Adversarial Text Perturbation Generation and Analysis

Jesus Guerrero*, Gongbo Liang† and Izzat Alsmadi‡
Computational, Engineering and Mathematical Sciences Department, Texas A&M, San Antonio
Email: *jguer017@jaguar.tamu.edu, † gliang@tamusa.edu, ‡ialsmadi@tamusa.edu

***Abstract*—With the evolution of applications of text generations in social networks, the genuineness of such text is questioned. Machine learning language based models such as GPT now can generate responses to complex questions which can be hardly distinguished from human-generated alternatives. In this scope, we utilized text-mutation to evaluate different text-based mutation operators that can be easily created and their impact on the output generated text. Our goal is to evaluate how can machine learning models distinguish those mutated versions from original versions. We reported results of several text-based mutation operators. We evaluated only a few examples of mutation operators and our goal is to eventually create a much larger list of operators. Those mutation operators can be used to distinguish human from machine-generated text.**

2023 3rd Intelligent Cybersecurity Conference (ICSC) | 979-8-3503-8217-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSC60084.2023.10349981

I. Introduction

Currently, text generation is widely used in different applications such as poetry, [Yi et al.(2017)Yi, Li, and Sun], machine writing, [Zhang et al.(2016)Zhang, Yao, and Wan], humans to machine dialogue, [Serban et al.(2016)Serban, Sordoni, Bengio, Courville, and Pineau] language to language translation [Prakash et al.(2016)Prakash, Hasan, Lee, Datla, Qadir, Liu, and Farri], document summary [Genest and Lapalme(2011)], headline or abstract generation, [Schick and Schütze(2021)]. Those applications can be classified into different categories, such as short versus long text generation applications. Short text generation applications include examples such as predicting next word or statement, image caption generation, short language translation, and documents summarization, [Chen(2021)], [Zhang and Huang(2019)]. Long text generation applications include long text story completion, review generation, language translation, poetry generation, and question answering, [Guo et al.(2018)Guo, Lu, Cai, Zhang, Yu, and Wang], [Guan et al.(2021)Guan, Mao, Fan, Liu, Ding, and Huang], [Hua and Wang(2020)]. Large language models such as open AI chat GPT-1 [Radford et al.(2018)Radford, Narasimhan, Salimans, Sutskever, et al.], 2 [Radford et al.(2019)Radford, Wu, Child, Luan, Amodei, Sutskever, et al.] and 3 [Brown et al.(2020)Brown, Mann, Ryder, Subbiah, Kaplan, Dhariwal, and Neelakantan] can be used to masquerade humans in many of those short and long text generation applications.

Machine-based text generation is recently expanded beyond "useful" application to "malicious" ones. In this context, malicious generated text indicates that users or receivers of this text assume that this text is generated by humans while in reality it is not. This is unlike most of the previously mentioned applications for text generation. Social media platforms provide users with an unprecedented opportunity to create and share various forms of information, e.g. text, video, and images. The mechanisms of information flow have been seen to impact the spread of both true and false news through social networks, [South et al.(2022)South, Smart, Roughan, and Mitchell]. Several recent studies showed how much social trolls and bots can drive misinformation, [South et al.(2017)South, Smart, Roughan, and Mitchell], [Alsmadi and O'Brien(2020)]. Social-bots and trolls are examples of text generators in Generative Adversarial Networks (GAN) models, [Alsmadi et al.(2021)Alsmadi, Ahmad, Nazzal, Alam, Al-Fuqaha, Khreishah, and Algosaibi].

In one classification to adversarial machine learning (AML) attacks, those attacks can be divided based on at which machine learning stage the attack is occurring (e.g. (1) learning/training stage, versus (2) testing stage).

A. Poisoning attacks: AML attacks on the learning stage: Manipulating the training data

Attackers can deliberately influence the training dataset to manipulate the results of a predictive model. A poisoning attack adds poisoned instances to the training set and introduces new errors into the model. If we consider one ML application, spam detection, filter of spam messages will be trained with adversary instances to incorrectly classify the spam messages as good messages leading to compromising of system's integrity. Alternatively, the Spam messages' classifier will be trained inappropriately to block the genuine messages thereby compromising system's availability, [Newsome et al.(2006)Newsome, Karp, and Song], [Perdisci et al.(2006)Perdisci, Dagon, Lee, Fogla, and Sharif], [Nelson et al.(2008)Nelson, Barreno, Chi, Joseph, Rubinstein, and Saini], [Rubinstein et al.(2009)Rubinstein, Nelson, Huang, Joseph, Lau, Rao, Taft, and Tygar], [Barreno et al.(2010)Barreno, Nelson, Joseph, and Tygar], [Biggio et al.(2012)Biggio, Nelson, and Laskov], [Newell et al.(2014)Newell, Potharaju, Xiang, and Nita-Rotaru].

B. Evasion attacks: AML attacks on the testing stage: Manipulating the testing data

In this attack, attackers try to evade the detection system by manipulating the testing data, resulting in a wrong model classification. The core of adversarial evasion attack is that

when an attacker can fool a classifier to think that a malicious input is actually benign, they can render a machine learning-based malware detector or intrusion detector ineffective, [Russu et al.(2016)Russu, Demontis, Biggio, Fumera, and Roli], [Zhang et al.(2020)Zhang, Wang, Liu, and Wang], [Sayghe et al.(2020)Sayghe, Zhao, and Konstantinou], [Shu et al.(2022)Shu, Xia, Williams, and Menzies]. [Chernikova and Oprea(2022)]

Our mutation-based approach in this paper can fall under the first category of the last classification (i.e. poisoning attacks). A mutation instance is a data instance or object that is created from an original genuine instance with a slight change. Such changes on machine learning stage can reduce accuracy of classification algorithms through generating fake instances that are very close to genuine instances. Mutation changes are typically introduced to simulate actual real-world faults or mistakes. There are several scenarios to implement our mutation-based AML:

- Two class labels (Human/Mutation text): Human generated text versus mutation generated text. Such experiments will test classifiers sensitivity to changes injected by mutations in comparison with original genuine text.
- Two class labels (Human/Adversarial, mutation instances are added to adversarial instances
- Two class labels (Human/Adversarial, mutation instances are added to Human instances
- Three class labels (Human/Mutation/Adversarial instances).

C. Mutation Testing in the Language Domain

The proposed mutation-based text generation is inspired by the advances in mutation analysis in software testing and the idea of "broiling frog syndrome" The well-controlled and close-ended environment makes generating precise text output possible. By using the proposed tool, researchers could test a language model by changing the input slightly and step-by-step.

Charm [Bravo-Santos et al.(2020)Bravo-Santos, Guerra, and Lara] is a chat-bot testing tool closely related to our mutation-based text generator that extended Botium [Jin et al.(2020)Jin, Jin, Zhou, and Szolovits]—a popular framework for chat-bot testing—by integrating eight mutation operators. Though both Charm and ours use mutation operators, Charm is proposed as an extension to Botium and relies on Botium to work. In addition, Charm only works for chat-bot testing.

Unlike Charm, ours is a general-purpose language generation method, which is designed to work alone. Our method can be used to analyze any type of language models that accept a sequence of text as input. In addition, the users are not limited by the pre-defined mutation operators. Our proposed mutation-based text generator is a general framework. The users could easily design their own mutation operators for their specific tasks within our framework.

In this paper, we will propose and evaluate mutation operators within the first category and leave investigating other categories in future papers.

The rest of the paper is organized as follows. Section II provides a summary of related research. Our paper goals and approaches are introduced in III. Section IV covers the experiments we performed to evaluate our proposed mutation operators. We have then a separate section, section V to compare with close contributions. Finally, Section VI provides some concluding remarks as well as future extensions or directions.

II. Related Works

Machine learning NLP-based classifiers can be influenced by words misspelling and all forms of adversarial text perturbations.

Literature survey indicated an increasing trend in using pre-trained models in machine learning, [Kumar et al.(2020)Kumar, Choudhary, and Cho], [Mathew and Bindu(2020)]. Word/sentence embedding models and transformers are examples of those pre-trained models. Adversarial models may utilize same or similar pre-trained models as well. In another trend related to text generation models, literature showed effort to develop universal text perturbations to be used in both black and white-box attack settings, [Alsmadi et al.(2022)Alsmadi, Aljaafari, Nazzal, AlHamed, Sawalmeh, and Khreishah]. The literature on adversarial text analysis is quite rich (e.g. see [Bravo-Santos et al.(2020)Bravo-Santos, Guerra, and Lara]). Our focus is in a selection of those papers on adversarial text generation relevant to poisoning attacks in general or mutation-based approaches in particular.

Word swapping (of semantically equivalent words) attack, [Alzantot et al.(2018)Alzantot, Sharma, Elgohary, Ho, Srivastava, and Chang] is similar to one example of our mutation operators where one word is swapped from original to mutation instances.
In [Alzantot et al.(2018)Alzantot, Sharma, Elgohary, Ho, Srivastava, and Chang],
[Sayghe et al.(2020)Sayghe, Zhao, and Konstantinou] and [Shi et al.(2022)Shi, Ge, and Zhao], words are swapped based on genetic algorithms. Genetic algorithms typically include 5 tasks in which mutation is one of them (Initial population, Fitness function, Selection, Crossover, and Mutation). As alternatives to genetic algorithms, word swapping attacks are also implemented using (1) neural machine translation in , [Ribeiro et al.(2018)Ribeiro, Singh, and Guestrin], (2) swarm optimization-based search algorithm, [Shi et al.(2022)Shi, Ge, and Zhao]. In [Alzantot et al.(2018)Alzantot, Sharma, Elgohary, Ho, Srivastava, and Chang], mutation instances are used to fool a sentiment analysis classifier.

Words substitution can be also role-based lexical substitutions, [Iyyer et al.(2018)Iyyer, Wieting, Gimpel, and Zettlemoyer], or entity-based text perturbation, [Liu and Chen(2022)].

In [Bhalerao et al.(2022)Bhalerao, Al-Rubaie, Bhaskar, and Markov] authors classified intentional and unintentional adversarial text perturbation into ten types, shown in Table I.

As we mentioned earlier, this work is closely related to AML poisoning attacks, [Newsome et al.(2006)Newsome,

Perturbation Type	Defense	Example
Combined Unicode	ACD	P.l.e.a.s.e.l.i.k.e.a.n.d. s.h.a.r.e
Fake punctuation	CW2V	Pleas.e lik,e abd shar!e th v!deo
Neighboring key	CW2V	Plwase lime and sharr the vvideo
Random spaces	CW2V	Pl ease lik e and sha re tl video
Replace Unicode	UC	Pleäse lïke and sharê the video
Space separation	ACD	Please l i k e and s h a r
Tandem character obfuscation	UC	PLE/\SE LIKE /\ND SH/\RE
Transposition	CW2V	Please like adn sahre
Vowel repetition and deletion	Mutation Testing in the Language Domain CW2V	Pls likee nd sharee
Zero-width space separation	ACD	Please like and share the video

TABLE I
ADVERSARIAL TEXT PERTURBATIONS, [BHALERAO ET AL.(2022)BHALERAO, AL-RUBAIE, BHASKAR, AND MARKOV]

Mutation Operator	Example	Definition
Randomization	Plz shr and hate film	Use all below mutation operators
Misspelling words	Plz sharr and like the vid	Misspell a few words
Deleting articles	Please share and like video	Delete a few articles, including starting ones
Random word with random word	Please roar and tree video	Replace a random word with another random word
Synonym replacement	Please disseminate and prefer the video	Replace a word with its synonym
Antonym replacement	Please hide and hate the video	Replace a word with its antonym
Replace ”a”, ”e”	Pls lik nd shar the video	Replace some a’s and e’s with epsilon & alpha

TABLE II
EXPERIMENTAL MUTATION OPERATORS

Karp, and Song], [Perdisci et al.(2006)Perdisci, Dagon, Lee, Fogla, and Sharif], [Nelson et al.(2008)Nelson, Barreno, Chi, Joseph, Rubinstein, and Saini], [Rubinstein et al.(2009)Rubinstein, Nelson, Huang, Joseph, Lau, Rao, Taft, and Tygar],
[Barreno et al.(2010)Barreno, Nelson, Joseph, and Tygar], [Biggio et al.(2012)Biggio, Nelson, and Laskov],
[Newell et al.(2014)Newell, Potharaju, Xiang, and Nita-Rotaru].
It is also related to AML text perturbations, [Vijayaraghavan and Roy(2019)], [Eger et al.(2019)Eger, Şahin, Rücklé, Lee, and Schulz], [Gao et al.(2018)Gao, Lanchantin, Soffa, and Qi], and [Li et al.(2018)Li, Ji, Du, Li, and Wang].

III. GOALS AND APPROACHES

According to a previous paper [Wolff and Wolff(2020)], a typical RoBERTa-based classifier mislabels synthetic text to human by very basic differences such as changing ’a’s to alpha or ’e’s to epsilon. This vulnerability can be used to trick detectors of synthetic text either intentionally or accidentally.

To compare synthetic text detectors sensitivity to mistakes or changes to human text we can break up these mutations into operators with the goal of supporting the creation of more generalized synthetic text detectors. Here, these operators will be introduced and be used to fine-tune RoBERTa’s [**?**] pre-trained model to detect mutations, such as the first scenario mentioned earlier, section I-B.

A. Approach: Use a finite set of operators for research customization

We introduce some examples of mutation operators to implement are in Table II. These are more advanced can be used for attacking a detector on a more granular level. For our research here however, we will be using more basic mutation operators such as these:

These 7 operators can replicate simple mistakes and changes which can happen to human written text, including the 2 operators used in previous cited works. These were chosen for their ease of implementation and usage in previous research. We will leave more advanced and numerous operators for future papers.

As for the implementation, most of these operators use word maps which iterate through each string replacing the words with the intended character, word insertion or deletion. Punctuation and excess special characters are removed for simplicity. Though there is different methods for the different mutation operators.

The random word operator is in fact a list of random words. Arbitrary words are chosen and replaced with a random word from the same list. Limits to the number of mutations should be added to limit the operator from completely fuzzing the string as well. The code from our GitHub, https://github.com/JesseGuerrero/Mutation-Based-Text-Detection has some written operators which can be viewed as examples.

In our implementation word maps are limited to 3000 of the most common words, synonyms and antonyms. These words can be pulled from any API service such as RapidAPI to get lists of words, synonyms, adverbs, verbs, etc.

B. Approach: Test mutations by Evasion attacks on neural network detectors

We will be testing these 7 operators against RoBERTa pre-trained models. Of these 7 operators we want to test how they will affect a previously researched synthetic text detector, how a fine-tuned mutation text detector will be affected and as well as the differences between the different mutation operators. Lastly we want to see how shorter text affects these results. More details on the next section, IV.

IV. EXPERIMENTS AND ANALYSIS

With these mutations, we can introduce mutation detection with a classic binary RoBERTa classifier. This part of the experiment is the extension portion of the previous author’s work mentioned before with an actual solution to the vulnerabilities in that paper.

A. Experiments Methodology

We used an existing RoBERTa classifier which is meant to classify synthetic and human generated text to test how it would classify mutated text. It is still the pre-trained binary model, however it is being retrained to detect mutation rather than synthetic text.

The data set used was the full COCO images data set where hand written captions are placed for each image. A total of 5 captions are human created per image. The captions were parsed into a re-usable format and were used to train the human portions of the model.

Across the training, testing and validation sets there were over 700,000 human texts used to train the model. Two models were made from this data set. The first was based on individual captions. The second was based on these 5 captions combined per unique image name for calling via a map.

Six operators were used as mutations for this classifier. They are; (1) replacing synonyms, (2) antonyms, (3) random words, (4) removing articles, (5) replacing a with alpha and e with epsilon, and lastly, (6) the most common misspellings.

The training data was duplicated for the mutation data sets. Over 700,000 texts were used for training and the same texts were re-used for mutations. This meant for individual captions there were over 1.4 million text instances with both mutation and human labels. For combined captions there were 1/5 of the total instances.

The data sets were selected as they were already labelled from COCO dataset. The training set went to training, the validation set went to validation and testing to testing. For training the mutation label, the mutation operators were used at run-time.

The operator was randomly chosen at run-time with a simple random function among 6 operators. Each operator was used so the classifier can learn to detect all 6 of these mutation types in one classifier.

So far as testing is concerned, the same testing data set from COCO was re-used with an operator manipulating a whole set. A total of 7 testing data sets were created for each of the 6 operators and a seventh randomized data set, like the mutations at run-time. This formed our metrics of how accurately the model can correctly label each instance of the mutation data sets as mutations and how accurately the model can label human text. In a total 8 operators, 1 human and a seventh randomizing the first six mutations.

B. Preliminary Results

If we were to apply these operators to the previously researched synthetic text we should get poor results for detecting mutated text. Given text derived from human text, though just modified, is still synthetic, we can see that mutation poses a vulnerability to detecting machine and human generated texts. Here are the results:

Operator Type	Accuracy
None	∼88.80%(1000 samples)
Randomized	∼01.00%(1000 samples)
Replace Alpha, Epsilon	∼01.01%(1000 samples)
Misspelling words	∼00.00%(1000 samples)
Delete articles	∼01.60%(1000 samples)
Synonym replacement	∼00.00%(1000 samples)
Replace random word	∼07.79%(1000 samples)
Antonym replacement	∼09.89%(1000 samples)

TABLE III
PRELIMINARY RESULTS

As we can see from 1000 samples the accuracy is quite poor when modifying the text. The original detector without mutations had a recall of over 97% detection of synthetic and human text in-distribution. For our research outside of the paper we have an out of distribution pure human text data set as 88% accurate as the 1st row in the table.

This means the detector is quite good at classifying human text out of distribution and is even good at detecting in-distribution synthetic text. The model does those things above human distinction which in the past was around 54% accurate. Our issue from our modeling is mutation from which the model does not perform well.

C. Experiments Results & Analysis

So far as the first run through with individual captions, the results were pretty good. A total of 2,490 texts were tested for the detector from the testing data set. In total overall the detector accuracy was about 91% and each epoch took 13 hours for a total of 4 epochs or 52 hours total.

Operator Type	Accuracy
None	∼71.48%(2490)
Randomized	∼99.83%(2490)
Replace alpha, epsilon	∼99.95%(2490)
Misspelling words	∼99.95%(2490)
Delete articles	∼59.87%(2490)
Synonym replacement	∼99.91%(2490)
Replace random word	∼100%(2490)
Antonym replacement	∼99.03%(2490)

TABLE IV
INDIVIDUAL CAPTIONS, SHORT LANGUAGE MODELING

The most inaccurate operator overall was always the ”delete articles operator”. This can be due to some semantic issues or just the difficulty of detecting what is *not* there rather than what *is there* to a RoBERTa classifier. For this first run, the other operators ranged from 59% accurate to 100% accurate, with human detection being 71%.

For the second run the captions for text chunks were combined per image. This meant all 5 captions were now one text and were fed to the training model. This means 1/5 th of the instances but more per text. The results were a slightly lower; total overall detector accuracy of 88% with 2490 texts being tested.

The epochs took about 2 hours each this way as well. A total of 10 epochs or 20 hours were used to finish the model training. Same as before, the delete articles operator was the weakest mutator and was in fact even weaker the second time.

Operator Type	Accuracy
None	∼93.65%(2490)
Randomized	∼98.96%(2490)
Replace alpha, epsilon	∼99.92%(2490)
Misspelling words	∼99.80%(2490)
Delete articles	∼25.42%(2490)
Synonym replacement	∼99.76%(2490)
Replace random word	∼98.43%(2490)
Antonym replacement	∼92.37%(2490)

TABLE V
COMBINED CAPTIONS, LONGER LANGUAGE MODELING

Besides the weakest operators, the other operators were all above 95%, much better than the original neural network. If we were to remove the lowest performing operator we would in fact have 95% accuracy for the 1st run and 97% accuracy for the 2nd. This means in reality the combined text may be the better approach to train.

D. Experiment: Models issues and weaknesses

The main issue of the evaluated models is possible a bias issue. It seems that models work mostly for semi in-distribution data sets. So accuracy is altered quite a bit by outside data sets. Both the individually captioned model and the grouped model were tested in an out of distribution data set. The individually captioned data set was 2.2% accurate for human detection and 100% accurate for Alpha/Epsilon mutation, while the grouped captioned data set was 55.7% accurate for human detection and 100% for Alpha/Epsilon mutation. The issue appears to be the out of distribution set may have had vocabulary that didn't exist in distribution and the detector defaults to the mutation label. Still in this sense, the longer the text-set the better is the performance.

In the future we can use this work flow with more diverse training data sets to generalize models to out of distribution and use these models to prevent simple mutation from fooling binary detectors. Lastly, the mutation operator "delete articles" seems like a great way to fool the classifier into mislabeling mutated text.

V. A Comparison Study

A. Machine Text Generation

Machine text generation is a field of study in Natural Language Processing (NLP) that combines computational linguistics and artificial intelligence that has progressed significantly in recent years. Neural network approaches are widely used for this task and keep dominant in the field. The state-of-the-art methods may include Transformers [Vaswani et al.(2017)Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, and Kaiser], BERT [Devlin et al.(2018)Devlin, Chang, Lee, and Toutanova], GPT-3 [Brown et al.(2020)Brown, Mann, Ryder, Subbiah, Kaplan, Dhariwal, and Neelakantan], RoBERTa [**?**], etc. The models are trained on a large amount of text data. For example, the GPT-2 model was trained on text scrapped from eight million web pages [Radford et al.(2019)Radford, Wu, Child, Luan, Amodei, Sutskever, et al.], and is able to generate human-like text. Due to the high text generation performance, such methods are very popular on tasks, such as image caption generation, text summarization, machine translation, moving script-writing, and poetry composition. However, the output of such methods is often open-ended.

Through this work, we propose a mutation-based text generation method that can be distinguished from the existing text generation method fundamentally. Unlike the neural network based methods, the mutation-based method generates output based on the given text under a given condition. The text is generated in a tightly controlled environment, and the output is closed-ended. The well-controlled environment makes the output of the mutation-based method suitable for serious security test tasks, such as machine learning model vulnerability tests and SQL injection defense and detection.

Given a text corpus (e.g., a sentence or a paragraph), $\mathcal{T}$, which contains an ordered set of words, $\mathcal{W} = \{w_1, w_2, ..., w_n\}$, and an ordered set of punctuation, $\mathcal{P} = \{p_1, p_2, ..., p_m\}$, a mutation operator, $\mu(\cdot)$ is used to generate the mutation-based text. For instance, given a character-level mutation operator, $\mu_c(\cdot)$:

$$\mathcal{W}' = \mu_c(w_i, \rho, \sigma), \tag{1}$$

where $\mathcal{W}'$ is the output of $\mu_c(\cdot)$, which replaces the letter ρ in w_i ($w_i \in \mathcal{W}$) with a mutation σ. Then, the final output of $\mathcal{T}$ is $\mathcal{T}' = \langle \mathcal{W}', \mathcal{P} \rangle$. For instance, assume $\mathcal{T} =$ `"Text generation is interesting!"`, $\mathcal{W} = \{\texttt{Text}, \texttt{generation}, \texttt{is}, \texttt{interesting}\}$, and $\mathcal{P} = \{\texttt{!}\}$, $w_i = \texttt{generation}$, and a character-level mutation operator $\mu_u(\cdot)$, where $\rho = \texttt{a}$ and $\sigma = \alpha$ (the Greek letter alpha). Then, the output text corpus, $\mathcal{T}'$ is generated as:

$$\begin{aligned} \mathcal{T}' &= \langle \mathcal{W}', \mathcal{P} \rangle \\ &= \langle \mu_c(w_i, \rho, \sigma), \mathcal{P} \rangle \\ &= \langle \mu_c(\texttt{generation}, \texttt{a}, \alpha), \mathcal{P} \rangle \\ &= \texttt{Text gener}\alpha\texttt{tion is interesting!} \end{aligned} \tag{2}$$

VI. Discussions, Conclusions and Future Directions

Automatic text generation techniques are adopted into various domains, from question-answering to AI-driven education. Due to the progress of neural network (NN) techniques, NN-based approaches dominate the field. Though advanced techniques may be applied to control text generation direction, the text is still generated in a widely open-ended fashion. For instance, a NN-based approach can generate a greeting message to greet a specific person. However, it is hard to control the exact wording used in the message. Such open-ended text generation might work fine for content generation tasks. However, due to lack of precise control, using open-ended text generation methods to systematically evaluate flaws in language analysis models may be non-trivial.

Unlike the existing language models, our proposed mutation-based text-generation framework provides a tightly controlled environment for text generation that extends text-generation techniques to the field of cyber security (i.e., flaws evaluation for language analysis models). The output of our framework can be used to systematically evaluate any machine learning models or software systems that use a sequence of text as input, such as SQL injection detection [Hlaing and Khaing(2020)] and software debugging [Zhao et al.(2022)Zhao, Su, Liu, Zheng, Wu, Kavuluru, Halfond, and Yu]. Researchers may also design their own mutation operators under our framework.

We demonstrated the proposed text-generation framework using the RoBERTa-based detector that is pre-trained for

separating human-written text from synthetic ones. Our experiments showed that the RoBERTa-based detector has a significant flaw. As a detection method, it is extremely vulnerable to simple adversarial attacks, such as replacing the English letter "a" with the Greek letter "α" or removing the articles—a, an, the—from a sentence. We also demonstrated that simply including the adversarial samples (i.e., the mutation texts) in the fine-tuning stage of the classifier would significantly improve the model robustness on such types of attack. However, we believe that this issue should be better addressed on the feature level since any changes at the text level will lead to changes in the tokenization stage that will eventually lead to a different embedding vector being fed into the classification network. Thus, one future direction of this work is reducing the distance between the original and mutation samples in the feature space. Some potentially useful methods might include using contrastive learning and siamese network [Koch et al.(2015)Koch, Zemel, Salakhutdinov, et al.], [Liang et al.(2021)Liang, Greenwell, Zhang, Xing, Wang, Kavuluru, and Jacobs] as well as dynamic feature alignment [Zhang et al.(2022)Zhang, Liang, and Jacobs], [Dong et al.(2020)Dong, Cong, Sun, Liu, and Xu].

Besides improving the robustness of the RoBERTa-based detector, we plan to continue to work on the development of the proposed mutation-based text generation framework. Currently, the framework only works in a two-step testing scenario. Users need to use the framework to generate the testing cases and feed the testing cases into the downstream model in separate steps. We plan to release a library that can be directly imported into any downstream applications. Tools for easily creating and editing mutation operators will also be created. In addition, a graphical user interface may also be developed.

In conclusion, we propose a general-purpose, mutation-based text generation framework that produces close-ended, precise text. The output of our framework can be used in various downstream applications that take text sequences as input, providing a systematic way to evaluate the robustness of such models. We believe the proposed framework offers a new direction to systematically evaluate language models that will be very useful to those who are seeking insightful analysis of such models.

REFERENCES

[Yi et al.(2017)Yi, Li, and Sun] Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. Generating chinese classical poems with rnn encoder-decoder. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data: 16th China National Conference, CCL 2017, and 5th International Symposium, NLP-NABD 2017, Nanjing, China, October 13-15, 2017, Proceedings 16*, pages 211–223. Springer, 2017.

[Zhang et al.(2016)Zhang, Yao, and Wan] Jianmin Zhang, Jin-ge Yao, and Xiaojun Wan. Towards constructing sports news from live text commentary. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1361–1371, 2016.

[Serban et al.(2016)Serban, Sordoni, Bengio, Courville, and Pineau] Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[Prakash et al.(2016)Prakash, Hasan, Lee, Datla, Qadir, Liu, and Farri] Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. Neural paraphrase generation with stacked residual lstm networks. *arXiv preprint arXiv:1610.03098*, 2016.

[Genest and Lapalme(2011)] Pierre-Etienne Genest and Guy Lapalme. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the workshop on monolingual text-to-text generation*, pages 64–73, 2011.

[Schick and Schütze(2021)] Timo Schick and Hinrich Schütze. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402, 2021.

[Chen(2021)] Meng Chen. Short text generation based on adversarial graph attention networks. In *2021 3rd International Conference on Artificial Intelligence and Advanced Manufacture*, pages 2889–2893, 2021.

[Zhang and Huang(2019)] Yaoqin Zhang and Minlie Huang. Overview of the ntcir-14 short text generation subtask: emotion generation challenge. In *Proceedings of the 14th NTCIR Conference*, 2019.

[Guo et al.(2018)Guo, Lu, Cai, Zhang, Yu, and Wang] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Guan et al.(2021)Guan, Mao, Fan, Liu, Ding, and Huang] Jian Guan, Xiaoxi Mao, Changjie Fan, Zitao Liu, Wenbiao Ding, and Minlie Huang. Long text generation by modeling sentence-level and discourse-level coherence. *arXiv preprint arXiv:2105.08963*, 2021.

[Hua and Wang(2020)] Xinyu Hua and Lu Wang. Pair: Planning and iterative refinement in pre-trained transformers for long text generation. *arXiv preprint arXiv:2010.02301*, 2020.

[Radford et al.(2018)Radford, Narasimhan, Salimans, Sutskever, et al.] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[Radford et al.(2019)Radford, Wu, Child, Luan, Amodei, Sutskever, et al.] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[Brown et al.(2020)Brown, Mann, Ryder, Subbiah, Kaplan, Dhariwal, and Neelakantan] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, and et. al. Neelakantan. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[South et al.(2022)South, Smart, Roughan, and Mitchell] Tobin South, Bridget Smart, Matthew Roughan, and Lewis Mitchell. Information flow estimation: a study of news on twitter. *Online Social Networks and Media*, 31:100231, 2022.

[South et al.(2017)South, Smart, Roughan, and Mitchell] Tobin South, Bridget Smart, Matthew Roughan, and Lewis Mitchell. Online social networks and media. 2017.

[Alsmadi and O'Brien(2020)] Izzat Alsmadi and Michael J O'Brien. How many bots in russian troll tweets? *Information Processing & Management*, 57(6):102303, 2020.

[Alsmadi et al.(2021)Alsmadi, Ahmad, Nazzal, Alam, Al-Fuqaha, Khreishah, and Algosaibi] Izzat Alsmadi, Kashif Ahmad, Mahmoud Nazzal, Firoj Alam, Ala Al-Fuqaha, Abdallah Khreishah, and Abdulelah Algosaibi. Adversarial attacks and defenses for social network text processing applications: Techniques, challenges and future research directions. *arXiv preprint arXiv:2110.13980*, 2021.

[Newsome et al.(2006)Newsome, Karp, and Song] James Newsome, Brad Karp, and Dawn Song. Paragraph: Thwarting signature learning by training maliciously. In *International Workshop on Recent Advances in Intrusion Detection*, pages 81–105. Springer, 2006.

[Perdisci et al.(2006)Perdisci, Dagon, Lee, Fogla, and Sharif] Roberto Perdisci, David Dagon, Wenke Lee, Prahlad Fogla, and Monirul Sharif. Misleading worm signature generators using deliberate noise injection. In *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 15–pp. IEEE, 2006.

[Nelson et al.(2008)Nelson, Barreno, Chi, Joseph, Rubinstein, and Saini] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, and et. al. Saini. Exploiting machine learning to subvert your spam filter. *LEET*, 8(1):9, 2008.

[Rubinstein et al.(2009)Rubinstein, Nelson, Huang, Joseph, Lau, Rao, Taft, and Tygar] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J Doug Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, pages 1–14, 2009.

[Barreno et al.(2010)Barreno, Nelson, Joseph, and Tygar] Marco Barreno, Blaine Nelson, Anthony D Joseph, and J Doug Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[Biggio et al.(2012)Biggio, Nelson, and Laskov] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[Newell et al.(2014)Newell, Potharaju, Xiang, and Nita-Rotaru] Andrew Newell, Rahul Potharaju, Luojie Xiang, and Cristina Nita-Rotaru. On the practicality of integrity attacks on document-level sentiment analysis. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, pages 83–93, 2014.

[Russu et al.(2016)Russu, Demontis, Biggio, Fumera, and Roli] Paolo Russu, Ambra Demontis, Battista Biggio, Giorgio Fumera, and Fabio Roli. Secure kernel machines against evasion attacks. In *Proceedings of the 2016 ACM workshop on artificial intelligence and security*, pages 59–69, 2016.

[Zhang et al.(2020)Zhang, Wang, Liu, and Wang] Fuyong Zhang, Yi Wang, Shigang Liu, and Hua Wang. Decision-based evasion attacks on tree ensemble classifiers. *World Wide Web*, 23(5):2957–2977, 2020.

[Sayghe et al.(2020)Sayghe, Zhao, and Konstantinou] Ali Sayghe, Junbo Zhao, and Charalambos Konstantinou. Evasion attacks with adversarial deep learning against power system state estimation. In *2020 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2020.

[Shu et al.(2022)Shu, Xia, Williams, and Menzies] Rui Shu, Tianpei Xia, Laurie Williams, and Tim Menzies. Omni: automated ensemble with unexpected models against adversarial evasion attack. *Empirical Software Engineering*, 27(1):1–32, 2022.

[Chernikova and Oprea(2022)] Alesia Chernikova and Alina Oprea. Fence: Feasible evasion attacks on neural networks in constrained environments. *ACM Transactions on Privacy and Security*, 25(4):1–34, 2022.

[Bravo-Santos et al.(2020)Bravo-Santos, Guerra, and Lara] Sergio Bravo-Santos, Esther Guerra, and Juan de Lara. Testing chatbots with charm. In *International Conference on the Quality of Information and Communications Technology*, pages 426–438. Springer, 2020.

[Jin et al.(2020)Jin, Jin, Zhou, and Szolovits] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025, 2020.

[Kumar et al.(2020)Kumar, Choudhary, and Cho] Varun Kumar, Ashutosh Choudhary, and Eunah Cho. Data augmentation using pre-trained transformer models. *arXiv preprint arXiv:2003.02245*, 2020.

[Mathew and Bindu(2020)] Leeja Mathew and VR Bindu. A review of natural language processing techniques for sentiment analysis using pre-trained models. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, pages 340–345. IEEE, 2020.

[Alsmadi et al.(2022)Alsmadi, Aljaafari, Nazzal, AlHamed, Sawalmeh, and Khreishah] Izzat Alsmadi, Nura Aljaafari, Mahmoud Nazzal, Shadan AlHamed, Ahmad Sawalmeh, and et. al. Khreishah, Abdallah. Adversarial machine learning in text processing: A literature survey. *IEEE Access*, 2022.

[Alzantot et al.(2018)Alzantot, Sharma, Elgohary, Ho, Srivastava, and Chang] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*, 2018.

[Shi et al.(2022)Shi, Ge, and Zhao] Dingmeng Shi, Zhaocheng Ge, and Tengfei Zhao. Word-level textual adversarial attacking based on genetic algorithm. In *Third International Conference on Computer Communication and Network Security (CCNS 2022)*, volume 12453, pages 272–276. SPIE, 2022.

[Ribeiro et al.(2018)Ribeiro, Singh, and Guestrin] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Semantically equivalent adversarial rules for debugging nlp models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[Iyyer et al.(2018)Iyyer, Wieting, Gimpel, and Zettlemoyer] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*, 2018.

[Liu and Chen(2022)] Zhengyuan Liu and Nancy Chen. Entity-based denoising modeling for controllable dialogue summarization. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 407–418, 2022.

[Bhalerao et al.(2022)Bhalerao, Al-Rubaie, Bhaskar, and Markov] Rasika Bhalerao, Mohammad Al-Rubaie, Anand Bhaskar, and Igor Markov. Data-driven mitigation of adversarial text perturbation. *arXiv preprint arXiv:2202.09483*, 2022.

[Vijayaraghavan and Roy(2019)] Prashanth Vijayaraghavan and Deb Roy. Generating black-box adversarial examples for text classifiers using a deep reinforced model. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 711–726. Springer, 2019.

[Eger et al.(2019)Eger, Şahin, Rücklé, Lee, and Schulz] Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, and et. al. Schulz. Text processing like humans do: Visually attacking and shielding nlp systems. *arXiv preprint arXiv:1903.11508*, 2019.

[Gao et al.(2018)Gao, Lanchantin, Soffa, and Qi] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE, 2018.

[Li et al.(2018)Li, Ji, Du, Li, and Wang] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018.

[Wolff and Wolff(2020)] Max Wolff and Stuart Wolff. Attacking neural text detectors. 2 2020. URL http://arxiv.org/abs/2002.11768.

[Vaswani et al.(2017)Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, and Kaiser] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, and et. al. Kaiser. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Devlin et al.(2018)Devlin, Chang, Lee, and Toutanova] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Hlaing and Khaing(2020)] Zar Chi Su Su Hlaing and Myo Khaing. A detection and prevention technique on sql injection attacks. In *2020 IEEE Conference on Computer Applications (ICCA)*, pages 1–6. IEEE, 2020.

[Zhao et al.(2022)Zhao, Su, Liu, Zheng, Wu, Kavuluru, Halfond, and Yu] Yu Zhao, Ting Su, Yang Liu, Wei Zheng, Xiaoxue Wu, Ramakanth Kavuluru, William GJ Halfond, and Tingting Yu. Recdroid+: Automated end-to-end crash reproduction from bug reports for android apps. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31 (3):1–33, 2022.

[Koch et al.(2015)Koch, Zemel, Salakhutdinov, et al.] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, page 0. Lille, 2015.

[Liang et al.(2021)Liang, Greenwell, Zhang, Xing, Wang, Kavuluru, and Jacobs] Gongbo Liang, Connor Greenwell, Yu Zhang, Xin Xing, Xiaoqin Wang, Ramakanth Kavuluru, and Nathan Jacobs. Contrastive cross-modal pre-training: A general strategy for small sample medical imaging. *IEEE Journal of Biomedical and Health Informatics*, 26(4):1640–1649, 2021.

[Zhang et al.(2022)Zhang, Liang, and Jacobs] Yu Zhang, Gongbo Liang, and Nathan Jacobs. Dynamic feature alignment for semi-supervised domain adaptation. *British Machine Vision Conference (BMVC)*, 2022.

[Dong et al.(2020)Dong, Cong, Sun, Liu, and Xu] Jiahua Dong, Yang Cong, Gan Sun, Yuyang Liu, and Xiaowei Xu. Cscl: Critical semantic-consistent learning for unsupervised domain adaptation. In *European Conference on Computer Vision*, pages 745–762. Springer, 2020.