Joint Character-Level Word Embedding and Adversarial Stability Training to Defend Adversarial Text

Hui Liu,[1,2] **Yongzheng Zhang,**[1,2] **Yipeng Wang,**[1,2] **Zheng Lin,**[1,2] **Yige Chen**[1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{liuhui, zhangyongzheng, wangyipeng, linzheng, chenyige}@iie.ac.cn

Abstract

Text classification is a basic task in natural language processing, but the small character perturbations in words can greatly decrease the effectiveness of text classification models, which is called character-level adversarial example attack. There are two main challenges in character-level adversarial examples defense, which are out-of-vocabulary words in word embedding model and the distribution difference between training and inference. Both of these two challenges make the character-level adversarial examples difficult to defend. In this paper, we propose a framework which jointly uses the character embedding and the adversarial stability training to overcome these two challenges. Our experimental results on five text classification data sets show that the models based on our framework can effectively defend character-level adversarial examples, and our models can defend 93.19% gradient-based adversarial examples and 94.83% natural adversarial examples, which outperforms the state-of-the-art defense models.

1 Introduction

Text classification is a basic task in natural language processing (NLP). In recent years, (Kim 2014; Zhang, Zhao, and LeCun 2015; Lai et al. 2015) apply deep neural networks to improve the text classification models. However, (Goodfellow, Shlens, and Szegedy 2014) show that small perturbations in test inputs can fool the state-of-the-art deep neural networks based classifiers. To improve the robustness of these deep neural networks based text classification models, many recent works (Liang et al. 2017; Gao et al. 2018; Li et al. 2018) have focused on adversarial examples, which are well-crafted by adding small perturbations to original examples, to fool the trained text classification models. For example, as shown in Table 1, we only swap two chars in the word "never" to generate an adversarial word "enver". This small perturbation can cause the well-trained text classification model to misclassify the entire sentence. In general, the generation methods of adversarial examples in text classification are mainly divided into three categories, which are character-level (char-level) adversarial examples (Gao et al. 2018), word-level adversarial examples (Papernot et al. 2016) and sentence-level adversarial examples (Liang et al. 2017). In this paper, we mainly discuss the char-level adversarial examples, which are the most frequently used method to fool the state-of-the-art text classification models (Gao et al. 2018; Li et al. 2018) in recent years. Adversarial examples of text classification reveal the vulnerability of the text classification models, and this vulnerability greatly hinders the application of text classification, such as phishing and spam detection systems.

Example	Input	*Prediction*
Original	never coming here again found hair in my to go food .	Negative
Adversarial	<u>enver</u> coming here again found hair in my to go food.	Positive
Original	i just called these guys talk about customer service, they were excellent .	Positive
Adversarial	i just called these guys talk about customer service, they were <u>exciellent</u> .	Negative

Table 1: The char-level adversarial examples in Yelp Review Polarity corpus.

However, the defense of char-level adversarial examples has two main challenges, namely out-of-vocabulary (OOV) words and distribution differences. The OOV words are caused by the word embedding mechanism, which is supposed to map the words in the vocabulary to fixed high-dimensional vectors. But when a word is out of the vocabulary, it will be replaced by the uninformed word UNK. In actual, there is a high probability that the adversarial words are OOV words to the word embedding based models, which brings the loss of many informative words. For example, as shown in Table 1, we only insert a char i into the word "excellent", and we will get an adversarial OOV word "exciellent" to the word embedding based model. Besides, the char-level adversarial examples disrupt the intrinsic structure of word including prefix and suffix, which causes the distribution difference between the training set and adversarial examples. This distribution difference violates the assumption of independent identical distribution in machine learning and leads to the poor performance of well-trained text classification models to adversarial examples.

To avoid the OOV words during the text processing, (Gao et al. 2018; Li et al. 2018) conduct a spell corrector to correct adversarial words before inputting the text data into the classification models. However, traditional spell correctors have been verified to be ineffective when the edit distance (Levenshtein 1966), a metric that measures the similarity of two words, between the original word and the adversarial word exceeds one (Gao et al. 2018). For instance, as shown in Table 1, the edit distance between the original word "never" and the adversarial word "enver" is two. The spell corrector tool autocorrector[1] cannot correct the adversarial word "enver" to "never" successfully. Besides, (Pruthi, Dhingra, and Lipton 2019) uses semi-character based RNN (ScRNN) (Sakaguchi et al. 2017) to correct the adversarial words, which performs very well in most of char-level adversarial sentences containing less than two adversarial words. But in our more challenging adversarial examples with many adversarial words, the ScRNN model shows poor performance when it tries to correct all the adversarial words. In addition, to ameliorate the distribution difference, (Belinkov and Bisk 2017; Cheng et al. 2018) propose word-based adversarial stability training, which is to train the model on both original texts and adversarial texts simultaneously to defend the char-level perturbations in neural machine translation. Unfortunately, this model training method does not solve the OOV words problem under char-level adversarial examples. Therefore, the spell corrector and word-based adversarial stability training cannot defend char-level adversarial examples directly.

In this paper, we propose a novel framework as shown in Figure 1, which jointly utilizes character (char) embedding and adversarial stability training to overcome the two challenges in char-level adversarial examples. To verify the effectiveness of our framework, we generate two kinds of char-level adversarial examples, namely the gradient-based adversarial examples and the natural adversarial examples, on five text classification data sets. We implement two models based on the framework and apply them on these two kinds of adversarial examples. The experimental results show that our framework can defend these char-level adversarial examples effectively and greatly outperforms the state-of-the-art models. To the best of our knowledge, this is the first work to defend char-level adversarial examples by jointly using char embedding and adversarial stability training.

We conclude our contributions as follows:

- We propose a new framework, which jointly utilizes char embedding and adversarial stability training, to defend char-level adversarial examples in text classification. This framework is easily applied to all the word-level text classification models to enhance their robustness.
- We construct two kinds of char-level adversarial examples, which can simulate both deliberate attacks and careless inputs, to evaluate the robustness of our framework.
- We implement two models based on our framework and conduct them on five text classification data sets. The experimental results show that both of these two models get remarkable defense effectiveness to the char-level adversarial examples.

[1]https://pypi.org/project/autocorrect/

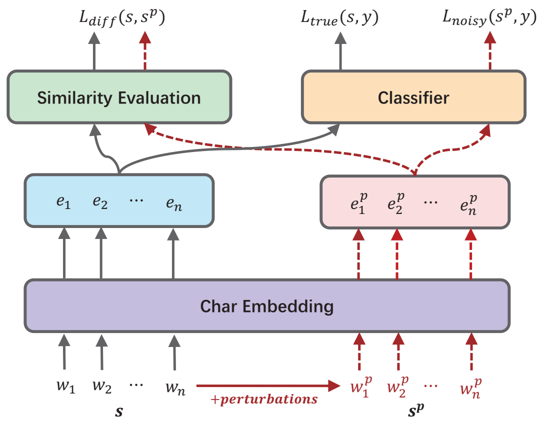


Figure 1: Our proposed framework to defend char-level adversarial examples.

2 Model

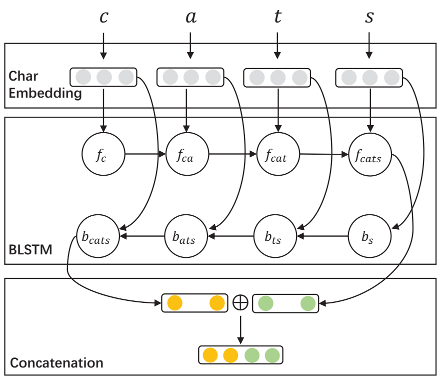
In this section, we will describe our proposed framework in detail.

Framework

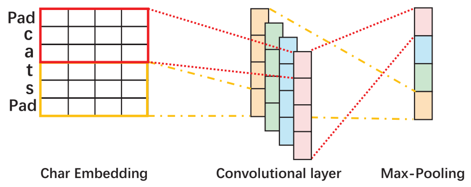
The basic purpose of our framework is to overcome the OOV words and distribution differences in char-level adversarial examples. As shown in Figure 1, our framework conducts char embedding to generate the char-level word representation $\boldsymbol{e_i}$ $(i = 1, 2, ..., n)$ for the word w_i in sentence s. The char-level word representation $\boldsymbol{e_i}$ is generated by fine-grained characters, which not only preserves the information of word w_i but also solves the problem of OOV words. Besides, to overcome the distribution difference, we develop the adversarial sentence s^p, which is obtained by adding some small perturbations to each word w_i in s, and generate the char-level word representation $\boldsymbol{e_i^p}$ for the word w_i^p in s^p. Subsequently, we employ three loss functions, $L_{diff}(s, s^p)$, $L_{true}(s, y)$ and $L_{noisy}(s^p, y)$ to generate the char embedding and apply the prior knowledge of adversarial examples to the classifier to ameliorate the distribution difference between the training set and adversarial examples. In the following parts, we will introduce the char embedding and the adversarial stability training in detail.

Char Embedding

Char embedding is to use deep neural network including Convolutional Neural Networks (CNN) and bi-directional Long Short-Term Memory (BLSTM) (Hochreiter and Schmidhuber 1997) to generate the char-level word representation through all the chars in the word. (Ling et al. 2015) has shown that char embedding can extract not only the syntactic and semantic information exactly like typical word embedding but also the prefix and suffix pattern of words to keep the information of the words as much as possible.



(a) BLSTM-Based Character Embedding



(b) CNN-Based Character Embedding

Figure 2: The process of generating the char-level word representation of the words "cats" by CharBLSTM and CharCNN.

Formally, for a word w_i, we convert each char in w_i to a d_C-dimensional vector $\boldsymbol{c_i^j}$ from char embedding matrix $\boldsymbol{C}$, and then w_i can be represented as a matrix:

$$\boldsymbol{W_i} = [\boldsymbol{c_i^1}, \boldsymbol{c_i^2}, ..., \boldsymbol{c_i^m}]_{m \times d_C} \tag{1}$$

where m is the num of chars in the word w_i. The char embedding use the matrix $\boldsymbol{W_i}$ as the input to generate the char-level word representation $\boldsymbol{e_i}$ for each word w_i.

In this paper, we adopt two kinds of char embedding methods, namely the BLSTM-based char embedding (CharBLSTM) and the CNN-Based char embedding (CharCNN), as shown in Figure 2. For CharBLSTM, the matrix $\boldsymbol{W_i}$ is the input of BLSTM, whose two final hidden vectors will be concatenated to generate $\boldsymbol{e_i}$. BLSTM extracts local and global dependency information from chars in both two directions. This means that the word representation e_i can capture local and global structure information of words. Besides, the four gates in BLSTM can selectively preserve information so that CharBLSTM can defend tiny char-level perturbation of char-level adversarial words. CharCNN use the matrix $\boldsymbol{W_i}$ as the input of a convolutional layer and a max-pooling layer to generate $\boldsymbol{e_i}$. Although CharCNN can only extract local information because of the single convolutional layer, the max-pooling layer in CharCNN can abandon unimportant information so that CharCNN can also defend tiny char-level perturbation as CharBLSTM.

Adversarial Stability Training

Although char embedding can defend char-level adversarial examples to some extent, the distribution difference between the training set and adversarial examples is still not conducive to defend char-level adversarial examples. In this paper, we utilize adversarial stability training to narrow this distribution difference. Specifically, for each sentence s in every training epoch, we add some small perturbation to each word w_i in s to generate its counterpart adversarial word w_i^p in adversarial example s^p. We randomly select one operation from the following four char-level perturbation types to generate w_i^p as (Gao et al. 2018):

1. **Swap**: Swap two adjacent chars in w_i.
2. **Substitution**: Substitute a random char in w_i with a random char.
3. **Deletion**: Delete a random char in w_i.
4. **Insertion**: Insert a random char in a random position in w_i.

After generating the new training samples, we also use three loss functions as (Cheng et al. 2018) to improve the robustness of our framework:

- $L_{true}(s, y)$ promotes the classifier to predict the correct label given the origin input sentence s.
- $L_{noisy}(s^p, y)$ promotes the classifier to predict the correct label given the noisy input sentence s^p.
- $L_{diff}(s, s^p)$ encourages the char embedding to generate similar char-level word representations $\boldsymbol{e_i}$ and $\boldsymbol{e_i^p}$ by s and s^p, respectively.

In summary, when given a training set S and its perturbed training set S^p, the adversarial stability training objective is:

$$\begin{aligned} L(\theta_{cle}, \theta_{cla}) = & \sum_{s \in S} L_{true}(s, y; \theta_{clm}, \theta_{cla}) + \\ & \alpha \sum_{s^p \in S^p} L_{noisy}(s^p, y; \theta_{clm}, \theta_{cla}) + \\ & \beta \sum_{s^p \in S^p} L_{diff}(s, s^p; \theta_{clm}) \end{aligned} \tag{2}$$

where θ_{clm} is the parameters of the char embedding, and θ_{cla} is the parameters of the classifier. α and β are the hyperparameters and will be discussed in Section 4. As shown in Figure 1, $L_{true}(s, y)$ and $L_{noisy}(s^p, y)$ are generated by the classifier, and $L_{diff}(s, s^p)$ is generated by the similarity evaluation. We will describe the classifier and the similarity evaluation in the following sections.

Classifier In this paper, except for the Char-CNN in (Zhang, Zhao, and LeCun 2015), our classifier is always BLSTM as shown in Figure 2a and followed by an one-layer fully connected MLP with a softmax activation function. The BLSTM takes the word representations as the input. So the loss $L_{true}(s, y)$ and $L_{noisy}(s^p, y)$ are binary or categorial cross-entropy loss with softmax activation:

$$L_{true}(S, Y) = \sum_{i=1}^{m} -log(Y_i \mid S_i)$$
$$L_{noisy}(S^P, Y) = \sum_{i=1}^{m} -log(Y_i \mid S_i^p) \tag{3}$$

Similarity Evaluation The basic idea of similarity evaluation is to make the char-level word representations e_i and e_i^p as close as possible when inputting with an origin word w_i and its noisy counterpart w_i^p. In this paper, we simply use mean square error loss to evaluate the similarity of two char-level word representations:

$$L_{diff}(w_i, w_i^p) = \frac{1}{n} \sum_{i=1}^{n} (f_{cle}(w_i) - f_{cle}(w_i^p))^2 \tag{4}$$

where f_{cle} is the char embedding function and n is the number of words in the sentence s. The similarity evaluation essentially promote the end-to-end deep neural networks to generate similar intermediate results by an extra loss function $L_{diff}(s, s^p)$. Therefore, the similarity evaluation is easy to be applied to other NLP tasks such as Sequence Labeling, Neural Machine Translation, Reading Comprehension and so on to improve their robustness to the char-level adversarial examples.

3 Adversarial Examples

The adversarial example is a reliable approach to evaluate the robustness of the deep learning based models. Given a trained classifier $C : S \to Y$, which can map a sentence $s \in S$ to its corresponding label $y \in Y$, the attacker adds some small perturbations to the sentence s to generate its adversarial example s^p, which will be mapped to a wrong label $y^p \in Y$ by the classifier C but can be mapped to the correct label y by humans. In this section, we generate two kinds of char-level adversarial examples, which are gradient-based adversarial examples and natural adversarial examples, to evaluate the robustness of our proposed framework.

Gradient-Based Adversarial Examples

The gradient-based adversarial examples use the cost gradient to select the important chars or words in the sentence s, i.e. the most influencing words or chars to the classification result in s, to craft the char-level adversarial examples. Specifically, for the word-level models, we sum the absolute value of the word embedding gradient($\boldsymbol{G_w}$) to indicate the importance of the word w:

$$I_w = \sum_{i=1}^{d_W} (|\boldsymbol{G_{w_i}}|) \tag{5}$$

where d_W is the dimension of word embedding in the word-level model. We randomly select a perturbation type from the four char-level perturbation types in Section 2 to modify the words one by one in order of the importance from high to low until the total edit distance reaches the defense difficulty controlling hyperparameter ed.

Dataset	SST	AG	DBP	Yelp P.	Yelp F.
# words	3740	5296	4783	5354	5348

Table 2: The statistic for the words that appear in both the vocabulary and the look-up table.

For char-level model, we use the same method to define the importance of the char c by its char embedding gradient($\boldsymbol{G_c}$):

$$I_c = \sum_{i=1}^{d_C} (|\boldsymbol{G_{c_i}}|) \tag{6}$$

Subsequently, we sorted all the chars in the sentence s by their importance, and only use substitution operation as (Gao et al. 2018) to change the sorted chars one by one until the total edit distance reaches ed. Besides, for word-level and char-level adversarial examples, we only modify each word one time as (Gao et al. 2018).

Natural adversarial examples

When people try to input text by handful typing, they may generate incorrect words called misspellings because of their carelessness or confusion. These misspellings are very common in social networks and various reviews. To further verify the robustness of our framework, we construct natural adversarial examples through Birkbeck[2], which is a spelling error corpus taken from native-speaker including 36133 misspellings of 6136 words. Meanwhile, we also use the "aspell" file, which contains 531 misspellings of 450 words by Atkinson[3], and the "wikipedia" file, which contains 2455 misspellings of 1922 words by Wikipedia editors. Subsequently, we covert all the words in these three misspelling data sets to lowercase and merge them to generate 38373 misspellings of 7392 words, which is used to build a look-up table of lexical replacements. We generate the natural adversarial example of sentence s by substituting each word in s with its random misspelling once the word is found in the look-up table. Besides, we count the number of words appearing both in the word vocabulary and look-up table in each data set as shown in Table 2, which can imply the difficulty of defense to the natural adversarial examples of each data set to some extent.

Now we have successfully built gradient-based adversarial examples and natural adversarial examples. Gradient-based adversarial examples can simulate deliberate attacks and are difficult to defend while natural adversarial examples can simulate people's careless inputs and are more common.

4 Experiments

Setup

Datasets We evaluate our framework on five public data sets: SST from (Socher et al. 2013), AG, DBPedia, and Yelp

[2]https://www.dcs.bbk.ac.uk/ ROGER/corpora.html

[3]http://aspell.net/test/batch0.tab

Dataset	SST	AG	DBP	Yelp P.	Yelp F.
# Training	6920	120K	560K	560K	650K
# Testing	1821	7.6K	70K	38K	50K
# Classes	2	4	14	2	5
# Avg_words	19	47	57	154	155
# Avg_characters	86	204	257	593	598

Table 3: The number of examples and length statistics of each data set.

P., Yelp F. from (Zhang, Zhao, and LeCun 2015). The number of samples, the average number of words, and the average number of characters are described in Table 3 in detail. For SST data set, we removed the samples labeled 2 according to (Pruthi, Dhingra, and Lipton 2019).

Model Configuration We first convert all input sentences into fixed-length sentences by padding and truncating and set the sentence length of SST, AG, and DBP to 100, the sentence length of Yelp P. and Yelp F. to 300 according to (Wang, Huang, and Deng 2018). Similarly, we convert the input words of all five data sets into words of fixed length 16. For the word-level model, we select 30,000 words with the highest frequency in the training set to build the word vocabulary and randomly initialize the word embedding in a fixed embedding dimension of 256. For the char embedding model, we use 70 characters, including English lowercase letters, digits, and some special characters, to build the char vocabulary as (Zhang, Zhao, and LeCun 2015) and randomly initialize the char embedding in a fixed embedding dimension of 64. Besides, we set the number of hidden units in CharLSTM to 128 and the number of filters in CharCNN to 256. Therefore, the word representations generated by all the experimental models have the same dimension. We set the number of hidden units in BLSTM classifier to 256, the loss parameters α and β to 1.0, and the edit distance ed in gradient-based adversarial examples to 30 for all models.

Training phase setting For all models, we adopt Adam optimizer (Kingma and Ba 2014) and set the minibatch size to 32. We set the learning rate of 0.0001 for all models. The SST data set has the pre-defined training and development set. For the other four data sets, we split the training set into a real training set and a development set at the ratio of 4 to 1. The model is tested on the development set every 1000 training steps. The early stopping patience is set to 20.

Models

Comparison Models We select two models, Word-BLSTM and Char-CNN, in (Gao et al. 2018) and three state-of-the-art defense models, Word-BLSTM-ATD, Word-BLSTM-ScRNN-10, and Word-BLSTM-ScRNN-78, in (Pruthi, Dhingra, and Lipton 2019) as the comparison models. In these five models, Char-CNN is a char-level model and both Word-BLSTM-ScRNN-10 and Word-BLSTM-ScRNN-78 use the char-level spell corrector.

- Word-BLSTM: A word-level model with bi-directional LSTM as Figure 2a and an one-layer fully connected MLP with a softmax activation function.
- Char-CNN: A CNN model in (Zhang, Zhao, and LeCun 2015), which contains a 9-layer convolutional network and uses one-hot encoding as the input.
- Word-BLSTM-ATD: A Word-BLSTM model followed by a context-aware spell corrector After the Deadline (AtD)[4].
- Word-BLSTM-ScRNN-10: A Word-BLSTM model followed by a deep learning based spell corrector ScRNN that containing a 10K words vocabulary. Word-BLSTM-ScRNN-10 is the state-of-the-art defense method to char-level adversarial examples.
- Word-BLSTM-ScRNN-78: A Word-BLSTM model same as Word-BLSTM-ScRNN-10 except that the ScRNN contains a 78K words vocabulary.

Our Models We implement two models, CharCNN-BLSTM-AST and CharBLSTM-BLSTM-AST, based on our proposed framework to evaluate the robustness of our framework.

- CharCNN-BLSTM-AST: A model that adopts CharCNN as the char embedding and uses the adversarial stability training.
- CharBLSTM-BLSTM-AST: Our model that adopts CharBLSTM as the char embedding and uses the adversarial stability training.

Experimental Results

Main Experimental Results We evaluate the seven models on two kinds of char-level adversarial examples of five data sets and presents the experimental results in Table 4. As shown in Table 4, CharBLSTM-BLSTM-AST performs best defense effectiveness among the seven models on both two kinds of char-level adversarial examples, which only drop 6.81% on average for gradient-based adversarial examples and drop 5.17% on average for natural adversarial examples. The defense effectiveness of CharCNN-BLSTM-AST is slightly weaker than CharBLSTM-BLSTM-AST but outperforms other comparison models, which is probably because CharCNN cannot extract global structure information of words compared with CharBLSTM. Through Word-BLSTM-ScRNN-10 and Word-BLSTM-ScRNN-78 make great improvement compared with Word-BLSTM, they get relatively poor results compared with our models as our two char-level adversarial examples are more complicated and difficult to defend than the char-level adversarial examples in (Pruthi, Dhingra, and Lipton 2019).

In detail, Word-BLSTM, Char-CNN, CharCNN-BLSTM-AST and CharBLSTM-BLSTM-AST perform much better for the natural adversarial examples than the gradient-based adversarial examples. However, Word-BLSTM-ATD and Word-BLSTM-ScRNN-78 perform better for the gradient-based adversarial examples and Word-BLSTM-ScRNN-10

[4]https://www.afterthedeadline.com/

(a) Gradient-Based Adversarial Examples

Dataset	Word-BLSTM			Char-CNN			Word-BLSTM-ATD			Word-BLSTM-ScRNN-10			Word-BLSTM-ScRNN-78			**CharCNN-BLSTM-AST**			**CharBLSTM-BLSTM-AST**		
	Ori	Grad	Dec	Ori	Grad	Dec	Ori	Grad	Dec	Ori	Grad	Dec	Ori	Grad	Dec	Ori	Grad	Dec	Ori	Grad	Dec
SST	81.22	50.36	38.00	70.07	57.28	18.25	80.45	67.55	16.03	79.02	66.83	15.43	79.19	67.82	14.36	77.21	64.80	16.07	74.85	67.22	10.19
AG	92.22	53.41	42.08	90.21	71.38	20.87	92.09	82.74	10.15	91.79	68.84	25.00	90.25	79.89	11.48	92.24	84.34	8.56	90.72	83.74	7.69
DBP	98.84	51.69	47.70	98.67	73.94	25.06	98.92	95.71	3.25	98.82	83.97	15.03	98.66	90.65	8.12	98.92	97.11	1.83	98.93	96.57	2.39
Yelp P.	95.75	82.61	13.72	94.93	84.92	10.54	96.19	92.60	3.73	95.91	91.23	4.88	95.48	91.16	4.52	96.16	92.02	4.31	95.92	92.13	3.95
Yelp F.	64.45	46.59	27.71	61.96	48.31	22.03	65.59	58.14	11.36	64.99	56.73	12.71	64.78	57.00	12.01	65.53	57.82	11.77	66.28	59.77	9.82
Mean	-	-	33.84	-	-	19.35	-	-	8.90	-	-	14.61	-	-	10.10	-	-	8.51	-	-	**6.81**

(b) Natural Adversarial Examples

Dataset	Word-BLSTM			Char-CNN			Word-BLSTM-ATD			Word-BLSTM-ScRNN-10			Word-BLSTM-ScRNN-78			**CharCNN-BLSTM-AST**			**CharBLSTM-BLSTM-AST**		
	Ori	Nat	Dec	Ori	Nat	Dec	Ori	Nat	Dec	Ori	Nat	Dec	Ori	Nat	Dec	Ori	Nat	Dec	Ori	Nat	Dec
SST	81.22	56.89	29.96	70.07	65.79	6.11	80.45	64.85	19.39	79.02	69.91	11.53	79.19	67.27	15.05	77.21	68.92	10.74	74.85	68.7	8.22
AG	92.22	82.50	10.54	90.21	84.91	5.88	92.09	84.41	8.34	91.79	85.04	7.35	90.25	81.57	9.62	92.24	89.45	3.02	90.72	88.8	2.12
DBP	98.84	75.03	24.09	98.67	85.72	13.12	98.92	90.11	8.91	98.82	88.75	10.19	98.66	87.32	11.49	98.92	96.9	2.04	98.93	97.66	1.28
Yelp P.	95.75	79.86	16.60	94.93	84.43	11.06	96.19	85.65	10.96	95.91	83.34	13.11	95.48	81.41	14.74	96.16	90.15	6.25	95.92	91.6	4.50
Yelp F.	64.45	41.95	34.91	61.96	47.17	23.87	65.59	48.54	25.99	64.99	47.71	26.59	64.78	46.36	28.43	65.53	54.57	16.73	66.28	57.22	13.67
Mean	-	-	23.22	-	-	12.01	-	-	14.72	-	-	13.75	-	-	15.87	-	-	6.88	-	-	**5.17**

Table 4: The main results for the gradient-based and natural adversarial examples. “Ori”, “Grad”, and “Nat” indicate the classification accuracy for the original test examples, the gradient-based adversarial examples and the natural adversarial examples, respectively. “Dec” indicates the accuracy reduction between the original test examples and the adversarial examples.

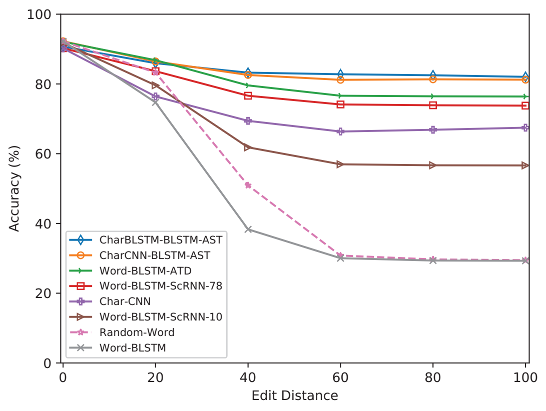


Figure 3: The experiment results for the gradient-based adversarial examples with different edit distance in AG dataset.

performs almost the same on those two kinds of adversarial examples. This is because the words in the gradient-based adversarial examples are allowed to change at most once, which is easy to correct by spell correctors because the edit distance between the original word and the counterpart adversarial word is only one or two. But the edit distance between the misspelling and the original word in natural adversarial examples is probably greater than 2, which causes the spell correctors to fail to correct the misspellings.

Experimental Result under Different Edit Distance To further demonstrate the robustness of our proposed model, we evaluate the classification accuracy of the seven models for the char-level adversarial examples in different edit distances. The experimental results are shown in Figure 3. According to Figure 3, our two models show good results at all edit distances. Furthermore, with the increase of the edit distance, our two models can still maintain ideal classification accuracy. In contrast, the classification accuracy of the other models decreases remarkably as the edit distance increases. Besides, to demonstrate the aggressiveness of our gradient-based adversarial examples, we generate some adversarial examples called Random Word by randomly changing the words and evaluate the classification accuracy of Word-BLSTM on these adversarial examples. The experimental results of Random Word are shown in Figure 3.

Experimental Results under Different Loss Parameters According to the training objective of Equation 2 in Section 2, the parameters α and β controls the weight of L_{noisy} and L_{diff} in the training phase and greatly affects the classification accuracy and robustness of our models. We conduct a grid parameter experiment to evaluate the interaction between the two parameters, where both α and β take 0, 0.5, 1.0, 2.0. We adopt AG as the data set and set the edit distance as 30 to evaluate the classification accuracy of CharCNN-BLSTM-AST model for the original examples and gradient-based adversarial examples, respectively. As shown in Figure 4, for both original and adversarial examples, our model performs unsatisfactorily when α takes a small value. When we set both α and β as 1, the model performs best for the original examples and shows competitive result for the adversarial examples. The robustness of the models reaches the highest when α takes 1 and β takes 2. Based on the exper-

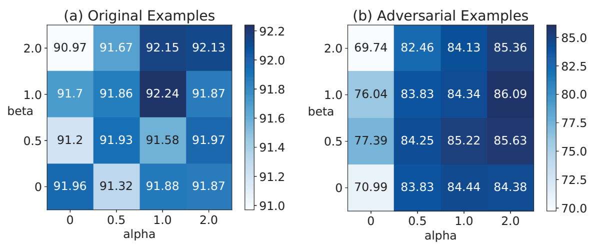


Figure 4: The classification accuracy of CharCNN-BLSTM-AST with different α and β for char-level adversarial examples.

Model	Original	Gradient	Natural
Word-BLSTM	92.22	53.41	82.50
CharCNN-BLSTM	91.96	70.99	87.88
CharCNN-BLSTM+L_{diff}	91.70	76.04	88.25
CharCNN-BLSTM+L_{noisy}	91.88	**84.44**	89.39
CharCNN-BLSTM+AST	**92.24**	84.34	**89.45**

Table 5: The experimental results of the incremental experiments.

imental results, we set both α and β as 1 for our models to perform best for the original examples because the original examples are the most common examples in practice.

Incremental Experiments We conduct incremental experiments to verify the robustness of various parts of our framework to char-level adversarial examples. Same as the loss parameters experiment, we adopt AG as the data set and set edit distance as 30. Based on the former experimental results, we set both α and β to 1. The experimental results are presented in Table 5. According to Table 5, CharCNN-BLSTM gets better results on both two kinds of char-level adversarial examples than Word-BLSTM. The results indicate using char embedding alone can improve the robustness of word-level models to char-level adversarial examples because char embedding can solve OOV words in word-level models to keep the information of words as much as possible. Using L_{diff} or L_{noisy} alone both can further improve the robustness of CharCNN-BLSTM. Finally, CharCNN-BLSTM-AST gets the best results on the original examples and the natural adversarial examples and very competitive result on the gradient-based adversarial examples.

Defense Examples Table 6 shows some natural adversarial examples in Yelp P.. We find that our models can defend char-level adversarial examples effectively comparing with Word-BLSTM.

5 Related Work

Adversarial examples are first proposed in the field of image processing (Goodfellow, Shlens, and Szegedy 2014; Carlini and Wagner 2017) to study the robustness of the deep neural networks based models. Subsequently, (Papernot et al. 2016) start to generate word-level adversarial examples in text classification. (Liang et al. 2017) also consider adversarial examples in text classification. They exploit multi-level perturbations simultaneously to fool the text classification models. In recent years, (Gao et al. 2018; Li et al. 2018; Ebrahimi et al. 2017) start to generate char-level adversarial examples to fool the state-of-the-art text classification models.

Contrasted to the adversarial example generation, there is less research of adversarial examples defense in NLP (Wang et al. 2019). (Gao et al. 2018; Li et al. 2018) propose using the spell corrector to defend adversarial examples, but (Gao et al. 2018) show that the spell corrector only works for tiny perturbed words, whose edit distance from the original word is only one. (Miyato, Dai, and Goodfellow 2016) add perturbation to the word embedding to improve the effectiveness of deep learning based text classification models. (Belinkov and Bisk 2017; Cheng et al. 2018) study the char-level perturbations in Neural Machine Translation(NMT). They both incorporate adversarial stability training with word embedding to defend the char-level perturbations. (Pruthi, Dhingra, and Lipton 2019) use ScRNN to defend char-level adversarial examples in text classification, but they only test their effectiveness on very simply adversarial examples, which is only allowed to change two words in a sentence at most.

In the area of char embedding, (Ling et al., 2016) first propose char embedding for both language models and part-of-speech tagging. (Ma and Hovy 2016; Lample et al. 2016) use CNN and BLSTM to construct char embedding for sequence labeling. (Seo et al. 2016; Tay et al. 2018; Wang et al. 2017) use char embedding in reading comprehension to capture char-level information.

6 Conclusion

In this paper, we propose a novel framework to defend char-level adversarial examples by jointly using char embedding and adversarial stability training, which can solve both the OOV words in char-level adversarial examples and the distribution difference between the training set and the adversarial examples. Experiment results on five text classification data sets show that the models based on our framework can both get competitive results on original test sets and defend gradient-based adversarial examples and natural adversarial examples effectively, which greatly outperform the state-of-the-art defense models to char-level adversarial examples. Besides, our framework does not use the special design for text classification, so it has the potential to be applied to any NLP tasks to improve their robustness to char-level adversarial examples.

7 Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable advices and insightful comments. This work was supported by the National Natural Science Foundation of China under grant numbers 61572496. The corresponding author is Yongzheng Zhang.

Example	***Input***	***Word-BLSTM***	***CharCNN-BLSTM-AST***
Original	very helpful and knowledgeable staff.	Positive	Positive
Adversarial	veyr helpfull are knowlegeable staff.	Negative	Positive
Original	terrible service!	Negative	Negative
Adversarial	terriable survice!	Positive	Negative
Original	horrible customer service. they only care money!	Negative	Negative
Adversarial	horible costomer servise. thy onle cerer moeny!	Positive	Negative

Table 6: Natural adversarial examples in Yelp Review polarity corpus.

References

Belinkov, Y., and Bisk, Y. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Carlini, N., and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57. IEEE.

Cheng, Y.; Tu, Z.; Meng, F.; Zhai, J.; and Liu, Y. 2018. Towards robust neural machine translation. *arXiv preprint arXiv:1805.06130*.

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

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

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

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

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lai, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, 707–710.

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

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

Ling, W.; Luís, T.; Marujo, L.; Astudillo, R. F.; Amir, S.; Dyer, C.; Black, A. W.; and Trancoso, I. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Ma, X., and Hovy, E. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Miyato, T.; Dai, A. M.; and Goodfellow, I. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.

Papernot, N.; McDaniel, P.; Swami, A.; and Harang, R. 2016. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM 2016-2016 IEEE Military Communications Conference*, 49–54. IEEE.

Pruthi, D.; Dhingra, B.; and Lipton, Z. C. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5582–5591. Florence, Italy: Association for Computational Linguistics.

Sakaguchi, K.; Duh, K.; Post, M.; and Van Durme, B. 2017. Robsut wrod reocginiton via semi-character recurrent neural network. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631–1642. Seattle, Washington, USA: Association for Computational Linguistics.

Tay, Y.; Luu, A. T.; Hui, S. C.; and Su, J. 2018. Densely connected attention propagation for reading comprehension. In *Advances in Neural Information Processing Systems*, 4906–4917.

Wang, W.; Yang, N.; Wei, F.; Chang, B.; and Zhou, M. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 189–198.

Wang, W.; Wang, L.; Tang, B.; Wang, R.; and Ye, A. 2019. Towards a Robust Deep Neural Network in Text Domain A Survey. *arXiv e-prints* arXiv:1902.07285.

Wang, S.; Huang, M.; and Deng, Z. 2018. Densely connected cnn with multi-scale feature attention for text classification. In *IJCAI*, 4468–4474.

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