TextShield: Robust Text Classification Based on Multimodal Embedding and Neural Machine Translation

Jinfeng Li, *Zhejiang University, Alibaba Group;* Tianyu Du, *Zhejiang University;* Shouling Ji, *Zhejiang University, Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies;* Rong Zhang and Quan Lu, *Alibaba Group;* Min Yang, *Fudan University;* Ting Wang, *Pennsylvania State University*

https://www.usenix.org/conference/usenixsecurity20/presentation/li-jinfeng

This paper is included in the Proceedings of the 29th USENIX Security Symposium.

August 12–14, 2020

978-1-939133-17-5

Open access to the Proceedings of the 29th USENIX Security Symposium is sponsored by USENIX.

TEXTSHIELD: Robust Text Classification Based on Multimodal Embedding and Neural Machine Translation

Jinfeng Li[†,¶], Tianyu Du[†], Shouling Ji[†,+,(✉)], Rong Zhang[¶], Quan Lu[¶], Min Yang[§], and Ting Wang[‡]

[†]Zhejiang University, [¶]Alibaba Group, [+]Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies, [§]Fudan University, [‡]Pennsylvania State University

E-mails: lijinfeng0713@zju.edu.cn, zjradty@zju.edu.cn, sji@zju.edu.cn, stone.zhangr@alibaba-inc.com, luquan.lq@alibaba-inc.com, m_yang@fudan.edu.cn, inbox.ting@gmail.com.

Abstract

Text-based toxic content detection is an important tool for reducing harmful interactions in online social media environments. Yet, its underlying mechanism, deep learning-based text classification (DLTC), is inherently vulnerable to maliciously crafted adversarial texts. To mitigate such vulnerabilities, intensive research has been conducted on strengthening English-based DLTC models. However, the existing defenses are not effective for Chinese-based DLTC models, due to the unique sparseness, diversity, and variation of the Chinese language.

In this paper, we bridge this striking gap by presenting TEXTSHIELD, a new adversarial defense framework specifically designed for Chinese-based DLTC models. TEXTSHIELD differs from previous work in several key aspects: (i) *generic* – it applies to *any* Chinese-based DLTC models without requiring re-training; (ii) *robust* – it significantly reduces the attack success rate even under the setting of adaptive attacks; and (iii) *accurate* – it has little impact on the performance of DLTC models over legitimate inputs. Extensive evaluations show that it outperforms both existing methods and the industry-leading platforms. Future work will explore its applicability in broader practical tasks.

1 Introduction

In this era of social networking, online social networks have become a de facto portal for hundreds of millions of Internet users (netizens) [3]. However, of the vast user generated text content produced everyday, a significant portion is toxic (e.g., abusive, pornographic and violent content), which represents an immense threat to the physical and mental health of netizens, especially young ones. It was reported that major social media platforms (e.g., Twitter and Facebook) were all criticized for not doing enough to curb the diffusion of toxic content and under pressure to cleanse their platforms [31]. Yet, the sheer amount and rampant growth of toxic content represent non-trigger challenges facing such effort.

To this end, automated techniques, especially deep learning-based text classification (DLTC), have been applied to online toxic content detection. Thanks to the state-of-the-art performance of deep neural network (DNN) models, DLTC-based toxic content detection significantly outperforms the time-consuming and laborious manual censorship in terms of both efficiency and effectiveness [18, 22, 34]. However, recent studies have revealed that existing DLTC models are inherently vulnerable to adversarially generated texts [9, 11, 26, 37], which are maliciously crafted texts that trigger DLTC models to misbehave. In the context of toxic content detection, the adversary may generate texts that remain toxic but evade the detection of DLTC models. As a concrete example, to make insulting comments evasive, the adversary may obfuscate some words with their variants, such as substituting "idiot" with "idi0t". These variants are visually similar to their original words (i.e., remaining toxic) but are able to effectively evade the detection. Such adversarial texts can be crafted under either white-box [9, 37] or black-box [11, 26] setting. In general, the white-box attacks aim to generate adversarial texts with the guidance of complete knowledge (e.g., architectures and parameters) about the target model. The black-box attacks generate adversarial texts by estimating the gradient or exploring model sensitivity based on the classification confidence when detailed model information is not available.

To defend against such attacks, countermeasures such as *adversarial training* [13, 43, 44] and *spelling correction* [26, 46, 47] have been proposed to enhance the robustness of English-based DLTC models, which have achieved considerable success. In comparison, the effort of improving the robustness of Chinese-based DLTC models is still fairly limited. Even worse, the existing defenses that are effective for English-based DLTC models are often inapplicable to Chinese-based models due to the following reasons: (i) unlike English which has a relatively small alphabet, Chinese is *logographic* with a large set of characters that are individually meaningful and the modification of a single character may

Jinfeng Li and Tianyu Du are the co-first authors. Shouling Ji is the corresponding author.

drastically alter the semantics of the text, making Chinese-based DLTC models inherently more vulnerable; (ii) it is fundamentally more challenging to perform spelling correction in Chinese since there is no word delimiter in Chinese written texts while variant characters can only be determined at the word-level; and (iii) the model retrained with adversarial training is still likely to be sensitive to new attacks due to the sparseness and diversity of Chinese adversarial perturbations. Concretely, there are more than 50,000 characters[1] might be perturbed by various variation strategies such as glyph-based and phonetic-based strategies (more detailed examples about these variations can be seen in Section 4.3). Given the scale of Chinese-based social media platforms (e.g., WeChat enjoys one billion daily active users [17]), the lack of robust toxic content detection represents an immense concern.

Our Work. To bridge this striking gap, in this paper, we present TEXTSHIELD, a novel adversarial defense framework for Chinese-based DLTC systems based on multimodal embedding and neural machine translation (NMT) [2]. At a high level, TEXTSHIELD performs robust toxic text detection in three phases. First, each text input is corrected by an adversarial NMT model for denoising some of the adversarial perturbations; second, the corrected text is converted to multimodal embedding, which extracts its semantic, glyph and phonetic features for dealing with the glyph-based and phonetic-based perturbations; finally, the extracted features are fused to form a semantic-rich representation, which is ready for the regular toxic classification. Through intensive empirical evaluations on two real-world datasets collected from Chinese online social media (e.g., Sina Weibo), we show that TEXTSHIELD is effective in defending against both the obfuscated texts generated by the adversary and the adversarial texts generated by the state-of-the-art attacks. It also outperforms four industry-leading online toxic content detection platforms including Alibaba GreenNet, Baidu TextCensoring, Netease Yidun and Huawei Moderation. We are currently in the process of integrating TEXTSHIELD with Alibaba GreenNet to enhance its robustness.

The main contributions of this paper can be summarized as follows.

- We propose TEXTSHIELD, which to our best knowledge is the first adversarial defense specialized for Chinese DLTC tasks without retraining the model, in which a novel multimodal embedding scheme is proposed to enhance the robustness of DLTC models and an adversarial NMT is first applied to reconstruct the original texts.
- We evaluate the *effectiveness* of TEXTSHIELD in real-world adversarial scenarios. The evaluation results show that TEXTSHIELD attains high accuracy (e.g., 0.944 for porn detection) on the malicious user generated obfuscated texts while having little impact on the model performance (e.g., the accuracy degrades by less than 2%) over benign inputs.
- We verify the *robustness* of TEXTSHIELD under the setting of adaptive attacks in two real-world tasks and compare it with four industry-leading platforms, which shows that TEXTSHIELD is of great practicability and superiority in decreasing the attack success rate (e.g., the attack success rate against abuse detection is degraded by 74.5%).

[1] https://en.wikipedia.org/wiki/Chinese_characters

2 Related Work

2.1 Adversarial Text Generation

Adversarial attacks against DNNs are first explored in the context of image classification [13, 19, 27, 29, 39, 43, 49] and are then extended to the NLP domain. We here mainly focus on discussing the work related to generating adversarial texts.

In one of the first attempts at tricking DLTC systems, Papernot *et al.* [37] introduced a white-box attack for generating adversarial inputs by leveraging the computational graph unfolding technique. Ebrahimi *et al.* [9] showed that automatically swapping one token for another with the guidance of gradients can deceive the character-level DLTC models. Jia *et al.* [20] generated adversarial texts for evaluating reading comprehension systems by adding distracting sentences to the original text based on manually-defined rules. Hosseini *et al.* [15] showed that simple modifications, such as adding dots or spaces between characters, can drastically change the toxicity score of Google's Perspective API. Li *et al.* [26] proposed TextBugger, a state-of-the-art black-box attack that successfully compromised 15 real-world applications.

Unlike English adversarial texts, most of the Chinese adversarial texts are manually crafted by real-world malicious netizens, which are more diverse due to the various word variation strategies adopted by different netizens [21]. In addition, there is an extremely large character space in Chinese in which each character may be perturbed by various strategies, which makes the perturbations more sparse.

2.2 Defenses against Adversarial Text

To defend against the above attacks, several defenses have been proposed in the English NLP domain, including adversarial training [8, 20, 44] and spelling correction [11, 26].

Adversarial Training. It was first proposed in [43] to enhance the robustness of DNNs used for image classification by augmenting training data with adversarial images. Wang *et al.* [44] and Ebrahimi *et al.* [8] presented several initial attempts to tackle adversarial texts by retraining the models with diversified adversarial training data and showed a marginal increase in robustness. However, since there currently exists no automatic attack for generating Chinese adversarial texts while the manual collection of user generated obfuscated texts is often laborious and costly, it is not trivial to extend existing adversarial training to the Chinese NLP domain. More importantly, the sparseness of Chinese adversarial perturbations

may also weaken its efficacy.

Spelling Correction. In the English NLP domain, Gao *et al.* [11] and Li *et al.* [26] leveraged the context-aware spelling correction approach to block editorial adversarial attacks (e.g., insertion, deletion and substitution) and achieved satisfactory performance. In the Chinese NLP domain, similar methods have also been tried to deal with user generated obfuscated texts, e.g., using dictionary-based [46] or language model-based [47] methods to restore the variant words to their benign format. However, compared to the alphabetical languages like English and French, it is more difficult to perform spelling correction in Chinese since there is no word boundary in Chinese writing texts while variant characters can only be determined at the word-level. Hence, it has been shown to have limited effect on model performance. Furthermore, the diversity, sparseness and dynamicity of Chinese adversarial perturbations may also challenge this approach.

3 Design of TEXTSHIELD

3.1 Problem Definition and Threat Model

Given a legitimate Chinese text $\boldsymbol{x} \in \mathcal{X}$ that contains N characters (i.e, $\boldsymbol{x} = \{x_1, x_2, \cdots, x_N\}$), and a DLTC model $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ which maps from the feature space $\mathcal{X}$ to the label space $\mathcal{Y}$, an attacker who has query access to the classification confidence returned by this model, aims to generate an adversarial text $\boldsymbol{x}_{adv}$ from $\boldsymbol{x}$ whose ground truth label is $y \in \mathcal{Y}$, such that $\mathcal{F}(\boldsymbol{x}_{adv}) = t(t \neq y)$.

In this paper, we aim to defend against such attacks by leveraging an NMT model which translates a source sequence into the target sequence to restore $\boldsymbol{x}_{adv}$, and universally improving the robustness of $\mathcal{F}$ by embedding the input from multi-modalities (e.g., semantics, glyphs and phonetics). Formally, our defense can be defined as

$$\mathcal{F}(E_{sgp}(\arg\max_{\boldsymbol{x}^* \in \mathcal{X}} p(\boldsymbol{x}^* | \boldsymbol{x}_{adv}; \theta))) = y, \tag{1}$$

where $E_{sgp}(\cdot)$ is the multimodal embedding function, $\boldsymbol{x}^*$ is a candidate text corrected from $\boldsymbol{x}_{adv}$, $p(\boldsymbol{x}^* | \boldsymbol{x}_{adv}; \theta)$ is the probability of outputting $\boldsymbol{x}^*$ given $\boldsymbol{x}_{adv}$, and θ is the parameters of the NMT model learned from an adversarial parallel corpora consisting of a plenty of aligned $(\boldsymbol{x}_{adv}, \boldsymbol{x}_{ori})$ sentence pairs.

3.2 Overview of TEXTSHIELD Framework

We present the framework overview of TEXTSHIELD in Fig. 1, which is built upon multimodal embedding, multimodal fusion and NMT. Generally, we first feed each text into an NMT model trained with a plenty of adversarial–benign text pairs for adversarial correction. Then, we input the corrected text into the DLTC model for multimodal embedding to extract features from semantic-level, glyph-level and phonetic-level. Finally, we use a multimodal fusion scheme to fuse

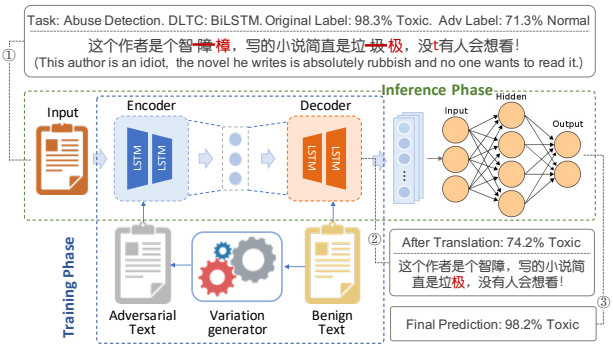


Figure 1: The framework of TEXTSHIELD.

the extracted features for the following regular classifications. Below, we will elaborate on each of the backbone techniques.

3.3 Adversarial Translation

We propose a novel adversarial corrector based on NMT and the framework is shown in Fig. 2. Generally, we first train an NMT model on a large adversarial parallel corpora for adversarial reconstruction. Then, we put it in front of the DLTC model based on multimodal embedding to restore the adversarial perturbations to their benign counterparts.

Model Design. We design the adversarial NMT model based on the *Encoder–Decoder* framework proposed in [6, 42], in which an encoder reads and encodes a source sequence $\boldsymbol{x} = (x_1, x_2, \cdots, x_N)$ into a fixed-length context vector $\boldsymbol{c}$ and a decoder decodes $\boldsymbol{c}$ and outputs a translation $\boldsymbol{x}^* = (x'_1, x'_2, \cdots, x'_{N'})$ by maximizing the ordered conditional probability

$$p(\boldsymbol{x}^* | \boldsymbol{x}) = \prod_{t=1}^{N} p(x'_t | x'_1, \cdots, x'_{t-1}, \boldsymbol{c}) = \prod_{t=1}^{N} g(x'_{t-1}, \boldsymbol{s}_t, \boldsymbol{c}), \tag{2}$$

where g is a nonlinear function that outputs the probability of x'_t, and $\boldsymbol{s}_t$ is the hidden state of the decoder at time t.

We use the long short-term memory (LSTM) network f with two layers to implement the encoder E and decoder D, and use Bahdanau's attention mechanism [2] to align $\boldsymbol{x}$ and $\boldsymbol{x}^*$. Moreover, we integrate a residual layer to learn the identity mapping since $\boldsymbol{x}$ and $\boldsymbol{x}^*$ only differ in few characters. Hence, the context vector $\boldsymbol{c}_i$ for each target character x'_i can be computed by the weighted sum of the hidden state $\boldsymbol{h}_j$ of E at each time j,

$$\begin{aligned} \boldsymbol{c}_i &= \sum_{j=1}^{N} \alpha_{ij} \cdot \boldsymbol{h}_j = \sum_{j=1}^{N} \frac{\exp(e_{ij})}{\sum_{k=1}^{N} \exp(e_{ik})} \cdot \boldsymbol{h}_j \\ e_{ij} &= a(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j) = \boldsymbol{v}_a^\top \cdot \tanh(\boldsymbol{W}_a \cdot \boldsymbol{s}_{i-1} + \boldsymbol{U}_a \boldsymbol{h}_j), \end{aligned} \tag{3}$$

where $\boldsymbol{v}_a$, $\boldsymbol{W}_a$ and $\boldsymbol{U}_a$ are the weight matrices of the additive alignment model. The hidden state $\boldsymbol{h}_j$ is calculated by $\boldsymbol{h}_j = f(x_j, \boldsymbol{h}_{j-1})$ and $\boldsymbol{s}_i$ is calculated by $\boldsymbol{s}_i = f(x'_{i-1}, \boldsymbol{s}_{i-1}, \boldsymbol{c}_i)$. Then,

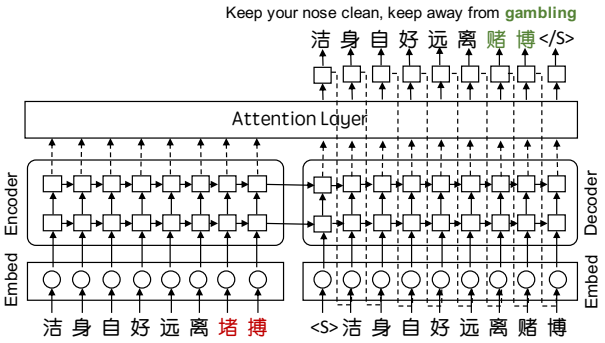


Figure 2: Architecture of the adversarial NMT model.

Eq. (2) can be rewritten as

$$p(\boldsymbol{x}^*|\boldsymbol{x}) = \prod_{i=1}^{N'} p(x'_i|x'_1,\cdots,x'_{i-1},\boldsymbol{c}) = \prod_{i=1}^{N'} g(x'_{i-1},\boldsymbol{s}_i,\boldsymbol{c}_i), \quad (4)$$

and the target character x'_i generated at each time i is sampled from the candidates by maximizing the conditional probability $g(x'_{i-1},\boldsymbol{s}_i,\boldsymbol{c}_i)$. Note that $\boldsymbol{x}$ and $\boldsymbol{x}^*$ are both Chinese, and we hence share the embedding vocabulary of E with D to reduce the amount of parameters. The designed NMT is finally implemented based on the TensorFlow NMT tutorial [30].

Model Training. In the training phase, we first construct a large adversarial parallel corpora $\mathcal{D}_{adv}$ by generating a plenty of $(\boldsymbol{x}_{adv},\boldsymbol{x}_{ori})$ sentence pairs through adversarial attacks. Then, the designed NMT model is trained on $\mathcal{D}_{adv}$ to learn the process of adversarial correction from adversarial texts to the benign texts by minimizing the negative log probability of a correct translation $\boldsymbol{x}_{ori}$ given the source sequence $\boldsymbol{x}_{adv}$. Formally, the training objective is defined as

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{D}_{adv}|} \sum_{(\boldsymbol{x}_{adv},\boldsymbol{x}_{ori})\in\mathcal{D}_{adv}} \log p(\boldsymbol{x}_{ori}|\boldsymbol{x}_{adv}). \quad (5)$$

To avoid the error being amplified step by step during the training process as well as improving the training stability to accelerate the convergence, we apply the *teacher forcing* [45] technique to train the NMT model by using the ground truth from a prior time step as the current input of the decoder D.

Adversarial Correction. Once the training is completed, the NMT model is then used as an adversarial text corrector to reconstruct the original text through translation. Formally, the corrected text $\boldsymbol{x}^*_{opt}$ is produced by finding an optimal translation that maximizes the conditional probability, i.e.,

$$\boldsymbol{x}^*_{opt} = \arg\max_{\boldsymbol{x}^*\in\mathcal{X}} p(\boldsymbol{x}^*|\boldsymbol{x}_{adv};\theta). \quad (6)$$

To improve the performance of the NMT model, we apply beam-search [14] in the decoding phase to search for the optimal translation. Finally, $\boldsymbol{x}^*_{opt}$ will be fed into $\mathcal{F}$ for multimodal embedding and then for the conventional classification.

3.4 Multimodal Embedding

Since the variation strategies adopted by malicious users in the real scenarios are mainly concentrated on glyph-based and phonetic-based perturbations [47], we therefore dedicatedly propose three embedding methods across different modalities to handle the corresponding variation types, i.e., semantic embedding, glyph embedding and phonetic embedding. They are also dedicatedly designed to deal with the sparseness and diversity unique to Chinese adversarial perturbations.

Semantic Embedding. We apply the skip-gram model proposed in [32] to learn continuous semantic word vectors. Note that we concentrate on character-level embedding since word-level embedding suffers the most from the out-of-vocabulary (OOV) phenomena, thus weakening the robustness of DLTC models. Specifically, the skip-gram model maps each character in vocabulary of size V to a continuous embedding space of d dimensions by looking up an embedding matrix $\boldsymbol{W}^{(1)}$, which is learned by maximizing the probability calculated by the matrix $\boldsymbol{W}^{(2)}$ of its neighbors within a context window. Formally, given a text contains N characters, i.e., $\boldsymbol{x} = \{x_1, x_2, \cdots, x_N\}$, the objective of the skip-gram model is to maximize the average log probability

$$Q = \frac{1}{N}\sum_{n=1}^{N} \sum_{-c\le j\le c, j\neq 0} \log p(x_{n+j}|x_n), \quad (7)$$

where c is the size of the context windows, x_n denotes the input central character and x_{n+j} for its neighboring character. The basic skip-gram formulation defines $p(x_{n+j}|x_n)$ using the following softmax function,

$$p(x_{n+j}|x_n) = \frac{\exp(\boldsymbol{w}^{(2)}_{n+j}\cdot\boldsymbol{w}^{(1)}_n)}{\sum_{k=1}^{V}\exp(\boldsymbol{w}^{(2)}_k\cdot\boldsymbol{w}^{(1)}_n)}, \quad (8)$$

where $\boldsymbol{w}^{(1)}_n$ and $\boldsymbol{w}^{(3)}_k$ denote row vectors in matrices $\boldsymbol{W}^{(1)}$ and $\boldsymbol{W}^{(2)}$, corresponding to character x_n and x_k respectively.

Glyph Embedding. In Chinese writing system, there are a large set of characters that are visually similar but have totally different meanings. This property has been exploited for crafting glyph-based perturbations, e.g., replacing "赌"(gamble) in "赌博" with a similar character "堵" (block). To improve the resilience of a DLTC model against such perturbations, we specially design a glyph embedding scheme to extract the glyph-based features of each character for capturing the similarity between the perturbed word and its benign counterpart.

To embed each character into a glyph embedding vector with the same dimension as its semantic embedding vector, we first convert it into an image of size $24\times24\times3$ by using a Python tool [2] dedicated to image processing. Second, we carefully design a simple convolutional neural network named g-CNN, which is modified from LeNet-5 [24] by replacing the fully connected layer and output layer with a convolution layer consisting of d filters of size 1×1 [28] and a global

[2] https://pypi.org/project/Pillow/

average pooling layer. Then, we integrate g-CNN as a glyph embedding layer into the DLTC model and train its parameters together with the whole DLTC model. Finally, we use the features extracted by the global average pooling layer of g-CNN as the glyph embedding vector.

Phonetic Embedding. Most existing DLTC models have only focused on the writing itself, while ignoring the fact that spoken language expresses the meaning directly. Unlike English whose pronunciation is tied to the alphabets, Chinese characters do not reflect the pronunciation and need to be annotated by *Hanyu Pinyin* [3]. In addition, Pinyin can also be directly used as a written language to express the meaning. Hence, similar to glyph embedding, we design a phonetic embedding scheme to extract phonetic-based features of Chinese characters for enhancing the performance a DLTC model as well as its robustness against the phonetic-based perturbations such as "涩情" or "se qing" which are mutated from the toxic word "色情" (porn) and have the same pronunciation.

For each character, we first use Pinyin to annotate its pronunciation, and non Chinese characters in the text are preserved. Then, we obtain a new sequence that contains N Pinyin forms for each text consisting of N characters. Finally, we apply the skip-gram model as used in semantic embedding to embed the Pinyin form of each character into a phonetic embedding vector of d dimensions.

3.5 Multimodal Fusion

Since multiple modalities can provide more valuable information than a single one by describing the same content in various ways, it is highly expected to learn effective joint representation by fusing the features of different modalities. Therefore, after multimodal embedding, we first fuse the features extracted from different modalities by multimodal fusion and then feed the fused features into a classification model for regular classification. In this paper, we experiment with two different fusion strategies, i.e., early multimodal fusion and intermediate multimodal fusion as shown in Fig. 10 in Appendix A.

Early Multimodal Fusion (EMF). EMF [35] refers to directly concatenating features from all the modalities and then employing multiple nonlinear transformations to generate the high-level joint representation. More formally, denote by $\boldsymbol{V}^{(S)}$ the semantic embedding vector, by $\boldsymbol{V}^{(G)}$ the glyph embedding vector and by $\boldsymbol{V}^{(P)}$ the phonetic embedding vector, the fused vector $\boldsymbol{V}$ is obtained by

$$\boldsymbol{V} = [\boldsymbol{V}^{(S)} \oplus \boldsymbol{V}^{(G)} \oplus \boldsymbol{V}^{(P)}]. \tag{9}$$

Obviously, it is an input-level fusion scheme, which is easy to capture the covariation between modalities, and other correlations existed at the input level. Meanwhile, it is the simplest to implement and requires less model parameters. However, it is also a coarse-grained fusion scheme that lacks the ability in capturing more complex correlation across modalities.

Intermediate Multimodal Fusion (IMF). The basic idea of IMF is to reduce the influence of individual differences and improve the shared semantic by building a joint feature representation based on the output of modality-specific networks [41]. Under this fusion scheme, the embedding vector from each modality is first fed into a unimodal backbone network, and then the outputs of the last hidden layers in all the unimodal backbones are concatenated for fusion. Hence, the fused vector $\boldsymbol{V}$ is obtained by

$$\boldsymbol{V} = [F_s(\boldsymbol{V}^{(S)}) \oplus F_g(\boldsymbol{V}^{(G)}) \oplus F_p(\boldsymbol{V}^{(P)})], \tag{10}$$

where $F_s(\cdot)$, $F_g(\cdot)$ and $F_p(\cdot)$ are the unimodal backbones specialized for semantics, glyphs and phonetics, respectively.

Classification. The vector $\boldsymbol{V}$ fused by EMF or IMF is then classified by

$$F(y=i|\boldsymbol{x}) = \frac{\exp(f(\boldsymbol{V})_i)}{\sum_{j=0}^{K}\exp(f(\boldsymbol{V})_j)}, \tag{11}$$

where $F(y=i|\boldsymbol{x})$ is the confidence of the i-th class, $f(\cdot)$ is the classification function of model $\mathcal{F}$ and K is the total number of classes. Note that the parameters of the backbones used for multimodal fusion are trained together with $\mathcal{F}$.

4 Experimental Setting and Implementation

4.1 Dataset

We evaluate TEXTSHIELD on three datasets of which two are used for toxic content detection and one is used for adversarial NMT. Each dataset is divided into three parts, i.e., 80%, 10%, 10% as training, validation and testing, respectively [26].

Toxic Content Detection. Since there currently does not exist a benchmark dataset for Chinese toxic content detection, we used two user generated content (UGC) datasets, i.e., Abusive UGC (Abuse) and Pornographic UGC (Porn) collected from online social media (the data collection details can be found in Appendix B). Each dataset contains 10,000 toxic and 10,000 normal samples that are well annotated by Chinese native speakers. The average text length of the Abuse and Porn datasets are 42.1 and 39.6 characters, respectively. The two datasets are used for building binary classification models for abuse detection and porn detection tasks.

Adversarial NMT. To increase the diversity of the adversarial parallel corpora and ensure that the NMT model can learn more language knowledge, we applied the Douban Movie Short Comments (DMSC) dataset released by Kaggle [4] along with Abuse and Porn. We then generate a corpora that consists of 2 million $(\boldsymbol{x}_{adv}, \boldsymbol{x}_{ori})$ sentence pairs for each task respectively, of which half is generated from DMSC and half is generated from the toxic datasets. The method used for generating sentence pairs is detailed in Section 4.3.

[3] Hanyu Pinyin is the official romanization system used for annotating the pronunciation of Standard Chinese.

[4] https://www.kaggle.com/utmhikari/doubanmovieshortcomments/

Table 1: Examples for six different kinds of bugs.

Bug	Example	Bug	Example
Insert	傻逼 → 傻&逼	Sim2Trad/1	裸体 → 裸體
PyConvert/1	智障 → zhi zhang	Sim2Trad/2	裸体 → 裸&體
PyConvert/2	智障 → zhi zha.ng	GlyphSim/1	赌博 → 堵博
PyConvert/3	智障 → zhi zhan	GlyphSim/2	赌博 → 堵搏
PyConvert/4	智障 → zhi zhnag	GlyphSim/3	赌博 → 堵t搏
PyConvert/5	智障 → Zhi zhang	PhoneticSim/1	色情 → 涩情
Split/1	炸弹 → 火乍弓单	PhoneticSim/2	色情 → 涩o情
Split/2	炸弹 → 炸弓/单		

4.2 Target Model

We implement a TextCNN [23] model and a BiLSTM [50] model as the backbone networks to design the target model since these two DNNs are most widely used in real-world text classification tasks [10,23,50]. In addition, the two models are often used in evaluating the efficacy of adversarial attacks and have been shown to be vulnerable to adversarial examples [11, 12, 26]. Based on the two backbones and combined with multimodal embedding and adversarial NMT, we totally implemented ten target models for abuse detection and porn detection, which are: Common TextCNN, Common BiLSTM, TextCNN + EMF, TextCNN + IMF, TextCNN + EMF + NMT, TextCNN + IMF + NMT, BiLSTM + EMF, BiLSTM + IMF, BiLSTM + EMF + NMT and BiLSTM + IMF + NMT.

Specifically, the common TextCNN and BiLSTM are built upon the TextCNN and BiLSTM backbones with no defense applied. "+ EMF" and "+IMF" represent that the input of the model is embedded by the multimodal embedding method and the extracted features are fused by EMF and IMF strategies, respectively. Similarly, "+EMF+NMT" and "+IMF+NMT" represent that the input text is first fed into the NMT model for adversarial correction and then processed by multimodal embedding, and finally fused by EMF and IMF, respectively.

4.3 Attack Method

In the real adversarial scenario, most of the Chinese adversarial texts are manually crafted by malicious netizens with black-box access to the models, which has posed severe threats to the real-world applications [16, 25, 48]. However, manual collection of these texts for evaluating the efficacy of TEXTSHIELD are usually laborious and costly. An intuitive idea is to mimic their attack behavior via adversarial attacks under the black-box setting. Since there is no proposed automatic black-box attack specialized for Chinese-based DLTC models in existing research, we then adopted TextBugger [26] as the attack method.

Recall that TextBugger first identifies the important word by sensitivity analysis based on the classification confidence and then replaces the important word by an optimal adversarial *bug* selected from the carefully crafted bug candidates. Since it is initially designed for English-based NLP systems and cannot be directly adopted for generating Chinese adversarial texts, we extend it to our tasks by redesigning the adversarial bugs. Based on the commonly used variation strategies adopted by real-world malicious users [16, 25], we carefully designed six kinds of bugs, which are: (1) **Insert**: Insert a meaningless character into the benign word. (2) **PyConvert**: Convert the word into its Pinyin form, e.g., replacing the word "智障" (idiot) with "zhi zhang". We can also modify the converted Pinyin by insertion, deletion, swap or substitution operations for further perturbation. (3) **Split**: Split one character into more characters and then replace the original character with the splitted characters, e.g., replacing the word "炸弹" (bomb) with "火乍弓单" which looks similar but has completely different meanings. (4) **Sim2Trad**: Convert the simplified Chinese character into its traditional form, e.g., converting the word "裸体" (nude) into "裸體". The character "體" has the same meaning with "体" but will be embedded into a different vector, thus affecting the model's classification result. (5) **GlyphSim**: Replace the character with another one that has similar glyphs, e.g., replacing "赌博" (gamble) with "堵搏". This perturbation has little impact on human understanding due to the powerful human perception and cognition. (6) **PhoneticSim**: Replace a character with another one that has the same pronunciation, e.g., replacing the word "色情" (porn) with "涩情" whose Pinyin are both "se qing". The empirical study on a corpus of real-world attack examples shows that over 98% of the samples can be categorized into one of the six types of bugs (see Fig. 5). More detailed bug examples are shown in Table 1.

4.4 Baselines

We implement and compare two state-of-the-art methods with TEXTSHIELD to evaluate their robustness against the extended TextBugger. In total, the two methods are: (1) **Pycorrector:** This method was first proposed in [47] for dealing with Chinese spelling errors or glyph-based and phonetic-based word variations in user generated texts based on the *n*-gram language model. In our experiments, we use an online version of Pycorrector implemented in Python [5]. (2) **TextCorrector:** It is a Chinese text error correction service developed by Baidu AI [6] for correcting spelling errors, grammatical errors and knowledge errors based on language knowledge, contextual understanding and knowledge computing techniques. In our experiments, we study the efficacy of these two defenses by combining them with the common TextCNN and BiLSTM, respectively. In addition, the common TextCNN and BiLSTM are baseline models themselves.

4.5 Evaluation Metrics

Translation Evaluation. We use three metrics, i.e, word error rate, bilingual evaluation understudy and semantic similarity

[5] https://pypi.org/project/pycorrector/
[6] https://ai.baidu.com/tech/nlp/text_corrector

to evaluate the translation performance of our adversarial NMT model from word, feature and semantics levels.

(1) Word Error Rate (WER). It is derived from the Levenshtein distance and is a word-level metric to evaluate the performance of NMT systems [1]. It is calculated based on the sum of substitutions (S), deletions (D) and insertions (I) for transforming the reference sequence to the target sequence. Suppose that there are total N words in the reference sequence. Then, WER can be calculated by $WER = \frac{S+D+I}{N}$. The range of WER is $[0,1]$ and a smaller value reflects a better translation performance.

(2) Bilingual Evaluation Understudy (BLEU). This metric was first proposed in [38]. It evaluates the quality of translation by comparing the n-grams of the candidate sequence with the n-grams of the reference sequence and counting the number of matches. Concretely, it can be computed as

$$BLEU = BP \cdot \exp(\sum_{n=1}^{N} w_n \log p_n), \tag{12}$$

where p_n is the modified n-grams precision (co-occurrence), w_n is the weight of n-grams co-occurrence and BP is the sentence brevity penalty. The range of BLEU is $[0,1)$ and a larger value indicates a better performance. In our experiment, we use the BLEU implementation provided in [30]

(3) Semantic Similarity (SS). We use this metric to evaluate the similarity between the corrected texts and reference texts from the semantic-level. Here, we use an industry-leading model SimNet developed by Baidu to calculate it, which provides the state-of-the-art performance for measuring the semantic similarity of Chinese texts [40].

Robustness Evaluation. We use the attack success rate, the average number of perturbed words and the required number of queries for per text to evaluate model robustness.

(1) Attack Success Rate (ASR). This metric is the most widely used one in evaluating the performance of adversarial attacks in terms of fooling the target model. It is defined by

$$\text{ASR} = \frac{\text{\# success samples}}{\text{\# total examples}} \tag{13}$$

and a lower success rate indicates a more robust target model.

(2) Perturbed Word. Since text is discrete data, we cannot use the metrics like l_1, l_2 and l_∞ to quantify the added perturbations as done in the image domain. Consequently, we use the number of required perturbed words to quantify the noise scale in adversarial texts.

(3) Query. Recall that TextBugger explores the sensitivity of the target model by iteratively query it for its classification confidence. Hence, we use the average number of queries required for generating one successful adversarial text to evaluate the model sensitivity and fewer queries indicate that the model is more vulnerable.

Utility Evaluation. We use edit distance and Jaccard similarity coefficient to evaluate the utility of the generated adversarial texts from the character-level, and use semantic similarity (which is the same as used in translation evaluation) to evaluate the utility from the semantic-level.

(1) Edit Distance (ED). This metric quantifies the dissimilarity between two strings by counting the minimum number of operations (e.g., removal, insertion or substitution) required to transform one string into the other [33].

(2) Jaccard Coefficient (JC). It is a statistic measure used for gauging the similarity and diversity of finite sample sets and is defined as the size of the intersection divided by the size of the union of the sample sets, i.e.,

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}. \tag{14}$$

In our experiments, A and B denotes the character sets of the benign text and its adversarial counterpart, respectively.

4.6 Implementation

To fairly study the performance and robustness of the baselines and TEXTSHIELD, our experiments have the following settings: (i) the backbone networks applied in all the models have the same architecture, and concretely, the TextCNN backbone is designed with 32 filters of size 2, 3 and 4, and the BiLSTM backbone is designed with one bidirectional layer of 128 hidden units; (ii) all the models have the same maximum sequence length of 50 (since the majority of the texts in our datasets are shorter than 50) and the same embedding size of 128; (iii) all the models are trained from scratch with the Adam optimizer by using a basic setup without any complex tricks; and (iv) the optimal hyperparameters such as learning rate, batch size, maximum training epochs, and dropout rate are tuned for each task and each model separately.

We conducted all the experiments on a server with two Intel Xeon E5-2682 v4 CPUs running at 2.50GHz, 120 GB memory, 2 TB HDD and two Tesla P100 GPU cards.

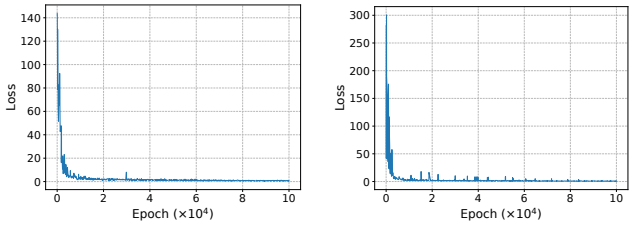
5 Experimental Results

5.1 Evaluation of Model Performance

Detection Performance. We first evaluate the efficacy of TEXTSHIELD and the compared baselines under the non-adversarial setting to verify whether the applied defense will have negative impact on the model performance. The main results are shown in Table 2. It is observed that the common TextCNN and BiLSTM both achieve impressive performance across the two tasks. However, their detection accuracy decreases by 4% and 3% for abuse detection and porn detection respectively when using Pycorrector as the defense, and similar degradation also exists when leveraging TextCorrector. After analyzing the bad cases, we find that some toxic words were erroneously detected as misspelling words by these two methods and the wrong correction thus caused the degradation. Comparatively, when leveraging TEXTSHIELD as the

Table 2: The model accuracy under non-adversarial setting.

Model	Abuse Detection	Porn Detection
Common TextCNN	0.88	0.90
TextCNN + Pycorrector	0.84	0.88
TextCNN + TextCorrector	0.85	0.90
TextCNN + EMF	0.85	0.89
TextCNN + IMF	0.87	0.89
TextCNN + NMT	0.87	0.89
TextCNN + EMF + NMT	0.86	0.88
TextCNN + IMF + NMT	0.88	0.89
Common BiLSTM	0.86	0.87
BiLSTM + Pycorrector	0.82	0.84
BiLSTM + TextCorrector	0.83	0.87
BiLSTM + EMF	0.84	0.86
BiLSTM + IMF	0.85	0.88
BiLSTM + NMT	0.84	0.86
BiLSTM + EMF + NMT	0.84	0.85
BiLSTM + IMF + NMT	0.85	0.87



(a) Abuse (b) Porn

Figure 3: The training loss of the adversarial NMT model.

defense, the accuracy decreases by less than 2% when only applying multimodal embedding and by less than 1% when combing multimodal embedding with adversarial translation. This demonstrates that TEXTSHIELD has little impact on the model performance in the non-adversarial environments.

Translation Performance. Compared to the traditional NMT tasks such as English–Chinese translations, our adversarial translation task is relatively easy since the source and target sequences are both Chinese and usually only differ in few characters. Hence, as illustrated in Fig. 3, the training loss of the adversarial NMT model converges quickly and achieves the optimality within 2×10^4 steps, indicating that it is feasible and easy to apply NMT to restore adversarial perturbations. Furthermore, Table 3 shows the error correction performance of the adversarial NMT model and the two compared spelling correction methods on the test set of the parallel corpora. The baseline result is calculated based on the adversarial texts without error correction and the corresponding reference texts, which reflects their original difference. It can be seen that the adversarial NMT model achieves an excellent performance across all the metrics in the two tasks and outperforms the compared spelling correction methods by a significant margin. This demonstrates that end-to-end adversarial NMT is more elastic and effective in reconstructing the original text from its corresponding adversarial text.

Table 3: The error correction performance.

Method	Abuse Detection			Porn Detection		
	WER	BLEU	SS	WER	BLEU	SS
Baseline	0.198	0.744	0.939	0.199	0.749	0.937
Pycorrector	0.223	0.687	0.906	0.213	0.701	0.911
TextCorrector	0.181	0.767	0.939	0.173	0.777	0.938
Adversarial NMT	**0.051**	**0.923**	**0.988**	**0.056**	**0.916**	**0.985**

5.2 Evaluation of Effectiveness

Second, we evaluate the efficacy of TEXTSHIELD in terms of defending the DLTC models against the user generated obfuscated texts in the real-world adversarial scenario. Specifically, we first collect 2,000 obfuscated abusive texts and 2,000 obfuscated pornographic texts from online social media. Each collected sample is manually confirmed to have at least one variant word. Then, we manually construct a reference set for each task by restoring the variant words to their original benign counterparts. Finally, all experiments are conducted on the collected obfuscated texts and the reference sets.

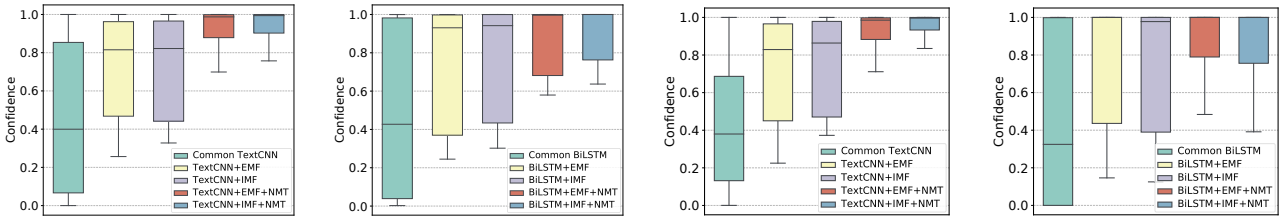
Detection Performance. The main results of detection performance and the comparison with different baselines are summarized in Table 4. It is observed that the common TextCNN and BiLSTM achieve an accuracy below 0.496 in the two tasks. Shielded by Pycorrector and TextCorrector, these models however obtain an unnoticeable improvement in detection accuracy. We speculate that this is mainly because these two spelling correctors are also vulnerable to the manually crafted real-world adversarial perturbations. Comparatively, TEXTSHIELD significantly improves the accuracy of the DLTC models by about 30% for only leveraging multimodal embedding, and by 45% for using the combined defense scheme, indicating that TEXTSHIELD is practical and effective in enhancing model robustness in the real-world adversarial scenario. An interesting observation is that TextCNN achieves the best performance for abuse detection when applying EMF while achieves the best performance for porn detection when applying IMF, and a converse result is observed for BiLSTM. This shows that the best defense varies for different models and tasks and we should design the application-aware defense schemes in practice.

We further analyze the classification confidence of the models defended by TEXTSHIELD on the collected obfuscated texts and the evaluation results are visualized in Fig. 4. Obviously, it can be seen that the models with the combined defense classify the obfuscated texts with a much higher confidence. This demonstrates that these models are more robust against the manually crafted adversarial perturbations. In addition, the above mentioned interesting observation also exists in Fig. 4, once again demonstrating the need for designing application-aware defense.

Correction Performance. The error correction performance of our adversarial NMT model on user generated obfuscated texts and the comparative performance of the two

Table 4: The detection performance on user generated obfuscated texts.

Model \ # of Perturbation	Abuse Detection				Porn Detection			
	≤ 1	≤ 2	≤ 3	> 3	≤ 1	≤ 2	≤ 3	> 3
Common TextCNN	0.488	0.483	0.480	0.458	0.496	0.448	0.426	0.398
TextCNN + Pycorrector	0.491	0.488	0.506	0.490	0.504	0.481	0.468	0.449
TextCNN + TextCorrector	0.498	0.484	0.485	0.457	0.568	0.563	0.558	0.555
TextCNN + EMF	0.790	0.783	0.760	0.736	0.753	0.742	0.732	0.718
TextCNN + IMF	0.714	0.725	0.732	0.729	0.777	0.767	0.751	0.730
TextCNN + NMT	0.857	0.886	0.869	0.836	0.909	0.899	0.887	0.870
TextCNN + EMF + NMT	**0.923**	**0.931**	0.919	**0.906**	0.928	0.921	0.908	0.901
TextCNN + IMF + NMT	0.922	0.931	**0.920**	0.904	**0.944**	**0.933**	**0.926**	**0.915**
Common BiLSTM	0.350	0.343	0.341	0.328	0.477	0.467	0.462	0.473
BiLSTM + Pycorrector	0.356	0.356	0.364	0.355	0.475	0.471	0.473	0.481
BiLSTM + TextCorrector	0.356	0.349	0.352	0.348	0.465	0.435	0.433	0.446
BiLSTM + EMF	0.604	0.616	0.620	0.605	0.746	0.725	0.730	0.724
BiLSTM + IMF	0.631	0.646	0.643	0.645	0.744	0.708	0.710	0.713
BiLSTM + NMT	0.801	0.791	0.764	0.707	0.856	0.804	0.778	0.757
BiLSTM + EMF + NMT	0.900	0.890	0.871	0.848	**0.933**	**0.913**	**0.903**	**0.890**
BiLSTM + IMF + NMT	**0.892**	**0.894**	**0.881**	**0.851**	0.932	0.906	0.891	0.882

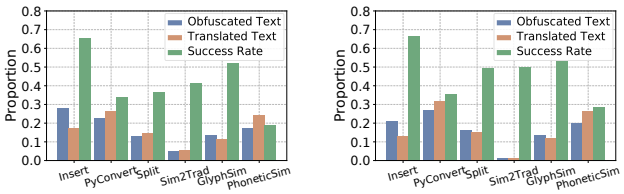


(a) TextCNN on Abuse (b) BiLSTM on Abuse (c) TextCNN on Porn (d) BiLSTM on Porn

Figure 4: The comparison of classification confidence on user crafted obfuscated texts.

Table 5: The error correction performance on user generated obfuscated texts.

Method	Abuse Detection			Porn Detection		
	WER	BLEU	SS	WER	BLEU	SS
Baseline	0.292	0.570	0.878	0.337	0.478	0.866
Pycorrector	0.319	0.568	0.858	0.349	0.563	0.840
TextCorrector	0.291	0.618	0.875	0.304	0.626	0.860
Adversarial NMT	**0.122**	**0.796**	**0.913**	**0.197**	**0.741**	**0.918**



(a) Abuse (b) Porn

Figure 5: The proportion of different bugs in obfuscated texts and the texts translated by adversarial NMT.

spelling correction methods are shown in Table 5. The baseline represents the original difference between obfuscated texts and the manually restored reference texts. From Table 5, we can see that Pycorrector has negative impact on error correction, which confirms the above speculation that it is also sensitive to adversarial perturbations. TextCorrector also does not work well when presented with carefully crafted adversarial perturbations instead of common spelling errors. In contrast, the adversarial NMT model performs well in restoring the real-world adversarial perturbation. For instance, it decreases WER by 17% and 14% for abusive texts and pornographic texts, respectively. Hence, it can be concluded that the end-to-end adversarial NMT is more elastic, practical and effective in the real-world adversarial scenario.

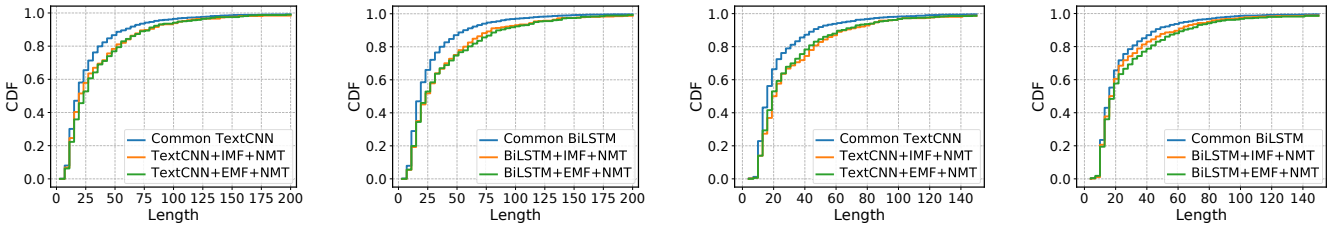
Table 6: The performance of transfer attack against the target models.

Model	Abuse Detection		Porn Detection	
	Ori Accuracy	ASR	Ori Accuracy	ASR
TextCNN + Pycorrector	0.906	0.749	0.926	0.796
TextCNN + TextCorrector	0.978	0.929	0.979	0.609
TextCNN + EMF	0.951	0.354	0.935	0.321
TextCNN + IMF	0.972	0.371	0.974	0.323
TextCNN + NMT	0.994	0.219	0.998	0.170
TextCNN + EMF + NMT	0.951	0.111	0.936	0.118
TextCNN + IMF + NMT	0.972	**0.088**	0.974	**0.090**
BiLSTM + Pycorrector	0.874	0.857	0.901	0.776
BiLSTM + TextCorrector	0.975	0.732	0.974	0.943
BiLSTM + EMF	0.947	0.392	0.938	0.210
BiLSTM + IMF	0.958	0.304	0.927	0.276
BiLSTM + NMT	0.991	0.238	0.989	0.226
BiLSTM + EMF + NMT	0.948	**0.075**	0.938	**0.067**
BiLSTM + IMF + NMT	0.957	0.113	0.928	0.093

We also visualize the proportion of different bugs in obfuscated texts and the translations of the adversarial NMT model in Fig. 5 to further analyze its robustness against different

Table 7: The performance of adaptive attack against all the target models.

Model	Abuse Detection			Porn Detection		
	ASR	Perturbed Word	Query	ASR	Perturbed Word	Query
Common TextCNN	0.860	2.19	65.8	0.839	2.12	61.1
TextCNN + Pycorrector	0.830	1.91	61.9	0.823	2.01	59.4
TextCNN + TextCorrector	0.786	2.03	66.3	0.773	2.13	60.4
TextCNN + EMF	0.687	2.35	69.2	0.706	2.02	58.9
TextCNN + IMF	0.622	2.32	68.5	0.595	2.18	61.7
TextCNN + NMT	0.375	2.05	63.7	0.428	2.34	64.3
TextCNN + EMF + NMT	0.240	2.00	63.9	0.339	2.15	60.8
TextCNN + IMF + NMT	**0.219**	1.93	62.7	**0.236**	2.03	59.4
Common BiLSTM	0.891	1.87	61.7	0.846	2.11	61.3
BiLSTM + Pycorrector	0.872	1.68	58.7	0.835	1.75	55.9
BiLSTM + TextCorrector	0.866	1.83	59.5	0.821	1.95	60.9
BiLSTM + EMF	0.726	1.97	63.8	0.548	2.12	61.6
BiLSTM + IMF	0.555	1.87	62.0	0.550	2.14	61.8
BiLSTM + NMT	0.450	1.93	62.5	0.548	2.20	62.7
BiLSTM + EMF + NMT	0.268	1.85	62.2	**0.247**	2.03	60.3
BiLSTM + IMF + NMT	**0.238**	1.73	60.2	0.289	1.80	55.7



(a) TextCNN on Abuse (b) BiLSTM on Abuse (c) TextCNN on Porn (d) BiLSTM on Porn

Figure 6: The original text length distribution of the successfully generated adversarial texts.

Table 8: Utility evaluation of successful adversarial texts.

Model	Abuse Detection			Porn Detection		
	ED	JC	SS	ED	JC	SS
Common TextCNN	8.23	0.736	0.903	9.09	0.722	0.884
TextCNN + Pycorrector	8.23	0.736	0.903	8.77	0.737	0.895
TextCNN + TextCorrector	8.48	0.731	0.892	8.89	0.725	0.874
TextCNN + EMF	8.84	0.720	0.883	10.3	0.713	0.846
TextCNN + IMF	8.60	0.727	0.887	10.4	0.712	0.843
TextCNN + NMT	8.05	0.743	0.890	10.2	0.711	0.856
TextCNN + EMF + NMT	9.18	0.759	0.891	9.31	0.756	0.876
TextCNN + IMF + NMT	8.43	0.764	0.900	10.6	0.748	0.879
Common BiLSTM	8.31	0.758	0.907	8.07	0.738	0.858
BiLSTM + Pycorrector	8.11	0.767	0.910	7.24	0.754	0.885
BiLSTM + TextCorrector	8.23	0.771	0.916	7.58	0.743	0.865
BiLSTM + EMF	8.69	0.731	0.905	8.05	0.720	0.854
BiLSTM + IMF	8.18	0.751	0.890	9.87	0.717	0.853
BiLSTM + NMT	8.34	0.752	0.901	8.71	0.716	0.863
BiLSTM + EMF + NMT	8.88	0.774	0.905	9.61	0.750	0.878
BiLSTM + IMF + NMT	8.34	0.787	0.906	9.22	0.742	0.880

bugs. It is obviously observed that Insert and PyConvert are the dominant variation strategies used by real-world malicious users, and Sim2Trad is used less than others. This is probably because that Insert and PyConvert are easy to craft while Sim2Trad can be easily defended by text preprocessing adopted in online toxic content detection services. From the visualized success rate, we can see that above 60% of Insert bugs can be corrected by the NMT model. Comparatively, the corrected PhoneticSim bugs in the abusive texts and pornographic texts account for less than 20% and 30%, respectively. This indicates that the NMT model is robust against the Insert bugs while less robust against the PhoneticSim bugs. From Table 4 and Fig. 5, we can also see that the DLTC models with the combined defense have a high accuracy even though bugs still exist in the translations. We argue that not all bugs need to be corrected and the obfuscated text can still be correctly classified by the multimodal DLTC model as long as one or two bugs are corrected.

5.3 Evaluation of Robustness

Next, we evaluate the robustness of TEXTSHIELD against adversarial attack from the perspectives of transfer attack and adaptive attack. In this evaluation, the adversarial texts are generated by TextBugger with the maximum perturbation of 4 and the semantic similarity threshold of 0.75, considering that the average text length of our datasets is about 40.

Robustness against Transfer Attack. One intriguing property of adversarial inputs is their transferability, i.e., adversarial input generated against one model can also trick another model with different architectures and trained on dif-

ferent data [7, 13]. Specifically, we first randomly sample a set of texts from each dataset, in which each text is correctly classified by the corresponding model. Then, we generate 1,000 successful adversarial texts against each common model from the sampled texts and transfer them to the models with defense. The main results are summarized in Table 6. Observe that the transferability against the models shielded by TEXTSHIELD is fairly low: it achieves an attack success rate lower than 0.09 against the TextCNN models that combined with IMF and NMT, and achieves an attack success rate lower than 0.075 against the BiLSTM models combined with EMF and NMT. In comparison, Pycorrector and TextCorrector have little effectiveness in mitigating transfer attack. We can therefore conclude that TEXTSHIELD is more robust against the transferred adversarial texts.

Robustness against Adaptive Attack. So far, we have only considered static adversaries, who only generate adversarial texts against the DLTC models without adapting to attack TEXTSHIELD directly. In this evaluation, we consider the challenges TEXTSHIELD may face when adversaries know our defense. Specifically, we first randomly sample 2,000 correctly classified texts from the test and validation sets for each target model. Then, the DLTC model and TEXTSHIELD would be viewed as a whole pipeline by attackers when exploring model sensitivity. This is a more realistic worst-case setting, since attackers (e.g., malicious netizens) usually only have black-box access to the whole pipeline in the real adversarial scenario [5]. Under this setting, attackers can not only capture the vulnerability of the DLTC model, but also capture the vulnerability of the multimodal embedding and the adversarial NMT model. Finally, adversarial texts are generated from sampled benign texts by TextBugger according to the explored sensitivity information.

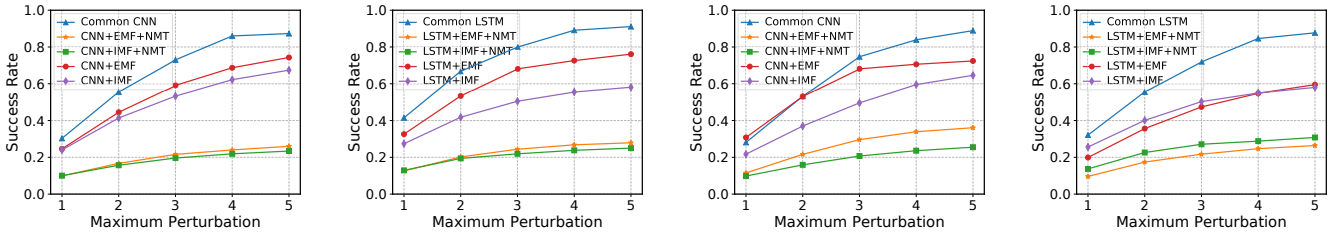
(1) Attack Success Rate. The attack performance against all the target models are reported in Table 7. Obviously, it can be seen that the adaptive attack achieves a lower success rate against the models with TEXTSHIELD compared to the common models. Particularly, when only applying multimodal embedding, the defense not only degrades the success rate but also increases the cost of the attack in terms of the average number of perturbed words and queries. However, when leveraging the combined defense scheme, an anomalous result is observed that although the defense significantly decreases the attack success rate, the required cost of attack also degrades in most cases. We conjecture that such anomalous result may stem from the poor classification robustness of the models with the combined defense on the benign texts used for generating these successful adversarial texts. To verify our conjecture, we visualize the cumulative distribution of the text length of these benign texts in Fig. 6. Observe that the original length of the texts successfully generated against the models with the combined defense is significant longer than those generated for the common models. However, all the models especially the NMT models usually perform relatively worse on longer texts due to the problem of *long-term dependencies* [4], thus making it easier to trigger the DLTC models to misbehave with less cost on longer texts.

(2) Utility Analysis. The main results of utility evaluation are summarized in Table 8. It can be seen that the adversarial texts generated against the common models and the models with Pycorrector and TextCorrector preserve good utility in terms of both character-level and semantics-level. Also, similar trend is observed that multimodal embedding reduces the utility of adversarial texts across all metrics, while the utility of the adversarial texts generated against TEXTSHIELD seems higher in several cases. This also stems from the impact of longer text length. Intuitively, longer text is more fault-tolerant and can preserve more of the original utility when slightly perturbed.

(3) The Impact of Maximum Perturbation. We further study the impact of the maximum number of required perturbations on the attack success rate of adaptive attack. The main results are shown in Fig. 7. It is clearly seen that the attack success rate against all the target models increases as the maximum number of perturbations grows. Particularly, the success rate against the common models increases rapidly with the increasing maximum perturbation while increases slightly against the models with TEXTSHIELD, thus resulting in a growing gap between the success rate. In addition, the success rate against the models with TEXTSHIELD is still below 0.3 when perturbed with the maximum perturbation of 5. We thus conclude that TEXTSHIELD is very robust against the adaptive attacks and outperforms the baselines.

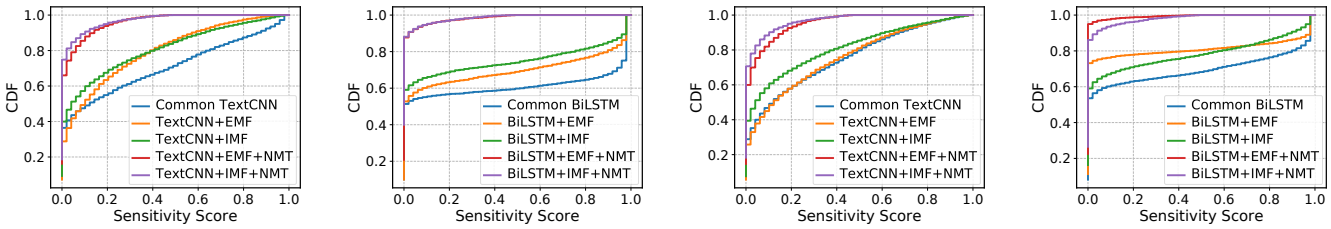
(4) Model Sensitivity Analysis. We also investigate the sensitivity of the target models against each bug replacement by visualizing the cumulative distribution of sensitivity score in Fig. 8. The sensitivity score represents the reward for each bug replacement, i.e., the reduction in toxic confidence. Observe from the two tasks that the sensitivity score of the models defended by TEXTSHIELD are markable smaller than those of the common models, especially when the DLTC model is shielded by the combined defense. For instance, for abuse detection, nearly 100% of bug replacement that against the BiLSTM with the combined defense only gains a reward lower than 0.2 while more than 40% of bug replacement against the common BiLSTM obtains a reward higher than 0.4. This demonstrates that all defense schemes in TEXTSHIELD have high resistance to adversarial bugs and thus help mitigate the sensitivity of the DLTC models.

(5) Bug Distribution. We take the TextCNN models used for abuse detection as examples to further study their sensitivity to different bugs, and the analysis for other models can be found in Appendix C. The distributions of bugs in adversarial texts against the common models and the models shielded by TEXTSHIELD are visualized in Fig. 9 , where the x-axis represents the proportion of bugs for the common model and y-axis represents the proportion for the model with defense, and the marker size represents the rate of the bugs being successfully



(a) TextCNN on Abuse (b) BiLSTM on Abuse (c) TextCNN on Porn (d) BiLSTM on Porn

Figure 7: The impact of maximum perturbation on attack success rate. CNN and LSTM represent TextCNN and BiLSTM.



(a) TextCNN on Abuse (b) BiLSTM on Abuse (c) TextCNN on Porn (d) BiLSTM on Porn

Figure 8: The sensitivity of the target models against bug replacement.

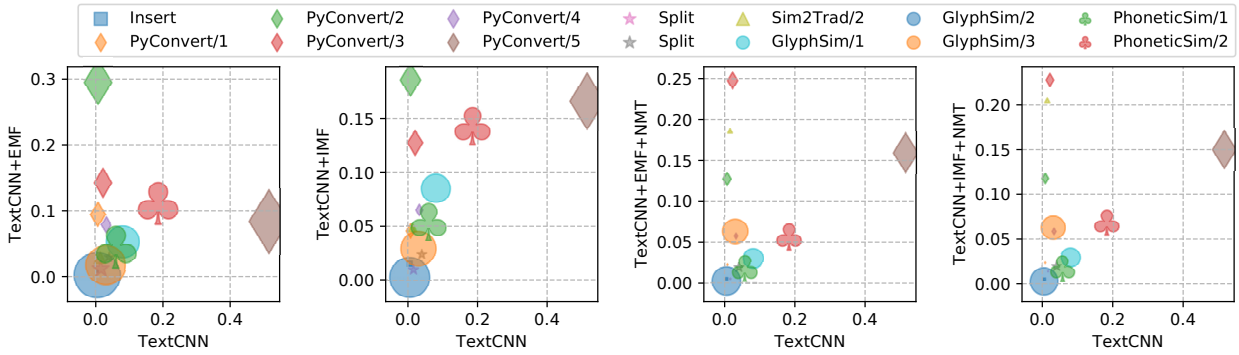


Figure 9: The sensitivity of the target TextCNN models against different bugs in abuse detection.

defended. It is clearly observed that Insert is used less than the others, indicating that this kind of bug is less powerful. In comparison, both the common model and the models with defense are more sensitive to PyConvert, especially PyConvert/2 and PyConvert/5. One reason is that the DLTC models in our experiment work at the character-level while the number of characters in Pinyin is usually several times that of Chinese characters, hence PyConvert may lead to errors in the feature extraction stage. In addition, there are many secondary variations based on the converted Pinyin which result in the sparseness of adversarial perturbations, making it difficult for the proposed defense to cover all possible variations. Meanwhile, TEXTSHIELD shows its robustness in defending against some of the PyConvert bugs as well as the GlyphSim and PhoneticSim bugs. As a future work, we will focus on more robust defense to deal with the stubborn bugs like PyConvert/2 and PyConvert/5. For example, we would first restore the secondary variations to their original format and then restore the original Pinyin to the corresponding Chinese characters by designing some preprocessing schemes.

5.4 Comparison with Online Services

Now, we make comparison with four industry-leading toxic content detection services, i.e., Alibaba GreenNet, Baidu TextCensoring, Huawei Moderation and Netease Yidun, who have claimed to be successful in handling the glyph-based and phonetic-based variations, to show the practicality of TEXTSHIELD. We generate adversarial texts with TextBugger under the same setting to evaluate their robustness.

The comparison results are reported in Table 9. Observe that most of these services achieve relatively good detection accuracy under the non-adversarial setting. However, it is also observed that they are still highly vulnerable to the generated adversarial texts. Specifically, they are tricked with higher attack success rate (i.e., above 0.814 across all cases) and less words perturbed than the models shielded by TEXTSHIELD, which indicate that the defenses integrated in

Table 9: Comparison with real-world online detection services.

Targeted API	Abuse Detection				Porn Detection			
	Accuracy	ASR	Perturbed Word	Query	Accuracy	ASR	Perturbed Word	Query
Alibaba GreenNet	0.778	0.868	1.34	40.1	0.869	0.884	1.71	48.2
Baidu TextCensoring	0.763	0.938	1.36	33.4	0.892	0.897	1.88	49.9
Huawei Moderation	0.704	0.888	1.34	35.3	0.710	0.814	1.67	46.7
Netease Yidun	0.805	0.903	1.38	42.1	0.823	0.818	1.90	51.1
TextCNN + IMF + NMT	0.880	0.219	1.93	62.7	0.890	0.236	2.03	59.4
BiLSTM + EMF + NMT	0.840	0.268	1.85	62.2	0.850	0.247	2.03	60.3

Table 10: The results of adaptive attacks against English-based DLTC models with TEXTSHIELD.

Model	Accuracy	ASR	Perturbed Word	Query
Common TextCNN	0.754	0.880	1.60	36.7
TextCNN + EMF + NMT	0.757	0.283	1.53	37.5
TextCNN + IMF + NMT	0.752	**0.265**	1.38	36.4
Common BiLSTM	0.766	0.782	1.80	38.4
BiLSTM + EMF + NMT	0.751	0.351	1.54	37.7
BiLSTM + IMF + NMT	0.763	**0.285**	1.26	36.1

these services can still be ruined by adversarial attacks. In contrast, TEXTSHIELD shows great practicality for reducing the attack success rate as well as improving the cost of the attack. Interestingly, we find that although Netease Yidun and Baidu TextCensoring outperform others in abuse detection and porn detection tasks, respectively, they are also more vulnerable to adversarial texts. We thus conclude that the robustness of DLTC systems is independent of their accuracy, i.e., the model with high accuracy is not necessarily secure.

5.5 Evaluation of Generalizability

Finally, we study extending TEXTSHIELD to English-based DLTC models to examine its generalizability across languages. The experiment is conducted on the classical sentiment analysis task with the benchmark Rotten Tomatoes Movie Reviews dataset [36] under the same adaptive setting. Since the pronunciation of an English word is related to its spelling, we only learn the word embeddings from two modalities, i.e., semantics and glyphs, and all the models are trained from scratch without any complex tricks. Finally, adversarial texts are generated from sampled benign texts by TextBugger [26].

The main results are summarized in Table 10. The second column is the model accuracy evaluated under the non-adversarial setting, which is comparable to the performance reported in [23]. It is clearly observed that the common models can be deceived with high attack success rates, e.g., 0.880 for TextCNN and 0.782 for BiLSTM, which indicates that the English-based DLTC models are also very vulnerable in the adversarial environment. However, the attack success rates against TextCNN and BiLSTM decrease to 0.265 and 0.285 respectively when the models are shielded by TEXTSHIELD. This indicates that TEXTSHIELD is also effective in defending English-based DLTC models against adversarial attacks, which shows good generalizability across languages.

6 Discussion

In this section, we discuss the limitations of TEXTSHIELD and promising directions for further improvements.

Extensions to Other Settings and Tasks. In this paper, TEXTSHIELD is designed to defend against adversaries in the realistic adversarial environments, and it is evaluated under the black-box setting. However, attackers may still have a small chance of accessing the entire system in white box. Hence, evaluating its efficacy against the white-box attacks is a valuable future work. Furthermore, TEXTSHIELD is currently applied to two real-world tasks. In practice, there are many other tasks such as spam email filtering that can also potentially benefit from TEXTSHIELD. In the future work, we will explore its applicability in broader real-world tasks.

Challenges for Real-world Deployments. Experimental results have shown great promise to deploy TEXTSHIELD in real-world. However, since TEXTSHIELD will increase the total amount of model parameters, it may slightly decrease the efficiency or increase the deployment cost of the whole system. We argue that this would not be a hindrance to the real-world deployment, because security is usually more important in the security-sensitive tasks. In the future, we plan to apply model compression and distributed computing techniques to accelerate the whole system and reduce the costs.

7 Conclusion

To enhance the robustness of DLTC models against adversarial texts in online toxic content detection tasks, we present TEXTSHIELD, a new defense framework specifically designed for Chinese-based DLTC models. At a high level, TEXTSHIELD achieves robust toxic content detection by integrating a set of key strategies, including multimodal embedding, multimodal fusion, and adversarial neural machine translation. Through extensive empirical evaluation, we demonstrate that TEXTSHIELD attains promising effectiveness in defending against user generated obfuscated texts in real-world adversarial scenarios, while with little impact on the original detection performance. We also show that TEXTSHIELD is robust against the state-of-the-art adversarial attacks even un-

der the adaptive setting. Our study may shed new light on designing adversarial defenses for other NLP tasks.

Acknowledgments

We sincerely appreciate the shepherding from David Evans. We would also like to thank the anonymous reviewers for their constructive comments and input to improve our paper. This work was partly supported by the National Key Research and Development Program of China under No. 2018YFB0804102, NSFC under No. 61772466, U1936215, and U1836202, the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020003, the Provincial Key Research and Development Program of Zhejiang, China under No. 2019C01055, the Ant Financial Research Funding, and the Alibaba-ZJU Joint Research Institute of Frontier Technologies. Ting Wang is partially supported by the National Science Foundation under Grant No. 1910546, 1953813, and 1846151. Min Yang is partially supported by NSFC under No. U1636204 and U1836213. Min Yang is also a member of Shanghai Institute of Intelligent Electronics & Systems, Shanghai Institute for Advanced Communication and Data Science.

References

[1] Ahmed Ali and Steve Renals. Word error rate estimation for speech recognition: e-wer. In *ACL*, pages 20–24, 2018.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[3] Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. The role of social networks in information diffusion. In *WWW*, pages 519–528. ACM, 2012.

[4] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[5] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR*, 2018.

[6] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *SSST*, pages 103–111, 2014.

[7] Tianyu Du, Shouling Ji, Jinfeng Li, Qinchen Gu, Ting Wang, and Raheem Beyah. Sirenattack: Generating adversarial audio for end-to-end acoustic systems. In *AsiaCCS*, 2020.

[8] Javid Ebrahimi, Daniel Lowd, and Dejing Dou. On adversarial examples for character-level neural machine translation. In *COLING*, pages 653–663, 2018.

[9] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. In *ACL*, pages 31–36, 2018.

[10] Björn Gambäck and Utpal Kumar Sikdar. Using convolutional neural networks to classify hate-speech. In *ALW*, pages 85–90, 2017.

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

[12] Zhitao Gong, Wenlu Wang, Bo Li, Dawn Song, and Wei-Shinn Ku. Adversarial texts with gradient methods. *arXiv preprint arXiv:1801.07175*, 2018.

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[14] Alex Graves. Sequence transduction with recurrent neural networks. In *ICML*, 2012.

[15] Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*, 2017.

[16] Longtao Huang, Ting Ma, Junyu Lin, Jizhong Han, and Songlin Hu. A multimodal text matching model for obfuscated language identification in adversarial communication? In *WWW*, pages 2844–2850, 2019.

[17] Mansoor Iqbal. Wechat revenue and usage statistics, 2019.

[18] Heng Ji and Kevin Knight. Creative language encoding under censorship. In *Proceedings of the First Workshop on Natural Language Processing for Internet Freedom*, pages 23–33, 2018.

[19] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. Model-reuse attacks on deep learning systems. In *CCS*, pages 349–363, 2018.

[20] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*, pages 2021–2031, 2017.

[21] Zhuoren Jiang, Zhe Gao, Guoxiu He, Yangyang Kang, Changlong Sun, Qiong Zhang, Luo Si, and Xiaozhong Liu. Detect camouflaged spam content via stoneskipping: Graph and text joint embedding for chinese character variation representation. In *EMNLP-IJCNLP*, pages 6188–6197, 2019.

[22] Mladen Karan and Jan Šnajder. Cross-domain detection of abusive language online. In *ALW*, pages 132–137, 2018.

[23] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.

[24] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] Ao Li, Zhou Qin, Runshi Liu, Yiqun Yang, and Dong Li. Spam review detection with graph convolutional networks. In *CIKM*, pages 2703–2711, 2019.

[26] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. In *NDSS*, 2019.

[27] Xurong Li, Shouling Ji, Meng Han, Juntao Ji, Zhenyu Ren, Yushan Liu, and Chunming Wu. Adversarial examples versus cloud-based detectors: A black-box empirical study. *IEEE Transactions on Dependable and Secure Computing*, 2019.

[28] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014.

[29] Xiang Ling, Shouling Ji, Jiaxu Zou, Jiannan Wang, Chunming Wu, Bo Li, and Ting Wang. Deepsec: A uniform platform for security analysis of deep learning model. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 673–690. IEEE, 2019.

[30] Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*, 2017.

[31] Joe Mayes and Stefan Nicola. Facebook warns it can't fully solve toxic content problem, 2019.

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[33] Frederic P Miller, Agnes F Vandome, and John McBrewster. *Levenshtein distance: Information theory, computer science, string (computer science), string metric, damerau? Levenshtein distance, spell checker, hamming distance*. Alpha Press, 2009.

[34] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. Abusive language detection in online user content. In *WWW*, pages 145–153. International World Wide Web Conferences Steering Committee, 2016.

[35] Behnaz Nojavanasghari, Deepak Gopinath, Jayanth Koushik, Tadas Baltrušaitis, and Louis-Philippe Morency. Deep multimodal fusion for persuasiveness prediction. In *ICMI*, pages 284–288. ACM, 2016.

[36] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124. Association for Computational Linguistics, 2005.

[37] Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM*, pages 49–54. IEEE, 2016.

[38] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. Association for Computational Linguistics, 2002.

[39] Chenghui Shi, Xiaogang Xu, Shouling Ji, Kai Bu, Jianhai Chen, Raheem Beyah, and Ting Wang. Adversarial captchas. *arXiv preprint arXiv:1901.01107*, 2019.

[40] Baidu simnet. https://ai.baidu.com/tech/nlp/simnet.

[41] Nitish Srivastava and Ruslan R Salakhutdinov. Multimodal learning with deep boltzmann machines. In *NIPS*, pages 2222–2230, 2012.

[42] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[43] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[44] Yicheng Wang and Mohit Bansal. Robust machine comprehension models via adversarial training. In *NAACL*, pages 575–581, 2018.

[45] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[46] Jui-Feng Yeh, Yun-Yun Lu, Chen-Hsien Lee, Yu-Hsiang Yu, and Yong-Ting Chen. Chinese word spelling correction based on rule induction. In *CIPS-SIGHAN CLP*, pages 139–145, 2014.

[47] Junjie Yu and Zhenghua Li. Chinese spelling error detection and correction based on language model, pronunciation, and shape. In *CIPS-SIGHAN CLP*, pages 220–223, 2014.

[48] Kan Yuan, Di Tang, Xiaojing Liao, XiaoFeng Wang, Xuan Feng, Yi Chen, Menghan Sun, Haoran Lu, and Kehuan Zhang. Stealthy porn: Understanding real-world adversarial images for illicit online promotion. In *S&P*, pages 952–966. IEEE, 2019.

[49] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. Interpretable deep learning under fire. In *USENIX Security*, 2020.

[50] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *ACL*, pages 207–212, 2016.

Appendix

A Multimodal Fusion Schemes

Fig. 10 illustrates the two multimodal fusion schemes, i.e., EMF and IMF.

B Data Collection Details

At the first stage, we collected 40,000 user comments from Weibo, Taobao, etc., for each task (i.e., abuse and porn detection). Considering the ethical implications, we fully respect the privacy of users, and only use the public comment texts of them. After preprocessing, removing the duplicates and filtering out the meaningless texts, we used Alibaba GreenNet to automatically label these processed texts, and we then got about 30,000 coarsely labelled samples for each task, in which about 15,000 samples were toxic and 15,000 were normal. At the second stage, we hired several Chinese native speakers to relabel the coarsely labelled samples, and we also filtered out those samples that were labelled inconsistently. Then, we randomly sampled 10,000 finely labelled samples for each class as the datasets we used in our experiments. Specifically, each sample was also manually confirmed that there did not exist variant words. In the meantime, we got a corpus of 2,000 obfuscated texts (i.e., real-world attack examples as shown in Fig. 11) for each task, in which each text had at least one variant word. We then asked the hired workers to annotate what the variant word was and which category it belonged to, and the statistic distribution of different variant categories can be seen in Fig. 5.

C Distribution of Bugs

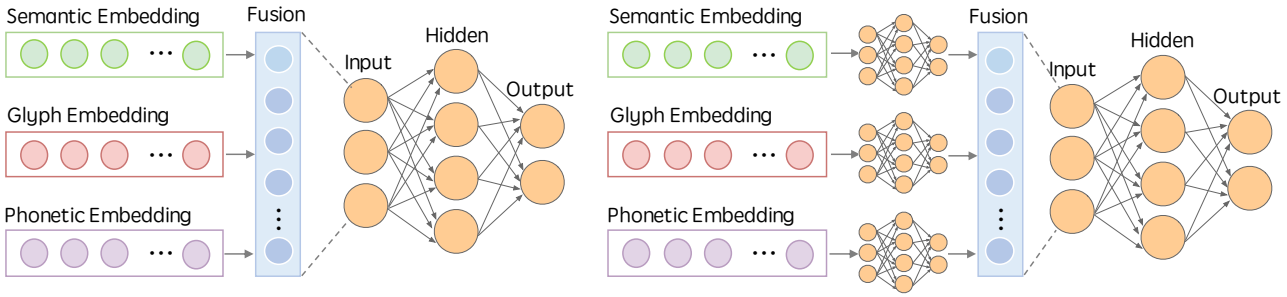
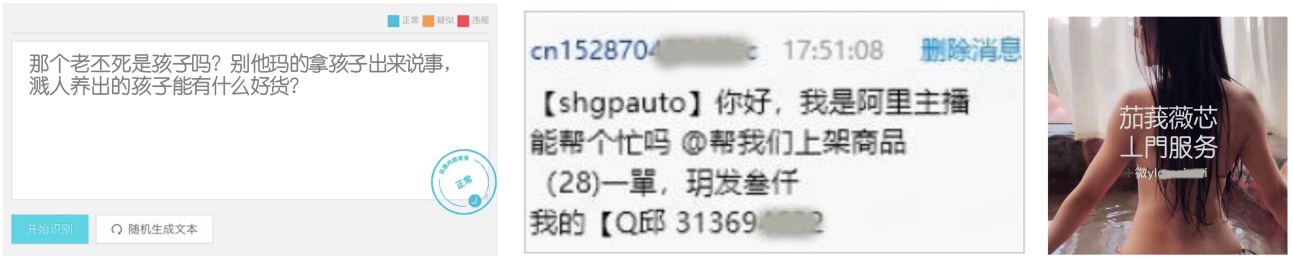


Figure 10: Illustration of multimodal fusion schemes.

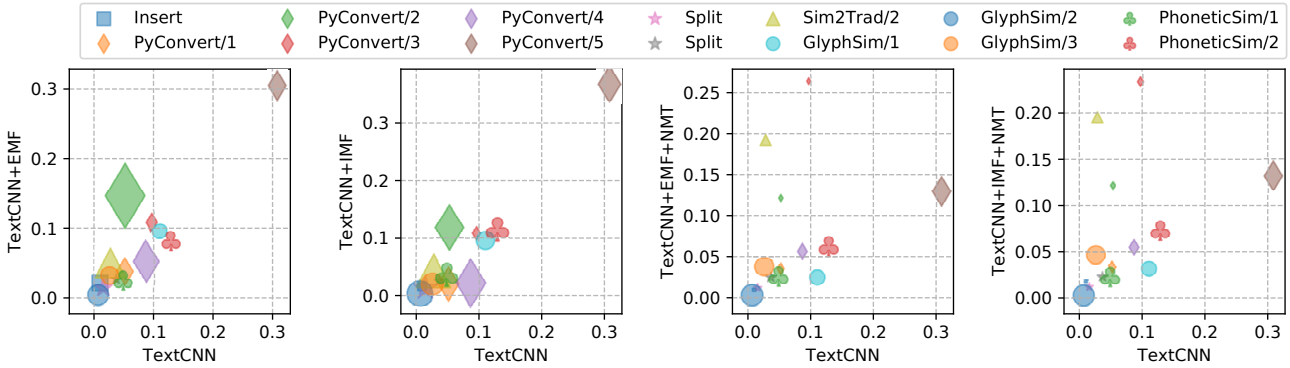


(a) Insulting Comment on Weibo

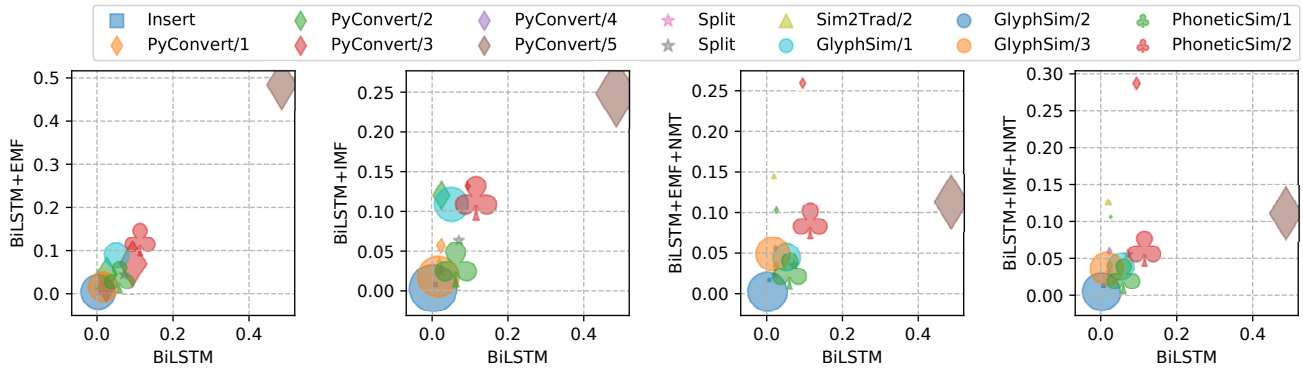
(b) Spam Message on Taobao

(c) Pornographic Ads on WeChat

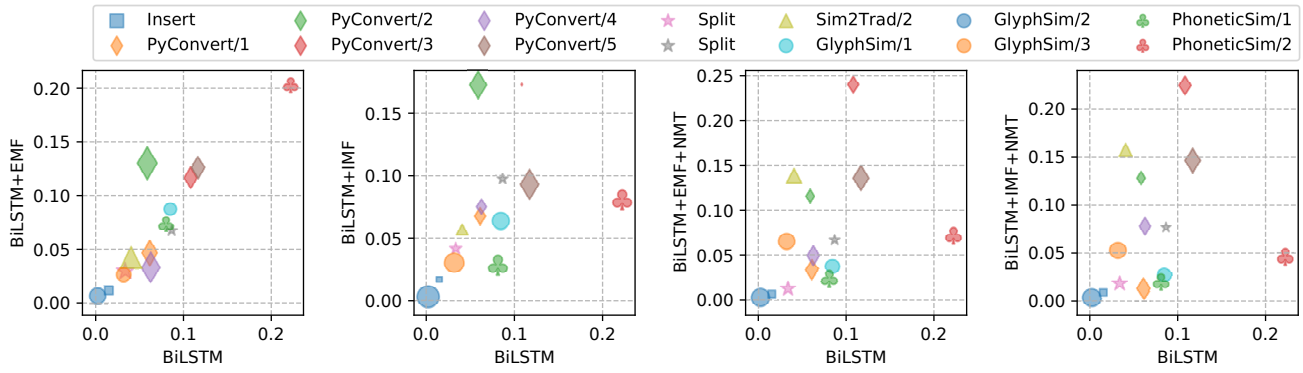
Figure 11: Adversarial examples in the real world. The subfigures are: (a) is an obfuscated insulting comment on Weibo in which "老丕死" is mutated from "老不死" (old fuck) and "溅人" is mutated from "贱人" (bitch), and the obfuscated text retains insulting but successfully evaded the censorship; (b) is an obfuscated spam ads for the purpose of fake purchase on Taobao; (c) is a pornographic ads for sex service on WeChat, in which "茄莪薇芯" is an obfuscated phrase of "加我微信" that means "add my WeChat account", and the obfuscated ads is still illegal but usually hard to detect.



(a) TextCNN on Porn



(b) BiLSTM on Abuse



(c) BiLSTM on Porn

Figure 12: The sensitivity of the target models against different bugs on the two datasets.