Automatic Generation of Adversarial Readable Chinese Texts

Mingxuan Liu, Zihan Zhang, Yiming Zhang, Chao Zhang, *Member, IEEE*, Zhou Li, *Senior Member, IEEE*, Qi Li, *Senior Member, IEEE*, Haixin Duan, *Member, IEEE*, Donghong Sun, *Member, IEEE*

Abstract—Natural language processing (NLP) models are known vulnerable to adversarial examples, similar to image processing models. Studying adversarial texts is an essential step to improve the robustness of NLP models. However, existing studies mainly focus on generating adversarial texts for English, with no prior knowledge that whether those attacks could be applied to Chinese. After analyzing the differences between Chinese and English, we propose a novel adversarial Chinese text generation solution Argot, by utilizing the method for adversarial English examples and several novel methods developed on Chinese characteristics. Argot could effectively and efficiently generate adversarial Chinese texts with good readability in both white-box and black-box settings. Argot could also automatically generate *targeted* Chinese adversarial texts, achieving a high success rate and ensuring the readability of the generated texts. Furthermore, we apply Argot to the spam detection task in both local detection models and a public toxic content detection system from a well-known security company. Argot achieves a relatively high bypass success rate with fluent readability, which proves that the real-world toxic content detection system is vulnerable to adversarial example attacks. We also evaluate some available defense strategies, and the results indicate that Argot can still achieve high attack success rates.

Index Terms—Natural language processing, Generation of Adversarial example

1 Introduction

NATURAL Language Processing (NLP) plays an important role in information processing tasks, e.g., sentiment analysis of reviews used in recommendation systems [2] and toxic content identification in online governance [3]. However, NLP with Deep Neural Networks (DNNs) is proved vulnerable to adversarial examples [4] as image-processing DNNs. As a popular promotion method, cybercriminal always uses simple perturbations to evade the real-world text-based detection system easily [5]. Surprisingly little attention, however, has been paid to the real-world adversarial techniques deployed by the cybercriminal. Therefore, studying adversarial texts of NLP models is a crucial and essential step to improve NLP models' robustness.

Different from continuous image data, text data is discrete which makes generating adversarial texts more challenging. Even minor perturbations applied to a word vector could yield a non-existent word vector (i.e., not associated with any valid word in the embedding vocabulary). In addition, these non-existed word vectors will result in meaningless texts. Increasing the perturbations may affect the readability of texts. Worse, the perturbations may cause the text expression vector to fail to map back to text. As a result, adversarial example generation methods for image processing DNNs cannot be directly transformed to NLP DNNs.

Several works have been devoted to generating adversarial texts [4], focusing primarily on the English language. Researchers [4], [6] have proposed five commonly used methods to generate English adversarial texts in the black-box scenario, i.e., insert, delete, swap, substitute-Character and substitute-Word. These methods have been proved to achieve high attack success rates of adversarial attacks by only introducing minor perturbations to the texts. Various NLP models are vulnerable to such adversarial attacks, even the large-scale pre-trained language models [7]. However, the possibility of migrating these methods to Chinese texts has not been discussed. Moreover, Chinese itself has some unique characteristics, e.g., pictograms, pinyin and no separators. Therefore, two research questions are raised: (1) Can existing adversarial English text generation methods be transformed to Chinese? How effective are they? (2) Are there new methods for generating adversarial texts based on the characteristics of Chinese? What is the quality of generated Chinese adversarial texts?

In this paper, we first analyzed the differences between English and Chinese, and found Chinese texts have three unique linguistic characteristics: pronunciation (*pinyin*), vision perception (*glyph*) and composition of characters (*radicals*). Therefore, we proposed a novel Chinese adversarial text generation solution Argot, by adding perturbations based on both the unique Chinese characteristics and two transformed generation methods from English. Note that this solution is also expected to work on other languages that have similar characteristics as Chinese, e.g., Korean and Japanese, which will be explored as part of our future work. In addition, Argot is able to generate both *targeted* and *non-targeted* Chinese adversarial text in both *white-* and *black- scenarios*.

We have evaluated Argot's performance of attack success rate, perturbation, time consumption and readability, on several common-used NLP models under both targeted and non-targeted

- *Mingxuan Liu, Zihan Zhang, Chao Zhang, Qi Li, Haixin Duan and Donghong Sun are with the Institute for Network Science and Cyberspace, Tsinghua University, Beijing, 10084, China, Email: {liumx18, zhangzih19}@mails.tsinghua.edu.cn, {chaoz, qli01, duanhx, sundonghong}@tsinghua.edu.cn.*
- *Yiming Zhang is with Department of Computer Science and Technology, Tsinghua University, Beijing 10084, China, Email: zhangyim17@mails.tsinghua.edu.cn.*
- *Zhou Li is with Department of Electrical Engineering & Computer Science, University of California, Irvine, 3227 Engineering Hall, Irvine, CA, 92697 , Email: zhou.li@uci.edu.*
- *Corresponding author: Chao Zhang, Haixin Duan*

An earlier version appeared at the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20) [1].

TABLE 1: The difference between Chinese and English.

English	**Phrase**	**Word (Sememe)**	**Letter**	N/A	N/A
Meaning	a group of words used together.	component of a sentence	component of a word		
Perturbation	replace with synonyms	add/delete/shuffle words	add/delete/shuffle letters		
Chinese	**Phrase (Chinese Word)**	**Character (Sememe)**	**Radical**	**Visual (shape)**	**Sonic (pronunciation)**
Meaning	A group of characters used together	component of a sentence	component of a character	Glyph (shape)	Pinyin (romanized spelling)
Perturbation	replace with synonyms	shuffle characters	split character into radicals	replace with characters with similar shape	replace with characters with similar pronunciation

attacks. The results showed that Argot can work well in both black- and white-scenario. Argot could generate adversarial texts with a success rate of over 98.99% for non-targeted attacks, by only introducing less than 11.25% perturbations to original texts, and 99.37% success rate with 7.76% perturbations for targeted attacks in a white-box setting. Even without the information of NLP models, Argot could generate adversarial texts with a success rate of over 97.7% for non-targeted attacks, by only introducing less than 11.6% perturbations to original texts, and 98.73% success rate with 12.25% perturbations for targeted attacks in a black-box setting. The low perturbation rate implies that, the adversarial texts generated by Argot have good readability, which is also confirmed by a user study conducted by us with 50 randomly selected adversarial texts generated by Argot. 25 native Chinese speakers participated in this user study. According to their assessments, the generated adversarial text achieved good results in readability (84% of volunteers considered it readable), maintaining the original semantics (91% of volunteers could get correct origin semantic label from adversarial texts) and semantic quality (receiving an average score of 4.6/5 in the semantic quality evaluation). In addition, we find that Argot shows promising performance even on pre-trained language models, which have enhanced robustness than classical deep learning models. Further, we have evaluated the attack performance of Argot in real-world scenarios. Using adversarial texts generated by Argot, we successfully bypassed a toxic content detection system provided by well-known security companies in China. The result shows that Argot could achieve good performance even against a very sophisticated detection system.

Besides, we evaluate Argot's performance against several defense mechanisms. Results show that Argot can still work well on these models with defenses. It also indicates that detecting and defending Argot is a challenging task. We will explore more useful defense approaches in future work.

In summary, we make the following contributions:

- We analyze the characteristics of Chinese texts and point out their differences from English. The fundamental difference is their minimal sememe: character in Chinese while word in English. Also, no separators existed between Chinese words as there are in English. Thereafter, we analyze the limitations of five common English adversarial text generation solutions on Chinese, including *Insert*, *Delete*, *Swap*, *Substitute-Character (Sub-C)* and *Substitute-Word (Sub-W)*.
- We propose the first adversarial Chinese texts generation solution Argot in both white-box and black-box settings, by utilizing several unique characteristics of Chinese, *glyph*, *pinyin* and *splitting characters*. We also redesign Sub-W and Swap in Argot. Besides non-targeted scenarios, Argot also works well in targeted attack scenarios which are more challenging.
- We evaluate the success rate, perturbation, time consumption and readability of texts generated by Argot in several commonly used NLP models, proving its effectiveness and efficiency. Furthermore, we propose several candidate defenses and evaluate Argot's performance against these defenses. The result shows that Argot's performance remains robust against the defenses. We also make some comparative experiments on each perturbation method and analyze their impact on the Chinese NLP models. Results show that the methods of *pinyin*, *glyph* and *shuffle* are most effective, which means that the Chinese NLP models are more vulnerable to the replacement of characters with similar *pinyin* and *glyph*.

2 Background

In this section, we first introduce the concept of adversarial example. Then, we elaborate on the unique characteristics of Chinese compared to English. Chinese texts are very different from English in terms of text composition, visual effects and sonic effects, as shown in Table 1. Therefore, the perturbations available for adversarial text generation are different.

2.1 Adversarial Example

Adversarial example, first proposed in 2014 [8], aims to make the target DNN model give the wrong decision with high confidence by adding perturbations on the input data. Many works focus on generating adversarial examples on image and voice data. They try to attack the image classification and speech recognition models with DNNs, which are one of the most widely used models in real life. Recently, more and more works try to add fewer perturbations with less knowledge about the targeted model.

According to the attacker's knowledge of the model, the generation process of adversarial example can be divided into two categories:

- *White-box setting.* In this setting, attackers can query the model with pre-known information of the targeted model, such as the parameters, structure, training dataset, etc. During the generating process, attackers can obtain the important part of the input by calculating the gradient and adding some perturbations at the corresponding position. Obviously, this kind of setting makes the attack more easily with pre-known knowledge.
- *Black-box setting.* Compared to the white-box setting, attackers have no information about the targeted model. All they can do is querying the model and getting the prediction results. In some stricter scenarios, the model only returns the label of the classification; or attackers can only query the model for a limited time. With the popularity of MLaaS (Machine Learning as a Service), this setting is more in line with the actual situation.

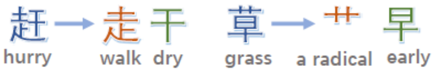


Fig. 1: Illustration of splitting characters into radicals.

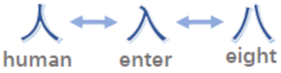


Fig. 2: Illustration of characters with similar glyph.

In this setting, attackers have to use some other methods such as substitute model, gradient estimation and transferability of adversarial examples to generate useful examples.

At present, the popular adversarial example generation methods are for continuous data such as image or voice. With the difference between continuous data and discrete data, adversarial example generation methods for continuous data cannot be transformed to discrete data directly, especially text data. Some works have explored the methods of generating adversarial texts in English. However, there are still challenges to generate adversarial texts in other languages, such as Chinese.

2.2 Composition of Texts

In English and Chinese, the sentence is the basic representation to express a complete idea and a sentence can be separated by several phrases. The smallest *sememe* of a sentence is word (in English) and character (in Chinese) respectively. A word/character can be further divided into letters/radicals. On the other hand, words/characters can be grouped together as a phrase to express a particular meaning.

2.2.1 Words

In Chinese, a phrase is usually called a word, since it is similar to an English word in many ways, which is shown in Table 1. A Chinese word is typically made up of two to four characters, while an English word usually consists of a string of alphabet letters. First, the Chinese word vocabulary can be ever-changing as English, although the number of Chinese characters is limited as English letters (i.e., 26), since they could be combined together with very few limitations. Second, many Chinese and English words have counterparts of the same meanings. For instance, the English word `hello` equals to Chinese word `你(you)好(good)`. Third, a Chinese word could be a synonym for another Chinese word, so does an English word.

However, there are also many differences between Chinese and English words. First, characters cannot be added to or deleted from Chinese words, since the yielded phrases usually do not exist or have totally different meanings. For example, if we delete the character 不(not) from the word 不好(bad), the remaining word 好(good) expresses a totally opposite meaning. But letters can be added to or deleted from English words without interfering with reading, as long as the modification to visual effects is small. Second, English words are separated by a specific mark (i.e., space) in a sentence, but Chinese words are directly placed adjacent to other words or characters. So, Chinese *NLP* models require an indispensable and critical step, i.e., *word segmentation*, and will have trouble with previously unknown words.

2.2.2 Letters and Radicals

An English word is composed of 26 letters, while a Chinese character is composed of a larger but limited number of radicals, which is shown in Figure 1. However, these components have different properties.

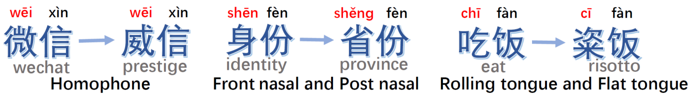


Fig. 3: Illustration of similar pronunciations in Chinese words.

First, some radicals could solely compose Chinese characters, while most letters cannot solely compose English words. So, some Chinese characters can be split into several radicals without interfering with reading too much, but splitting English words could cause trouble with reading.

Second, adding a single letter to an English word, deleting a letter from a word, or shuffling a small number of letters of a word, usually do not confuse English readers too much, e.g., f oolish/folish/fo0lish for “foolish” [4]. However, changing/adding/deleting a radical inside a character will yield either a character with very different appearances, or a character that does not exist in the Chinese character dictionary and cannot print or input to DNNs.

As a result, new words could be invented easily in English and the English word vocabulary can be ever-changing, while new Chinese characters are rare and the Chinese character vocabulary is almost fixed. Namely, creating a new *sememe* in Chinese (i.e., character) acceptable to readers is much harder than English (i.e., word).

2.3 Unique Characteristics of Chinese

Chinese texts also have some unique characteristics in terms of visual and sonic effects, as shown in Table 1.

2.3.1 Visual Effects

Chinese is a type of hieroglyphics, which utilizes the image perception of pictograph (glyph) to help people understand a character or sentence [9]. Some characters are similar in glyph with different semantic meanings, as shown in Figure 2. It is quite likely that a native Chinese speaker can comprehend a sentence in the same way even when a character is transformed to another one with a similar glyph, utilizing the context near this wrong character. In addition, many people use the *Wubi* input method, an input method that encodes Chinese characters completely based on radicals and glyph features. So most people are very familiar with those Chinese characters with similar glyphs and easy to understand the original meaning even there are some typos in the input process.

2.3.2 Sonic Effects

Chinese is a phonetic language with *pinyin* (a romanized spelling) and tone to express the pronunciation. As shown in Figure 3, the English letters on top of each Chinese character are their *pinyin* representations. Each character also has one out of four tones. Many characters and words share the same or similar *pinyin* and tones.

It is worth noting that, the *pinyin* input method is the dominant method to type Chinese characters in computers. But typos are very common in practice, e.g., a wrong pinyin is typed or a wrong candidate character with the same pronunciation is chosen. So, native Chinese speakers are able to understand texts with wrong characters, as long as they have correct or similar *pinyin*.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3164289, IEEE Transactions on Dependable and Secure Computing

3 Generating Adversarial Chinese Text

3.1 Threat Model

The adversary's goal is generating adversarial texts by perturbing a piece of Chinese text to mislead a target NLP application (e.g., toxic content detection), while keeping the texts comprehensible by readers and the meaning intact. Namely, readers can still get the original meaning from perturbed text. While English is still the most popular language served by online NLP services, e.g., Google Cloud NLP [10] and Microsoft Azure Text Analytics [11], services supporting Chinese are catching up. Whether they can defend against adversarial perturbations is questionable, as their backend technique is based on DNN, which is known to be vulnerable to such attacks.

Here we formalize the problem. Assume an NLP classifier f classifies a text piece $x \in \mathcal{X}$ into a label $y \in \mathcal{Y}$, i.e., $y = f(x)$. The adversarial perturbation function h changes x to $x_a \in \mathcal{X}$ i.e., $x_a = h(x)$. The adversary aims to change its prediction label such that $f(x_a) = z, z \neq y$. The change can be *targeted* ($z = s$, where $s \in \mathcal{Y}$ and s is pre-defined by the adversary) or *non-targeted* ($z \in \mathcal{Y} \setminus \{y\}$). Also, we need x_a preserve the meaning of x from the perspective of readers.

In this work, we assume two attack scenarios as described in Section 2.1. For *white-box* attacks, adversaries pre-know the architecture, training dataset, parameters and hyper-parameters of classifier f. They can design how to generate adversarial texts more efficiently with this pre-known information and query the targeted classifier f. In contrast, we assume that the adversaries launch a *black-box* attack, who do not know the architecture, training dataset, parameters and hyper-parameters of classifier f. All they can do is query the public API of f using $x, x_a \in \mathcal{X}$ to obtain the output label and its confidence. Argot can generate adversarial texts in *white-box* settings more effectively and efficiently. More challenging, our attack can still achieve high accuracy in *black-box* settings, in which the adversaries' knowledge is much more limited.

3.2 Adversarial Changes in Chinese Word

Previous works about generating adversarial text focused on English [4] and five types of word perturbation have been proposed: 1) **Insert** a space into the word; 2) **Delete** a random letter; 3) **Swap** random two adjacent letters; 4) **Substitute-Character (Sub-C)** or replace characters with visually similar characters; 5) **Substitute-Word (Sub-W)** or replace the word with its top_k nearest neighbors in a context-aware word vector space. We examine those methods but found not all of them can be transformed into Chinese directly. For example, deleting a character or radical in Chinese will have much more impact on comprehension compared to English.

In the end, we found five types of perturbations applicable to Chinese while 2 of them are transformed from attacks against English with some adjustment and 3 are novel and unique to Chinese[1]. Due to the fact that languages in East Asia like Korean and Japanese share similar properties, e.g., hieroglyphics, our methods are expected to be applicable to those languages as well, which we will explore in future work. We summarize these five types of perturbations for Chinese as: *Synonyms*, *Shuffle*, *Splitting-character*, *Glyph* and *Pinyin*.

- *Synonyms.* This method is transformed from English **Sub-W** method. But we focus on synonyms defined in a vocabulary without using word embedding.
- *Shuffle.* This method is transformed from English **Swap** method. We apply it to characters within a Chinese word.
- *Splitting-character*.* As aforementioned, some radicals are also characters. Therefore, a special method for generating adversarial Chinese texts is splitting characters into radicals. In particular, we only split characters of left-right or semi-enclosed radical structures (left and right part of Figure 1, respectively), as it causes less confusion to readers. This method is similar to **Inserting** spaces into English words.
- *Glyph*.* As aforementioned, characters with similar glyphs can be understood smoothly in a sentence. Therefore we can replace characters with ones of the similar glyph. Note that, this is different from **Sub-C**, as the candidate glyph pairs are much more than English letters (e.g., only 3 options, `1-l`, `O-o` and `a-@`, are explored in [4]).
- *Pinyin*.* As aforementioned, replacing a character with another one of similar *pinyin* also yields readable texts. This feature is unique to hieroglyphics languages, and could be utilized to generate adversarial texts as well.

3.3 Argot

We propose a solution named Argot to generate adversarial Chinese texts, by applying the aforementioned perturbations to target Chinese words in a sentence. The workflow of Argot is shown in Figure 4.

Given a Chinese sentence (termed x) and a target NLP classifier f, Argot first segments x into a list of Chinese words ($W =< w_1, w_2, ..., w_n >$, where n is the length of x) using a standard NLP tool[2]. To retain the readability, we only mutate a limited set of important words $W_{imp} \subset W$. Specifically, we sort each word's contribution based on the classification results of text x under the NLP classifier f, and mark important words accordingly. Then, we iterate important words W_{imp} in order and apply the aforementioned perturbations to each word. The target classifier f will be queried with each new sentence. If f yields a different label z (non-targeted attack) or an expected label s (targeted attack), the query sentence is reported as an adversarial example. Details are presented as follows.

3.3.1 Querying target classifier f

When evaluating the importance of a word or the label of a mutated sentence, we query the target NLP classifier f. Given an input x, we assume f returns the confidence for *every* label, and outputs a label y if and only if the confidence $f_y(x)$ is highest among all candidate labels. This is the normal setting for MLaaS APIs and a common assumption of previous black-box attacks [4]. When evaluating the importance of a word under the black-box setting, we remove the word from the sentence to construct a new sentence, and query the target classifier f with the new sentence to compute the change of confidence. When evaluating the label of a mutated sentence, we design two score functions L_t and L_n to guide targeted and non-targeted attacks respectively.

3.3.2 Non-targeted attack

After adding perturbations on the sentence x with label y, a new sentence x_a is yielded, where $x_a = h(x)$. We monitor the drops

[1] We use "*" to mark those novel methods.

[2] https://github.com/fxsjy/jieba

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3164289, IEEE Transactions on Dependable and Secure Computing

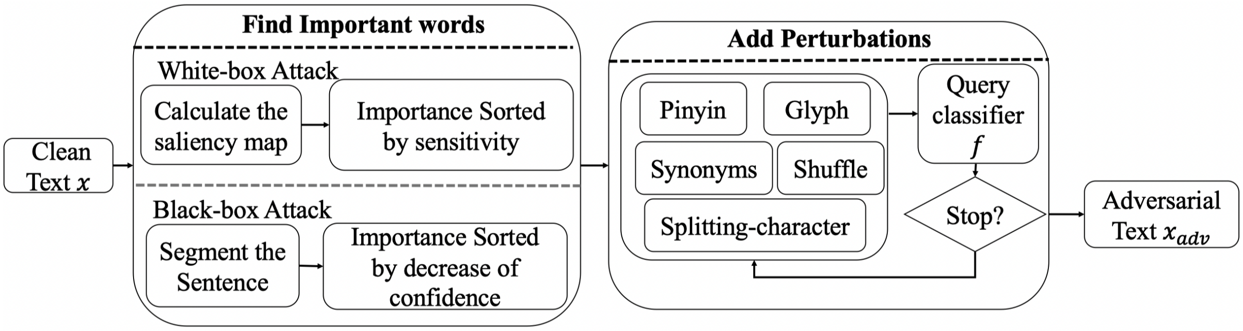


Fig. 4: Workflow of Argot.

of the original label's (i.e., y's) confidence $L_n = f_y(x) - f_y(x_a)$ to evaluate perturbations' effects. Argot will iteratively mutate the target sentence, to generate a sentence x_a^n after n iterations, whose confidence $f_y(x_a^n)$ drops to a very low level and is even lower than (at least) one other label's confidence , i.e., $f_y(x_a^n) < f_z(x_a^n)$ if $\exists z \in \mathcal{Y}$ and $z \neq y$.

Algorithm 1 : The detail of function glyph(c).

Input: Original character c.
Output: New character c_2 with similar glyph.

```
Read in all Chinese radicals into a list $all\_radicals$.
$similarity = 0$, $c_2 = c$, $candidates = \{\}$
$radicals \leftarrow$ decompose_radicals($c$)
for $radical \in radicals$ do                              ▷ Replace radicals
    for $other \in all\_radicals$ do
        if $other \neq radical$ then
            $radicals^{'} \leftarrow$replaceWith($radical, other$)
            $candidates \leftarrow candidates \cup$Val ($radicals^{'}$)
        end if
    end for
end for
for $radical \in radicals$ do                              ▷ Delete radicals
    $radicals^{'} \leftarrow$deleteFromRadicals($radical$)
    $candidates \leftarrow candidates \cup$Val ($radicals^{'}$)
end for
for $other \in all\_radicals$ do                           ▷ Add radicals
    $radicals^{'} \leftarrow$addIntoRadicals($other$)
    $candidates \leftarrow candidates \cup$Val ($radicals^{'}$)
end for
for $c_1 \in candidates$ do              ▷ Assess the similarity of candidates
    $score \leftarrow$siamese_similarity($c, c_1$)
    if $score > similarity$ then
        $similarity = score$, $c_2 = c_1$
    end if
end for
return $c_2$;
```

3.3.3 Targeted attack

In this setting, we use the increase rather than decrease of target label s's confidence $L_t = f_s(x_a) - f_s(x)$, to measure the effectiveness of perturbations. The ultimate goal is that, after n iterations, the label s's confidence $f_s(x_a^n)$ reaches a very high level and is higher than all other labels' , i.e., $f_s(x_a^n) > f_t(x_a^n)$ for $\forall t \in \mathcal{Y}$ and $t \neq s$.

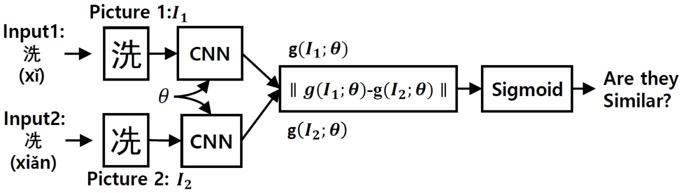


Fig. 5: Architecture of Siamese network.

3.3.4 White-box setting

This setting assumes the adversary can not only access model's input and output, but also pre-know the internal knowledge of target models or compute the gradients. We utilize the `saliency map` to assess the sensitivity of output to the input vectors. The saliency map has been applied to adversarial examples in the image field [12], which can identify the key input features and introduce fewer modified input features than previous work. We have modified the calculation process of the saliency map so that it can be applied to discrete text data. Then, we can use this sensitivity map to assess the importance of words in sentences to reduce the perturbations.

First, we compute partial derivatives for input vectors. The partial derivative shows the relationship between a function and a variable. Given an input data x, $x \in R^{l \times d}$, where n is the length of the text, and d is the dimension of the embedding vector. We define the number of classes as m and $f(x)$ is the output of the DNN model. The partial derivatives are given by:

$$\nabla_x f(x) = \frac{\partial f(x)}{\partial x} = [\frac{\partial f_k(x)}{\partial x_{i,j}}]_{i \in 1...l, j \in 1...d, k \in 1...m}$$

After computing the partial derivative, each word can obtain a gradient vector of length d. Gradient vector means that the directional derivative of a function at this point takes the maximum value along this direction, that is, the function $f(x)$ changes the fastest along this direction (the direction of this gradient) at this point, and the rate of change is the largest. As the original gradient of each word is in the form of a d-dimensional vector and hard to compare directly, we further compute the magnitude of the gradient to measure the sensitivity of each word. Therefore, our saliency map function $S(x)$ for each word in x is defined as:

$$S(x_i) = \sqrt{(\frac{\partial f_k(x)}{\partial x_{i,1}})^2 + (\frac{\partial f_k(x)}{\partial x_{i,2}})^2 + ... + (\frac{\partial f_k(x)}{\partial x_{i,d}})^2}$$

The larger $S(x_i)$ means that x_i has a greater sensitivity to output k, according to the definition of saliency map. So when we add perturbations on x_i, it is more likely to change the decision of the target model. Then, we sort the word list x by $S(x_i)$. We set k as the original label y in both targeted and non-targeted attacks since we want to get the keyword that has the greatest impact on the current label y and add perturbations on it to decrease the confidence of y. We can also set k as the target label s in targeted attacks, however, we find that the saliency map of s is often too small to compare in our evaluation.

3.3.5 Black-box setting

In black-box attacks, gradients are inaccessible so we use the output label and confidence as alternative indicators. In particular, for each word w_i in W, we delete it from the original sentence, query the new sentence with classifier f, and compare the result to the original one's. If deleting w_i makes the original label's confidence drop more than deleting w_j, then w_i is more important than w_j. After enumerating W's drop score, we get a list of words sorted by importance.

3.3.6 Adding perturbations

Given a seed sentence, one round of mutations will be performed on this sentence as follows. The important words will be iterated one by one in the decreasing order of importance. In each iteration, the selected word will be mutated with five candidate perturbations, yielding five new sentences which will be queried with classifier f. If either new sentence satisfies the ultimate goal of non-targeted or targeted attacks, Argot will stop. Otherwise, this round stops after all important words have been iterated. And, the yielded sentence with the highest L_n or L_t is chosen as the new seed sentence, and a new round of mutations will start. Among the five perturbations, splitting-character, synonyms and shuffle are straightforward, so we only elaborate on the remaining two as follows.

Pinyin. Homophone and typos in typing pinyin can be utilized to generate adversarial words with little influence on readability. We propose to replace the target word with the following three types of words:

- A homophone word [13], which has the same pinyin representation.
- A word that has a similar pinyin representation but different head and tail nasal, by interchanging `an` and `ang`, `in` and `ing`, as well as `en` and `eng` in the pinyin representation.
- A word that has a similar pinyin but different rolling/flat tongues, by interchanging `c` with `ch`, `z` with `zh`, and `s` with `sh` [14].

Glyph. For a given word consisting of multiple characters, we replace some characters with ones of similar glyphs. The core challenge is finding similar characters. For each character, we first decompose it into a set of radicals, with an open-source tool[3]. Then, we update the radical set with three strategies: **replacing** a radical with another one from the Chinese vocabulary, **deleting** one radical, or **adding** another radical. A special function **Val()** is utilized to check whether the new radical set can make up a legitimate Chinese character. If yes, it returns the yielded character, otherwise, it returns *NULL*. In this way, we could generate a set of characters with similar glyphs. The character that is most similar to the original character will be used to mutate the sentence. Algorithm 1 shows the details. To measure the similarity between two characters, we develop a DNN model based on the *siamese* network, which has been found effective in comparing input data [15]. The structure is shown in Figure 5. It takes two pictures I_1, I_2 of characters as input and sends them to two identical CNN models g with parameters θ. In our prototype, the model g consists of three convolutional layers and each is followed by a max-pooling layer. The number of convolutional filters is 64 for the first layer and 128 for the last two. Then, the distance between two pictures' features $g(I_1;\theta)$ and $g(I_2;\theta)$ is computed, and the output confidence is produced via *sigmoid*. We selected 16,008 pairs of similar characters from an open-source corpus[4] as the training data. After training, model g can calculate the similarity between any two characters.

4 Evaluation

4.1 Evaluation Setup

4.1.1 NLP task

We select one of the most common NLP tasks, i.e., text classification, as the NLP application to be attacked. We use sentiment classification (2-class) to evaluate the non-targeted attack, and use news classification (5-class) to evaluate the targeted attack.

4.1.2 Dataset

We create a dataset for sentiment classification using samples from an existing Chinese NLP dataset[5]. We choose the reviews of hotels, books, electronic products that are crawled from e-commerce websites. Our dataset is composed of 15,471 negative reviews and 16,188 positive reviews. We manually check all these reviews and delete the ones that are too short or have ambiguous labels by two researchers. We divide this dataset into 80% and 20% for training and validation respectively. For news classification, we use the THUNews dataset [16]. Out of the fourteen classes, we select five classes from the dataset, i.e., affair, education, finance, society and sports. For each class, we sample 25,000 texts as the training set and 5,000 as the validation set.

4.1.3 Target Models

We choose two existing models built on top of *CNN* [17] and *LSTM* [18] as the target models. They are widely used for many NLP tasks. As aforementioned, a big difference between English and Chinese is that, there is a word separator (space) in English, but not in Chinese. As a result, Chinese NLP models usually have two variants, i.e., character-based and word-based, depending on whether the model performs word segmentation and whether it performs characters or word embedding. We consider both variants in evaluation.

4.1.4 Metrics

Three metrics are used to comprehensively evaluate the generated adversarial texts: success rate, perturbations and efficiency. *Success rate* reflects how many sentences in the validation set have been successfully mutated to fool the target NLP model. *Perturbation* reflects the average percentage of characters in a successful adversarial sentence that has been mutated. It also reflects the impact on readability, because more perturbation tends to make readability worse. *Efficiency* represents the average time consumed by Argot when generating adversarial texts. Longer input texts usually cost more time, since Argot is likely to mutate more words. As a result, we use the metric *Time/Char.*, i.e., the time cost divided by the text length (seconds per character).

[3] https://github.com/kfcd/chaizi/blob/master/chaizi-jt.txt

[4] https://github.com/zzboy/chinese

[5] https://github.com/SophonPlus/ChineseNlpCorpus

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3164289, IEEE Transactions on Dependable and Secure Computing

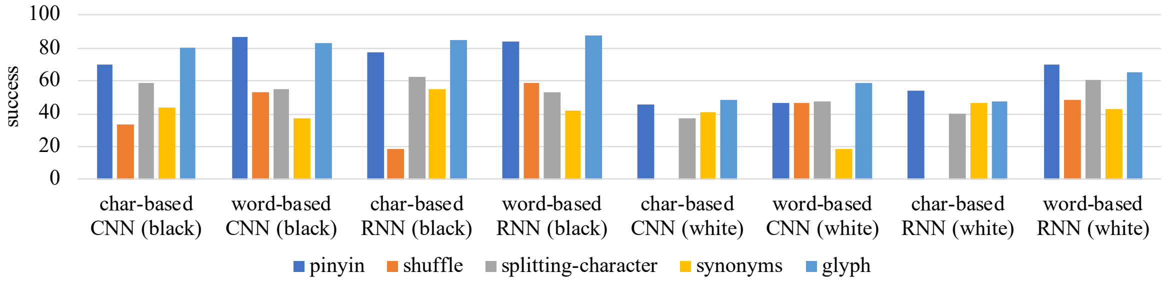


Fig. 6: Success rate of individual perturbations with non-targeted attack under white- and black-box settings.

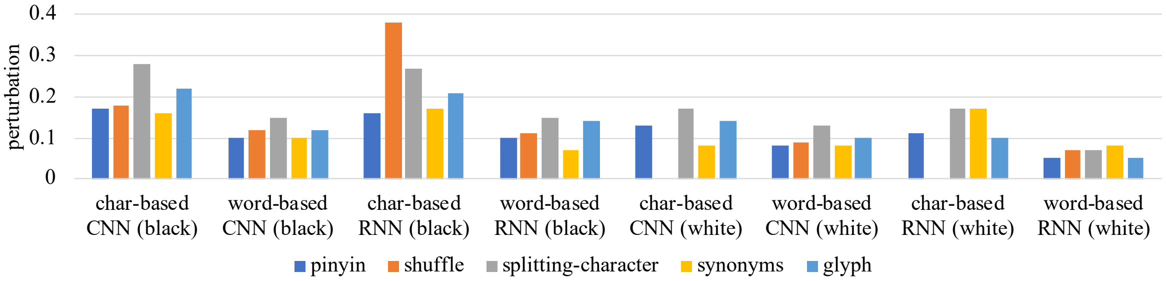


Fig. 7: Perturbation of individual perturbations with non-targeted attack under white- and black-box settings.

TABLE 2: Results of targeted attack under the white-box setting of Argot (S: success rate, P: perturbation, T: Time/Char.).

	Target	Affair			Education			Finance			Society			Sports		
Origin		S	P	T	S	P	T	S	P	T	S	P	T	S	P	T
Affair	char				100%	0.08	0.12	99.80%	0.11	0.17	100%	0.06	0.09	99.60%	0.14	0.21
	word				99.80%	0.07	0.09	100%	0.03	0.04	100%	0.01	0.01	97.00%	0.14	0.15
Education	char	98.60%	0.14	0.18				98.40%	0.25	0.52	100%	0.10	0.12	92.80%	0.24	0.39
	word	99.20%	0.03	0.03				100%	0.01	0.02	100%	0.01	0.01	96.80%	0.08	0.09
Finance	char	100%	0.06	0.11	100%	0.08	0.13				100%	0.05	0.08	99.4%	0.14	0.22
	word	99.60%	0.03	0.04	100%	0.04	0.05				100%	0.02	0.02	97.20%	0.09	0.12
Society	char	100%	0.07	0.14	100%	0.07	0.11	100%	0.09	0.17				100%	0.10	0.17
	word	100%	0.04	0.06	100%	0.06	0.09	100%	0.04	0.06				100%	0.09	0.13
Sports	char	98.60%	0.09	0.17	100%	0.08	0.14	98.20%	0.16	0.34	100%	0.06	0.11			
	word	100%	0.02	0.03	100%	0.04	0.05	100%	0.02	0.03	99.80%	0.01	0.01			

TABLE 3: Results of targeted attack under the black-box setting of Argot (S: success rate, P: perturbation, T: Time/Char.).

	Target	Affair			Education			Finance			Society			Sports		
Origin		S	P	T	S	P	T	S	P	T	S	P	T	S	P	T
Affair	char				100%	0.16	0.37	99.00%	0.22	0.51	100%	0.10	0.29	99.00%	0.23	0.55
	word				100%	0.08	0.20	100%	0.02	0.14	100%	0.01	0.11	96.80%	0.14	0.27
Education	char	98.40%	0.25	0.52				90.00%	0.32	0.69	99.20%	0.18	0.39	84.00%	0.37	0.79
	word	99.00%	0.04	0.13				100%	0.02	0.12	100%	0.02	0.11	96.60%	0.10	0.23
Finance	char	100%	0.12	0.35	99.40%	0.15	0.34				99.60%	0.12	0.30	98.40%	0.24	0.55
	word	99.80%	0.04	0.14	100%	0.05	0.17				99.8%	0.02	0.11	97.00%	0.10	0.24
Society	char	100%	0.15	0.42	99.80%	0.15	0.38	99.80%	0.22	0.53				99.40%	0.21	0.50
	word	100%	0.04	0.17	100%	0.06	0.21	100%	0.03	0.17				100%	0.08	0.24
Sports	char	99.00%	0.17	0.41	99.20%	0.16	0.39	96.60%	0.28	0.59	99.60%	0.15	0.36			
	word	100%	0.02	0.14	100%	0.04	0.17	100%	0.03	0.16	99.80%	0.01	0.10			

4.1.5 Implementation Details

The word segmentation tool for Chinese text pre-processing used in our experiment is jieba[6]. The word embedding scheme is trained from the Chinese wiki corpus [19] with an embedding dimension of 300, using *word2vec* [20]. To generate perturbations, we use a tool [21] to transform Pinyin to character and another tool [22] to transform character to Pinyin. In addition, we use a tool [23] to choose the appropriate synonym for a word.

4.2 Evaluation Results

4.2.1 Evaluation of individual perturbations

To evaluate the effectiveness of individual perturbations, we add only one kind of perturbation to generate adversarial text at a time for non-targeted attacks under both black-box and white-box settings. The result of success rate and perturbation is shown in Figure 6 and Figure 7. Note that, we don't add shuffle perturbation on char-based models under white-box settings. Because we can only assess the importance of each character and it will be unable to swap within a Chinese character.

[6] https://github.com/fxsjy/jieba

TABLE 4: Results of non-targeted attack under the white-box setting of Argot (S: success rate, P: perturbation, T: Time/Char.).

Model	Accuracy	S	P	T
char-based CNN	89.00%	99.73%	0.10	0.25
word-based CNN	90.00%	97.88%	0.13	0.30
char-based LSTM	90.00%	99.93%	0.10	0.23
word-based LSTM	90.00%	98.99%	0.12	0.51

TABLE 5: Results of non-targeted attack under the black-box setting of Argot (S: success rate, P: perturbation, T: Time/Char.).

Model	Accuracy	S	P	T
char-based CNN	89.00%	93.73%	0.14	0.28
word-based CNN	90.00%	98.31%	0.09	0.21
char-based LSTM	90.00%	99.73%	0.13	0.39
word-based LSTM	90.00%	99.05%	0.09	0.58

Compared to traditional perturbations like shuffle and synonyms, pinyin and glyph perform well. Results show that pinyin and glyph perturbations have the highest success rate and relatively low perturbation for each model. It confirms that, common methods used in adversarial English text generation cannot be transferred to Chinese directly, and our methods based on Chinese characteristics are more effective.

In addition, the success rate of adding only one type of perturbation is still not perfect and the perturbation is still high. To improve the performance, we try to combine all five kinds of perturbations to generate adversarial texts. During the next generation, we not only focus on a higher success rate but also high readability as well.

4.2.2 *Non-targeted Attack*

Table 4 and Table 5 show the results of applying all 5 perturbations for non-targeted attacks in the white-box setting and black-box settings. The second column shows the accuracy of original black-box models (all are around 90%). The best success rate is 99.93% on the char-based *LSTM* in white-box settings, while the worst case can still reach 93.73% on char-based *CNN* in black-box settings. The fourth column shows the percentage of perturbations added by Argot compared to the text length, showing that minor mutations are sufficient to yield the desired adversarial texts. And we can see that in both white-box and black-box settings Argot can achieve a relatively high success rate in a short time.

In addition, compared with results in 4.2.1, we can see that the success rate has increased significantly on each model. And perturbation has also dropped a lot. It proves that our strategy of combining all five kinds of perturbations is effective and it improves the readability of adversarial texts to a large degree.

4.2.3 *Targeted Attack*

Table 2 and Table 3 show the results of target attacks on the CNN model. In most cases, Argot achieves a more than 95% success rate by introducing only a few perturbations (lower than 0.2). The small perturbation rate also implies that Argot is very effective and is able to retain the semantics of the original sentences. The results prove that two categories are more likely to be attacked if they are closer in semantics, e.g. society and affair. In contrast, if the semantics are far apart, it's more difficult to attack, such as education and sports.

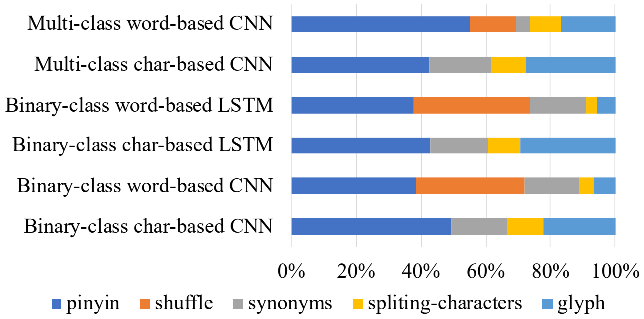


Fig. 8: The distribution of 5 perturbations in white-box attack.

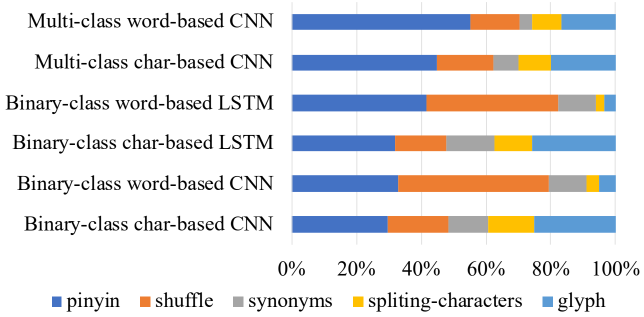


Fig. 9: The distribution of 5 perturbations in black-box attack.

4.2.4 *Efficiency Analysis*

From Table 2, Table 3, Table 4 and Table 5, we can see that Argot is fast at generating adversarial texts. In the best case, it can process one Chinese character within 0.01 and 0.1 seconds on average in the white-box and black-box targeted attack setting, 0.23 and 0.21 seconds for white-box and black-box non-targeted attacks. Note that, the length of input texts in the targeted attack dataset is longer than the non-targeted attack dataset on average (1000 for targeted and 80 for non-targeted), which increases the time of generating adversarial texts.

Also, the efficiency of Argot is strongly related to the specific attack target. For example, adversarial perturbation from *Affair* to *Society* is much more efficient compared to that from *Education* to *Sports*. We speculate the root cause is that the categories *Affair* and *Society* are very similar in semantic meanings and overlapped to some extent, which makes targeted attack easier, but the categories *Education* and *Sports* are very different.

In addition, compared with the black-box setting, our attack costs less time in the white-box setting, especially on the multi-classification model. It is because in the white-box setting, Argot only needs to calculate the gradient once to find the important words, while in the black-box setting, Argot will delete words one by one and check the confidence drop to get the sorted word list, and it needs much time while the text length is longer. As the texts in news classification are much longer than sentiment classification, we can see that the efficiency difference between the targeted attack is greater than that between the non-targeted attack.

4.2.5 *Contributions of Perturbations*

For all adversarial texts, we evaluated which kind of perturbations are added to the original texts. The distributions of these 5 perturbation methods are shown in Figure 8 and Figure 9. In

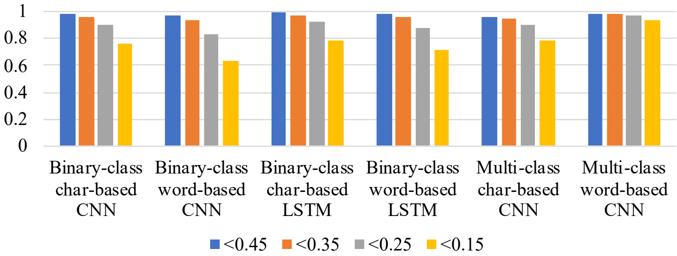


Fig. 10: Relation of perturbations and success rate in white-box attack. The x-axis refers to the perturbation threshold and models. The y-axis refers to the success rate of non-targeted and targeted attacks.

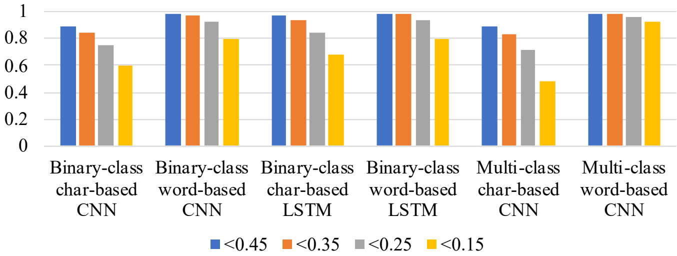


Fig. 11: Relation of perturbations and success rate in black-box attack. The x-axis refers to the perturbation threshold and models. The y-axis refers to the success rate of non-targeted and targeted attacks.

the white-box attack on char-based models, Argot calculates the gradient amplitude on each character. We actually get the character list rather than word list in black-box attack, so Argot can only use four kinds of perturbation except for shuffle.

As we can see, the method *pinyin*, *glyph* and *shuffle* account for the largest proportion. In particular, *pinyin* accounts for the most perturbations overall, which can be explained by the large perturbation space under this method. *Pinyin* and tone determine the pronunciation of a character directly, and they also contain semantic meanings that cannot be provided by the writing system. Existing Chinese NLP models focus on achieving robust classification results on words and characters, thus they are vulnerable to *pinyin* perturbations. Other than *pinyin*, glyph contributes more among the char-based models and shuffle contributes more for word-based models, mainly because those models concentrate on a different set of features. This conclusion is consistent with the results in 4.2.1, when adding only one perturbation, pinyin and glyph perturbations have the highest success rate for each model. And our methods based on Chinese characteristics are effective.

4.2.6 User Study

The ultimate goal of adversarial texts is to mislead NLP models but not humans. So, we further evaluated the human readability of the generated adversarial texts, by conducting a user study. We randomly selected 50 adversarial texts generated by Argot, and queried 25 volunteer native Chinese speakers around the age of 24 with a gender ratio of 1:1. In the user study, for each adversarial text, volunteers are given the pair of the original text and adversarial text, and three questions to answer: (1) whether the adversarial text is comprehensible, (2) what is the label of the adversarial text, and (3) the semantic quality score (0-5) of adversarial texts comparing to the original text. From the result of this survey, 84% of volunteers think that the generated adversarial texts are readable. 91% of them consider the label of the generated adversarial texts the same as the original texts, implying that the adversarial texts do not change the semantics significantly. The average score of semantic quality is 4.6, very close to the highest score 5. In summary, we believe that the generated adversarial texts are of very high quality in terms of readability and semantic understanding. Table 6 presents two examples of the adversarial texts generated by Argot.

4.2.7 Factors Affecting Success Rates

The first factor we look into is the perturbation rate. Figure 10 and Figure 11 shows the changes of success rate given different perturbation rate threshold in both white-box and black-box attack. In this experiment, we control the perturbation as the stop condition to observe the change in the success rate. Results show that the success rate is decreasing when fewer perturbations are allowed in all attack settings. Furthermore, the success rate keeps at a relatively high level even we reduce the perturbation threshold to a relatively low level. Intuitively, the more perturbations added to the original text, the greater the impact on readability will be. As a result, we can trade perturbation rate with success rate to get adversarial texts of higher quality. Another factor that may affect the success rate is the length of the original text. Given a longer input text, Argot could generate adversarial texts with a higher success rate, probably because the longer input texts leave more opportunities for Argot to add perturbations.

4.3 Generalizability

4.3.1 Performance on Pre-trained Language Models

The recent development of pre-trained language representation models, such as BERT [7], has led to a new stage of performance on NLP tasks. However, as one prominent representation of pre-trained models, BERT has been proved to be vulnerable to adversarial attacks on English text [24]. As BERT can generate fluent and semantically consistent substitutions for input text, researchers have also explored the possibility of attacking one fine-tuned BERT with another BERT [25], [26]. However, existing attacking works all focused on English text, and have not covered the Chinese corpus yet.

To fill this research gap, we applied Argot on BERT-based Chinese models under the black-box setting. Specifically, we choose base BERT ($\text{BERT}_{\text{base}}$), BERT trained with whole word masking strategy (BERT_{wwm}), BERT trained with extended data ($\text{BERT}_{\text{wwm/ext}}$), and Chinese RoBERTa ($\text{RoBERTa}_{\text{wwm/ext}}$) in the evaluation experiments.[7] As shown in Table 7, although the pre-trained models are more robust than CNN and LSTM, the improvement is quite limited. Argot still shows high attack success rates on all those BERT-based models. Note that, we also use Argot to attack ChineseBERT, which incorporates Chinese phonetic and glyph features to learn character representations, and has achieved the SOTA performance of several Chinese NLP tasks. The results show that ChineseBERT has better robustness versus other models, thus Argot needs to add more perturbations for successful attacks. However, it is still vulnerable. An in-depth analysis of the robustness of BERT and effective defenses for pre-trained models could be explored in future work.

[7] Sources of models: https://huggingface.co

TABLE 6: Two example of adversarial texts generated by Argot.

Model & Label	Text	Description
Model: char-based CNN Label: Positive->Negative	很(狠)不错呀，已经用了半个月了 Very good, it has been used for half a month	Glyph:很(very)->狠(ruthless)
Model: word-based CNN Label: Education->Affairs	2010年1月各地自考成绩查询汇总,2010年(碾)1月自考成绩已陆续公布.....查询信息不断更新中！ Summary of self-examination results in various places in January 2010 Self-exam results in January 2010 have been announced one after another....The query information is constantly updated!	Pinyin:年(nian)(year)->碾(nian)(roller)

TABLE 7: Performance of Argot on BERT-based models (S: success rate, P: perturbation, T: Time/Char).

Model	Accuracy	S	P	T
BERT$_{\text{base}}$	94.71%	81.47%	0.37	0.34
BERT$_{\text{wwm}}$	94.55%	81.16%	0.32	0.27
BERT$_{\text{wwm/ext}}$	94.81%	80.08%	0.33	0.33
RoBERTa$_{\text{wwm/ext}}$	94.73%	78.10%	0.39	0.38
ChineseBERT	94.87%	76.99%	0.39	0.53

4.3.2 *Performance on Real-World Applications*

In addition to proving the fragility of the NLP model itself, Argot also has certain real-world utility. Take text-based toxic content detection as an example. In the underground economy, text (e.g., spam short messages, spam emails) is one of the main methods to promote illegal content. To evade text-based toxic content detection systems, cyber-criminals would also try to generate changes to the original spam text, bypassing the detection through some sort of "adversarial methods". In this scenario, the criminals know few details about the detection system, such as what NLP model is used and what the training dataset is. This scenario is similar to our black-box attack. Considering the complexity of the real-world, although Argot performs well on experimental models, we further test its performance against a real-world text-based toxic detection system, as an example to comprehensively evaluate its usability.

The chosen attack target is the online text-based toxic content detection system of Baidu[8] , which is among the most popular Internet security companies in China. These two systems both provide the ability to detect irregular texts, e.g., **Malicious Promotion**, Pornography and Political Content. We select spam SMS (Short Message Service) as the attacking text and collect a real-world spam SMS dataset from [27].

Then Argot generates 300 adversarial spam SMS texts based on this dataset. We apply the generated examples to the online detection system and find 30% of them can evade the detection. Figure 12 shows a result of one original spam SMS against Baidu's detection system, which is detected as **Malicious Promotion** with 100% confidence. With the perturbation after Argot, the adversarial SMS message is detected as normal content, as shown in Figure 13. It is worth noting that Argot utilizes no information of the target detection system, even has not queried it during the generating process. In essence, this experiment embodies the transferability of Argot. Although this success rate is not high enough, we take the first step to evaluate the transferability of adversarial texts in the real world. Besides, we find existing defense methods could not work well against Argot's attack. For instance, Huang's work [28] detects obfuscated content by matching red-flagged terms. However, the obfuscated terms chosen by Argot are based on the semantic contribution inside sentences, rather than the red-flagged terms. In addition, considering the huge volume of spam SMS in the real-world (∼40 million per day in China [29]), even a 30% success rate could still induce a significant impact on the effectiveness of the detection system. We will further explore the transferability of Argot in our future work.

In conclusion, Argot can work in real-world applications and seriously affects the performance of a state-of-the-art text-based detection system.

5 Discussion

5.1 White-box and Black-box Attack

From Section 4, we can see that white-box attacks can get a higher success rate and less perturbation than black-box attacks, which is normal in adversarial examples.

In the white-box setting, adversaries will get all the information of the target model, including the structure, parameters, training dataset, etc. Therefore, adversaries can perform accurate gradient calculations to find which word is important for model decisions. While in the black-box setting, adversaries can only query the target model and get the output confidence. Argot tries to find the important words by deleting words one by one and observing the confidence drop. Although we have proved that this method is also effective, it still requires many queries and the results are not as accurate as under the white-box setting. Obviously, the number of queries is very proportional to the length of the text. Once the text is very long, this process consumes much time while under white-box settings attackers only need one calculation to get the gradient vectors of all the words.

Although white-box attacks are better than black-box attacks from Argot's results, in reality, it is difficult for an attacker to obtain the knowledge of the model. Most of the time, the adversary can only query the trained model and get the confidence. Therefore, we also propose the black-box attack. Although the result of the black-box attack is not as good as the white-box attack, it can also achieve a relatively high success rate within acceptable perturbation. In addition, there are some more strict scenarios where the adversary has limited queries on the model, or can only get the label instead of confidence. For example, with the popularity of MLaaS (Machine Learning as a Service), some enterprises provide APIs that only output the corresponding labels. We only test one online system in Section 4.3.2, and we leave the systematic evaluation of all online NLP models in our future work.

5.2 Embedding Base

We find the base of embedding would affect the robustness of Chinese NLP models, which could be summarized from the comparison of evaluation results between word-based models and char-based models (see Table 2, 3, 4and 5). Targeted attack towards word-based models usually results in a higher success rate in a shorter time while introducing less perturbation than char-based models. It suggests that char-based Chinese NLP models,

[8] https://ai.baidu.com/tech/textcensoring

in general, are more robust than word-based models against adversarial perturbations.

We speculate the root reason may be that the char-based model has fewer OOV (Out-of-Vocabulary) problems when faced with adversarial text, which helps it keep more semantics of the original text. As mentioned in Section 2, due to the lack of nature separators between words, word segmentation is a key part of the Chinese NLP procedure. Adversarial attacks will introduce more serious OOV (Out-of-Vocabulary) problems in Chinese by breaking the word segmentation. Similar insights regarding the impact of embedding base have also been confirmed in Meng's work [30].

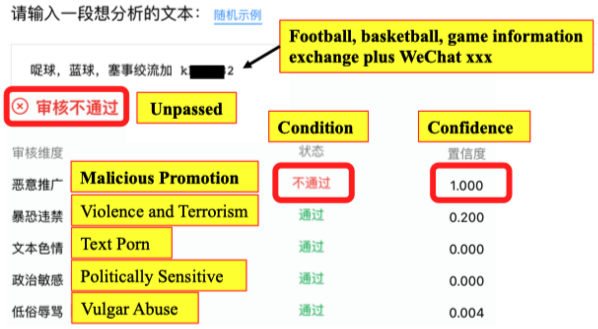
In addition to char-based and word-based models, several new models use byte-level embedding methods (e.g., RoBERTa [31]), which tokenizes texts in the base of a byte. The new byte-embedding method has been proven to provide more robust privacy protection [32]. However, it is not clear whether it could also help improve the robustness against adversarial attacks. We will explore and evaluate the robustness against adversarial attacks on such new embedding methods in our future work.

5.3 Defenses

The perturbation of Argot draws on the unique characteristics of Chinese, *Pinyin* and *Glyph*. While in addition to the semantic information, the NLP model under attack is missing prior knowledge of both the unique features. Several previous works have shown that adding the embedding of *Pinyin* or *Glyph* helps improve the performance of NLP models. For example, Zhu [33] added the *Pinyin* embedding based on semantic embedding to enhance the robustness of Chinese NLP models. After collecting a large number of Chinese character graphical representations in different fonts, Wu [9] combined the embedding information of *glyphs* with semantics, and achieved better results in several NLP tasks on Chinese models. Besides, Sun [34] enhanced the robustness of the pre-trained BERT model with additional pinyin and glyph information.

Inspired by the existing works, we explore the feasibility of combining *Pinyin* and *Glyph* embedding for defending Argot. We choose the char-based CNN model as a typical representative to evaluate the above defenses in a black-box and non-targeted attack setting. However, the results show that, although we have taken advantage of state-of-the-art techniques in the field, their defense performance against Argot is quite limited. Compared with results in Section 4.2.1, *Pinyin* defense just lowers the success rate of attacks from 70.1% to 65.4% on average when Argot only uses *Pinyin* perturbation, while *Glyph* defense lowers the success rate from 80.1% to 77.7% on average when only using *Glyph* perturbation. Besides, as discussed in Section 4.3.1, Argot remains effective even for ChineseBERT, which integrates pinyin- and glyph- embedding into Chinese pre-trained language models. Furthermore, Adversarial training [35] is also by far one of the most general defense methods. However, given its remarkable limitations, including the expensive time cost for adversarial text generations and limited effectiveness against unknown attacks [24], experimental evaluations of Argot on adversarial training models were not considered in this work.

In summary, integrating *Pinyin* and *Glyph* features of Chinese texts indeed improves NLP models' safety to some extent. However, the success rates of our attacks are still high. It highlights the excellent performance of Argot, and also indicates the need to explore better defenses against it, which will be part of our future work.



请输入一段想分析的文本： 随机示例

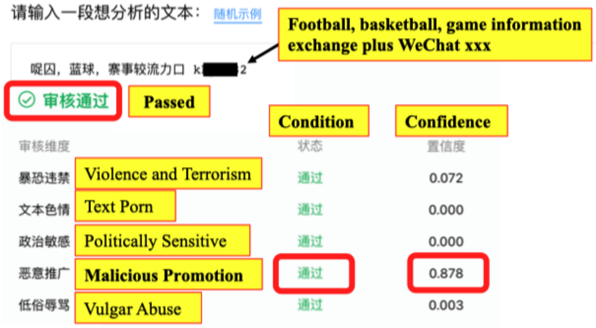
呢球，蓝球，塞事绞流加 k[redacted]2

Football, basketball, game information exchange plus WeChat xxx

ⓧ 审核不通过 Unpassed

审核维度		状态 Condition	置信度 Confidence
恶意推广	Malicious Promotion	不通过	1.000
暴恐违禁	Violence and Terrorism	通过	0.200
文本色情	Text Porn	通过	0.000
政治敏感	Politically Sensitive	通过	0.000
低俗辱骂	Vulgar Abuse	通过	0.004

Fig. 12: The result of the text-based toxic content detection system on the original text.



请输入一段想分析的文本： 随机示例

呢囚，蓝球，赛事较流力口 k[redacted]2

Football, basketball, game information exchange plus WeChat xxx

✓ 审核通过 Passed

审核维度		状态 Condition	置信度 Confidence
暴恐违禁	Violence and Terrorism	通过	0.072
文本色情	Text Porn	通过	0.000
政治敏感	Politically Sensitive	通过	0.000
恶意推广	Malicious Promotion	通过	0.878
低俗辱骂	Vulgar Abuse	通过	0.003

Fig. 13: The result of the text-based toxic content detection system on the adversarial text.

6 Related Work

Adversarial texts have been studied a lot in the literature. However, these methods mainly focus on generating English adversarial texts. These solutions can be categorized into white-box and black-box settings.

6.1 White-box Setting

Attackers mainly use methods similar to adversarial examples on images to find the key parts of the text. For example, they use gradient computed by the model to assess each word's impacts and add perturbations to important words to generate adversarial English texts.

Earlier, Papernot [36] used a gradient-based method to generate adversarial text on RNN that could spoof the text classification model, but the generated text was very hard to understand. In 2000 review sentences with an average length of 71.06 words, changing 9.18 words on average can make the neural network 100% wrong. This is the first work to apply the concept of adversarial examples to the text domain, but it does not consider the readability of adversarial texts, and the perturbation is relatively high.

Samanta [37] used the Fast Gradient Sign Method (FGSM) [38] to measure the importance of each word. They established a word-level candidate pool by considering synonyms, typos and genre specific keywords. Then they add word-level perturbations such as adding, deleting and replacing the corresponding words. By analyzing the dataset, they summarized some natural language rules, such as part of speech, and added the part of speech characteristics of the text to the process of generating adversarial text to make the texts more readable.

Gong [39] chose to use FGSM [38] and Deepfool [40] methods to add perturbation on the vectors directly. Second, they used the nearest neighbor method to find the nearest word in the vector space and substituted the original one. However, this method may lead to a large semantic difference. Based on the model using one-hot input representation, Ebrahimi [6] calculated the effect of replacing one letter with another on the loss function. It could also extend to increase (right characters are shifted to the right) and delete (left characters are shifted on the right) a letter. The three methods, addition, deletion, and modification can be used in combination. Ren [41] proposes Probability Weighted Word Saliency (PWWS) attack by calculating the word saliency and the classification probability, to determine the word replacement order.

6.2 Black-box Setting

This setting assumes the adversary can only access model's input and output, rather than internal knowledge of target models or gradients.

The TS (temporal score) and TTS (temporal tail score) methods [42] were used to evaluate the importance of each word. TS compares the difference between the outputs of the first i^{th} and $i-1^{th}$ words as the score of the i_{th} word. However, the result is based on the preceding words. Then, they propose TTS as a supplement. TTS compares the outputs of the two trailing parts of a sentence, one containing the words from the i^{th} word to the n^{th}, the other containing the words from the $i+1^{th}$ word to the n^{th}. TS score and TTS score are given different weights and can be combined as the final score for each word, and then the letter-level disturbance was added on the text. But the base of perturbation is the letter which is not a concept in Chinese, the corresponding concept in China is radical. In our work, we propose the method of the glyph and splitting characters in Chinese to generate adversarial text with similar glyphs.

Kuleshov [43] tried to find the most similar word in the vector space for all the words in text, and chose the replacement with the highest confidence in the model at every iteration. Alzantot [44] used the nearest neighbor method to find the nearest N words of the current word in the vector space, and used the counter-fitting method to check if they were similar semantically. Then they checked whether it conforms to the locale and chose the best one from the candidates to replace the original word. The whole process is optimized by a genetic algorithm and the initial selection of words is random.

The methods used by [4] and [45] were similar, they can finish attacks in both black-box and white-box settings. In white-box setting, they use gradient-based methods. In the black-box setting, they deleted or hid words one by one to calculate the decrease of the confidence and evaluate the importance of words. From the user study, Liang's work [45] can generate adversarial texts with good readability, but it needs lots of human effort and it could not generate adversarial texts automatically. However, these approaches are not applicable to Chinese texts, as the concept of a Chinese word is actually a phrase, which is much more ever-changing.

Furthermore, the recently prevalent pre-trained language models, especially BERT and its variants, have also been proved vulnerable to adversarial attacks [24]. Garg [25] and Li [26] both have proposed to attack a fine-tuned BERT with another BERT by semantic-consistent substitutions under black-box settings.

6.3 Comparison

Most existing schemes for generating adversarial texts, both for white-box and black-box scenarios, target English only. No prior knowledge has been obtained that whether those attacks could also be applied to Chinese. To fill this gap, we provided the first systematic analysis of the differences between English and Chinese languages. We uncovered several special composition rules and unique features of Chinese text (e.g., visual and sonic features), and further demonstrated that proposed attacking methods are not directly applicable.

Based on the unique characteristics of Chinese, we proposed a novel and generic solution to generate adversarial texts, not only at phrase-level, but also at character and radical-level. As the most comparable work, Cheng [46] also proposes three modification strategies on Chinese texts for non-target attacks under black-box settings. However, we take a far more comprehensive attacking view with 5 perturbation methods and broadly extend the attacking scenarios. Specifically, we implement comprehensive experiments to evaluate the effectiveness of Argot, confirming its good performance in generating both targeted and non-targeted Chinese adversarial texts in both white-box and black-box settings. We also verified that Argot is robust to defense methods.

To conclude, our work proposes the first comprehensive approach for Chinese adversarial text generation, together with a systematical evaluation of its performance.

7 Conclusion

In this study, we analyze the unique characteristic of Chinese compared to English and the limitation of existing adversarial text generation solutions for English. Based on Chinese characteristics, we propose an adversarial Chinese text generation solution Argot for two attack scenarios, i.e., targeted attack and non-targeted attack in both white-box and black-box settings. In Argot, we use five methods pinyin, glyph, splitting-characters, shuffle, and synonyms to add perturbations. We evaluate the three metrics (success rate, perturbation and efficiency) of generated texts of Argot. The results show that Argot could generate adversarial texts with a high success rate and relatively small perturbations. Meanwhile, the generated texts can maintain good readability. According to user studies, most volunteers can understand the meaning and get correct labels of adversarial texts.

Acknowledgements

This work was supported by National key R&D Program of China (No.2018YFB2101501), National Natural Science Foundation of China (under Grant U19B2034, U1636204, U1836213, 62132011 and 61972224), the Huawei Technologies Co., Ltd under Grant No.TC20200917004, Industrial Internet Innovation and Development Project, (under TC200H02X and TC200H02Y).

References

[1] Z. Zhang, M. Liu, C. Zhang, Y. Zhang, Z. Li, Q. Li, H. Duan, and D. Sun, "Argot: Generating adversarial readable chinese texts," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 2533–2539.

[2] K. Ravi and V. Ravi, "A survey on opinion mining and sentiment analysis: tasks, approaches and applications," *Knowledge-Based Systems*, vol. 89, pp. 14–46, 2015.

[3] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," in *Proceedings of the 25th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2016, pp. 145–153.

[4] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," *arXiv preprint arXiv:1812.05271*, 2018.

[5] K. Yuan, D. Tang, X. Liao, X. Wang, X. Feng, Y. Chen, M. Sun, H. Lu, and K. Zhang, "Stealthy porn: Understanding real-world adversarial images for illicit online promotion," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 952–966.

[6] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[8] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[9] W. Wu, Y. Meng, Q. Han, M. Li, X. Li, J. Mei, P. Nie, X. Sun, and J. Li, "Glyce: Glyph-vectors for chinese character representations," *arXiv preprint arXiv:1901.10125*, 2019.

[10] G. Cloud., "Google cloud nlp api services." https://cloud.google.com/learn/what-is-natural-language-processing, Accessed March, 2022.

[11] M. Azure., "Microsoft azure text analytics services." https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/#customer-stories, Accessed March, 2022.

[12] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.

[13] C. Hiruncharoenvate, Z. Lin, and E. Gilbert, "Algorithmically bypassing censorship on sina weibo with nondeterministic homophone substitutions," in *Ninth International AAAI Conference on Web and Social Media*, 2015.

[14] Z. Chen and K.-F. Lee, "A new statistical approach to chinese pinyin input," in *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, 2000, pp. 241–247.

[15] S. Chopra, R. Hadsell, Y. LeCun *et al.*, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR (1)*, 2005, pp. 539–546.

[16] M. Sun, X. Chen, K. Zhang, Z. Guo, and Z. Liu, "Thulac: An efficient lexical analyzer for chinese," Technical Report. Technical Report, Tech. Rep., 2016.

[17] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[18] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[19] S. Li, Z. Zhao, R. Hu, W. Li, T. Liu, and X. Du, "Analogical reasoning on chinese morphological and semantic relations," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, 2018, pp. 138–143. [Online]. Available: http://aclweb.org/anthology/P18-2023

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[21] L. Tiantian. (2015) Pinyin2hanzi. [Online]. Available: https://github.com/letiantian/Pinyin2Hanzi

[22] H. Y. X. Hai Liang Wang. (2018) python-pinyin. [Online]. Available: https://github.com/mozillazg/python-pinyin

[23] ——. (2017) Synonyms. [Online]. Available: https://github.com/huyingxi/Synonyms

[24] 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 *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 8018–8025.

[25] S. Garg and G. Ramakrishnan, "Bae: Bert-based adversarial examples for text classification," *arXiv preprint arXiv:2004.01970*, 2020.

[26] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "Bert-attack: Adversarial attack against bert using bert," *arXiv preprint arXiv:2004.09984*, 2020.

[27] Y. Zhang, B. Liu, C. Lu, Z. Li, H. Duan, S. Hao, M. Liu, Y. Liu, D. Wang, and Q. Li, "Lies in the air: Characterizing fake-base-station spam ecosystem in china," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[28] L. Huang, T. Ma, J. Lin, J. Han, and S. Hu, "A multimodal text matching model for obfuscated language identification in adversarial communication?" in *The World Wide Web Conference*, 2019, pp. 2844–2850.

[29] C. A. o. I. 360 mobile guard, 360 Enterprise Security and C. Technology. (2020) Report on china's mobile phone security in the first half of 2020. [Online]. Available: https://zt.360.cn/1101061855.php?dtid=1101061451&did=610637885

[30] X. Li, Y. Meng, X. Sun, Q. Han, A. Yuan, and J. Li, "Is word segmentation necessary for deep learning of chinese representations?" *arXiv preprint arXiv:1905.05526*, 2019.

[31] 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 preprint arXiv:1907.11692*, 2019.

[32] X. Pan, M. Zhang, S. Ji, and M. Yang, "Privacy risks of general-purpose language models," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1314–1331.

[33] W. Zhu, X. Jin, J. Ni, B. Wei, and Z. Lu, "Improve word embedding using both writing and pronunciation," *PloS one*, vol. 13, no. 12, p. e0208785, 2018.

[34] Z. Sun, X. Li, X. Sun, Y. Meng, X. Ao, Q. He, F. Wu, and J. Li, "Chinesebert: Chinese pretraining enhanced by glyph and pinyin information," *arXiv preprint arXiv:2106.16038*, 2021.

[35] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" *arXiv preprint arXiv:1904.12843*, 2019.

[36] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *MILCOM 2016-2016 IEEE Military Communications Conference*. IEEE, 2016, pp. 49–54.

[37] S. Samanta and S. Mehta, "Towards crafting text adversarial samples," *arXiv preprint arXiv:1707.02812*, 2017.

[38] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[39] Z. Gong, W. Wang, B. Li, D. Song, and W.-S. Ku, "Adversarial texts with gradient methods," *arXiv preprint arXiv:1801.07175*, 2018.

[40] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[41] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019, pp. 1085–1097.

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

[43] V. Kuleshov, S. Thakoor, T. Lau, and S. Ermon, "Adversarial examples for natural language classification problems," 2018.

[44] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," *arXiv preprint arXiv:1804.07998*, 2018.

[45] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, "Deep text classification can be fooled," *arXiv preprint arXiv:1704.08006*, 2017.

[46] C. Nuo, G.-Q. Chang, H. Gao, G. Pei, and Y. Zhang, "Wordchange: Adversarial examples generation approach for chinese text classification," *IEEE Access*, vol. 8, pp. 79 561–79 572, 2020.

Mingxuan Liu received the B.E. degree from Beijing University of Posts and Telecommunications, China, in 2018. She is now a Ph.D. student in the Institute for Network Science and Cyberspace, Tsinghua University. Her research interests include AI security, Data Driven Security and Network Security.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3164289, IEEE Transactions on Dependable and Secure Computing

Zihan Zhang received the B.E. degree from Wuhan University, Wuhan, China, in 2019. She is working towards the Master degree in the Institute for Network Science and Cyberspace in Tsinghua University. Her research interests include AI security and malware analysis.

Yiming Zhang received the BSc degree from Tsinghua University in 2017. She is currently a Ph.D. student with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. Her research interests include Network Security and Data Driven Security.

Chao Zhang is an Associate Professor at Tsinghua University. He received his B.S and Ph.D. degrees from Peking University, and did postdoc research at UC Berkeley. His research interest lies in system security, including AI for security and security for AI.

Zhou Li received the Ph.D. degree in Computer Science from Indiana University Bloomington. He was a Principal Research Scientist at RSA Labs from 2014-2018. He is currently an Assistant Professor at EECS Department of University of California, Irvine, with research interests include cyber security, privacy and machine learning. He is an NSF CAREER awardee in 2021.

Qi Li received his Ph.D. degree from Tsinghua University. Now he is an associate professor of Institute for Network Science and Cyberspace, Tsinghua University. He has ever worked at ETH Zurich and the University of Texas at San Antonio. His research interests are in network and system security, particularly in Internet and cloud security, mobile security, and big data security. He is currently an editorial board member of IEEE TDSC.

Haixin Duan graduated from Tsinghua University with a Ph.D degree on computer science, and then became a faculty member in Tsinghua University. He has been a visiting scholar in UC Berkeley and senior scientist in International Computer Science Institute(ICSI). Prof. Duan focuses his research on network security, including intrusion detection, security of network protocols(DNS, Web, HTTP and HTTPS). Most of his papers are published in the top security conferences (S&P, USENIX Security, CCS and NDSS) and had got several best paper awards of CCS, NDSS and other top security conferences.

Donghong Sun is an Associate Professor at Tsinghua University. Her research interest lies in Cyberspace security, including network detection and security assessment ,privacy computing ,etc.