RoCBert: Robust Chinese Bert with Multimodal Contrastive Pretraining

Hui Su[1]*, **Weiwei Shi**[1]*, **Xiaoyu Shen**[2]*
Xiao Zhou[1], **Tuo Ji**[1], **Jiarui Fang**[1], **and Jie Zhou**[1]
[1]Pattern Recognition Center, Wechat AI, Tencent Inc, China
[2]Saarland Informatics Campus
aaronsu@tencent.com

Abstract

Large-scale pretrained language models have achieved SOTA results on NLP tasks. However, they have been shown vulnerable to adversarial attacks especially for logographic languages like Chinese. In this work, we propose RoCBert: a pretrained Chinese Bert that is robust to various forms of adversarial attacks like word perturbation, synonyms, typos, etc. It is pretrained with the contrastive learning objective which maximizes the label consistency under different synthesized adversarial examples. The model takes as input multimodal information including the semantic, phonetic and visual features. We show all these features are important to the model robustness since the attack can be performed in all the three forms. Across 5 Chinese NLU tasks, RoCBert outperforms strong baselines under three blackbox adversarial algorithms without sacrificing the performance on clean testset. It also performs the best in the toxic content detection task under human-made attacks.

1 Introduction

Large-scale pretrained models, by finetuning on sufficient annotated data, have been able to approach or even surpass human performance on many benchmark testsets (Peters et al., 2018; Radford et al.; Devlin et al., 2019; Liu et al., 2019; Brown et al., 2020). However, even pretrained with huge amounts of text, the models are still vulnerable under adversarial attacks like synonyms, word deletion/swapping, misspelling, etc (Li et al., 2019a; Jin et al., 2020; Sun et al., 2020a; Eger and Benz, 2020). These adversarial examples occur frequently in the real-world scenario and can be made either naturally (e.g., typos) or maliciously (e.g., to avoid auto detection of toxic content) [1].

Attacker	Text
phonetic	克(kè)比的精神值得永远学习
visual	科此的精神值得永远学习
character split	禾斗匕匕的精神值得永远学习
synonym	我(wǒ)科(kē)的精神值得永远学习
synonym + phonetic	蜗(wō)壳(ké)的精神值得永远学习
to pinyin	kebi的精神值得永远学习
to pinyin + unicode	keb1的精神值得学习永远
swap	科比的精神值得永远习学
insertion	科比的精神九值得永远学习
deletion	科比的神值得永远学习
Original: 科(kē)比(bǐ)的精神值得永远学习	
Translation: Kobe's spirit is worth studying forever.	

Table 1: Examples of various attackers. Contents in the brackets are corresponding pinyins of Chinese characters.

The lack of robustness with them can easily lead to large performance drop when testing in the noisy real-world traffic. The issue is particularly outstanding for logographic languages like Chinese since the attack can be either with the glyph character, pinyin (the romanized phonetic representations) or a combination of them (Wang et al., 2020; Li et al., 2020d; Zhang et al., 2020; Nuo et al., 2020). We show some examples in Table 1. The word "科比(Kobe)" can be replaced with synonyms, phonetically or visually similar words. The attacker can also replace the character with its pinyin then continue the attack in the alphabet-level ("keb1" in the table). The isolation of semantics and phonetics, and the rich set of glyph characters in written Chinese makes the attacking forms much more diverse than in alphabetic languages like English.

Current research works usually adopt two ways to defend adversarial attacks: (1) Run spell checking to correct the written errors before feeding to the prediction model (Pruthi et al., 2019; Li et al., 2020b; Mozes et al., 2021), and (2) Adversarial training, which adds adversarial example to the training data (Zang et al., 2020; Li et al., 2020a; Liu et al., 2020). For the former, Chinese spell checking itself is even a more difficult task because it requires the model to accurately recover the original text. Any tiny errors of the spell checking can

*Equal contribution.

[1]The concept of adversarial examples is quite wide. In this paper, we focus on adversarial examples that do *NOT* change the original semantics (Mozes et al., 2021)

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 921 - 931

lead to unpredicted model behaviors. For the latter, it is hard for the model to adapt to all adversarial variants only in the finetuning stage, especially when the training data is sparse (Meng et al., 2021).

To address the above challenges, we propose RoCBert, a Robust Chinese Bert pretrained with the contrastive learning objective by maximizing the label consistency under various adversarial examples. The adversarial examples are synthesized from an algorithm that encapsulates common types of attacks. We also consider combinatorial attacks where multiple types of attacks can be added on top of each other, which has never been considered in previous research. To defend attacks in all levels, we incorporate multimodal information into the encoder. The phonetic and visual features are inserted into one self-attention layer then dynamically fused in later layers. Across 5 standard NLU tasks and one toxic content detection task, we show the pretrained model achieves new SOTAs under various adversarial attackers.

In short, our contribution are (1) We propose pretraining a robust Chinese Bert with adversarial contrastive learning, such that the model can perform well on not only clean testbeds, but also adversarial examples. (2) The model is pretrained with synthesized adversarial examples covering combinations of semantic, phonetic and visual attacks. It takes as input multimodal features to handle all levels of possible attacks. (3) The pretrained model outperforms strong baselines across 5 NLU tasks and 1 toxic content detection task under various adversarial attackers. (4) We perform an extensive ablation studies for pretraining options and have a wide comparison with popular defending methods, which we hope will benefit future research.

2 Related Work

Adversarial attack There have been a lot of works showing the vulnerability of NLP models under adversarial examples (Li et al., 2020c; Garg and Ramakrishnan, 2020; Zang et al., 2020), which are understandable by humans yet lead to significant model prediction drops. There are usually two types of attacks: (1) semantic equivalent replacement, which can be synthesized by replacing words based on vector similarity (Jin et al., 2020; Wang et al., 2020), WordNet synonyms (Zang et al., 2020), masked prediction from pretrained models (Li et al., 2020c; Garg and Ramakrishnan, 2020; Li et al., 2020d), etc. (2) noise injection, which can be synthesized by adding/deleting/swapping words (Li et al., 2019a; Gil et al., 2019; Sun et al., 2020a), replacing words with phonetically or visually similar ones (Eger et al., 2019; Eger and Benz, 2020). For logographic languages like Chinese, the noise can be much more complex as it can be injected on both the glyph characters or romanized pinyins (Zhang et al., 2020; Nuo et al., 2020).

Adversarial defense The most common way of adversarial defense is adversarial training, which simply appends synthesized adversarial examples into the training data (Zang et al., 2020; Li et al., 2020a). Nonetheless, it relies only on the limited labeled training data. In contrast, the proposed RoCBert is pretrained on billions of text and can better adapted to diverse adversarial variants. Another popular way is to first remove the noise with off-the-shelf spell checkers, then feed the corrected text into the model (Li et al., 2020b). However, Chinese spell checking requires fully recovering the correct text and current model performances are far from satisfactory (Liu et al., 2021; Xu et al., 2021; Wang et al., 2021a). Any tiny error in the spell checking process can lead to unpredicted model behaviors. It also incurs significant latency to model prediction. RoCBert does not add additional latency and can perform well even if fully recovery is difficult due to its consistency-maximization pretraining objective. There have also been works on pretraining more robust models through virtual adversarial training and noise regularization (Yoo and Qi, 2021; Wang et al., 2021b; Meng et al., 2021), but they perform poorly on man-made attacks.

3 Adversarial Example Synthesis

3.1 Attacking Chinese Characters

As we focus on Chinese in this paper and Chinese characters are much more diverse than in alphabetical languages, we design the following 5 Chinese-specific attacking algorithms first.

phonetic: Replace a Chinese character with a random homonym (ignoring diacritics). For polyphones, we consider the 2 most common pinyins [2].

Visual: Replace Chinese characters with their visually similar characters (with the similarity table in the Kanji Database Project) [3].

Character Split: Split one character into two parts with every part still being (or visually similar to) a valid Chinese character. We follow the Chinese

[2] https://unicode.org/charts/unihan.html

[3] http://kanji-database.sourceforge.net/

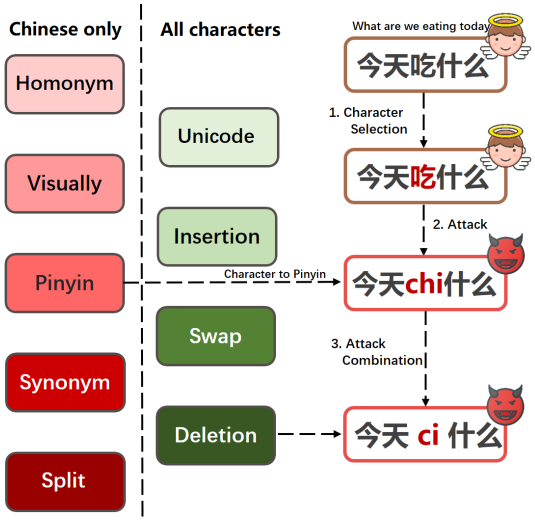

Figure 1: Adversarial example synthesis process.

splitting dictionary [4], which contains 17,803 splitting ways for Chinese characters in total.

Synonym: Segment Chinese characters into words with the jieba tokenizer [5], then randomly replace the word with one of its synonyms. Two words are treated as synonym if they share a similarity score of over 0.75 [6]. We only replace adjectives or nouns as we find other words can be hardly replaced without changing the semantics.

Character to Pinyin: Replaces the character into its pinyin representation (without diacritics).

3.2 Attacking Other Characters

Apart from Chinese characters, there are often other characters like the pinyin, numbers, punctuations and foreign words in the Chinese corpus. The following 4 types of attacks apply to not only Chinese characters, but also all other characters.

Unicode: Randomly sample one of the visually similar unicodes as a replacement [7].

Random Insertion: Sample one character from the vocabulary set, then randomly insert the character to the left or right of the current character.

Swap: Swap the character with its neighbor.

Deletion: Delete the character directly.

Examples of all types of attacks are in Table 1.

3.3 Synthesis Process

The synthesis process of adversarial examples is as follow: Given one sentence, we first select several characters to attack. For each selected character, we then combine the above mentioned character-level attacking algorithms [8] to get its attacked form.

Attack Ratio: The attack ratio γ decides how many characters we will attack. Let n_c be the number of characters in the sentence, we define γ as:

$$\begin{aligned} \gamma &= \min(\max(\text{int}(\epsilon), 1), n_c) \\ \epsilon &\sim \mathcal{N}(\max(1, 0.15 n_c), 1) \end{aligned} \tag{1}$$

where the int function rounds ϵ into the closest integer. The intuition is that we want to attack 15% of the characters on average [9]. If the sentence is short, we will make sure to attack at least one character. We insert normal Gaussian noise on top of the average ratio to add some randomness.

Character Selection: There have been many research works showing that attacking informative words is more effective than random words (Li et al., 2019a; Sun et al., 2020a). Therefore, we decide the chance of one character c_i being selected based on its informativeness in the sentence. Let $w(c_i)$ denote the word c_i belongs to, the informative score for c_i is counted as the difference of the language model loss after deleting $w(c_i)$ (denoted as $\mathcal{L}(\triangledown w(c_i))$ (Li et al., 2016) [10]. The chance that c_i will be selected to be attacked is:

$$p(c_i) = \frac{e^{\mathcal{L}(\triangledown w(c_i))}}{|w(c_i)| \sum_{j=1}^{n_w} e^{\mathcal{L}(\triangledown w_j)}} \tag{2}$$

where n_w is the number of words in the sentence. $|w(c_i)|$ means the number of characters in $w(c_i)$ such that characters in the same word have equal chances to be selected.

Attack Combination: There can be combinations of attacks for one character. For example, we can transfer one Chinese character into its pinyin then continue to attack it in the alphabet level ("to pinyin + unicode" in Table 1). We define it as a sequential process where a new attack can be added on top at each step. Specifically, the new character $\tilde{c}$ after all the attack combinations applied to c is:

$$\begin{aligned} \tilde{c} &= A_{S(c)} \circ \cdots \circ A_2 \circ A_1(c) \\ p(S(c) = k) &= q(1-q)^{k-1} \end{aligned} \tag{3}$$

[4] https://github.com/kfcd/chaizi

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

[6] https://github.com/chatopera/Synonyms

[7] http://www.unicode.org/Public/security/revision-03/confusablesSummary.txt

[8] For synonym replacement which applies in the word level, we apply it on the word that the selected character belongs to.

[9] The ratio is chosen by manual annotation. 15% is the highest ratio we can attack without hurting human reading.

[10] We use ChineseGPT (Zhang et al., 2021) as the language model, so "word" here means the subword token defined in the vocabulary of ChineseGPT.

wher $\circ$ means applying a new attacking algorithm A to the output of the last step. At each step i, the attacking algorithm A_i is randomly selected from all algorithms that are applicable to the output from step $i-1$. $S(c)$ is the number of attacking steps applied to c, which follows an exponentially decay function. We set $q = 0.7$ empirically. The full process of adversarial example synthesis is illustreated in Figure 1.

4 Multimodal Contrastive Pretraining

With the above-mentioned algorithm to sample adversarial examples, we can pretrain the model with the multimodal contrastive learning objective.

4.1 Multimodal Features

We follow the standard Bert architecture (Devlin et al., 2019) as our backbone, based on which we integrate phonetic and visual features for input text.

Feature Representation: For every character c in our vocabulary, apart from the standard semantic embedding $Se(c)$, we include two more vectors $Ph(c)$ and $Vi(c)$ to encode its phonetic and visual features respectively. If c is not a Chinese character, it has its own phonetic vector. Otherwise, $Ph(c) = \sum_{k \in pinyin(c)} Ph(k)$ where $pinyin(c)$ is its pinyin sequence. $Vi(c)$ is extracted from its 32×32 image $I(c)$. The image is in simsun (宋体) for Chinese characters and arial for others, the default fonts for most online text. $Vi(c)$ is defined as:

$$Vi(c) = \text{LayerNorm}(M^T \text{ResNet18}(I(c))) \quad (4)$$

M is a learnable matrix and we utilize Resnet18 (He et al., 2016) to map $I(c)$ into a one-dimentional vector (freezed during training).

Visual Representation Pretrain: To get an reasonable initialization, we add another pretraining stage only for the visual representation. Phonetic representations are randomly initialized [11]. M in Eq 4 is pretrained with the same contrastive loss as in Eq 5. The positive sample for the character c is its visually adversarial form $\tilde{c} = \mathcal{A}(c)$. $\mathcal{A} \sim \mathbb{U}$(visual, character split, unicode), which means uniform sampling from the three visual attacking algorithms mentioned in §3. If c is split into two characters c_1 and c_2, we sum the visual representation of the two split characters $Vi(\tilde{c}) = Vi(c_1) + Vi(c_2)$. The negative samples are all other characters in the same batch. After training, visually similar characters will be close in their representation space.

Feature Integration: A straightforward way to integrate these multimodal features is to fuse them before fed to the encoder (Sun et al., 2021; Liu et al., 2021). However, three features will be given equal weights and the model cannot dynamically attend to only useful features. Another way is a two-step encoding which first decides the weight, then encode with selective attention (Xu et al., 2021), but it will significantly slow down the system. We propose a lightweight fusion method *layer-insert*, which insert multimodal features in only one encoder layer. Let $H^k(i)$ denote the representation of the ith word in the kth layer, we insert by:

$$\begin{aligned} W_1 &= K_1^T H^k(i) H^k(i) V_1 \\ W_2 &= K_2^T H^k(i) Ph(i) V_2 \\ W_3 &= K_3^T H^k(i) Vi(i) V_3 \\ H^k(i) &= \frac{W_1 H^k(i) + W_2 Ph(i) + W_3 Vi(i)}{W_1 + W_2 + W_3} \end{aligned}$$

where $Ph(i)$ and $Vi(i)$ are the phonetic and visual representations and K_j/V_j are learnable matrices. Intuitively we can use the layer 0 to $k-1$ to decide the weights of three multimodal representations and use the rest layers for sentence representation learning. It allows dynamic fusion according to sentence context yet adds marginal complexity.

4.2 Model Loss

The model loss has two components: the contrastive learning loss and the standard masked language model (MLM) loss.

Contrastive Learning: The idea of contrastive learning (Chen et al., 2020; Kim et al., 2021) is that the representation space should be made closer for similar (positive) samples and farther for dissimilar (negative) samples. For each sentence, we treat its adversarial form (obtained from the algorithm in §3) as positive and all the other sentences in the same batch as negative. Given a batch with N sentences, the loss to the ith sentence s_i is:

$$\mathcal{L}_c(i) = -\log \frac{e^{sim(s_i, \tilde{s_i})/\tau}}{\sum_{j=1}^{N} e^{sim(s_i, s_j)/\tau}}, \quad (5)$$

where τ is a temperature hyperparameter and $\tilde{s_i}$ is the adversarial example synthesized from s_i. We set $\tau = 0.01$ based on our pilot experiments and

[11] We show in Section 5.3 that pretraining is necessary for visual features not but for phonetic features.

define $sim(s_i, \tilde{s}_i)$ as $\frac{h_i^\top \tilde{h}_i}{\|h_i\| \cdot \|\tilde{h}_i\|}$, which is the cosine similarity in their representation space h_i and $\tilde{h}_i$.

Mix with MLM: We mix the contrastive learning loss with the standard masked language model (MLM) loss (Devlin et al., 2019) to enable both sentence and word level representation learning. We use a character-based tokenizer because (1) Chinese characters as themselves stand for individual semantic units (Li et al., 2019b) and (2) char-based models are much more robust under noisy and adversarial scenarios (El Boukkouri et al., 2020). For Chinese characters, we use two masking strategies – Whole Word Masking (WWM) and Char Masking (CM) because a large number of words in Chinese consist of multiple characters (Cui et al., 2019; Sun et al., 2021). The contrastive learning loss and the MLM loss are equally weighted.

5 Experiments

5.1 Experiment Setup

Model Details We use a vocabulary size of 16224, out of which 14642 are Chinese characters. We provide two versions of RoCBert: base and large. The base version has 12 layers/heads with 768 hidden neurons. It is trained for 600k steps with a batch size of 4k, learning rate of 1e-4 and warmup rate of 25k steps. The large version has 48 layers and 24 attention heads with 1024 hidden neurons. It is trained for 500K steps with a learning rate of 3e-4, warmup of 70K steps and batch size of 8k.

Pretraining Details Following the common practice, we pretrain our model on 2TB text extracted from a mixture of THUCTC [12], Chinese Wikipedia and Common Crawl. Models are trained on 64 NVIDIA V100 (32GB) GPUs with FP16 and ZERO-stage-1 optimization (Rasley et al., 2020). To make better use of the GPU, we train our model with PatricStar [13] which applies a dynamic memory scheduling with a chunk-based memory management module (Fang et al., 2021). The memory management offloads everything but the current computing part of the model to CPUs. This results in training a much larger model within the same hardware environment. The chunk-based memory management takes advantage of the linear structure of the transformer-based model, so that it will inherently prefetch the upcoming layers to GPUs.

Baseline Models We compare our model with SOTA pretrained Chinese models: (1) MBert-Chinese (Devlin et al., 2019), (2) Bert-wwm (Cui et al., 2019), (3) MacBert (Cui et al., 2020), (4) Ernie-gram (Sun et al., 2019, 2020b) and (5) ChineseBert (Sun et al., 2021). BERT-wwm continues pretraining from MBert-Chinese with the Whole Word Masking pretraining strategy. MacBERT applies the MLM-As-Correlation (MAC) pretraining strategy as well as the sentence-order prediction (SOP) task. ERNIE-gram adopts various masking strategies including token-level, phrase-level and entity-level masking to pretrain BERT on largescale heterogeneous data. Chinese-Bert is pretrained with the glyph and phonetic features.

Tasks We test our model on 5 standard Chinese NLU tasks and one toxic detection tasks. The 5 NLU tasks are: (1) ChnSentiCorp, Chinese sentiment classification with 2k training data [14], (2) TNEWS: news title classification with 50k training data, (3) AFQMC: question matching with 34k training data, (4) CSL, keyword recognition from paper abstracts with 20k training data ChnSentiCorp: 2k (Xu et al., 2020) and (5) CMNLI, Chinese Multi-Genre NLI with 390k data (Conneau et al., 2018). Toxic detection can server as a task with "human-made" attacks in contrast with the synthesized ones. It is collected from user interactions (written) with a popular online conversational platform, where users sometimes use various man-made attacks to avoid automatic system filtering of junk ads, porn and abusive information. We manually annotate 50k user inputs and identify 2k toxic contents (positive), out of which 90% are in adversarial forms. We randomly sample 2k negative text then split the whole into train/dev/test with 8:1:1.

Attacker We test the model performance under three different attackers (all untargeted as we do not need restrictions to the target class): (1) *ADV*, our own attacking algorithm, (2) *TextFooler* (Jin et al., 2020), a black-box algorithm replacing important words with semantically similar ones and (3) *Argot* (Zhang et al., 2020), a black-box attacking algorithm considering Chinese-specific features. We set the maximum attacking ratio for all the three algorithms as 20%. TextFooler is originally designed for English, we reimplement it with corresponding pretrained Chinese-version models.

[12] https://github.com/thunlp/THUCTC

[13] https://github.com/Tencent/PatrickStar

[14] We use the small version of training data to test the few-shot capability of models.

Model	Clean	ADV	TextFooler	Argot
		Base		
MBert	91.16	58.57	62.29	46.65
Bert-wwm	91.27	59.28	63.22	44.52
MacBert	91.33	59.72	63.18	44.34
Ernie-gram	90.76	57.81	60.20	42.71
ChineseBert	91.01	60.07	65.73	47.78
RoCBert	**91.45**	**81.62**	**83.11**	**68.40**
		Large		
MacBert	92.05	55.92	45.75	41.83
RoCBert	**92.58**	**83.17**	**85.74**	**69.40**

Table 2: Performance on ChnSentiCorp

Model	Clean	ADV	TextFooler	Argot
		Base		
MBert	56.84	53.76	42.05	40.18
Bert-wwm	57.44	54.12	45.25	40.76
MacBert	57.53	54.41	45.10	41.94
Ernie-gram	57.30	52.58	43.02	41.16
ChineseBert	57.65	55.74	51.01	50.27
RoCBert	**58.64**	**57.14**	**52.05**	**52.21**
		Large		
ChineseBert	59.65	55.92	50.75	51.83
RoCBert	**59.98**	**59.17**	**54.74**	**54.46**

Table 3: Performance on TNEWS

Model	Clean	ADV	TextFooler	Argot
		Base		
MBert	74.07	72.04	57.69	51.24
Bert-wwm	75.07	72.40	57.58	51.05
MacBert	74.79	72.08	57.37	50.78
Ernie-gram	75.42	71.07	56.81	50.34
ChineseBert	73.77	72.59	57.92	52.41
RoCBert	**75.48**	**74.11**	**62.95**	**62.16**
		Large		
Ernie-gram	76.35	70.92	58.04	50.64
RoCBert	**77.48**	**76.43**	**65.85**	**64.97**

Table 4: Performance on AFQMC

5.2 Experiment Results

Chinese NLU Results We show the results on 5 Chinese NLU tasks in tables 2 to 6. For every task, we report the model accuracy measured in the clean testset and the adversarial testsets under 3 adversarial algorithms *ADV*, *TextFooler* and *Argot*. We report the performance of all base-version models for a fair comparison. We select the best-performed base-version model to test its large-version performance and compare it with RoCBert. As can be seen, our attacking algorithm *ADV* do not affect much on TNEWS, AFQMC and CSL because they rely more on the global sentence structure instead of individual words. On tasks like sentiment classification and NLI, single words contribute mostly to the model decision and therefore the attacking can lead to significant performance drop. *Argot* and *TextFooler* lead to more drop compared with *ADV* because they explicitly select words that affect the model decisions most while *ADV* selects words to attack based on the general language model scores. *Argot* is more effective than *TextFooler* because it tailors its character replacement to consider Chinese-specific features. Overall RoCBert *outperforms other models over all attacking algorithms on all the 5 tasks*. Even in the clean dataset, it performs the best on 4 out of the 5 tasks. *ChineseBert* performs the second under various attacks because it also considers multimodal features during its pretraining same as RoCBert, which further confirms the importance of using mulimodal features in Chinese language pretraining.

Toxic Content Detection Results We train all models in the toxic content detection task. As can be seen in Table 7, RoCBert *outperforms all other models over 4 metrics*. This confirms the its effectiveness at capturing the true semantics regardless of its adversarial form. The difference among models is smaller because they have all been finetuned on this task. All models can get adapted to different forms of attacks in the training process while the tables 2 to 6 are testing the zeroshot generalization to unknown attacks.

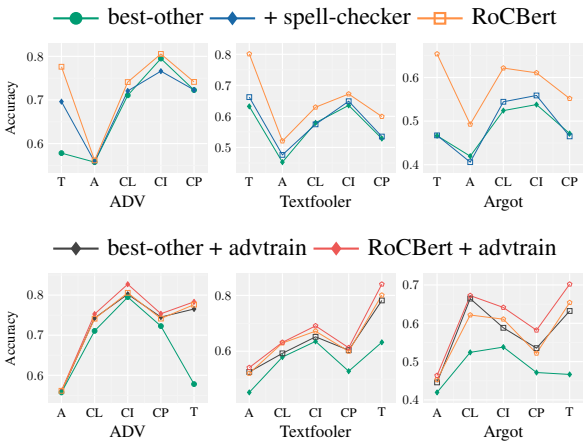

Figure 2: Defending Method Comparison on CI(CMNLI) CP(ChnSentiCorp), T(TNEWS), A(AFQMC) and CL(CSL).

Defending Method Comparison We further compare RoCBert with two other popular ways of defending adversarial attack: (1) run a spell-checker before fed to the model and (2) adversarial training (*advtrain*) which augments training data with adversarial examples. We add these two de-

Model	Clean	ADV	TextFooler	Argot
			Base	
MBert	81.83	78.28	61.06	52.40
Bert-wwm	81.50	79.08	61.68	53.41
MacBert	81.97	78.34	61.75	52.35
Ernie-gram	82.70	79.53	63.54	53.66
ChineseBert	81.77	78.69	61.27	53.79
RoCBert	**83.83**	**82.56**	**69.29**	**63.07**
			Large	
Ernie-gram	83.05	79.42	61.85	57.43
RoCBert	**85.28**	**83.59**	**70.13**	**66.38**

Table 5: Performance on CSL

Model	Clean	ADV	TextFooler	Argot
			Base	
MBert	80.53	69.57	50.21	45.52
Bert-wwm	80.79	68.54	50.46	44.26
MacBert	81.01	69.94	49.86	42.07
Ernie-gram	**82.22**	68.83	50.77	44.69
ChineseBert	81.42	72.27	52.85	47.15
RoCBert	81.27	**74.14**	**59.95**	**55.17**
			Large	
Ernie-gram	82.36	70.11	52.45	45.82
RoCBert	**82.38**	**76.83**	**60.26**	**56.64**

Table 6: Performance on CMNLI

fending methods on top of the best-performed base model (on clean testsets) in different tasks: ChineseBert for TNEWS, Ernie-gram for AFQMC, CSL and CMNLI, MacBert for ChnSentiCorp. We apply the spell-checker in Cheng et al. (2020). The results are visualized in Figure 2. We can see that spell checking improves the performance only marginally and sometimes even hurt the performance (best-other under ADV in CI). The reason could be that the spell checker performs poorly for out-of-domain adversarial examples. The errors could be propagated and further reduce the performance. *advtrain* can significantly benefit the performance, but note that it explicitly "peeps" at the adversarial algorithm applied in the testset while RoCBert is not aware of the testing adversarial algorithm. Nevertheless, it is still comparable and in some cases even outperforms *advtrain*. By combining RoCBert and advtrain, the model robustness can be further improved.

5.3 Ablation Study

We perform a set of ablation studies to understand the choice of different components in RoCBert. All models in this section are pretrained with the *same base architecture and hyperparameters for one epoch on 1M sampled training text* then tested in TNEWS. The results are shown in Table 8.

Model	Acc	Precision	Recall	F1
		Base		
MBert	85.11	87.12	81.35	84.13
Bert-wwm	85.70	87.30	81.37	84.23
MacBert	85.26	87.24	81.35	84.19
Ernie-gram	85.94	87.43	81.38	84.29
ChineseBert	85.52	87.29	81.36	84.22
RoCBert	**87.10**	**89.26**	**83.14**	**86.42**
		Large		
Ernie-gram	87.30	88.96	82.57	85.64
RoCBert	**88.49**	**90.36**	**84.25**	**87.20**

Table 7: Performance on Toxic Detection

Setting	Clean	ADV	Textfooler	Argot
Best	55.38	**52.23**	**47.72**	**44.59**
		Model Loss		
MLM	54.63	48.58	38.63	33.75
Contrastive	54.97	50.73	41.80	39.25
		Tokenization		
bpe	**55.40**	48.64	38.19	35.67
char-cnn	53.23	49.45	44.37	41.44
		Multimodal		
- vis-pretrain	53.29	51.18	44.42	40.08
- vis	53.35	51.20	45.45	41.86
- pho	54.71	51.18	46.02	42.08
+ pho-pretrain	54.96	51.95	47.03	43.56
		Architecture		
Sum	54.63	52.06	46.57	43.27
Concatenate	55.13	52.14	46.69	43.84
Two-step	55.09	51.81	45.39	42.67
		Insert Layer		
Layer 0	55.10	52.02	47.46	44.27
Layer 4	54.63	51.95	47.35	44.33
Layer 7	54.43	51.76	46.83	44.08
Layer 10	54.25	50.98	46.20	43.56

Table 8: Ablation studies on *TNEWS* with different settings. *Best* indicates the best setting used in RoCBert.

Loss To study the effects of the loss function used in the pretraining stage. We tried two other settings: (1) *contrastive only*, where the model is pretrained only with the contrastive learning loss in Eq 5 and (2) *MLM-only*, where the model is pretrained only with the MLM objective as in standard Bert. We can see that both options lowers down the model performance. By combining both loss, the model can be robust under adversarial attacks without affecting the performance in clean data.

Tokenization It has been widely demonstrated that char-based tokenization is preferred for Chinese characters (Li et al., 2019b), but it is rather unclear how we should model pinyins and non-Chinese words. We try different tokenization methods for non-Chinese characters: (1) *bpe* (Sennrich et al., 2016). We set the vocabulary as 20k and train the split on the training data (after convert-

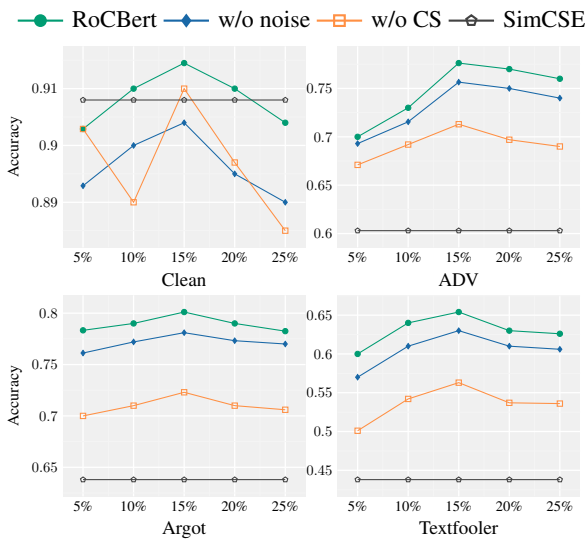

Figure 3: Ablation study with varying attacking ratio, w/o Gaussian noise, w/o character selection (CS) and SimCSE.

ing all Chinese characters into pinyin). (2) *char-cnn* (Zhang et al., 2015), which process each character individually but get the pinyin embedding with a char-cnn. The best setting in RoCBert used *char-sum* which processes each character individually and set the pinyin embedding as the sum of its character embeddings. We can see that bpe hurt the performance. This might be because the bpe split is trained on clean data only. For adversarial examples, the letters in pinyins can be easily perturbed and break its vocabulary. Char-based tokenization is more robust under adversarial attacks. Char-cnn does not lead to improvement here, probably because there are a limited combination of letters in Chinese pinyins ($\sim$ 400), each pinyin can usually be uniquely identified by its bag of characters without the need of order information.

Multimodal feature We tried removing the visual feature pretraining as mentioned in §4 and observe the performance drop (-vis-pretrain). It is even worse than removing the visual feature completely (-vis), suggesting the pretraining for visual features is essential, without which the model can be hard to learn meaningful visual features. The phonetic feature is less crucial than visual features but also brings positive improvement. By adding a pretraining stage for the phonetic features too (+pho-pretrain), the improvement is very marginal. As the phonetic features are also based on character embeddings, it might be easier for the model to automatically learn the phonetic features compared with the visual features.

Multimodal integration We compare our proposed *layer-insert* with three other ways of integrating multimodal features: (1) *sum* (Liu et al., 2021), which sums the multimodal embeddings, (2) *concatenation*, which concatenate(Sun et al., 2021), which concatenate the multimodal embeddings then fuse with an MLP layer, (3) *two-step* (Xu et al., 2021), which first determine the weight of different embeddings then fuse to the encoder. We can see that RoCBert performs best with only marginal computational overhead by updating the encoder representation in one layer.

Insert Layer We further analyze the effects of the insertion layer. Our best setting inserts the multimodal features in layer 1 for the base model and layer 3 for the large model. From Table 8, we can see that when inserting them in the upper layer 4,7 and 10, the performance gradually drops, suggesting an earlier insert is helpful for the model to incorporate these features in-depth. However, inserting them in layer 0 is also worse since the model can only learn weight among multimodal features solely from bag of words.

Attacking Algorithm We change the settings in our attacking algorithm to see the effects in Figure 3. We can see the attacking ratio can neither be too small nor too large. 15% is a sweet spot for pretraining. The Gaussian noise added in Eq 1 also brings positive effects consistently, suggesting we should not use a fixed attacking ratio in the pretraining stage. The character selection is also crucial and removing it significantly reduces the performance. To show whether it is necessary to adopt our attacking algorithm with complex combinations of attacking forms. We further compare with pretraining the model with SimCSE (Gao et al., 2021), an algorithm which uses drop out as the noise instead of our adversarial examples. We can see that SimCSE is rarely helpful under different attacks. This suggests it is important to define rule-based attacking algorithms to better fit the real-world attacks. *General drop-out regularizations cannot adapt well to complex real-world attacks.*

6 Conclusion

We present RoCBert: the first pretrained Chinese language model that is robust under various forms of adversarial attacks. It is pretrained with the multimodal contrastive learning objective and achieves the best performance on 5 Chinese NLU tasks un-

der three different attacking algorithms without negative effects on clean testsets. It also significantly outperforms the others in the toxic content detection task. Extensive ablation studies are provided to benefit future research.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Xingyi Cheng, Weidi Xu, Kunlong Chen, Shaohua Jiang, Feng Wang, Taifeng Wang, Wei Chu, and Yuan Qi. 2020. Spellgcn: Incorporating phonological and visual similarities into language models for chinese spelling check. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 871–881.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for chinese natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 657–668.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Steffen Eger and Yannik Benz. 2020. From hero to z\'eroe: A benchmark of low-level adversarial attacks. *arXiv preprint arXiv:2010.05648*.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text processing like humans do: Visually attacking and shielding nlp systems. *arXiv preprint arXiv:1903.11508*.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. Characterbert: Reconciling elmo and bert for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915.

Jiarui Fang, Yang Yu, Zilin Zhu, Shenggui Li, Yang You, and Jie Zhou. 2021. Patrickstar: Parallel training of pre-trained models via a chunk-based memory management. *arXiv preprint arXiv:2108.05818*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. *arXiv preprint arXiv:2004.01970*.

Yotam Gil, Yoav Chai, Or Gorodissky, and Jonathan Berant. 2019. White-to-black: Efficient distillation of black-box adversarial attacks. *arXiv preprint arXiv:1904.02405*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. 2021. Self-guided contrastive learning for bert sentence representations. *arXiv preprint arXiv:2106.07345*.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020a. Contextualized perturbation for textual adversarial attack. *arXiv preprint arXiv:2009.07502*.

Jinfeng Li, Tianyu Du, Shouling Ji, Rong Zhang, Quan Lu, Min Yang, and Ting Wang. 2020b. Textshield: Robust text classification based on multimodal embedding and neural machine translation. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1381–1398.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019a. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020c. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*.

Linyang Li, Yunfan Shao, Demin Song, Xipeng Qiu, and Xuanjing Huang. 2020d. Generating adversarial examples in chinese texts using sentence-pieces. *arXiv preprint arXiv:2012.14769*.

Xiaoya Li, Yuxian Meng, Xiaofei Sun, Qinghong Han, Arianna Yuan, and Jiwei Li. 2019b. Is word segmentation necessary for deep learning of chinese representations? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3242–3252.

Hui Liu, Yongzheng Zhang, Yipeng Wang, Zheng Lin, and Yige Chen. 2020. Joint character-level word embedding and adversarial stability training to defend adversarial text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8384–8391.

Shulin Liu, Tao Yang, Tianchi Yue, Feng Zhang, and Di Wang. 2021. Plome: Pre-training with misspelled knowledge for chinese spelling correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2991–3000.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Zhao Meng, Yihan Dong, Mrinmaya Sachan, and Roger Wattenhofer. 2021. Self-supervised contrastive learning with adversarial perturbations for robust pretrained language models. *arXiv preprint arXiv:2107.07610*.

Maximilian Mozes, Pontus Stenetorp, Bennett Kleinberg, and Lewis Griffin. 2021. Frequency-guided word substitutions for detecting textual adversarial examples. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 171–186.

Cheng Nuo, Guo-Qin Chang, Haichang Gao, Ge Pei, and Yang Zhang. 2020. Wordchange: Adversarial examples generation approach for chinese text classification. *IEEE Access*, 8:79561–79572.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Danish Pruthi, Bhuwan Dhingra, and Zachary C Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. 2020a. Adv-bert: Bert is not robust on misspellings! generating nature adversarial samples on bert. *arXiv preprint arXiv:2003.04985*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020b. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8968–8975.

Zijun Sun, Xiaoya Li, Xiaofei Sun, Yuxian Meng, Xiang Ao, Qing He, Fei Wu, and Jiwei Li. 2021. Chinesebert: Chinese pretraining enhanced by glyph and pinyin information. *arXiv preprint arXiv:2106.16038*.

Baoxin Wang, Wanxiang Che, Dayong Wu, Shijin Wang, Guoping Hu, and Ting Liu. 2021a. Dynamic connected networks for chinese spelling check. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2437–2446.

Boxin Wang, Boyuan Pan, Xin Li, and Bo Li. 2020. Towards evaluating the robustness of chinese bert classifiers. *arXiv preprint arXiv:2004.03742*.

Dong Wang, Ning Ding, Piji Li, and Hai-Tao Zheng. 2021b. Cline: Contrastive learning with semantic negative examples for natural language understanding. *arXiv preprint arXiv:2107.00440*.

Heng-Da Xu, Zhongli Li, Qingyu Zhou, Chao Li, Zizhen Wang, Yunbo Cao, Heyan Huang, and Xian-Ling Mao. 2021. Read, listen, and see: Leveraging multimodal information helps chinese spell checking. *arXiv preprint arXiv:2105.12306*.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, et al. 2020. Clue: A chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772.

Jin Yong Yoo and Yanjun Qi. 2021. Towards improving adversarial training of nlp models. *arXiv preprint arXiv:2109.00544*.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. *arXiv preprint arXiv:1910.12196*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

Zhengyan Zhang, Xu Han, Hao Zhou, Pei Ke, Yuxian Gu, Deming Ye, Yujia Qin, Yusheng Su, Haozhe Ji, Jian Guan, et al. 2021. Cpm: A large-scale generative chinese pre-trained language model. *AI Open*, 2:93–99.

Zihan Zhang, Mingxuan Liu, Chao Zhang, Yiming Zhang, Zhou Li, Qi Li, Haixin Duan, and Donghong Sun. 2020. Argot: Generating adversarial readable chinese texts.