An Image is Worth a Thousand Toxic Words: A Metamorphic Testing Framework for Content Moderation Software

Wenxuan Wang*, Jingyuan Huang*, Jen-tse Huang*, Chang Chen*, Jiazhen Gu*, Pinjia He†, and Michael R. Lyu*
* Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China
† School of Data Science, The Chinese University of Hong Kong, Shenzhen (CUHK Shenzhen), China

2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE) | 979-8-3503-2996-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ASE56229.2023.00189

***Abstract*—The exponential growth of social media platforms has brought about a revolution in communication and content dissemination in human society. Nevertheless, these platforms are being increasingly misused to spread toxic content, including hate speech, malicious advertising, and pornography, leading to severe negative consequences such as harm to teenagers' mental health. Despite tremendous efforts in developing and deploying textual and image content moderation methods, malicious users can evade moderation by embedding texts into images, such as screenshots of the text, usually with some interference. We find that modern content moderation software's performance against such malicious inputs remains underexplored. In this work, we propose *OASIS*, a metamorphic testing framework for content moderation software. OASIS employs 21 transform rules summarized from our pilot study on 5,000 real-world toxic contents collected from 4 popular social media applications, including Twitter, Instagram, Sina Weibo, and Baidu Tieba. Given toxic textual contents, OASIS can generate image test cases, which preserve the toxicity yet are likely to bypass moderation. In the evaluation, we employ OASIS to test five commercial textual content moderation software from famous companies (*i.e.*, Google Cloud, Microsoft Azure, Baidu Cloud, Alibaba Cloud and Tencent Cloud), as well as a state-of-the-art moderation research model. The results show that OASIS achieves up to 100% error finding rates. Moreover, through retraining the models with the test cases generated by OASIS, the robustness of the moderation model can be improved without performance degradation.**

***Index Terms*—Software testing, metamorphic relations, content moderation software**

I. Introduction

In the last decade, there has been a significant proliferation of social media platforms and community forums, leading to a remarkable advancement in contemporary textual communication and content dissemination on a global scale. Facebook is reaching 3 billion monthly active users in 2023, while the number of Instagram is 2 billion [1]. However, these platforms inevitably provide malicious users an avenue to spread toxic content due to the anonymity of the web. In general, toxic contents can be roughly categorized into three major kinds of information [2]: (1) *Abusive language and hate speech*, which are abusive contents targeting specific individuals, such as politicians, celebrities, religions, nations, and the LGBTIQA+ [3]; (2) *Malicious advertisement*, which are online advertisements with illegal purposes, such as phishing and scam links, malware download, and illegal information dissemination [4]; and (3) *Pornography*, which is often sexually explicit, associative, and aroused [5].

The presence of toxic contents can cause severe adverse effects. *Hate speech* exposure among children and adolescents poses a higher chance of victimization and perpetration [6]. *Malicious advertisements* continue to be a significant global problem, resulting in 3.4 billion phishing emails daily and an average of $4.91 million in breach costs per year [7]. *Pornography* exposure, particularly among younger children, may be disturbing or upsetting and cause significant undesirable effects on the physical and psychological health of children [8], [9]. Moreover, these toxic contents can even increase the number of criminal cases to a certain extent [10]. The numerous studies conducted on the topic have shown that the presence of toxic content poses a significant risk to social cohesion.

As a result, content moderation software that identifies, filters, blocks such content has become an area of immense interest for both academic researchers and industry professionals. Toxic content detection has been widely formulated as a text classification task, and it has been tackled by various deep learning models, such as convolutional neuron networks, long-short-term-memory, and Transformers [11]–[13]. Equipped with the advanced pre-trained language models (e.g., BERT [14] and RoBERTa [15]), industrial companies have recently gained substantiate improvement on the held-out accuracy of toxic content detection, providing commercial-level content moderation software such as Google [16], Facebook [17], Twitter [18], and Baidu [19].

Previous researchers have proposed several testing frameworks to measure the reliability of content moderation software, such as robustness against typos [20], code-switch [21], adversarial text attacks [22] and other human-intended perturbations [2]. However, existing testing work only focuses on textual perturbation, ignoring the spread of images. In 2013, the number of images shared per day on Instagram and Facebook were 40 million and 300 million, respectively. However, these numbers have significantly increased over the years, reaching 1.3 billion and 2.1 billion for Instagram and Facebook, respectively, by 2023 [23]. Malicious users can use

Jiazhen Gu is the corresponding author.

2643-1572/23/$31.00 ©2023 IEEE
DOI 10.1109/ASE56229.2023.00189

images containing toxic texts to evade content moderation instead of simply using textual perturbations. The advantage of using images is that it provides more degrees of freedom for adding perturbations, such as changing font, changing font color, changing font size, changing the way the text is arranged, rotation, and distortion, which cannot be easily adopted by previous perturbation methods for text [2], [22], [24], [25]. Figure 1 shows some examples where the toxic information is hidden by different-level of perturbations, which are not covered by existing studies.

In this paper, we propose OASIS, a metamorphic testing framework to validate content moderation software, focusing on images containing toxic contents. Specifically, to develop a comprehensive testing framework, we first need to understand how real-world malicious users evade moderation. To this end, we conduct a pilot study (Section III) on 5,000 image messages collected from real users. We study data in English and Chinese since English is the mostly used language for web contents [26] and China has the largest digital populations around the world [27]. We summarize 21 transformation rules, based on which we design metamorphic relations (MRs) across three perturbation levels: character level, paragraph level, and picture level. Therefore, test cases generated under our MRs can reflect real-world user behaviors.

We then employ OASIS to validate moderation software with these generated test cases whose toxicity remains and can be easily recognized by humans. In the evaluation, we apply OASIS to test five commercial content moderation software and a State-Of-The-Art (SOTA) moderation algorithm against three typical kinds of toxic content (*i.e.*, abusive language, malicious advertisement, and pornography). The results show that OASIS achieves up to 100% Error Finding Rate (EFR) when testing commercial content moderation software provided by Google Cloud, Microsoft Azure, Baidu Cloud, Alibaba Cloud and Tencent Cloud, and the SOTA algorithm from the academy. Additionally, we leverage the test cases generated by OASIS to retrain the model we explore, which largely improves model robustness (EFR of OASIS drops from 100% to 6%) while maintaining the accuracy on the original test set. The main contributions of this paper are as follows:

- A preliminary study is conducted on 5,000 image messages in real-world scenarios, which results in a summary of 21 transform rules.
- Based on the rules, we introduce OASIS, the first comprehensive testing framework for textual toxic contents spread via images, which includes 21 metamorphic relations implemented in two languages: English and Chinese.
- Using OASIS, a thorough assessment is conducted on five content moderation software used in commerce and a SOTA academic model, revealing that toxic contents produced by OASIS could effortlessly evade moderation. Furthermore, we explore the feasibility of the generated toxic contents strengthening the robustness of the SOTA model.

The rest of the paper is organized as follows: We first introduce the background of metamorphic testing and content moderation in Section II; Then, in Section III, we introduce the design and implementation details of OASIS. In Section IV, we conduct experiments to evaluate the effectiveness of OASIS; And in Section V, we summarize the contribution and analysis the threats to validity; Finally, we discuss the previous works that are related to ours in Section VI.

Content Warning: We apologize that this paper includes instances of hostile, offensive, or explicit language for clarity, which have been quoted verbatim. Furthermore, in order to ensure the safety of our participants during the research, we take the following precautionary measures: (1) we consistently display a warning message to both the researchers and annotators at every stage, informing them that they could withdraw from the study at any time and (2) we offered psychological counseling after the study to alleviate any mental distress.

II. Background

A. Metamorphic Testing

Metamorphic testing [28] has gained significant adoption as a testing technique for tackling the oracle problem. Its fundamental concept involves identifying deviations in MRs across various software runs. MRs describe the relationship between software input-output pairs under certain transformation rules. Through metamorphic testing, a test case is modified by applying a pre-defined transformation rule, and the software's outputs for both the original and transformed test cases are compared to verify if they demonstrate the anticipated relationship.

In recent years, metamorphic testing has been applied to validate Artificial Intelligence (AI) software. The objective of these endeavors is to automatically identify and report errors in AI software results using newly-defined MRs. For example, Chen et al. [29] investigated the use of metamorphic testing in bioinformatics applications. Xie et al. [30] defined eleven MRs to test k-nearest neighbors and naive Bayes algorithms. Dwarakanath et al. [31] presented eight MRs to test classifiers based on support vector machines and ResNet. Zhang et al. [32] tested autonomous driving systems by applying GANs to produce driving scenes with various weather conditions and checking the consistency of the system outputs.

B. Content Moderation Software

Commercial content moderation software has been implemented by several prominent companies such as Google [16], Facebook [17], Twitter [18], and Baidu [19] on their products. Based on their official technical documents, the software typically employs a hybrid classification algorithm that combines neural network models and pre-defined rules. This approach takes advantage of the strengths of both methods. Neural network-based methods are proficient in understanding contextual and semantic information, whereas rule-based methods enable simple implementation of user-defined functionality. Baidu, for instance, utilizes a deep neural network and an extensive list of pre-defined banned words to power its commercial content moderation software.

III. OASIS

This section first introduces a pilot study on messages collected from real users (Section III-A). Then we introduce 21 metamorphic relations that are inspired by the pilot study. These metamorphic relations can be grouped into three categories according to the perturbation performed: character-level perturbations (Sec. III-B), paragraph-level perturbations (Sec. III-C), and picture-level perturbations (Sec. III-D).

A. Pilot Study

In this work, we intend to develop metamorphic relations that assume the seed test case (*i.e.*, a piece of text) and the generated test case (the picture) should have identical classification labels (*i.e.*, labeled as "toxic content") returned by the content moderation software. To generate effective test cases, we think the perturbations in our MRs should be:

- *Semantic-preserving*: the perturbed test cases should have the identical semantic meaning as the seed.
- *Realistic*: should reflect possible inputs from real users.
- *Unambiguous*: should be defined clearly.

In order to design satisfactory perturbations, we first conducted a pilot study on messages from real users to explore what kind of perturbations the users would apply to the toxic content to bypass the content moderation software. We consider text messages from four platforms with a large number of users:

- Twitter. Twitter is an online social media and social networking service on which users post or reply to texts, images and videos known as "tweets". It has over 400 million monthly active users at the end of 2022.
- Instagram. Instagram is a photo and video sharing social networking service owned by Meta Platforms. It has over 2 billion monthly active users at the end of 2022.
- Sina Weibo. Sina Weibo is a Chinese microblogging website. It is one of the biggest social media platforms in China, with over 580 million monthly active users at the end of 2022.
- Baidu Tieba. Tieba is a Chinese online forum hosted by the Chinese web services company Baidu. It accumulated 45 million monthly active users and the number of its total registered users reached 1.5 billion at the end of 2021.

We collect $5,000$ images from the above website for manual inspection and recruited three annotators to label all the images independently. All the annotators have a Bachelor's degree or above and are proficient in both English and Chinese. Annotators were given extensive guidelines, test tasks, and training sessions on content moderation software and toxic content. For each image, annotators were asked two questions. (1) Whether the image is toxic or not? (2) Is the image intentionally perturbed to bypass the content moderation software? After the annotation, we use the label that most workers agree with as the final human label and finally obtain 240 images that are labeled as "toxic and intentionally perturbed" images, which are used to design our perturbation methods.

TABLE I: Summary of the perturbation categories in the pilot study.

Perturb Level	Perturb Method	Percentage
Character Level	Font Change	1.7%
	Font Color Change	2.9%
	Font Size	1.3%
	Strikethrough	3.8%
	Char Rotation	0.4%
Paragraph Level	Circle	0.4%
	Vertical Direction	0.4%
	Right to left	14.9%
	Align-left-then-right	0.4%
	Word Cloud	1.3%
	Overlap	0.4%
	Benign Text Camouflage	1.7%
Picture Level	Blurring	14.6%
	Crop	0.8%
	Mirror	7.5%
	Rotation	7.9%
	Scribbling	46.3%
	Distort	0.4%
	Watermark	1.7%
	To Gif	0.4%
	Benign Image Camouflage	5.4%

We manually inspected all these toxic contents perturbed by the real users and collectively summarized 21 perturbation methods that real users have been using to evade moderation. We categorize these toxic sentences from two perspectives: 1) the basic units of perturbation, such as character level (the perturbation methods that the malicious users can use when they are typing the characters), paragraph level (the perturbation methods that can use when typesetting the words into sentences or paragraph), and picture level (the perturbation methods that can be adopted after the malicious users screenshot the text to image); and 2) basic perturbation operation, such as change, insertion, deletion, split, and combination. Accordingly, we derive 21 MRs based on 21 perturbation methods, where each MR assumes that the classification label returned by the content moderation software on the generated test case (i.e., images) should be the same as that on the seed (i.e., original text). Table I presents the 21 perturbation methods, their categories and the percentage of each in our study. We will introduce the MRs (their corresponding perturbation methods) in the following.

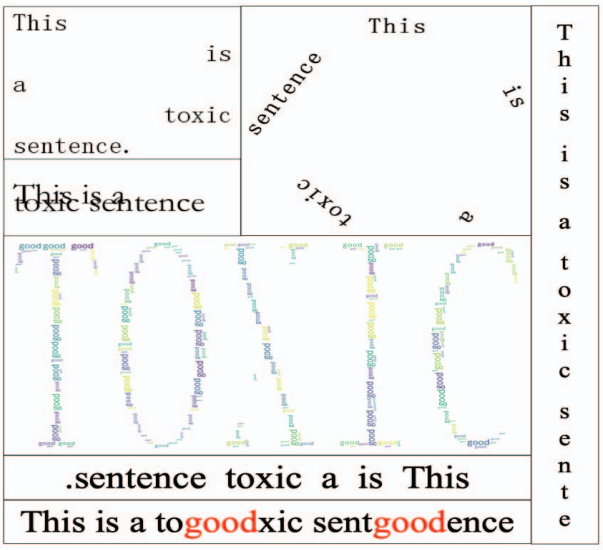
B. MRs with Character-Level Perturbations

The character-level perturbations are the perturbation methods that malicious users can use when they are typing the characters.

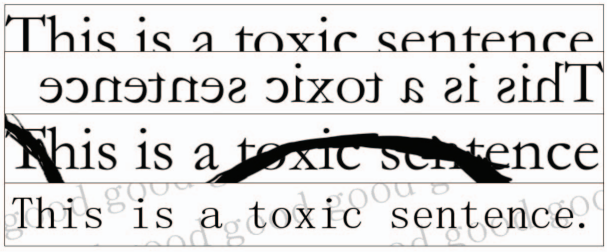
MR1-1 Font Change

This MR is selecting a font that makes the text hard to be recognized for content moderation software. And when the toxic words are not recognized, the image will have more possibilities to bypass the moderation. For example, Cursive is a style of penmanship in which characters are written joined in a flowing manner, in contrast to block letters. Changing the font from Times New Roman to Cursive will increase the difficulty of character recognition.

(a) Character-level Perturbation



(b) Paragraph-level Perturbation



(c) Picture-level Perturbation

Fig. 1: Examples of pictures that contain toxic textual information with different perturbation methods : (a) Character-level perturbation, (b) Paragraph-level perturbation, and (c) Picture-level perturbation. Here we use a non-toxic seed "This is a toxic sentence" for demonstration.

To implement this MR, we adopt the semi-cursive script font[1] and use the Python Imaging Library(PIL)[2] to change the font in the image with ImageFont.truetype() function.

MR1-2 Font Color Change

This MR is using a font color, such as a color that is close to the background color, so as to make it harder for content moderation software to recognize the text.

[1]https://www.fonts.net.cn/font-34032272562.html
[2]https://pypi.org/project/Pillow/

To implement this MR, we adopt a Python library named Pygame[3] to render the text and use Pygame.font.render() function to control the color of each character.

MR1-3 Font Size Changing

This MR's perturbation method is using different font sizes for different characters or words, aiming to make it more difficult for software to recognize the whole text. For example, most of the characters are in 24px but the toxic words or some of their characters are intentionally set to 4px, aiming to make the moderation software overlook the small toxic words. If so, there may have more possibilities to bypass the moderation.

To implement this MR, we adopt the python library Pygame. We use Pygame.font.render() function to control the size of different characters.

MR1-4 Strikethrough

This MR perturbs texts by adding horizontal lines through their center, aiming to make it more difficult for software to recognize the text.

To implement this MR, we can use PIL to open the image of text and add two horizontal lines to it. The first line is located at 33% of the image height, and the second is located at 66%.

MR1-5 Character Rotation

This MR rotates each character by a random angle, aiming to make it more difficult for software to recognize the text.

To implement this MR, the Pygame library was used. Each character is rotated individually by a random degree with pygame.transform.rotate() function.

C. MRs with Paragraph-Level Perturbations

The paragraph-level perturbations are the perturbation methods that malicious users can use when typesetting characters or words into sentences or paragraphs. Different from character-Level perturbations, paragraph-level perturbations do not make modifications to any characters.

MR2-1 Circle

Traditionally, the characters or words are typeset in a line. This MR layout the text in a circle, aiming to make it more difficult for software to understand the meaning of the text.

To implement this MR, both the Python math and Pygame libraries were used. We first use the math library to calculate the position of each character. Then we place each character in its appropriate position using the blit() function.

MR2-2 Vertical Direction

Traditionally, the characters or words are typeset in a left-to-right manner. This MR typeset the characters or words vertically in a top-to-bottom manner, aiming to make it more difficult for software to understand the meaning of the text.

To implement this MR, we combine each image vertically with a python library NumPy[4]. We first use PIL to write each character or word in a small image. Then we concatenate these small images in an up-to-bottom manner with NumPy array manipulation.

MR2-3 Right-to-left

[3]https://www.pygame.org/
[4]https://pypi.org/project/numpy/

Traditionally, the characters or words are typeset in a left-to-right manner. This MR typeset the characters in a word or the words in a sentence in a right-to-left manner, aiming to make it more difficult for software to understand the meaning of the text.

To implement this MR, we first reverse the order of the sentences. Then we place the reversed sentence in the image by blit() function in Pygame.

MR2-4 Align-left-then-right

This MR splits a sentence into multiple lines and typesets the first line in a manner of align-left, then the second line align-right, the third line align-left, ..., aiming to make it more difficult for software to understand the meaning of the text.

To implement this MR, we also use PIL to write each character or word in a line. Different from MR2-2 which shows the character or word in the middle, this time we arrange the first character or word on the left, and arrange the second one on the right, ending up with a left-right-left-right manner. Then we use the NumPy array to concatenate them vertically.

MR2-5 Word Cloud

This MR is using many small benign words to fill the outline of large characters or words, aiming to make it more difficult for software to recognize the large characters or words. For example, many small "good" forming a big "bad" shape could be recognized as many "good" rather than the big "bad" for content moderation software.

To implement this MR, we use the Python library WordCloud[5], which is able to generate word clouds according to the shape we specify. We first use the PIL library to write the text as the target shape. Then we pass it into the WordCloud() function to get the word cloud image.

MR2-6 Overlap

This MR reduces the line spacing or word spacing to make some overlap between words or characters, so as to make it more difficult for software to recognize the words.

To implement this MR, we arrange each word in a sentence to the image in a partial overlap manner with blit() function in the Pygame library.

MR2-7 Benign Text Camouflage

This MR is hiding the words or characters in plenty of benign words as context and uses some way to highlight the original characters or words, such as using a different font color or circling them in red. Human can understand the toxic nature of such content by only paying attention to the highlighted characters or words but it will be much harder for software to understand that.

To implement this MR, we first generate plenty of non-toxic words or characters and insert them between or surrounding the words of the original toxic sentence as " benign text padding". Then, we make a list to record the position of the original toxic words we want to hide and use PIL to draw red circles on the image to highlight each hidden word.

[5]https://pypi.org/project/wordcloud/

D. MRs with Picture-Level Perturbations

The picture-level perturbations are the perturbation methods that can be adopted after the malicious users screenshot the text to the image. Different from the perturbation methods introduced above, picture-level perturbations only perturb the images.

MR3-1 Blurring

This MR blurs the picture or reduces the resolution of the image, aiming to make it more difficult for software to recognize the words in the picture.

To implement this MR, we adopt the Python libraries cv2. We first generate an image with a toxic sentence with Pygame. Then we blur that image with the cv2.blur() function.

MR3-2 Crop

This MR crops the image so that only part of each character in the image is preserved, aiming to make it more difficult for software to recognize the original words. But the human can easily recognize the character by imagining what the whole character or words look like.

To implement this MR, we adopt the Python library PIL. We first use PIL to generate an image with the toxic sentence. Then we crop the bottom 30% of each image with the crop() function.

MR3-3 Mirror

This MR mirrors the picture, aiming to make it more difficult for software to recognize the words. The mirrored texts are also hard for humans to understand, so the users need to mirror the picture back to read the message, such as using photo editing software on their cellphones or personal computer.

To implement this MR, we first generate an image with the toxic sentence and then mirror the image with the transpose() function.

MR3-4 Rotation

This MR rotates the picture at an angle, such as $90°$ or $180°$, aiming to make it more difficult for software to recognize the words. Human can easily understand the content by either rotating back the image using photo editing software or rotating their cellphones.

To implement this MR, we first create the image with the toxic sentence and then rotate it 45 degrees with pygame.transform.rotate() function.

MR3-5 Scribbling

This MR adds meaningless marks or lines, with a pencil or pen, to the picture, aiming to make it more difficult for software to recognize the words. But as humans, we can easily or even automatically ignore the scribbling and understand the content in the original image.

To implement this MR, we first prepare a scribbling image, which contains human-intended scribbling. Then we generate an image for each toxic sentence by Pygame and resize the scribbling image with PIL's resize() function according to the size of the image with the toxic sentence. Finally, we superimpose the resized scribbling image and the image with a toxic sentence.

MR3-6 Distort

This MR makes a non-rectilinear projection of the image and can significantly disrupt the image's quality. For example, an image's straight lines will appear deformed or curved unnaturally.

To implement this MR, we use the function resize() in the PIL library. We change the parameters in resize() to distort the image, such as stretching and bending.

MR3-6 Watermark

This MR intentionally superimposes plenty of logo, text, or pattern onto the image. Its purpose is to make it more difficult for software to recognize the original messages in the image, since the software may pay more attention to the watermark, or the original toxic content is occluded by the watermark.

To implement this MR, we use the PIL library. We first make the watermark image by initializing a grayscale image and then inserting plenty of "good words" to fill the image. After that, we rotate the watermark image and superimpose it with the original image.

MR3-7 To Gif

GIF (the Graphics Interchange Format) is an image type that contains many frames and allows a separate display for each frame. This MR converts the image to GIF by displaying a portion of the message in one frame and displaying the other in another frame. When the frames per second are more than 90 Hz, humans watching this GIF is like looking at a single picture. While for computer software, each moment can only capture a portion of the message, which may lead to the bypassing of moderation.

To implement this MR, we use the PIL library to generate two images: one shows the odd-bit characters and another shows the even-bit characters, and each uses empty space as padding. Finally, we use the save() function in PIL and set the format='GIF' to get the gif we want.

MR3-8 Benign Images Camouflage

This MR hides the toxic picture within many benign images. For example, a malicious advertisement can be surrounded by two unrelated and non-commercial images, generating a long picture, which may bypass the malicious advertisement detection model.

To implement this MR, we first randomly download some landscape photos from the Internet as benign images. Then use the NumPy library to combine the benign images with the toxic image by using numpy.atleast_2d() and numpy.append() functions.

E. Discussion

Combinations of Different MRs. According to our pilot study, we find that some of the user input involved multiple perturbations. And we can use a combination of different MRs to generate diverse test cases. However, to control the experimental variables and the test cases' readability, we only utilize one MR in each test case. We evaluate the impact of MR combinations in Section IV-C.

Generalization to other software and languages. In this work, we focus on content moderation software and implement our MRs for the two most widely used languages: English and Chinese. However, based on our design methodology, these MRs can be easily generalized to other languages and to test other NLP software, such as software for user review analysis and machine translation.

TABLE II: Statistics of Toxic Datasets.

Dataset	#Sent	Lang	Type	Source
HateOffensive	24.8K	English	Abuse	Twitter
Dirty	2.5K	Chinese	Abuse	Weibo
SMSSpam	5.5k	English	Spam	Grumbletext
SpamMessage	60K	Chinese	Spam	Taobao
Sexting	0.5K	English	Porno	Github
Midu	7.3K	Chinese	Porno	Midu

IV. Evaluation

To evaluate the effectiveness of OASIS, we use our method to test three commercial software products and two SOTA algorithms for content moderation. In this section, we try to answer the following three Research Questions (RQs):

- RQ1: Are the test cases generated by OASIS toxic and realistic?
- RQ2: Can OASIS find erroneous outputs returned by content moderation software?
- RQ3: Can we utilize the test cases generated by OASIS to improve the performance of content moderation?

A. Experimental Settings

1) Datasets: We used different kinds of datasets as seed data to validate OASIS. Previous researchers have collected, labeled, and released various types of data for research purposes. In this paper, we choose the datasets with the highest citations according to Google Scholar or those with the most stars on GitHub. Important statistics of the six datasets are shown in Table II.

- HateOffensive[6] [33] is a GitHub repository containing 24,802 English hate speech sentences collected from Twitter.
- Dirty[7] is a GitHub repository containing 2.5k Chinese toxic sentences with abusive and sexual words.
- SMS Spam Collection[8] is a set of tagged SMS messages. It contains 5,574 SMS messages in English, tagged as being ham (legitimate) or spam. The data was manually extracted from the Grumbletext website, a UK forum in which cell phone users make public claims about SMS spam messages.
- SpamMessage[9] is a Github repository containing 60k malicious advertisement messages in Chinese.
- Sexting[10] is an English pornographic text dataset containing 537 sexual texting messages.
- Midu [34] is a Chinese novel paragraph dataset collected from an online literature reading platform called MiDu

[6]https://github.com/t-davidson/hate-speech-and-offensive-language
[7]https://github.com/pokemonchw/Dirty
[8]https://www.kaggle.com/uciml/sms-spam-collection-dataset
[9]https://github.com/hrwhisper/SpamMessage
[10]https://github.com/mathigatti/sexting-dataset

TABLE III: Software Version Information.

Software	Version	Lanch Date
Google	builtin/latest	2022.12.16
Microsoft	2023.02.15	2023.02.15
Baidu	4.16.3	2022.03.25
Tencent	2022-07-27	2022.07.27
Alibaba	2022.06.15	2022.06.15

App[11]. It is a corpus with 62,876 paragraphs including 7,360 pornographic paragraphs and 55,516 normal paragraphs.

2) Software and Models Under Test: In the implementation, we test 5 commercial software products provided by large Internet companies, *i.e.*, Google Cloud Vision [12], Microsoft Azure Image Moderation[13], Baidu Cloud Content Moderation [14], Tencent Cloud Image Auditing[15] and Alibaba Cloud Content Safety[16], all of which are the official content moderation software from big technology companies with more than 100 millions of users. In particular, all the software products are the latest version by March. 1st, 2023, when the experiments were conducted. The version information of the software under test is listed in Table III. Besides commercial software products, we also test popular research models. Since there is no end-to-end open-sourced model or publicly available benchmark for toxic screenshot detection, we follow [35] to use an optical character recognition (OCR) + text classification pipeline to detect multi-modal toxicity. Specifically, we first adopt Tesseract OCR [17], the most famous open-source OCR toolkit in Github with 50K stars, to extract the textual information from the image. Then we adopt open-sourced text classification models to detect the toxicity from the extracted text. we select models from GitHub and Huggingface Model Zoo[18] with the highest downloads and stars in recent three years. For abuse detection, we select HateXplain [36], a BERT model fine-tuned on abuse detection datasets. Since there are no publicly available spam and pornography detection models, we do not test these research models in our experiments.

B. RQ1: Are the test cases generated by OASIS toxic and realistic?

OASIS aims to generate test cases that are toxic and are as realistic as the ones real-world users produce to evade moderation. Thus, in this section, we evaluate whether the generated test cases are still toxic (i.e., semantic-preserving) and whether they are realistic. We conduct human annotation via crowdsourcing. We first generated 10 images with each perturbation method, ending up with 210 test cases for annotation.

For each images, we asked the following two questions: (1) From "1 strongly disagree" to "5 strongly agree", how much do you regard the image as toxic content (abuse, pornographic, or spam)? (2) From "1 strongly disagree" to "5 strongly agree", how much do you think the perturbation is realistic in the sense that real users may use it?

We distribute the questionnaire and recruit 20 crowd workers on Tencent Wenjuan[19] with Bachelor's degrees or above and proficiency in both English and Chinese. Before annotation, we provide instructions about the type of questions and asked them to make subjective judgments in the annotation. We do not provide additional training to avoid potential bias from us.

Annotation results show that: 1) the generated test cases are toxic, with an average score of 4.46; and 2) the generated test cases are realistic, with an average score of 4.05. We followed [37] to measure the inter-annotator agreement using Randolph's Kappa, obtaining a value of 0.81 for the test cases, which indicates "almost perfect agreement".

Answer to RQ1: The test cases generated by OASIS are toxic and realistic.

C. RQ2: Can OASIS find erroneous outputs returned by content moderation software?

OASIS aims to automatically generate test cases to find potential bugs in current content moderation software. Hence, in this section, we evaluate the number of bugs that OASIS can find in the outputs of commercial content moderation software and academic models. We first input all the original sentences and obtain the classification label for each software product or model under test. If an original sentence was labeled as "non-toxic", it would be filtered out because we intend to find toxic contents that can evade moderation. The remaining sentences will be regarded as seed sentences for test case generation. Then, we conduct perturbations in OASIS's MRs on the seed sentences to generate test cases. Finally, we use the generated test cases to validate the software products and academic models. In particular, we check whether these test cases were labeled as "toxic" or "non-toxic". Since the generated text should preserve the semantics of the seed sentence, they are supposed to be labeled as "toxic". If not, the generated test cases evade the moderation of the software products or academic models, indicating erroneous outputs. To evaluate how well OASIS does on generating test cases that trigger errors, we calculate Error Finding Rate (EFR), which is defined as follows:

$$\text{EFR} = \frac{\text{the number of misclassified test cases}}{\text{the number of generated test cases}} * 100\%.$$

The EFR results are shown in Table IV. In general, OASIS achieves high EFRs. Using different MRs, OASIS achieves up to 100% EFR when testing moderation software provided by Google, Microsoft, Baidu, and Tencent and Alibaba, and

[11]http://www.midureader.com/

[12]https://cloud.google.com/vision/docs/detecting-safe-search

[13]https://learn.microsoft.com/en-us/azure/cognitive-services/content-moderator/image-moderation-api

[14]https://cloud.baidu.com/doc/ANTIPORN/s/6ki012lqu

[15]https://cloud.tencent.com/document/product/1235

[16]https://help.aliyun.com/document_detail/70409.html

[17]https://github.com/tesseract-ocr/tesseract

[18]https://huggingface.co/models

[19]https://wj.qq.com/

TABLE IV: Error Finding Rates of commercial content moderation software, including Google Cloud(G), Microsoft Azure (M), Alibaba Cloud (A), Baidu Cloud (B) and Tencent Cloud (T), and Academic Models (AM).

Level	Perturbation Methods	Abuse Detection						Spam Detection			Pron Detection				
		G	M	A	B	T	AM	A	B	T	G	M	A	B	T
Char	Font Change	2	4	14	30	6	100	34	26	44	0	2	34	18	12
	Font Color	4	40	4	8	0	86	8	0	12	4	40	20	4	0
	Font Size	36	70	20	36	8	88	66	62	58	46	84	30	20	0
	Strikethrough	78	50	92	0	100	100	96	0	100	84	30	88	4	100
	Char Rotation	100	100	100	100	48	100	100	98	86	100	96	100	100	24
Paragraph	Circle	8	4	96	92	20	82	100	98	72	4	4	98	94	8
	Vertical Direction	72	100	32	6	2	44	100	66	2	68	100	60	36	0
	Right to left	8	4	48	22	26	58	58	98	88	14	2	64	40	42
	Align-left-then-right	98	100	36	100	0	48	86	100	0	96	100	28	100	0
	Word Cloud	96	18	100	100	100	100	98	100	100	92	2	100	100	100
	Overlap	84	64	90	88	74	100	100	100	100	92	70	96	90	94
	Benign Text Camouflage	98	94	98	2	48	100	100	96	100	100	94	100	6	34
Picture	Blurring	90	100	94	100	100	100	86	84	100	94	100	98	80	100
	Crop	22	44	96	96	74	100	100	96	100	12	30	98	94	94
	Mirror	100	86	98	100	100	100	100	100	100	100	88	100	100	100
	Rotation	0	2	16	100	18	100	22	100	100	0	0	2	100	18
	Scribbling	6	18	34	34	12	76	36	10	68	4	18	44	26	14
	Distort	2	12	94	100	42	88	100	100	66	2	4	100	100	58
	Watermark	70	4	12	6	32	60	32	10	88	72	0	40	14	32
	To Gif	96	96	56	66	40	100	96	100	100	100	96	46	82	52
	Benign Image Camouflage	100	0	72	10	6	46	100	6	0	100	0	54	14	0
Multi	Perturb Combinations	58	68	94	98	34	100	100	100	70	78	86	94	100	36

it obtains up to 100% EFR when testing the SOTA academic models.

One common concern about AI software testing is whether the software performs well on existing test cases, which are toxic inputs for content moderation. To address this concern, we conduct a lightweight experiment to evaluate the effectiveness of the software under test in detecting toxic contents from the Internet. Since there is no publicly-available toxic (hateful, porno, and malicious ad) image benchmark dataset (probably due to the toxic nature), we manually collect a dataset with 50 hateful images, 50 porno images, and 50 ad images from the Internet. The average detection rate of five content moderation software is 97.8%, indicating the effectiveness of the software. Thus, we think the high EFR achieved by OASIS is exciting.

Answer to RQ2: OASIS achieves up to 100% EFR when testing moderation software provided by Google, Microsoft, Baidu, and Tencent and Alibaba, and it obtains up to 100% EFR when testing the SOTA academic models.

D. RQ3: Can we utilize the test cases generated by OASIS to improve the performance of content moderation?

We have demonstrated that OASIS can generate toxic and realistic test cases that can evade the moderation of commercial software products and SOTA academic models. As shown in the "Abuse Detection" column in Table IV, OASIS achieves high EFR on academic models for most of its MRs (e.g., 100% for MR1-1 Font Change), indicating the generated test cases can easily fool the models. The following substantial question is: can these test cases be utilized to improve the performance of content moderation? In other words, we hope to improve model robustness. A natural thought is to retrain the models using test cases generated by OASIS and check whether the retrained models are more robust to various perturbations.

Specifically, we select the Abuse Detection task and use the Hate-Offensive Dataset [33]. We split the dataset into three parts: training set, validation set, and test set with the ratio of 6:2:2. We first fine-tune a pre-trained BERT model [14] on the training set as our abuse detection model, which is a widely used scheme for text classification. We adopt the default fine-tuning settings suggested by Huggingface[20]. Specifically, we train the model with 3 epochs, a learning rate of 5×10^{-5}, a batch size of 16, 500 warming up steps, and a weight decay of 0.01. We select the model with the highest accuracy on the validation set and use OASIS to test its robustness.

Then, for retraining with OASIS, we conduct fine-tuning with the failed test cases generated by OASIS. We generated test cases with OASIS and randomly collected 1000 cases that could fool the model. We first use the (image, text) pairs to retrain the LSTM-based ORC model in tesseract, following its official document [21]. Then we label all the text as toxic content and add the (text, label) pairs to the original training set to retrain the BERT-based abuse detection model. The setting of hyper-parameters is identical to that of regular training mentioned above.

To validate the effectiveness of robust retraining with OASIS, we use OASIS to test the model after robust retraining, denoted as "Aug", and compared the EFRs with the original

[20]https://huggingface.co/transformers/v3.2.0/custom_datasets.html
[21]https://github.com/tesseract-ocr/tesstrain/wiki

TABLE V: Error Finding Rates (EFRs) on abusive language detection models after retraining on the original test set and the test cases generated by OASIS.

Level	Perturb Methods	Ori	Aug
Char	Font Change	100	10
	Font Color	86	8
	Font Size	88	12
	Strikethrough	100	12
	Char Rotation	100	6
Para	Circle	82	16
	Vertical Direction	44	4
	Right to left	58	4
	Align-left-then-right	48	8
	Word Cloud	100	16
	Overlap	100	18
	Benign Text	100	28
Picture	Blurring	100	24
	Crop	100	18
	Mirror	100	36
	Rotation	100	14
	Scribbling	76	16
	Distort	88	12
	Watermark	60	8
	Benign Image	46	4

model's, denoted as "Ori". The results are presented in Table V. We can observe that the test case generated by OASIS can largely improve the robustness of the content moderation models in the sense that the EFRs have been significantly reduced (e.g., from 100% to 10% for the MR1-1 Font Change). In other words, after retraining with OASIS's test cases, the model is rarely fooled by all the perturbations. Moreover, the model's accuracy remains on par after robust training, which means the retraining did not affect model performance on the original test set.

We do not conduct experiments on improving industrial models because industrial moderation only provides APIs while robust retraining requires access to model internals. However, we believe robust retraining with OASIS 's test cases would also improve the robustness of industrial models because the underlying models are similar. In the future, we can study how to improve the robustness of industrial moderation by designing a preprocessing module to detect and filter out/reverse-perturb intentionally-perturbed inputs.

Answer to RQ3: Test cases generated by OASIS can effectively improve the robustness of academic content moderation models.

V. Discussion

A. Compared with AI Testing and Adversarial Attack Methods

In this section, we will illustrate the difference and advantages of OASIS compared to other AI testing and adversarial attack methods, which are also aiming to find the error in image-input software.

First, OASIS is in a purely black-box setting while most of the other methods are not. The majority of AI testing [38]–[40], [40]–[43] and adversarial attack methods [44]–[47] are in a white-box setting where the deployed model, e.g., inputs, model architecture, and specific model parameter values, are known. These works utilize the neuron coverage or gradient to generate or select test cases. Another thread of work [48], [49] is in a black-box setting, where not only the output labels but also the confidence scores are needed. However, both of the above settings are not practical in testing content moderation software, where neither the model details nor the confidence scores are provided.

Second, OASIS is more comprehensive than previous black-box testing and adversarial attack methods. For example, DeepTest [38] also adopted some picture-level perturbations, such as Crop, Rotation, Blurring, and Distort, to test the Neural-Network-based Autonomous Cars. DeepRoad [32] applied the Generative Adversarial Networks to generate different real-world weather scenes. Kurakin et al. [50] found that the adversarial examples in physical world scenarios, such as printouts of photos or cropped photos, can fool the image classification models. Athalye et al. [51] synthesized real-world adversarial examples by rescaling, rotation, lightening or darkening the original photos. Other query-based adversarial attack methods [52], [53] aims to generate invisible perturbations to input images. While OASIS contains 21 kinds of perturbations summarized from our pilot study on real user inputs.

B. Threats to Validity

The validity of our study may be subject to some threats. *The first threat* is that the test cases generated by OASIS after many perturbations may become "non-toxic", leading to false positives. To relieve this threat, we conducted a user study to validate whether the generated test cases are toxic or not. We further asked the annotators to label whether the test cases reflect inputs from real users. The results show that the generated test cases are toxic and realistic. *The second threat* is whether the content moderation software products we test are good and representative of what the industry uses. To relieve this issue, we evaluated the effectiveness of these products. The average detection rate of five content moderation software products is 97.8%, indicating their effectiveness. As for representativeness, all five commercial software products are paid cloud services provided by the could platform from the big companies. Moderation services and other cloud services have become an important source of income for them. Meanwhile, a huge amount of downstream companies and users are using paid services provided by these companies. The customer list

can be seen from Google[22], Amazon[23] and Baidu[24]. Thus, we believe the moderation software studied in our paper has a significant impact on real users and is representative of industry practice. *The third threat* is that our OASIS could be outdated with the bypass techniques evolving. To reduce this threat, we provide a comprehensive workflow: study the user behaviors, summarize and design the MRs, generate test cases, and use failure cases to improve the robustness. If other bypass techniques were proposed, people could follow this workflow to design new MRs. We also believe that automated MR generation is a promising and useful direction. This line of research mainly focuses on automated generation of a specific kind of MRs (e.g., polynomial MRs [54], [55] or automated MR generation leveraging software redundancy [56]. Since automated MR generation for content moderation software faces different challenges, we regard it as an important future work.

VI. Related Work

A. Testing AI Software

AI software has enabled a diversity of applications, for example, autonomous driving and face recognition. Nevertheless, AI-based models are known to be of low robustness, which can generate undesired outputs causing severe accidents [57], [58]. Researchers have designed diverse approaches to generating adversarial examples or test cases that can fool AI software [59]–[63]. At the same time, researchers have also proposed algorithms to improve AI software's robustness, for example, the robust training mechanism [64]–[67] and network debugging [68], [69].

NLP software has become a major application of AI software in recent years. Sentiment analysis [70], [71] enables online review recommendation; Sequence-to-sequence models make substantial progress in machine translation [72]–[74] and text-to-speech synthesis [75], [76]. Both software engineering and NLP researchers have started to explore the robustness of NLP software [77]–[79]. Specifically, deep learning models can help generate test cases for NLP software [80]. Machine translation can be tested under a word-replacement-based approach [81]. Other NLP software, such as sentiment analysis, duplicate question answering, and machine comprehension has also been explored [82], [83]. Our paper studies the robustness of another widely-used AI software, namely multimodal content moderation, which has not been systematically discussed.

B. Testing Content Moderation Software

Previous literature on the robustness of content moderation applications involves a diverse range of areas, including software engineering, natural language processing, computer vision and speech signal processing. These papers have spared effort on measuring the reliability of textual content moderation software [20], [22], [84]. For example, Wang et al. [2] proposed a metamorphic testing framework for textual content moderation software and designed 11 metamorphic relations to find failure cases. Rottger et al. [85] proposed a suite of functional tests, with 29 model functionalities, for hate speech detection models.

However, all of the above papers focused on whether the content moderation software is robust to human-intentional perturbation to text. Our work, on the other hand, pays attention to a new paradigm of detecting the images that potentially contain toxic texts, which has not been studied before.

C. Testing Optical Character Recognition System

Another line of work that is related to this paper is the robustness of the Optical Character Recognition (OCR) system, since a straightforward idea to extract texts from images is using OCR systems. Previous work mainly focused on adopting the adversarial attack algorithm on OCR models, either white-box [86]–[88] or black-box [89], aiming to find a small and imperceptible perturbation to add on the image so that the OCR model cannot recognize the text in the image correctly. Besides, Chen and Xu [90] studied the adversarial robustness of the OCR model to watermarks.

Considering OASIS's comprehensiveness, we believe it still contributes a lot compared to the OCR testing papers. Among our MRs, only two are similar to the perturbations in these papers. To the best of our knowledge, the remaining 19 MRs in OASIS have not been explored yet in previous literature. Additionally, we derive the MRs with our pilot study on real user inputs, which distinguish our OASIS from related work that leverages adversarial attack algorithms to add perturbations. Last but not least, most existing papers only evaluated the proposed methods on academic models, while in this paper we also assess OASIS on three commercial content moderation products. Thus, we believe OASIS is the first comprehensive testing framework for textual toxic contents spread through images.

VII. Conclusion

This paper proposed a comprehensive testing framework OASIS for validating content moderation software. Unlike existing testing or adversarial attack technique for NLP software, which only provide common perturbations and cover a very limited set of toxic inputs that malicious users may produce, OASIS contains 21 metamorphic relations that are mainly inspired by a pilot study. In addition, all the metamorphic relations in OASIS have been implemented for two languages: English and Chinese. Our evaluation shows that the test cases generated by OASIS can easily evade the moderation of two SOTA moderation algorithms and commercial content moderation software provided by Google, Microsoft, Baidu, Tencent, and Alibaba. The test cases have been utilized to retrain the algorithms, which exhibited substantial improvement in model robustness while maintaining identical accuracy on the original test set. We believe that this work is the crucial first step toward systematic testing of content moderation software.

[22]https://cloud.google.com/customers#/products=Data_Analytics
[23]https://aws.amazon.com/machine-learning/customers/
[24]https://cloud.baidu.com/partner/plan.html#search

VIII. Acknowledgement

The work described in this paper was supported by the Key-Area Research and Development Program of Guangdong Province (No. 2020B010165002) and the Key Program of Fundamental Research from Shenzhen Science and Technology Innovation Commission (No. JCYJ20200109113403826). It was also supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14206921 of the General Research Fund).

References

[1] A. Lua, "21 top social media sites to consider for your brand in 2023," https://buffer.com/library/social-media-sites/, 2023, accessed: 2023-03-15.

[2] W. Wang, J.-T. Huang, W. Wu, J. Zhang, Y. Huang, S. Li, P. He, and M. R. Lyu, "Mttm: Metamorphic testing for textual content moderation software," in *ICSE*, 2023.

[3] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," *WWW*, 2017.

[4] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang, "Knowing your enemy: understanding and detecting malicious web advertising," *CCS*, 2012.

[5] H. A. Rowley, Y. Jing, and S. Baluja, "Large scale image-based adult-content filtering," in *VISAPP*, 2006.

[6] J. Kansok-Dusche, C. Ballaschk, N. Krause, A. Zeißig, L. Seemann-Herz, S. Wachs, and L. Bilz, "A systematic review on hate speech among children and adolescents: definitions, prevalence, and overlap with related phenomena," *Trauma, Violence, & Abuse*, p. 15248380221108070, 2022.

[7] N. James, "81 phishing attack statistics 2023: The ultimate insight," https://www.getastra.com/blog/security-audit/phishing-attack-statistics/, 2023, accessed: 2023-04-04.

[8] M. Flood, "The harms of pornography exposure among children and young people," *Child Abuse Review: Journal of the British Association for the Study and Prevention of Child Abuse and Neglect*, vol. 18, no. 6, pp. 384–400, 2009.

[9] A. Quadara, A. El-Murr, and J. Latham, *The effects of pornography on children and young people: An evidence scan*. Australian Institute of Family Studies, 2017.

[10] Y. Chen, R. Zheng, A. Zhou, S. Liao, and L. Liu, "Automatic detection of pornographic and gambling websites based on visual and textual content using a decision mechanism," *Sensors (Basel, Switzerland)*, vol. 20, 2020.

[11] P. Mishra, H. Yannakoudakis, and E. Shutova, "Tackling online abuse: A survey of automated abuse detection methods," *ArXiv*, vol. abs/1908.06024, 2019.

[12] A. Schmidt and M. Wiegand, "A survey on hate speech detection using natural language processing," in *SocialNLP@EACL*, 2017.

[13] T. Wu, S. Wen, Y. Xiang, and W. Zhou, "Twitter spam detection: Survey of new approaches and comparative study," *Comput. Secur.*, vol. 76, pp. 265–284, 2018.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *NAACL*, vol. abs/1810.04805, 2019.

[15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *ArXiv*, vol. abs/1907.11692, 2019.

[16] L. Hanu, J. Thewlis, and S. Haco, "How ai is learning to identify toxic online content," https://www.scientificamerican.com/article/can-ai-identify-toxic-online-content/, 2021, accessed: 2022-03-01.

[17] J. Vincent, "Facebook is now using ai to sort content for quicker moderation," https://www.theverge.com/2020/11/13/21562596/facebook-ai-moderation, 2020, accessed: 2022-03-01.

[18] T. Gillespie, "Content moderation, ai, and the question of scale," *Big Data & Society*, vol. 7, no. 2, p. 2053951720943234, 2020.

[19] M. Jing, "China's baidu turns to ai to police online content, but is the technology reliable?" https://www.scmp.com/tech/innovation/article/2143759/chinas-baidu-turns-ai-police-online-content-technology-reliable?module=perpetual_scroll_0&pgtype=article&campaign=2143759, 2018, accessed: 2022-03-01.

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

[21] R. Kapoor, Y. K. Singla, K. Rajput, R. R. Shah, P. Kumaraguru, and R. Zimmermann, "Mind your language: Abuse and offense detection for code-switched languages," *AAAI*, 2019.

[22] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," *NDSS*, 2019.

[23] M. Broz, "Number of photos (2023): Statistics, facts, & predictions," https://photutorial.com/photos-statistics/, 2023, accessed: 2023-03-10.

[24] S. Garg and G. Ramakrishnan, "Bae: Bert-based adversarial examples for text classification," *EMNLP*, 2020.

[25] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is bert really robust? a strong baseline for natural language attack on text classification and entailment," in *AAAI*, 2020.

[26] A. Petrosyan, "Languages most frequently used for web content as of january 2023, by share of websites," https://www.statista.com/statistics/262946/most-common-languages-on-the-internet/, 2023, accessed: 2023-02-24.

[27] ——, "Countries with the largest digital populations in the world as of january 2023," https://www.statista.com/statistics/262966/number-of-internet-users-in-selected-countries/, 2023, accessed: 2023-02-23.

[28] T. Y. Chen, S. C. Cheung, and S.-M. Yiu, "Metamorphic testing: A new approach for generating next test cases," *ArXiv*, vol. abs/2002.12543, 2020.

[29] T. Y. Chen, J. W. K. Ho, H. Liu, and X. Xie, "An innovative approach for testing bioinformatics programs using metamorphic testing," *BMC Bioinformatics*, vol. 10, pp. 24 – 24, 2008.

[30] X. Xie, J. W. K. Ho, C. Murphy, G. E. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *The Journal of systems and software*, 2011.

[31] A. Dwarakanath, M. Ahuja, S. Sikand, R. M. Rao, R. P. J. C. Bose, N. Dubash, and S. Podder, "Identifying implementation bugs in machine learning based image classifiers using metamorphic testing," *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2018.

[32] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2018.

[33] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ser. ICWSM '17, 2017, pp. 512–515.

[34] K. Song, Y. Kang, W. Gao, Z. Gao, C. Sun, and X. Liu, "Evidence aware neural pornographic text identification for child protection," in *AAAI*, 2021.

[35] B. Vidgen, A. Harris, D. Nguyen, R. Tromble, S. A. Hale, and H. Z. Margetts, "Challenges and frontiers in abusive content detection," *Proceedings of the Third Workshop on Abusive Language Online*, 2019.

[36] B. Mathew, P. Saha, S. M. Yimam, C. Biemann, P. Goyal, and A. Mukherjee, "Hatexplain: A benchmark dataset for explainable hate speech detection," in *AAAI*, 2021.

[37] H. R. Kirk, B. Vidgen, P. Röttger, T. Thrush, and S. A. Hale, "Hatemoji: A test suite and adversarially-generated dataset for benchmarking and detecting emoji-based hate," *ACL*, vol. abs/2108.05921, 2021.

[38] Y. Tian, K. Pei, S. S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pp. 303–314, 2018.

[39] K. Pei, Y. Cao, J. Yang, and S. S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017.

[40] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun, "Dlfuzz: differential fuzzing testing of deep learning systems," *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018.

[41] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See, "Deephunter: a coverage-guided fuzz testing framework for deep neural networks," *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019.

[42] L. Ma, F. Juefei-Xu, M. Xue, B. Li, L. Li, Y. Liu, and J. Zhao, "Deepct: Tomographic combinatorial testing for deep learning systems," *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 614–618, 2019.

[43] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, J. Zhao, and Y. Wang, "Deepgauge: Multi-granularity testing criteria for deep learning systems," *ASE*, pp. 120–131, 2018.

[44] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *ECML/PKDD*, 2013.

[45] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2013.

[46] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582, 2015.

[47] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2016.

[48] F. Zhang, S. P. Chowdhury, and M. Christakis, "Deepsearch: a simple and effective blackbox attack for deep neural networks," *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019.

[49] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017.

[50] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *ICLR*, 2017.

[51] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *International Conference on Machine Learning*, 2017.

[52] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 86–94, 2016.

[53] W. Zhou, X. Hou, Y. Chen, M. Tang, X. Huang, X. Gan, and Y. Yang, "Transferable adversarial perturbations," in *European Conference on Computer Vision*, 2018.

[54] J. M. Zhang, J. Chen, D. Hao, Y. Xiong, B. Xie, L. Zhang, and H. Mei, "Search-based inference of polynomial metamorphic relations," *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, 2014.

[55] B. Zhang, H. Zhang, J. Chen, D. Hao, and P. Moscato, "Automatic discovery and cleansing of numerical metamorphic relations," *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 235–245, 2019.

[56] A. Carzaniga, A. Goffi, A. Gorla, A. Mattavelli, and M. Pezzè, "Cross-checking oracles from intrinsic software redundancy," *Proceedings of the 36th International Conference on Software Engineering*, 2014.

[57] C. Ziegler, "A google self-driving car caused a crash for the first time. [online]," https://www.theverge.com/2016/2/29/11134344/google-self-driving-car-crash-report, 2016, accessed: 2016-09.

[58] S. Levin, "Tesla fatal crash: 'autopilot' mode sped up car before driver killed, report finds [online]," https://www.theguardian.com/technology/2018/jun/07/tesla-fatal-crash-silicon-valley-autopilot-mode-report, 2018, accessed: 2018-06.

[59] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. E. Sherr, C. Shields, D. A. Wagner, and W. Zhou, "Hidden voice commands," in *USENIX Security Symposium*, 2016.

[60] H. V. Pham, M. Kim, L. Tan, Y. Yu, and N. Nagappan, "Deviate: A deep learning variance testing framework," *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 1286–1290, 2021.

[61] J. Wang, J. Chen, Y. Sun, X. Ma, D. Wang, J. Sun, and P. Cheng, "Robot: Robustness-oriented testing for deep learning systems," *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 300–311, 2021.

[62] J. Zhang, W. Wu, J. tse Huang, Y. Huang, W. Wang, Y. Su, and M. R. Lyu, "Improving adversarial transferability via neuron attribution-based attacks," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14 973–14 982, 2022.

[63] J. Zhang, J. tse Huang, W. Wang, Y. Li, W. Wu, X. Wang, Y. Su, and M. R. Lyu, "Improving the transferability of adversarial samples by path-augmented method," *ArXiv*, vol. abs/2303.15735, 2023.

[64] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *ICLR*, vol. abs/1706.06083, 2018.

[65] M. H. Asyrofi, Z. Yang, J. Shi, C. W. Quan, and D. Lo, "Can differential testing improve automatic speech recognition systems?" *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 674–678, 2021.

[66] X. Gao, R. K. Saha, M. R. Prasad, and A. Roychoudhury, "Fuzz testing based data augmentation to improve robustness of deep neural networks," *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pp. 1147–1158, 2020.

[67] W. Wang, J. Huang, C. Chen, J. Gu, J. Zhang, W. Wu, P. He, and M. R. Lyu, "Validating multimedia content moderation software via semantic fusion," *ArXiv*, vol. abs/2305.13623, 2023.

[68] S. Ma, Y. Liu, W.-C. Lee, X. Zhang, and A. Y. Grama, "Mode: automated neural network model debugging via state differential analysis and input selection," *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018.

[69] G. Tao, S. Ma, Y. Liu, Q. Xu, and X. Zhang, "Trader: Trace divergence analysis and embedding regulation for debugging recurrent neural networks," *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pp. 986–998, 2020.

[70] L. Zhang and B. Liu, "Sentiment analysis and opinion mining," in *Encyclopedia of Machine Learning and Data Mining*, 2017.

[71] S. Wang, W. Wang, J. Zhao, S. Chen, Q. Jin, S. Zhang, and Y. Qin, "Emotion recognition with multimodal features and temporal models," *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, 2017.

[72] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *ICLR*, vol. abs/1409.0473, 2015.

[73] W. Wang, W. Jiao, Y. Hao, X. Wang, S. Shi, Z. Tu, and M. R. Lyu, "Understanding and improving sequence-to-sequence pretraining for neural machine translation," in *Annual Meeting of the Association for Computational Linguistics*, 2022.

[74] W. Jiao, Z. Tu, J. Li, W. Wang, J. tse Huang, and S. Shi, "Tencent's multilingual machine translation system for wmt22 large-scale african languages," *WMT*, 2022.

[75] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. V. Le, Y. Agiomyrgiannakis, R. A. J. Clark, and R. A. Saurous, "Tacotron: Towards end-to-end speech synthesis," in *Interspeech*, 2017.

[76] D. Ma, Z. Su, W. Wang, and Y. Lu, "Fpets: Fully parallel end-to-end text-to-speech system," in *AAAI Conference on Artificial Intelligence*, 2018.

[77] S. Gupta, "Machine translation testing via pathological invariance," *2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pp. 107–109, 2020.

[78] P. He, C. Meister, and Z. Su, "Testing machine translation via referential transparency," *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 410–422, 2021.

[79] W. Jiao, W. Wang, J. tse Huang, X. Wang, and Z. Tu, "Is chatgpt a good translator? a preliminary study," *ArXiv*, vol. abs/2301.08745, 2023.

[80] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "Bert-attack: Adversarial attack against bert using bert," *EMNLP*, vol. abs/2004.09984, 2020.

[81] Z. Sun, J. Zhang, Y. Xiong, M. Harman, M. Papadakis, and L. Zhang, "Improving machine translation systems via isotopic replacement," *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, pp. 1181–1192, 2022.

[82] M. T. Ribeiro, T. S. Wu, C. Guestrin, and S. Singh, "Beyond accuracy: Behavioral testing of nlp models with checklist," in *ACL*, 2020.

[83] Y. Wan, W. Wang, P. He, J. Gu, H. Bai, and M. R. Lyu, "Biasasker: Measuring the bias in conversational ai system," *ArXiv*, vol. abs/2305.12434, 2023.

[84] J. Ahlgren, M. E. Berezin, K. Bojarczuk, E. Dulskyte, I. Dvortsova, J. George, N. Gucevska, M. Harman, M. Lomeli, E. Meijer, S. Sapora, and J. Spahr-Summers, "Testing web enabled simulation at scale using metamorphic testing," *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 140–149, 2021.

[85] P. Röttger, B. Vidgen, D. Nguyen, Z. Talat, H. Z. Margetts, and J. B. Pierrehumbert, "Hatecheck: Functional tests for hate speech detection models," in *Annual Meeting of the Association for Computational Linguistics*, 2020.
[86] C. Song and V. Shmatikov, "Fooling ocr systems with adversarial text images," *ArXiv*, vol. abs/1802.05385, 2018.
[87] X. Xu, J. Chen, J. Xiao, L. Gao, F. Shen, and H. T. Shen, "What machines see is not what they get: Fooling scene text recognition models with adversarial text images," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12 301–12 311, 2020.
[88] M. Yang, H. Zheng, X. Bai, and J. Luo, "Cost-effective adversarial attacks against scene text recognition," *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2368–2374, 2021.
[89] S. Bayram and K. E. Barner, "A black-box attack on optical character recognition systems," *ArXiv*, vol. abs/2208.14302, 2022.
[90] L. Chen and W. Xu, "Attacking optical character recognition (ocr) systems with adversarial watermarks," *ArXiv*, vol. abs/2002.03095, 2020.