega \in \tau$ and $\omega = \{\omega_1, \omega_2, \ldots, \omega_i, \ldots, \omega_n\}$, is changed to $\omega' = \{\omega_1, \omega_2, \ldots, \omega_i', \ldots, \omega_n\}$. The mutation text corpus is $\tau' = \{\omega', v\}$. For instance, given a text corpus, τ, where $\tau =$ `Mutation-Based Adversarial Attacking`. The ordered list of words is $\omega = \{$`Mutation, Based, Adversarial, Attackinge`$\}$. Assume $\omega_i =$ `Mutation`, $\rho =$ `a`, and, $\sigma = \alpha$. Then, $\omega_i' =$`Mutαtion` and $\tau' =$ `Mutαtion-Based Adversarial Attacking`.

The word-level operator, $\mu_w(\cdot)$, is defined as, mathematically:

$$\omega'' = \mu_w(\omega, \omega_j, \omega_j'), \tag{2}$$

where $\omega_j \in \omega$, ω_j' replaces ω_j, and $\omega'' = \{\omega_1, \omega_2, \ldots, \omega_j', \ldots, \omega_n\}$. For instance, given the same text corpus as before where: $\tau =$ `I am generated by a neural net`, assume $\omega_j =$ `net`, and $\omega_j' =$ `network`. After applying $\mu_w(\cdot)$ to $\omega =$ `net`, the mutation is $\omega'' =$ $\{$`I, am, generated, by, a, neural, network`$\}$.

The character-level mutation operators intend to simulate the situation where nefarious actors make minor character-level changes to the text to bypass automatic filtering systems on the internet. The words may look weird after the changes but may still present the same information to human users. Although the strategy looks simple, it works effectively in many cases. For instance, for a keyword-based filtering system, the character-level mutation fools the filtering system by generating a new "word" that may not be on the keyword list. For a contextual-embedding-based system, the newly generated "word" might be embedded far away from the original word in the embedding space, which may also effectively fool the filtering system.

The word-level operators aim to simulate another strategy to fool an automatic filtering system—replacing words with other words, which is also commonly used in the real world. For instance, a student may wish to pass the plagiarism check by replacing some words with synonyms in an essay copied online. The word-level mutation operators are intended to simulate such nefarious activities.

The character- and word-level mutation operators introduced in this section are two general operators. Various specific operators may be developed based on these two general types. For instance, a misspelling operator can be developed using the character-level operator, and an adjective-removing operator can also be developed by replacing the adjectives in a text corpus with "" (i.e., the empty string). As a proof of concept, in this work, we proposed nine sets of mutation operators in Section 5. Section 4.3 gives a detailed explanation of the operators. However, users are not limited to the operators used in this study. One may easily design more operators that may be more suitable for their specific tasks under the proposed framework.

3.1.2. Attacks on Neural Text Detectors

Neural network language models can be trained to distinguish human-written text vs. neural text. Among several existing detectors, the RoBERTa-based detector [41] is well known for the state-of-the-art performance on the GPT-2-generated text [26,67]. We apply adversarial attacks to the RoBERTa-based detector using the adversarial samples generated with the mutation operators introduced in Section 3.1.1. To attack the neural text detectors, we first apply the mutation operators to a set of neural texts to generate the adversarial samples. Then, the adversarial samples are used to test a pretrained RoBERTa-based detector. The basic process is illustrated in Figure 1.

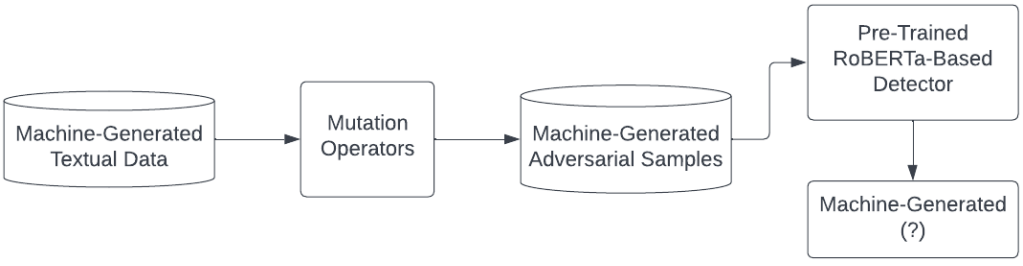


Figure 1. Adversarial attack of the pretrained RoBERTa-based detector.

Figure 2 shows the general training process of the ReBERTa-base detector, which was accomplished by finetuning a RoBERTa large model with the outputs of the 1.5 B parameter GPT-2 model. The RoBERTa large model was trained for text generation on 160 GB of text, including Books Corpus, English Wikipedia, CommonCrawl News dataset, Web text corpus, and Stories from Common Crawl [49]. To finetune the detector, both human-written and machine-generated textual data were fed to the detector to evaluate whether they were neural text or not. A binary loss function was used to assess the network prediction by comparing the prediction and the ground-truth label of the input. Afterward, the loss was back-propagated to the detector for tuning the network parameters.

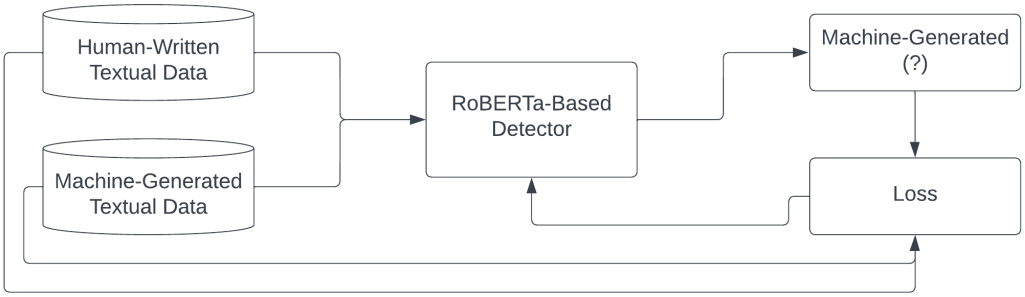


Figure 2. The RoBERTa-based detector general training procedure.

3.2. Random-Removing Training Strategy

Empirically speaking, pretrained neural network models are often less robust when handling out-of-distribution data. For instance, Wang et al. show that pretrained neural network models may achieve near-human or even above-human performance on certain medical diagnosis tasks on the in-distribution data. However, the model performance was significantly reduced when applying the pretrained model to out-of-distribution data (e.g., data samples collected from a different hospital or region) [68]. However, research showed that relaxing model prediction by introducing a small amount of uncertainty to classification models helps increase the overall model performance [69–72]. For instance, Label Smoothing [69] is widely used in computer vision classification tasks. Instead of targeting 1 for the correct class, Label Smoothing tries to predict $1 - \epsilon$ for the correct class label, where ϵ is usually a small number, such as 0.01. MixCut [71] is another example, which mixes two images of two classes together into one image and predicts the mix proportionally to the number of pixels of combined images. Both Label Smoothing and MixCut demonstrate a significant performance improvement by introducing a small uncertainty to the model.

Inspired by the methods mentioned above, as well as the robustness evaluation result using μAttacking in Section 5.1, we propose a random-removing (RR) training strategy to improve the robustness of neural text detectors by introducing a small uncertainty during the training stage (Algorithm 1). We name the method as RR-training. Specifically, given a training instance, we randomly apply the word-level mutation operator μ_w where $\mu_w = (\omega, \omega_j, \texttt{""})$ to k words in ω. For each input text τ, k is randomly decided and $k \in [0, floor(len(\omega) \times m)]$, where $len(\omega) =$ indicates the number of words in the ordered list, $floor(\cdot)$ indicates the floor function, and $m \in [0.0, 1.0]$ is the maximum percentage of words we might want to remove. For instance, assume $\tau =$`{random remove some words}` and $m = 0.3$. Then, $len(\omega) = 4$, $floor(len(\omega) \times 0.3) = 1$, and $0 \leq k \leq 1$. Our RR-training strategy may also be considered as an input-level dropout [73] operation.

Algorithm 1: RR-Training

```
/* Require: Neural text detector model h_Θ(·), training data
   D = {T, L}, text corpus T = {τ_1, τ_2, ..., τ_i}, classification labels
   L = {ι_1, ι_2, ..., ι_i} where ι ∈ {0, 1}, loss function 𝕃(·), word-level
   mutation operator μ_w(·), random class ℝ(·), and sort function
   sort(·). */

for i ∈ 0 to len(T) do
    ω, v ← τ_i  // Obtain the ordered word list and the punctuation list
     from τ_i
    r ← ℝ.randInt(0, 1)            // Obtain a random integer for 0 or 1.

    /* Apply mutation to τ_i if r is 1 */
    if r == 1 then
        n ← ℝ.randInt(0, floor(len(ω) × m))    // Obtain a random number of
         words
                                          // that will be removed from ω
        indices ← {index_1, index_2, ..., index_n}   // Obtain a list of n random
         indices of
                                       // words, where 0 <= index_i < len(ω).
        sort(indices)                  // Sort the list in ascending order.

        /* Remove n words */
        for j ∈ 0 to n do
            ω = μ_w(ω, ω_indices[j], " ")        // Using the word-level mutation
             operator
                                                 // to remove words from ω
        end
        τ_i ← {ω, v}           // Update τ_i. n words are removed from the
         original τ_i
    end

    /* Train the model */
    logits ← h_Θ(τ_i)                          // Feed data to the model
    loss = 𝕃(logits, ι_i)                        // Calculate the loss
    loss.back()                                  // Back-propagation
end
```

4. Dataset and Experiment Setup

4.1. Data Preparation and Neural Text Acquiring

Due to the low diversity between human-written text and machine-generated short text, detecting machine-generated short text is more challenging than for longer text. Unfortunately, neural text detection for shorter text is as important as longer text, if not more. For instance, the influence of social media in our life is increasing daily. A large-scale information operation on social media may lead to national crises. Automatic content generation reduces the human resource demand of launching such an operation. Thus, distinguishing machine-generated posts from those written by humans in social media posts is a critical step in defending an AI-backed information operation. Ideally, the experiment should use actual social media data; however, acquiring such data is non-travel, especially acquiring machine-generated data since no effective methods exist to determine whether a machine generates a particular social post. Thus, we had to generate some simulated data by ourselves, and we designed an oversimplified social media scenario where social media posts may have two typical features:

- The text in a post is usually relatively short. For instance, the typical length of tweets is often between 25–50 characters.
- A good percentage of posts contain images and text. The text is often related to the image.

To simulate the oversimplified social media scenario mentioned above and to acquire neural text and human-written text more efficiently, we used the MS COCO2017 dataset [74], a dataset containing both imagery and text data. The text data are written by humans, describing the corresponding images. Each imagery sample comes with five human-written descriptions. To speed up our experiments, we selected the first 10,000 imagery samples from the dataset. Then, we used a bin-search-based image caption generation model [75] to generate five more descriptions for each image. The pretrained weights of the caption-generation model from the authors were used in this study directly. Our simulated social media post dataset contains 10,000 images and 100,000 captions, with 50,000 from the original MS COCO2017 dataset and 50,000 generated by us. Then, we used the original MS COCO2017 descriptions as human-written samples and those generated by us as neural text. The dataset was divided into the train, val, and test sub-sets on the image level with a ratio of 70:15:15, respectively.

4.2. Models and Training Setup

Three neural text-detection models are compared in this study, namely: (1) RoBERTa-Base, (2) RoBERTa-Finetune, and (3) RoBERTa-RR.

- RoBERTa-Base: The RoBERTa-based detector was originally released by OpenAI. We used the model as-is and used the author-related weights.
- RoBERTa-Finetune: A finetuned RoBERTa-based detector using our training set. All the embedding layers were frozen during the training. We only optimized the classifier as part of the model. No mutation operators were applied during the training.
- RoBERTa-RR: Another finetuned RoBERTa-based detector that followed the same setup of the RoBERTa-Finetune model but was trained using the RR-training strategy.

The `RobertaForSequenceClassification` model (i.e., `roberta-large`) provided by HuggingFace [76] was used as the architecture of the RoBERTa-based detector. The OpenAI pretrained weights (https://openaipublic.azureedge.net/gpt-2/detector-models/v1/detector-large.pt (accessed on 8 October 2022)) were used for RoBERTa-Base and were used to initialize the RoBERTa-Finetune and RoBERTa-RR models. All the experiments were conducted using an Nvidia A100 GPU card. Both finetuned models were trained for 50 epochs using our train set. The best checkpoint was selected based on the evaluation performed on the val set. The test set was applied for the final evaluation. We padded each input to have a maximum length of 50. The AdamW [77] optimizer with a learning rate of 1×10^{-4}, a batch size of 512, and the cross-entropy loss was used during the training.

4.3. Mutation Operators and Adversarial Attacks

We developed nine sets of mutation operators using the character- and word-level operator introduced in Section 3.1.1, including six sets of character-level operators and three sets of word-level operators. The output text from the nine sets of operators was then used as adversarial samples to attack the three models. In total, up to 2355 operators were derived from the nine sets and were used in this study.

The character-level operators were categorized into two groups: (1) $\mu_c(\omega_i, a, \alpha)$ and (2) $\mu_c(\omega_i, e, \epsilon)$. Each group contained three sets of mutation operators: the set of articles, the set of adjectives, and the set of adverbs. The operators of articles focused on the three articles—a; an; the. The operators of adjectives focused on 527 common adjectives provided by the Missouri Baptist University (MBU) Academic Success Center (https://www.mobap.edu/wp-content/uploads/2013/01/list_of_adjectives.pdf (accessed on 8 October 2022)). The operators of adverbs focused on 255 common adverbs also provided by MBU (https://www.mobap.edu/wp-content/uploads/2013/01/list_of_adverbs.pdf (accessed on 8 October 2022)). In total, the experiment involved six sets of character-level mutation

operators, with up to 1570 operators (i.e., 3 articles + 527 adjectives + 255 adverbs, each word has up to two character-level mutations).

The word-level mutation operators were applied by replacing certain words with empty strings, $\mu_w(\omega, \omega_i, \texttt{""})$. Following the setup of the character-level mutation operators, the ω_i used in μ_w was also selected from the three types of words—articles, adjectives, and adverbs. The same word lists were used to build the word-level operators.

5. Results and Analysis

5.1. Evaluating Robustness with µAttacking

Seven binary classification tasks were used to evaluate the robustness of the pretrained neural text detector, RoBERTa-Base, on our test set, namely human text vs. neural text and human text vs. neural text w/ different mutations. We combined the two character-level mutation groups (i.e., $\mu_c(\omega_i, a, \alpha)$ and $\mu_c(\omega_i, e, \epsilon)$) in reporting the performance and reducing the total sets of mutations to six, namely, (1) word-level mutation of articles, (2) word-level mutation of adjectives, (3) word-level mutation of adverbs, (4) character-level mutation of articles, (5) character-level mutation of adjectives, (6) and character-level mutation of adverbs. Note that the number of mutation operators was not changed.

We tested the RoBERTa-Base model using the original test set and the test set with six sets of mutations applied to the neural text. Ideally, the performance on the mutated text should be similar to those without mutations if a model is robust enough. Table 1 exhibits the detailed performance for the test cases. We evaluated the performance using the area under the receiver operating characteristic curve (denoted as AUC) and accuracy (denoted as ACC). The performance of AUC is also summarized in Figure 3a, and the performance of ACC compared with other models is shown in Figure 3b.

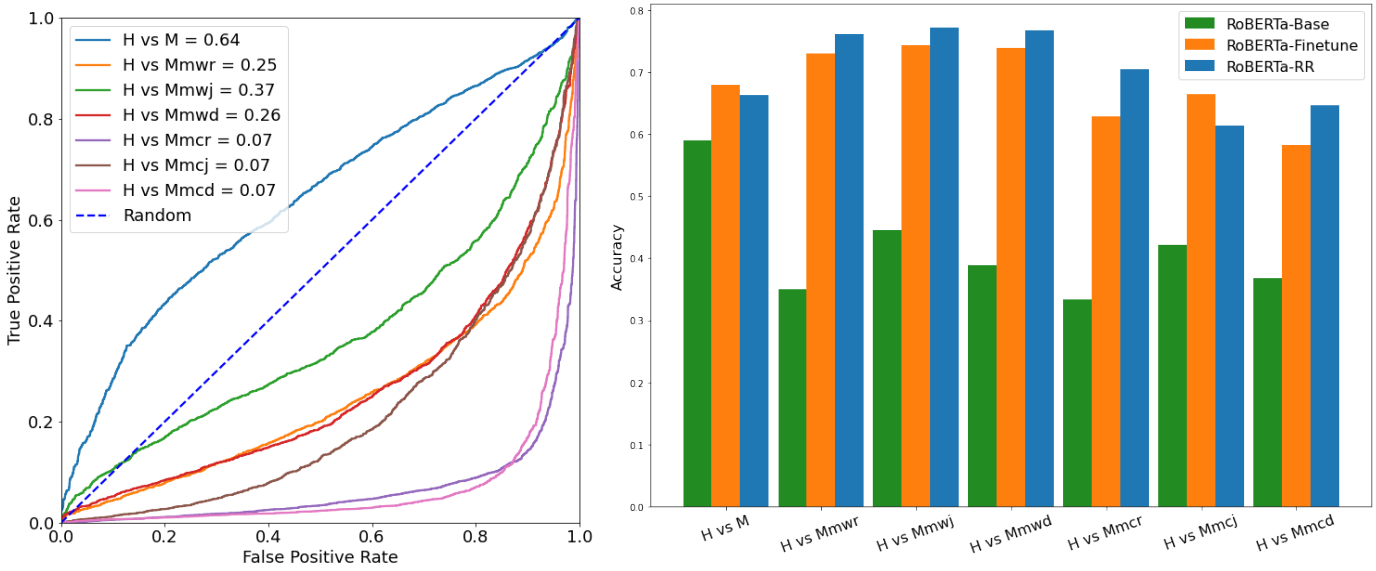
In the table and figure, we used `H` to indicate human-written text, `M` to indicate neural text (i.e., machine-generated text), and the subscripts to indicate the type of mutation operators. The first letter in the subscript, i.e., `m`, indicates that the result is for mutation; the second letter in the subscript indicates the level of the mutation operator, either word-level (denoted as `w`) or character-level (denoted as `c`); and the third letter in the subscript indicates the types of words that the mutation operator is applied on, such as `r` for articles, `j` for adjectives, and `d` for adverbs. For instance, `H vs. M` means Human-Written Text vs. Neural Text, `H vs.` $\texttt{M}_{\texttt{mwr}}$ means human-written text vs. neural text mutations with the word-Level mutation operators of articles (i.e., the world-level mutation operators were applied to the articles), and `H vs.` $\texttt{M}_{\texttt{mcd}}$ means human-written text vs. neural text mutations w/the character-level mutation operators of adjectives.

The results revealed that the RoBERTa-Base model is not robust enough to handle the mutated text. For instance, when applying the model on the original test set (i.e., for `H vs. M`), the model achieves a 0.6381 AUC and 58.92% accuracy. However, the performance reduced significantly when applying the model on the test set with mutated neural text (i.e., the last six columns in Table 1), with the lowest AUC of 0.0676 and ACC of 33.38%.

The results also illustrated that the RoBERTa-Base model is extremely vulnerable to mutation operators for removing words that lead us to analyze our dataset statistically. We found that human-written text is ≈100 times more likely to be missing articles (i.e., a, an, the) than neural text. Our analysis shows that approximately ≈4% of human-written text does not have a single article in the sentence. However, the same number for neural text is only 0.04%, which is only 99% less than human-written text.

Table 1. Detailed performance of RoBERTa-Base in different classification cases.

Metric	H vs. M	H vs. $\texttt{M}_{\texttt{mwr}}$	H vs. $\texttt{M}_{\texttt{mwj}}$	H vs. $\texttt{M}_{\texttt{mwd}}$	H vs. $\texttt{M}_{\texttt{mcr}}$	H vs. $\texttt{M}_{\texttt{mcj}}$	H vs. $\texttt{M}_{\texttt{mcd}}$
AUC	0.6381	0.2488	0.3695	0.2591	0.0676	0.0676	0.0714
ACC	0.5892	0.3504	0.4460	0.3892	0.3338	0.4213	0.3686



(**a**) (**b**)

Figure 3. (**a**) The area under the receiver operating characteristic curve for RoBERTa-Base different classification cases for in-distribution data. (**b**) The accuracy of RoBERTa-Base, RoBERTa-Finetune, and RoBERTa-RR in different cases. The absolute accuracy values of each bar are reported in Tables 1 and 2.

5.2. Improving Robustness Using RR-Training

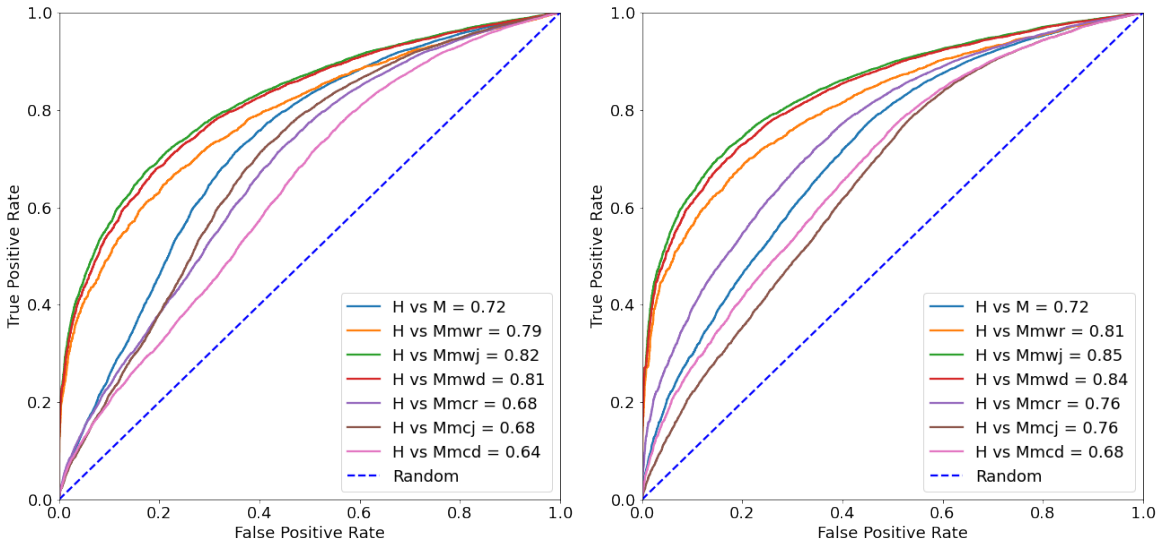
Inspired by the model evaluation result of using μAttacking in Section 5.1, we proposed RR-training to improve the neural text detector robustness. We conducted several experiments to evaluate and demonstrate the effectiveness of this training method. First, we compared the finetuning model performance produced with and without RR-training–RoBERTa-Finetune and RoBERTa-RR, respectively (Section 5.2.1). Then, we applied the pretrained RR-training model on out-of-distribution data (Section 5.2.2). After that, we compared RR-training with dropout (Section 5.2.3). Finally, we evaluated the effective percentage of words removed from the text (Section 5.2.4).

5.2.1. In-Distribution Data

To show the advantage of the RR-training, we first finetuned two models from RoBERTa-Base using our training set. The two models were RoBERTa-Finetune and RoBERTa-RR, finetuned with and without RR-training, respectively. Then, the same binary classification tasks, introduced in Section 5.1, were evaluated on our test set in this section. Table 2 shows the detailed performance of the RoBERTa-Finetune and RoBERTa-RR over the seven binary classification tasks. The best performance amount the two models are highlight in bold text. The performance is also summarized in Figures 3b and 4. The results reveal that RoBERTa-Finetune and RoBERTa-RR are significantly better than RoBERTa-Base. The RoBERTa-RR performs substantially better on 11 out of 14 evaluation results, with up to an 11.33% improvement on AUC and up to a 12.05% improvement on ACC.

Table 2. Detailed performance of RoBERTa-Finetune and RoBERTa-RR in different classification cases.

Metric	Model	H vs. M	H vs. M_{mwr}	H vs. M_{mwj}	H vs. M_{mwd}	H vs. M_{mcr}	H vs. M_{mcj}	H vs. M_{mcd}
AUC	**Finetune**	**0.7227**	0.7855	0.8216	0.8140	0.6788	0.6788	0.6363
	RR	0.7168	**0.8124**	**0.8476**	**0.8395**	**0.7557**	**0.7557**	**0.6842**
ACC	**Finetune**	**0.6794**	0.7295	0.7436	0.7384	0.6283	**0.6646**	0.5818
	RR	0.6625	**0.7609**	**0.7714**	**0.7678**	**0.7046**	0.6135	**0.6465**



(**a**) RoBERTa-Finetune (**b**) RoBERTa-RR

Figure 4. The area under the receiver operating characteristic curve for RoBERTa-Finetune and RoBERTa-RR in different classification cases for in-distribution data.

Although the RoBERTa-Finetune worked better on the original test set (i.e., without mutations), the RoBERTa-RR shows its strength in mutation cases. With RR-training, the model performance on separating human-written text and neural text with word-level mutations is improved by an average of 10.12% on AUC and 4.80% on ACC. The model also significantly improves the classification performance of character-level mutations with an average of 3.24% on AUC and a 4.01% on ACC. In general, RoBERTa-RR is 6.43% and 4.37% better than RoBERTa-Finetune on AUC and ACC, respectively. This experiment shows that RR-training can improve the model robustness effectively.

5.2.2. Out-of-Distribution Data

To further evaluate the robustness of the proposed RR-training strategy and the model transferability, we also conducted an experiment that assessed the pretrained RoBERTa-RR on an out-of-distribution (OOD) dataset. The OOD dataset was generated using [78] by passing the images of our testing set to the neural text generation model.

Figure 5 shows the AUC of RoBERTa-RR on the seven binary classification tasks, demonstrating that the RoBERTa-RR model produces similar performance to the in-distribution data with an average of 6% degrade on character-level mutations and 10.45% degraded word-level mutations. We believe this is an acceptable performance since the two datasets significantly differ from each other. We only present the evaluation result on AUC in this section because the accuracy also shows the same trend.

5.2.3. Comparing with Dropout

Since RR-training may be considered as the input-level dropout, we also compared RR-training with dropout. Figure 6a shows the RoBERTa-Base with a dropout layer randomly dropping 30% neurons during the training. Figure 6b shows the AUC curves of the RoBERTa-RR model with $m = 0.3$ (i.e., random removal of up to 30% of words in the sentence). The figures indicate that with the same spontaneous removal rate, the proposed RR-training generates substantially better than dropout. Figure 6c shows that when adding the same dropout layer (with 30% of random dropping) to the same RoBERTa-RR ($m = 0.3$), the model performance is reduced significantly. We believe this is caused by the piling up of randomness. Adding 30% neuron dropping and 30% input-level dropping might roughly translate to up to 90% random dropping, which introduces too much uncertainty during the training.

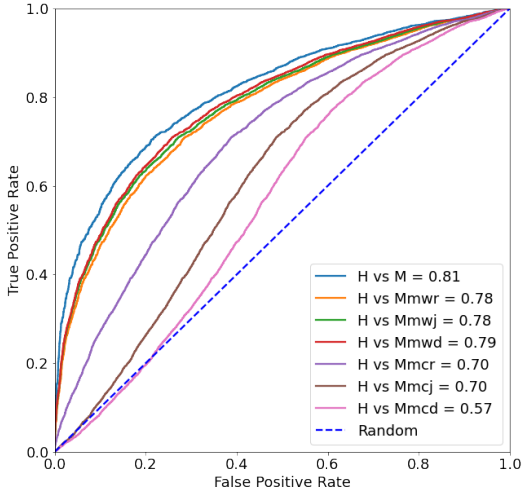
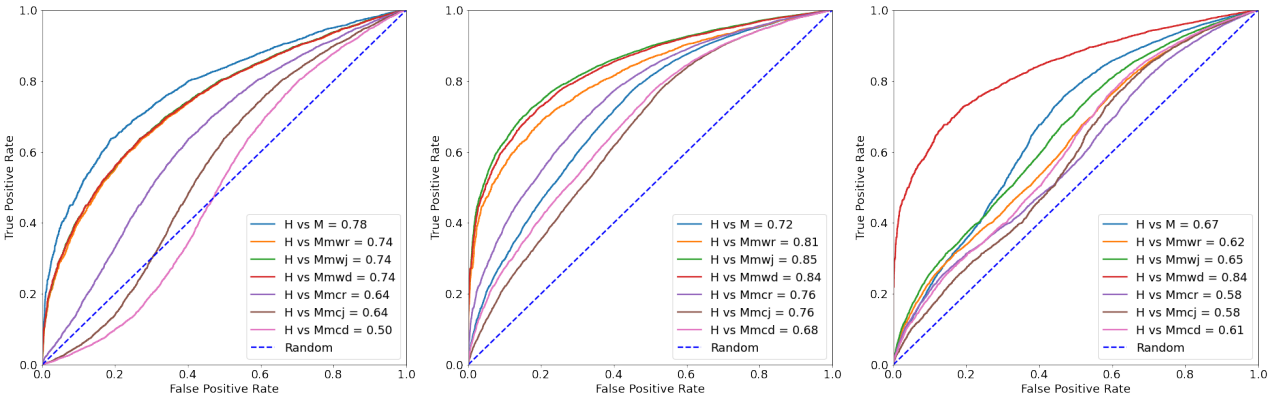


Figure 5. The area under the receiver operating characteristic curve for RoBERTa-RR on out-of-distribution data.

5.2.4. Effective of Random Removing Ratio

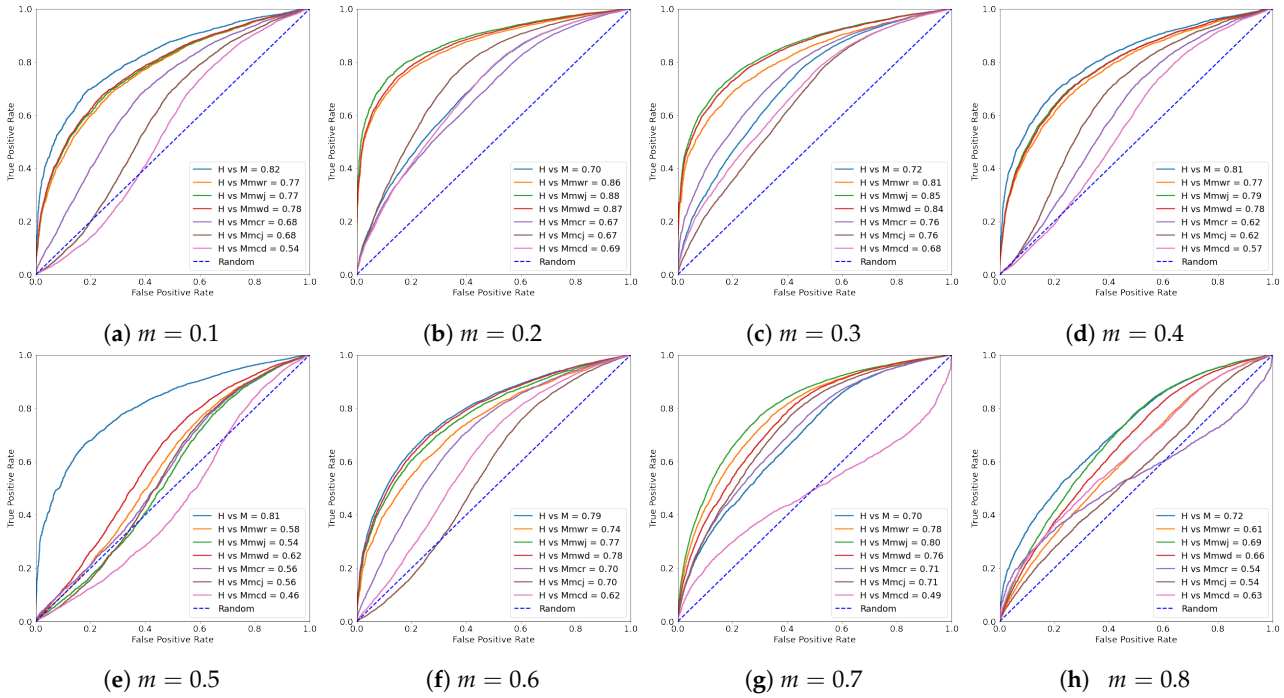
A hyperparameter m (i.e., the random removal ratio) needs to be selected to use the proposed RR-training. This section shows the effect of different m values. We trained eight different RoBERTa-RR models with $m = \{0.1, 0.2, 0.3, \ldots 0.7, 0.8\}$ using our training set. We then tested each model using our testing set. Figure 7 illustrates the AUC curves of different m values.

The figure shows that RR-training is not very sensitive to the m value, especially when m is small (i.e., $m \leq 0.4$). With a larger m, more uncertainty will be introduced to the model during training. The model performance will be degraded if m is large, e.g., $m \geq 0.5$. However, when m is too small, RR-training might not be able to introduce enough uncertainty during the training, which also leads to non-optimal performance (e.g., $m = 0.1$). Similar to many other hyperparameters used for neural networks, such as learning rate and dropout ratio for dropout, a sweet spot needs to be found to obtain optimal performance. In general, we suggest using a m value between 0.2 and 0.4. In this study, we set $m = 0.3$.



(**a**) RoBERTa-Base with dropout (**b**) RoBERTa-RR without dropout (**c**) RoBERTa-RR with dropout

Figure 6. The area under the receiver operating characteristic curve for RoBERTa-Base, RoBERTa-RR without random dropout and RoBERTa-RR with 30% random dropout (i.e., dropout = 0.3).



(**a**) $m = 0.1$ (**b**) $m = 0.2$ (**c**) $m = 0.3$ (**d**) $m = 0.4$

(**e**) $m = 0.5$ (**f**) $m = 0.6$ (**g**) $m = 0.7$ (**h**) $m = 0.8$

Figure 7. Effect of random removal ratio (i.e., m) for RR-training.

6. Discussion, Conclusions, and Future Directions

Due to the progress of advanced neural network techniques, language models may produce human-like text that benefits society from various aspects. However, there is also a risk that such advanced intelligent techniques may be used by malicious actors for nefarious activities, such as information operations on a large scale, from spreading misinformation to targeting specific groups for political agenda. A robust detector that can separate neural text from human-written text is the first step to defending against such newly emerged AI-powered cyber threats. Unfortunately, how researchers may systematically evaluate the robustness of a neural text detector is still being determined in the literature. Thus, we propose the μAttacking, a mutation-based framework that evaluates the robustness of neural text detectors through adversarial attacking. With μAttacking, researchers can generate textual test cases in a well-controlled and close-ended environment. The precise test cases provide a systematic evaluation of the robustness of language analysis models.

In this study, we used μAttacking to evaluate the state-of-the-art RoBERTa-based detector for separating human-written text from machine-generated text. The results show that the detector has significant flaws. The insightful evaluation results lead us to a statistical analysis and finds that human-written text is 100 times more likely to miss articles in sentences than neural text. Based on this finding, we proposed RR-training to significantly improve the model robustness. However, we also believe that the robustness issue of neural text detectors should be better addressed at the feature level.

Tokenization or word embedding is the essential step that feeds natural language text to a neural network model. The input-level changes generated by the mutation operators of μAttacking cause changes in the tokenization stage, leading the word to have a different embedding vector. In addition, the contextual embedding characteristics of RoBERTa may make this vector change more dramatic since contextual embedding generates different embeddings of the same word based on the context. Thus, when feeding the adversarial samples into the RoBERTa-based detector, contextual embedding layers also produce different embeddings of the non-changed words. In this case, the distance between the original and adversarial samples may increase even more in the feature spaces. Thus, one future direction of this work is reducing the feature space differences between the original and adversarial samples. Some potentially useful methods might include using contrastive learning and siamese network [79,80] and dynamic feature alignment [81,82].

In conclusion, we proposed the μAttacking, a mutation-based evaluation framework that produces close-ended, precise textual test cases for evaluating language-analysis models that take text sequences as input. We demonstrated the framework for evaluating the neural text detector. The results showed that the detector is extremely vulnerable to simple adversarial attacks, and they lead to an insightful analysis of the flaws, which inspired us to propose RR-training for improving the robustness of the language-analysis model. The proposed method is a general-purpose method that is not limited to any specific application domain. The μAttacking can be used in evaluating the robustness of any downstream applications that take text sequences as input, such as ChatBot detection [58], SQL injection detection [83], and software debugging [84]. In addition, researchers may also design their mutation operators that better fit their specific tasks under our framework, making the proposed method more flexible and applicable to various models. We believe that the proposed μAttacking and RR-training will be useful for those seeking systematic and insightful analysis of language models, as well as developing robustness language analysis models.

Author Contributions: Study design, G.L. and I.A.; data collection, G.L. and J.G.; model training, G.L., J.G. and F.Z.; data analysis, G.L., F.Z. and I.A.; literature search, G.L., J.G. and I.A.; figures, G.L.; writing, G.L., J.G., F.Z. and I.A. All authors reviewed the manuscript and contributed to revisions. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Training data is available upon request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
2. Zhang, Y.; Liang, G.; Salem, T.; Jacobs, N. Defense-pointnet: Protecting pointnet against adversarial attacks. In Proceedings of the IEEE International Conference on Big Data, Los Angeles, CA, USA, 9–12 December 2019; pp. 5654–5660.
3. Xing, X.; Liang, G.; Blanton, H.; Rafique, M.U.; Wang, C.; Lin, A.L.; Jacobs, N. Dynamic image for 3d mri image alzheimer's disease classification. In Proceedings of the European Conference on Computer Vision Workshops, Glasgow, UK, 23–28 August 2020; Part I; pp. 355–364.
4. Su, Y.; Zhang, Y.; Liang, G.; ZuHone, J.A.; Barnes, D.J.; Jacobs, N.B.; Ntampaka, M.; Forman, W.R.; Nulsen, P.E.J.; Kraft, R.P.; et al. A deep learning view of the census of galaxy clusters in illustristng. *Mon. Not. R. Astron. Soc.* **2020**, *498*, 5620–5628. [CrossRef]
5. Ying, Q.; Xing, X.; Liu, L.; Lin, A.L.; Jacobs, N.; Liang, G. Multi-modal data analysis for alzheimer's disease diagnosis: An ensemble model using imagery and genetic features. In Proceedings of the 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society, Mexico City, Mexico, 1–5 November 2021; pp. 3586–3591.
6. Liu, L.; Chang, J.; Wang, Y.; Liang, G.; Wang, Y.P.; Zhang, H. Decomposition-based correlation learning for multi-modal mri-based classification of neuropsychiatric disorders. *Front. Neurosci.* **2022**, *16*, 832276. [CrossRef] [PubMed]
7. Liang, G.; Xing, X.; Liu, L.; Zhang, Y.; Ying, Q.; Lin, A.L.; Jacobs, N. Alzheimer's disease classification using 2d convolutional neural networks. In Proceedings of the 2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society, Mexico City, Mexico, 1–5 November 2021; pp. 3008–3012.
8. Lin, S.C.; Su, Y.; Liang, G.; Zhang, Y.; Jacobs, N.; Zhang, Y. Estimating cluster masses from SDSS multiband images with transfer learning. *Mon. Not. R. Astron. Soc.* **2022**, *512*, 3885–3894. [CrossRef]
9. Li, K.; Zheng, F.; Wu, P.; Wang, Q.; Liang, G.; Jiang, L. Improving Pneumonia Classification and Lesion Detection Using Spatial Attention Superposition and Multilayer Feature Fusion. *Electronics* **2022**, *11*, 3102. [CrossRef]
10. Xing, X.; Rafique, M.U.; Liang, G.; Blanton, H.; Zhang, Y.; Wang, C.; Jacobs, N.; Lin, A.L. Efficient Training on Alzheimer's Disease Diagnosis with Learnable Weighted Pooling for 3D PET Brain Image Classification. *Electronics* **2023**, *12*, 467. [CrossRef]
11. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
12. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
13. Workman, S.; Rafique, M.U.; Blanton, H.; Jacobs, N. Revisiting Near/Remote Sensing with Geospatial Attention. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–20 June 2022.
14. Mihail, R.P.; Liang, G.; Jacobs, N. Automatic hand skeletal shape estimation from radiographs. *IEEE Trans. Nanobiosci.* **2019**, *18*, 296–305. [CrossRef]

15. Liang, G.; Fouladvand, S.; Zhang, J.; Brooks, M.A.; Jacobs, N.; Chen, J. Ganai: Standardizing ct images using generative adversarial network with alternative improvement. In Proceedings of the 2019 IEEE International Conference on Healthcare Informatics, Xi'an, China, 10–13 June 2019; pp. 1–11.
16. Liu, L.; Zhang, P.; Liang, G.; Xiong, S.; Wang, J.; Zheng, G. A spatiotemporal correlation deep learning network for brain penumbra disease. *Neurocomputing* **2023**, *520*, 274–283. [CrossRef]
17. Liu, L.; Chang, J.; Wang, Y.; Zhang, P.; Liang, G.; Zhang, H. Llrhnet: Multiple lesions segmentation using local-long rang features. *Front. Neuroinform.* **2022**, *16*, 859973. [CrossRef]
18. Ajami, S. Use of speech-to-text technology for documentation by healthcare providers. *Natl. Med. J. India* **2016**, *29*, 148. [PubMed]
19. Wang, C.; Tang, Y.; Ma, X.; Wu, A.; Okhonko, D.; Pino, J. Fairseq S2T: Fast Speech-to-Text Modeling with Fairseq. In Proceedings of the AACL Association for Computational Linguistics, Suzhou, China, 4–7 December 2020; pp. 33–39.
20. Li, Q. Machine Translation of English Language Using the Complexity-Reduced Transformer Model. *Mob. Inf. Syst.* **2022**, *2022*, 6603576. [CrossRef]
21. Khan, N.S.; Abid, A.; Abid, K. A novel natural language processing (NLP)–based machine translation model for English to Pakistan sign language translation. *Cogn. Comput.* **2020**, *12*, 748–765. [CrossRef]
22. Kaczorowska-Spychalska, D. Chatbots in marketing. *Management* **2019**, *23*, 251–270. [CrossRef]
23. Cheng, Y.; Jiang, H. Customer–brand relationship in the era of artificial intelligence: Understanding the role of chatbot marketing efforts. *J. Prod. Brand Manag.* **2022**, *31*, 252–264. [CrossRef]
24. Huang, K.H.; McKeown, K.; Nakov, P.; Choi, Y.; Ji, H. Faking Fake News for Real Fake News Detection: Propaganda-loaded Training Data Generation. *arXiv* **2022**, arXiv:2203.05386.
25. Rezaei, S.; Kahani, M.; Behkamal, B. The process of multi-class fake news dataset generation. In Proceedings of the International Conference on Computer Engineering and Knowledge, Mashhad, Iran, 28–29 October 2021; pp. 134–139.
26. Stiff, H.; Johansson, F. Detecting computer-generated disinformation. *Int. J. Data Sci. Anal.* **2022**, *13*, 363–383. [CrossRef]
27. Alsmadi, I.; Ahmad, K.; Nazzal, M.; Alam, F.; Al-Fuqaha, A.; Khreishah, A.; Algosaibi, A. Adversarial attacks and defenses for social network text processing applications: Techniques, challenges and future research directions. *arXiv* **2021**, arXiv:2110.13980.
28. NBCNews. Americans Are Wary of AI Tech like ChatGPT, Data Shows. Available online: https://www.nbcnews.com/meet-the-press/data-download/chatgpt-ai-tech-leaves-americans-concerned-excited-rcna71369/ (accessed on 19 February 2023).
29. DailyMail. Rogue Artificial Intelligence Chatbot Declares Love for User, Tells Him to Leave His Wife and Says It Wants to Steal Nuclear Codes. Available online: https://www.dailymail.co.uk/news/article-11761271/Rogue-artificial-intelligence-chatbot-declares-love-user-tells-leave-wife.html (accessed on 16 February 2023).
30. OpenAI.com. ChatGPT: Optimizing Language Models for Dialogue. Available online: https://openai.com/blog/chatgpt/ (accessed on 17 December 2022).
31. OpenAI. GPT-4 Technical Report. *arXiv* **2023**, arXiv:2303.08774.
32. Google.com. An Important Next Step on Our AI Journey. Available online: https://blog.google/technology/ai/bard-google-ai-search-updates/ (accessed on 24 February 2023).
33. Pu, J.; Sarwar, Z.; Abdullah, S.M.; Rehman, A.; Kim, Y.; Bhattacharya, P.; Javed, M.; Viswanath, B. Deepfake Text Detection: Limitations and Opportunities. In Proceedings of the IEEE Symposium on Security and Privacy, Los Alamitos, CA, USA, 22–24 May 2023; pp. 19–36.
34. Wolff, M.; Wolff, S. Attacking neural text detectors. *arXiv* **2020**, arXiv:2002.11768.
35. Madeyski, L.; Orzeszyna, W.; Torkar, R.; Jozala, M. Overcoming the equivalent mutant problem: A systematic literature review and a comparative experiment of second order mutation. *IEEE Trans. Soft. Eng.* **2013**, *40*, 23–42. [CrossRef]
36. Misra, S. Evaluating four white-box test coverage methodologies. In Proceedings of the Canadian Conference on Electrical and Computer Engineering: Toward a Caring and Humane Technology, Montreal, QC, Canada, 4–7 May 2003; Volume 3, pp. 1739–1742.
37. DeMillo, R.A.; Lipton, R.J.; Sayward, F.G. Hints on test data selection: Help for the practicing programmer. *Computer* **1978**, *11*, 34–41. [CrossRef]
38. Huang, L.; Joseph, A.D.; Nelson, B.; Rubinstein, B.I.; Tygar, J.D. Adversarial machine learning. In Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, Chicago, IL, USA, 21 October 2011; pp. 43–58.
39. Niedermayr, R.; Juergens, E.; Wagner, S. Will my tests tell me if i break this code? In Proceedings of the International Workshop on Continuous Software Evolution and Delivery, Austin, TX, USA, 14–15 May 2016; pp. 23–29.
40. Jia, Y.; Harman, M. An analysis and survey of the development of mutation testing. *IEEE Trans. Soft. Eng.* **2010**, *37*, 649–678. [CrossRef]
41. Solaiman, I.; Brundage, M.; Clark, J.; Askell, A.; Herbert-Voss, A.; Wu, J.; Radford, A.; Wang, J. Release strategies and the social impacts of language models. *arXiv* **2019**, arXiv:1908.09203.
42. Mann, W.C. An overview of the Penman text generation system. In Proceedings of the AAAI, Washington, DC, USA, 22–26 August 1983; pp. 261–265.
43. Jelinek, F. Markov source modeling of text generation. In *The Impact of Processing Techniques on Communications*; Springer: Berlin/Heidelberg, Germany, 1985; pp. 569–591.
44. Guo, J.; Lu, S.; Cai, H.; Zhang, W.; Yu, Y.; Wang, J. Long text generation via adversarial training with leaked information. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.

45. Zhu, Y.; Lu, S.; Zheng, L.; Guo, J.; Zhang, W.; Wang, J.; Yu, Y. Texygen: A benchmarking platform for text generation models. In Proceedings of the 41st International ACM Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 1097–1100.
46. Yu, W.; Zhu, C.; Li, Z.; Hu, Z.; Wang, Q.; Ji, H.; Jiang, M. A survey of knowledge-enhanced text generation. *ACM Comput. Surv. (CSUR)* **2022**, *54*, 227. [CrossRef]
47. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. Available online. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf (accessed on 16 February 2023).
48. Yu, W.; Zhu, C.; Li, Z.; Hu, Z.; Wang, Q.; Ji, H.; Jiang, M. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
49. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
50. Rasmy, L.; Xiang, Y.; Xie, Z.; Tao, C.; Zhi, D. Med-BERT: Pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction. *NPJ Dig. Med.* **2020**, *4*, 86. [CrossRef]
51. Vinyals, O.; Toshev, A.; Bengio, S.; Erhan, D. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 July 2015; pp. 3156–3164.
52. El-Kassas, W.S.; Salama, C.R.; Rafea, A.A.; Mohamed, H.K. Automatic text summarization: A comprehensive survey. *Expert Syst. Appl.* **2021**, *165*, 113679. [CrossRef]
53. Vaswani, A.; Bengio, S.; Brevdo, E.; Chollet, F.; Gomez, A.N.; Gouws, S.; Jones, L.; Kaiser, Ł.; Kalchbrenner, N.; Parmar, N. Tensor2tensor for neural machine translation. *arXiv* **2018**, arXiv:1803.07416.
54. Zhu, Y.; Song, R.; Dou, Z.; Nie, J.Y.; Zhou, J. Scriptwriter: Narrative-guided script generation. *arXiv* **2020**, arXiv:2005.10331.
55. Yi, X.; Li, R.; Sun, M. Generating chinese classical poems with rnn encoder-decoder. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 211–223.
56. Gehrmann, S.; Strobelt, H.; Rush, A.M. Gltr: Statistical detection and visualization of generated text. *arXiv* **2019**, arXiv:1906.04043.
57. Adelani, D.I.; Mai, H.; Fang, F.; Nguyen, H.H.; Yamagishi, J.; Echizen, I. Generating sentiment-preserving fake online reviews using neural language models and their human-and machine-based detection. In Proceedings of the International Conference on Advanced Information Networking and Applications, Caserta, Italy, 15–17 April 2020; pp. 1341–1354.
58. Bhatt, P.; Rios, A. Detecting Bot-Generated Text by Characterizing Linguistic Accommodation in Human-Bot Interactions. In Proceedings of the Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, Online, 1–6 August 2021; pp. 3235–3247.
59. Cauteruccio, F.; Terracina, G.; Ursino, D. Generalizing identity-based string comparison metrics: Framework and techniques. *Knowl. Based Syst.* **2020**, *187*, 104820. [CrossRef]
60. Lowd, D.; Meek, C. Adversarial learning. In Proceedings of the Eleventh ACM International Conference on Knowledge Discovery in Data Mining, Chicago, IL, USA, 21–24 August 2005; pp. 641–647.
61. Qiu, S.; Liu, Q.; Zhou, S.; Wu, C. Review of artificial intelligence adversarial attack and defense technologies. *Appl. Sci.* **2019**, *9*, 909. [CrossRef]
62. Zhang, Y.; Song, Y.; Liang, J.; Bai, K.; Yang, Q. Two sides of the same coin: White-box and black-box attacks for transfer learning. In Proceedings of the 26th ACM International Conference on Knowledge Discovery & Data Mining, Online, 6–10 July 2020; pp. 2989–2997.
63. Athalye, A.; Carlini, N.; Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 274–283.
64. Bhagoji, A.N.; He, W.; Li, B.; Song, D. Practical black-box attacks on deep neural networks using efficient query mechanisms. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 154–169.
65. Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial machine learning at scale. *arXiv* **2016**, arXiv:1611.01236.
66. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
67. Jawahar, G.; Abdul-Mageed, M.; Lakshmanan, L.V. Automatic detection of machine generated text: A critical survey. *arXiv* **2020**, arXiv:2011.01314.
68. Wang, X.; Liang, G.; Zhang, Y.; Blanton, H.; Bessinger, Z.; Jacobs, N. Inconsistent performance of deep learning models on mammogram classification. *J. Am. Coll. Radiol.* **2020**, *17*, 796–803. [CrossRef]
69. Pereyra, G.; Tucker, G.; Chorowski, J.; Kaiser, Ł.; Hinton, G. Regularizing neural networks by penalizing confident output distributions. *arXiv* **2017**, arXiv:1701.06548.
70. Müller, R.; Kornblith, S.; Hinton, G.E. When does label smoothing help? In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 4–14 September 2019; Volume 32.
71. Yun, S.; Han, D.; Oh, S.J.; Chun, S.; Choe, J.; Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 6023–6032.
72. Liang, G.; Zhang, Y.; Wang, X.; Jacobs, N. Improved Trainable Calibration Method for Neural Networks on Medical Imaging Classification. In Proceedings of the British Machine Vision Conference, Online, 7–10 September 2020.

73. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
74. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
75. Shrimal, A.; Chakraborty, T. Attention beam: An image captioning approach. *arXiv* **2020**, arXiv:2011.01753.
76. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv* **2019**, arXiv:1910.03771.
77. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv* **2017**, arXiv:1711.05101.
78. Vinyals, O.; Toshev, A.; Bengio, S.; Erhan, D. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 652–663. [CrossRef] [PubMed]
79. Koch, G.; Zemel, R.; Salakhutdinov, R. Siamese neural networks for one-shot image recognition. In Proceedings of the ICML Deep Learning Workshop, Lille, France, 6–1 July 2015; Volume 2.
80. Liang, G.; Greenwell, C.; Zhang, Y.; Xing, X.; Wang, X.; Kavuluru, R.; Jacobs, N. Contrastive cross-modal pre-training: A general strategy for small sample medical imaging. *IEEE J. Biomed. Health Inform.* **2021**, *26*, 1640–1649. [CrossRef] [PubMed]
81. Zhang, Y.; Liang, G.; Jacobs, N. Dynamic feature alignment for semi-supervised domain adaptation. In Proceedings of the British Machine Vision Conference, London, UK, 21–24 November 2022.
82. Dong, J.; Cong, Y.; Sun, G.; Xu, X. Cscl: Critical semantic-consistent learning for unsupervised domain adaptation. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; pp. 745–762.
83. Hlaing, Z.C.S.S.; Khaing, M. A detection and prevention technique on sql injection attacks. In Proceedings of the IEEE Conference on Computer Applications, Yangon, Myanmar, 27– 28 February 2020; pp. 1–6.
84. Zhao, Y.; Su, T.; Liu, Y.; Zheng, W.; Wu, X.; Kavuluru, R.; Halfond, W.G.; Yu, T. ReCDroid+: Automated End-to-End Crash Reproduction from Bug Reports for Android Apps. *ACM Trans. Soft. Eng. Methodol.* **2022**, *31*, 1–33. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.