Learning Multi-Level Dependencies for Robust Word Recognition

Zhiwei Wang,[1]* **Hui Liu,**[1] **Jiliang Tang,**[1] **Songfan Yang,**[2] **Gale Yan Huang,**[2] **Zitao Liu**[2]†
[1]Michigan State University, {wangzh65, liuhui7, tangjili}@msu.edu
[2]TAI AI Lab, TAL Education Group, {yangsongfan, galehuang, liuzitao}@100tal.com

Abstract

Robust language processing systems are becoming increasingly important given the recent awareness of dangerous situations where brittle machine learning models can be easily broken with the presence of noises. In this paper, we introduce a robust word recognition framework that captures multi-level sequential dependencies in noised sentences. The proposed framework employs a sequence-to-sequence model over characters of each word, whose output is given to a word-level bi-directional recurrent neural network. We conduct extensive experiments to verify the effectiveness of the framework. The results show that the proposed framework outperforms state-of-the-art methods by a large margin and they also suggest that character-level dependencies can play an important role in word recognition. The code of the proposed framework and the major experiments are publicly available[1].

Introduction

Most of the widely used language processing systems have been built on neural networks that are highly effective, achieving the performance comparable to humans (Devlin et al. 2018; Yang et al. 2019; Yu et al. 2018). They are also very brittle, however, as they could be easily broken with the presence of noises (Belinkov and Bisk 2017; Zhao, Dua, and Singh 2017; Ebrahimi et al. 2017). However, the language processing mechanism of humans are very robust. One representative example is the following *Cambridge* sentence:

> *Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.*

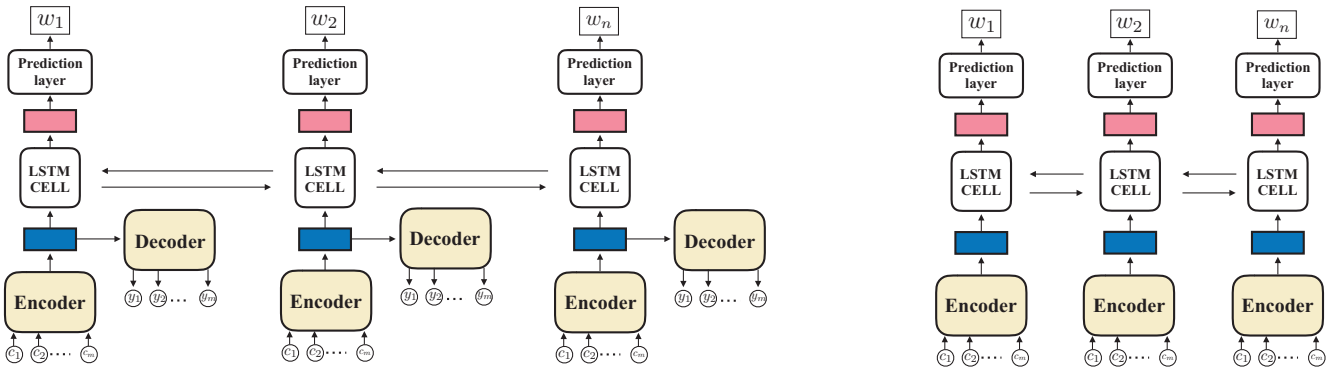
In spite of the fact that a human can read the above sentence with little difficulty, it can cause a complete failure to existing natural language processing systems such as Google Translation Engine [2]. Building robust natural language processing systems is becoming increasingly important nowadays given severe consequences that can be made by adversarial samples (Xu et al. 2019c): carefully misspelled spam emails that fool spam detection systems (Fumera, Pillai, and Roli 2006) deliberately designed input sentences that force chatbot to emit offensive language (Wolf, Miller, and Grodzinsky 2017; Dinan et al. 2019; Liu et al. 2019), etc. Thus, in this work, we focus on building a word recognition framework which can denoise the misspellings such as those shown in the *Cambridge* sentence. As suggested by psycholinguistic studies (Rayner, White, and Liversedge 2006; Davis 2012), the humans can comprehend text that is noised by jumbling internal characters while leaving the first and last characters of a word unchanged. Thus, an ideal word recognition model is expected to emulate robustness of human language processing mechanism.

The benefits of such framework are two-folds. The first is its recognition ability can be straightforwardly used to correct misspellings. The second is its contribution to the robustness of other natural language processing systems. By serving as a denoising component, the word recognition framework can firstly clean the noised sentences before they are inputted into other natural language processing systems (Pruthi, Dhingra, and Lipton 2019; Zhou et al. 2019).

From the human perspective, there are two types of information that play an essential role for us to recognize the noised words (Perea et al. 2015). The first is the character-level dependencies. Take the word '*wlohe*' in the *Cambridge* sentences as an example, it is extremely rare to see a 'w' sits next to an 'l' in an English word. Instead, it is more natural with 'wh'. Thus, it is quite easy for humans to narrow down possible correct forms of '*wlohe*' to be '*whole*' or '*whelo*'. To ensure that it should be '*whole*', we often need the second type of information: context information such as '*but the wrod as a wlohe.*', which is denoted as word-level dependencies in this paper. Intuitively, an effective word recognition framework should capture these multi-level depen-

*Work was done when interned at TAL AI Lab
†The corresponding author

[1]https://github.com/DSE-MSU/MUDE
[2]https://translate.google.com/



(a) Train Stage. (b) Test Stage

Figure 1: The graphical illustration of the proposed framework: MUDE.

dencies. However, multi-level dependencies are rarely exploited by the exiting works such as scRNN (Sakaguchi et al. 2017). Hence, we propose a framework MUDE that is able to fully utilize **mu**lti-level **de**pendencies for the robust word recognition task. It integrates a character-level sequence-to-sequence model and a word-level sequential learning model into a coherent framework. The major contributions of our work are summarized as follows:

- We identify importance of character-level dependencies for recognizing a noised word;
- We propose a novel framework, MUDE, that utilizes both character-level and word-level dependencies for robust word recognition task;
- We conduct extensive experiments on various types of noises to verify the effectiveness of MUDE.

For the rest of the paper, we firstly give a detailed description of MUDE. Then we conduct experiments to verify its effectiveness. We will also show that MUDE is able to achieve the state-of-the-art performance no matter what type of noise presents and outperforms the widely used baselines by a large margin. Next, we introduce the relevant literature in the related work section, followed by a conclusion of current work and discussion of possible future research directions.

The Proposed Framework: MUDE

In this section, we describe MUDE that is able to capture both character-level and word-level dependencies. The overall architecture is illustrated in Figure 1. It consists of 3 major components: a sequence-to-sequence model, a bi-directional recurrent neural network and a prediction layer. Next we will detail each component.

Learning Character-level Dependencies

As mentioned previously, there exist sequential patterns in the characters of a word. For example, vocabulary roots such as *cur* and *bio* can be found in many words. In this subsection, we propose a sequence-to-sequence model to learn a better representation of a given word by incorporating character-level dependencies. The model consists of an encoder and a decoder, which we will describe next.

Encoder Let $\hat{w} = c_1, c_2, \cdots c_m$ be a sequence of characters of a given noised word $\hat{w}$. We firstly map each character c_i to a d_c-dimensional character embedding as follows:

$$x_i = \mathbf{E}o_i \tag{1}$$

where $\mathbf{E} \in \mathbb{R}^{C \times d_c}$ is the embedding matrix given that the total number of unique characters is C. $o_i \in \mathbb{R}^C$ is the one-hot representation of c_i. Since there could some noise in $\hat{w}$, the sequential order of c_i can be misleading. Thus, instead of using a sequential learning model such as recurrent neural network, we choose the multi-head attention mechanism (Vaswani et al. 2017) to model the dependencies between characters without considering their order. To do so, we add a special character c_0 whose final representation will be used as the representation of the word.

Specifically, the multi-head attention mechanism will obtain a refined representation for each character in $\hat{w}$. Next, without the loss of generality, we will use c_i as an example to illustrate. To obtain the refined representation of c_i, x_i will firstly be projected into query space and x_j $\forall j \in \{0, 1, \cdots, m\}$ will be projected into key and value spaces as follows:

$$\begin{aligned} x_i^q &= \mathbf{Q}x_i \\ x_j^k &= \mathbf{K}x_j \quad \forall j \in \{0, 1, \cdots, m\} \\ x_j^v &= \mathbf{V}x_j \quad \forall j \in \{0, 1, \cdots, m\} \end{aligned} \tag{2}$$

where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ are the projection matrices for query, key, and value spaces, respectively. With x_i^q, x_j^k and x_j^v, the refined representation e_i of c_i can be calculated as the weighted sum of x_j^v:

$$e_i = \sum \alpha_j x_j^v \tag{3}$$

where α_j is the attention score that is obtained by the following equation:

$$\alpha_0, \alpha_1, \cdots, \alpha_m = \sigma^s\left(\frac{x_i^{qT} x_0^k}{\sqrt{d}}, \frac{x_i^{qT} x_1^k}{\sqrt{d}}, \cdots, \frac{x_i^{qT} x_m^k}{\sqrt{d}}\right)$$

Where σ^s is the softmax function. To capture the dependencies of characters from different aspects, multiple sets of projection matrices are usually used, which will result in multiple sets of x_i^q, x_j^k and x_j^v, and thus e^i. To be concrete, assume that there are h sets of projection matrices, from Eq. (2) and Eq. (3), we can obtain h e_is, which are denoted as $\{e_i^1, e_i^2, \cdots, e_i^h\}$. With this, the refined representation of c_i is obtained by the concatenation operation:

$$z_i = concatenation(e_i^1, e_i^2, \cdots, e_i^h) \tag{4}$$

where z_i is the new representation of c_i and contains dependency information of c_i to other characters in $\hat{w}$ from h aspects.

Following (Vaswani et al. 2017), we also add a positional-wise feedforward layer to z_i as follows:

$$p_i = \mathbf{W}^2 ReLU(\mathbf{W}^1 z_i) \tag{5}$$

where $\mathbf{W}^1$ and $\mathbf{W}^2$ are the learnable parameters. p_i is the final representation of c_i. Note that we can have several above mentioned layers stacked together to form a deep structure.

At this point, we have obtained the refined representation vector for each character and we use that of the special character c_0 as the representation of given noised word, which is denoted as w^c

Decoder To capture the sequential dependency in the correct words, the Gated Recurrent Unit (GRU) which has achieved great performance in many sequence learning tasks (Xu et al. 2019b; Andermatt, Pezold, and Cattin 2016; Xu et al. 2019a) is used as the decoder. To be specific, in the decoding process, the initial hidden state h_0 of GRU is initialized with the noised word presentation $\hat{w}$. Then at each time stamp t, GRU will recursively output a hidden state h_t given the hidden state h_{t-1} at the previous time stamp. Due to the page limitation, we do not show the details of GRU, which is well described in (Cho et al. 2014). In addition, each hidden state will emit a predicted character c_t^p. The decoding process will end when the special character denoting the end of word is emitted. Concretely, the whole decoding process is formally formulated as follows:

$$\begin{aligned} h_0 &= w^c \\ h_t &= GRU(h_{t-1}) \\ p_t &= \sigma^s(\mathbf{W}^p h_t) \\ c_t^p &= \arg\max_i(p_t[i]) \end{aligned} \tag{6}$$

where $\mathbf{W}^p \in \mathbb{R}^{C \times d}$ is a trainable parameter. $p_t \in \mathbb{R}^C$ gives the emission probability of each character and $p_t[i]$ denotes the i^{th} entry of vector p_t.

Sequence-to-sequence Loss To train the previously described character-level sequence-to-sequence model, we define the loss function as follows:

$$\mathcal{L}_{seq2seq} = -\sum_i^m p_i[y_i] \tag{7}$$

where y_i is the index of the ground truth at position i of the correct word w. By minimizing $\mathcal{L}_{seq2seq}$, the designed sequence-to-sequence model can learn a meaningful representation that incorporates character-level sequential dependencies for the noised word. Next, we will describe the framework component that captures the word-level dependencies.

Capturing Word-level Dependencies

From the human perspective, it is vitally important to consider the context of the whole sentences in order to understand a noised word. For example, it would be very hard to know 'frist' means 'first' until a context 'the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae.' is given. Thus, to utilize the context information and word-level dependencies, we design a recurrent neural network (RNN) to incorporate them in the noised word representation. Specifically, the word presentations obtained from character-level encoder will be passed into a bi-directional long short-term memory (LSTM). Concretely, given a sequence of word presentations $S = \{w_1^c, w_2^c, \cdots, w_n^c\}$ obtained from character-level dependencies, we calculate a sequence of refined word representation vectors $\{w_1, w_2, \cdots, w_n\}$ as follows:

$$\begin{aligned} w_1^f, w_2^f, \cdots, w_n^f &= LSTM_{forward}(w_1^c, w_2^c, \cdots, w_n^c) \\ w_1^b, w_2^b, \cdots, w_n^b &= LSTM_{backward}(w_1^c, w_2^c, \cdots, w_n^c) \\ w_1, w_2, \cdots, w_n &= w_1^f || w_1^b, w_2^f || w_2^b, \cdots, w_n^f || w_n^b \end{aligned} \tag{8}$$

where $\|$ denotes concatenation. $LSTM_{forward}$ indicates that w^cs are processed from w_1^c to w_n^c, while $LSTM_{backward}$ processes word presentations in an opposite direction, namely, from w_n^c to w_1^c. Comparing to original LSTM where only forward pass is performed, bi-directional LSTM can include both 'past' and 'future' information in the representation of w_i.

With the aforementioned procedure, the representation of each word now incorporates both character-level and word-level dependencies. Thus, the correct word is predicted as follows:

$$\begin{aligned} p_i^w &= \sigma^s(\mathbf{W}^w w_i) \\ w_i^p &= \arg\max_i(p_t^w[i]) \end{aligned} \tag{9}$$

where $\mathbf{W}^w \in \mathbb{R}^{V \times d_w}$ is a trainable matrix and V is the size of the vocabulary that contains all possible words. Moreover, $p_i^w \in \mathbb{R}^V$ is the probability distribution over the vocabulary for the i^{th} word in a sentence.

Word Prediction Loss To effectively train MUDE for correct word prediction, similar to character-level sequence-to-sequence model, we define the following objective function:

$$\mathcal{L}_{pred} = -\sum_i^n p_i^w[y_i^w] \tag{10}$$

where y_i^w is the index of the i^{th} correct word.

Training Procedure

So far we have described MUDE which includes a character-level sequence-to-sequence model and a word-level sequential learning model. To train both models simultaneously, we design a loss function for the whole framework as follows:

$$\mathcal{L} = \mathcal{L}_{pred} + \beta \mathcal{L}_{seq2seq} \tag{11}$$

where β is a hyperparameter that controls the contribution of the character-level sequence-to-sequence model. Since the major goal of the framework is to predict the correct word given the noised word, we decrease the value of β gradually as the training proceeds to allow the optimizer increasingly focus on improving the word prediction performance.

Test Stage As shown in Figure 1, in the test stage, we simply remove the decoder of the sequence-to-sequence model and only keep the encoder in the framework.

Experiment

In this section, we conduct extensive experiments on the spell correction task to verify the effectiveness of MUDE. Next, we firstly introduce the experimental settings, followed by the analysis of the experimental results.

Experimental Settings

Data We use the publicly available Penn Treebank (Marcus, Santorini, and Marcinkiewicz 1993) as the dataset. Following the previous work (Sakaguchi et al. 2017), we firstly experiment on 4 different types of noise: Permutation (PER), Deletion (DEL), Insertion (DEL), and Substitution (SUB), which only operate on the internal characters of words, leaving the first and last characters unchanged. Table 1 shows a toy example of a noised sentence. These 4 types of noise can cover most of the realistic cases of misspellings and commonly tested in previous works (Belinkov and Bisk 2017; Pruthi, Dhingra, and Lipton 2019). For each type of noise, we construct a noised dataset from the original dataset by altering all the words that have more than 3 characters with corresponding noise. We use the same training, validation and testing split in (Sakaguchi et al. 2017), which contains 39,832, 1,700 and 2,416 sentences, respectively.

Table 1: Toy examples of noised text

Noise Type	Sentence
Correct	An example of noised text
PER	An epaxmle of nsieod txet
DEL	An examle of nosed tet
INS	An edxample of nmoised texut
SUB	An exsmple of npised test

Baselines To show the effectiveness of MUDE, we compare it with two strong and widely used baselines. The first is Enchant [3] spell checker which is based on dictionaries. The second one is scRNN (Sakaguchi et al. 2017). It is a recurrent neural network based word recognition model and has achieved previous state-of-the-art results on spell correction tasks. This baseline only considers the sequential dependencies in the word level with a recurrent neural network and ignores that of character level. Note that other baselines including CharCNN (Sutskever, Martens, and Hinton 2011) have been significantly outperformed by scRNN. Thus, we do not include them in the experiments.

Implementation Details Both scRNN and MUDE are implemented with Pytorch. The number of hidden units of word representations is set to be 650 as suggested by previous work (Sakaguchi et al. 2017). The learning rate is chosen from $\{0.1, 0.01, 0.001, 0.0001\}$ and β in Eq (11) is chosen from $\{1, 0.1, 0.001\}$ according to the model performance on the validation datasets. The parameters of MUDE are learned with stochastic gradient decent algorithm and we choose RMSprop (Tieleman and Hinton 2012) to be the optimizer as it did in (Sakaguchi et al. 2017). To make the comparison fair, scRNN is trained with the same settings as MUDE.

Comparison Results

The comparison results are shown in Table 2. There are several observations can be made from the table. The first is that model based methods (scRNN and MUDE) achieve much better performance than dictionary based one (Enchant). This is not surprising as model based methods can effectively utilize the sequential dependencies of words in the sentences. Moreover, MUDE consistently outperforms scRNN in all cases, which we believe attributes to the effective design of MUDE to capture both character and word level dependencies. More detailed analysis of contribution brought by the character-level dependencies will be shown later in this section. In addition, we observe that the difficulty brought by different types of noise varies significantly. Generally, for model based methods, permutation and insertion noises are relatively easier to deal with comparing to deletion and substitution noises. We argue this is because the former ones do not lose any character information. In other words, the original character information is largely preserved with permutation and insertion. On the contrary, both deletion and substitution can cause information loss, which makes it harder to recognize the original words. This again demonstrate how important the character-level information is. Finally, the results also show that in more difficult situations where deletion or substitution noises present, the advantages of the MUDE become even more obvious. This clearly suggests the effectiveness of the MUDE.

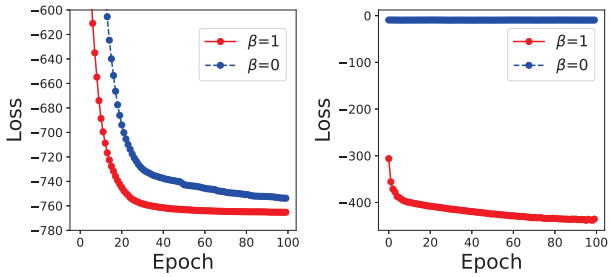
Next, we take one step further by removing the constraint that the noise will not affect the first and last characters of each word. More specifically, we define 4 new types of noise that are W-PER, W-DEL, W-INS, and W-SUB, which stand for altering a word by permuting the whole word, deleting, inserting, and substituting characters in any position of the word. Similarly, for each type of new noise, we construct a noised dataset. The results are shown in Table 3.

From the table, we observe that firstly, the performance of

[3] https://abiword.github.io/enchant/

Table 2: Performance comparison with different types of noise in terms of accuracy (%). Best results are highlighted with bold numbers.

Method	INT	DEL	INS	SUB
Enchant	72.33	71.23	93.93	79.77
scRNN	98.23	91.55	95.95	87.09
MUDE	**98.81**	**95.86**	**97.16**	**90.52**



(a) Prediction loss. (b) Seq2Seq loss.

Figure 2: Learning curve of MUDE in the training procedure with different β values.

nearly all methods decreases comparing to that of Table 2. This suggests the new types of noise are more difficult to handle, which is expected as they cause more variations of noised words. In fact, without keeping first and last characters of each words, it also becomes a difficult task for human to comprehend the noised sentences (Rayner, White, and Liversedge 2006). Secondly, MUDE still achieves higher accuracy than other baselines, which is consistent with observations from Table 2. More importantly, as the difficulty of the task increases, the advantages of MUDE over scRNN also become more obvious. Take the noise of substitution for example, in Table 2, MUDE has around 3.5% absolute accuracy gain over scRNN. When more difficult noise (W-SUB) comes, the performance gain of MUDE becomes 4% as shown in Table 3. Such observation is also consistent with previous findings.

In summary, both Table 2 and 3 clearly demonstrate the robustness of MUDE and its advantages over scRNN which can not utilize the character-level dependencies. Thus, in the next subsection, we conduct analysis on the contribution of character-level dependencies to gain better understanding of MUDE.

Table 3: Performance comparison different type of noise in terms of accuracy (%). Best results are highlighted with bold numbers.

Method	W-PER	W-DEL	W-INS	W-SUB
Enchant	59.08	69.84	93.77	77.23
scRNN	97.36	89.99	95.96	81.12
MUDE	**98.75**	**93.29**	**97.10**	**85.17**

Parameter Analysis

In this subsection, we analyze the contribution of character-level dependencies to better word representations by showing the model performance with different β values, which controls the contribution of character-level sequence-to-sequence loss. Specifically, we let the β be 0 and 1. When β is 0, MUDE will totally ignore the character-level dependencies; When β equals to 1, MUDE achieve best accuracy in validation set. The prediction loss and seq2seq loss during the training stage with different β values are shown in Fig 2. Note that the trends in Fig 2 are similar in all of the cases with the different types noise and we only show that of W-PER case due to the page limitation.

As the upper sub-figure shows, when $\beta = 1$ the prediction loss converges faster and at a lower value comparing to that of case when $\beta = 0$. For the seq2seq loss, it remains constant value when $\beta = 0$ as the model does not learn anything regarding seq2seq task. On the other hand, when $\beta = 1$, the seq2seq loss stably decreases, suggesting that the MUDE is trained to obtain better representation of each word. The obvious positive correlation between these two losses clearly demonstrates the importance of learning character-level dependencies in misspelling correction tasks.

Generalization Analysis

In this subsection, we conduct experiments to understand generalization ability of MUDE. Concretely, we train the framework on one type of noise and test it with a dataset that presents another type of noise. The results are shown in Table 4.

From results, we have the following two observations. Firstly, between datasets with similar type of noise, MUDE generalizes quite well (e.g. trained on W-PER and tested on PER), which is not surprising. However, the MUDE trained on one type of noise performs much worse on other types of noise that are very different. These observations suggest that it is hard for MUDE to generalize between noises, which we argue is possibly because of the small overlap between distributions of each type of noise.

Thus, in the next, we apply the commonly used adversarial training method by augmenting all types of noise to train MUDE and test it on each type noise individually. As W-* (*$\in$ {PER, DEL, INS, SUB}) includes the *, in this experiment, we only combine W-* instead of all types of noise. We denote the new constructed training dataset as W-ALL. The results are shown in Table 5. It can be observed from table that the MUDE trained on W-ALL has much better generalization ability (i.e., the mean value is much higher). In addition, it is interesting to see that performance of the MUDE decreases slightly in relatively easy cases where permutation or insertion noise presents while increasing a lot in difficult cases where deletion or substitution noise presents.

Case Study

In this subsection, we take the *Cambridge* sentences which are not the training set as an example to give a qualitative illustration of MUDE's misspelling correction performance.

Table 4: Generalization analysis results. The best result are highlighted. MEAN shows the average value of each row.

		Test Noise								
		PER	W-PER	DEL	W-DEL	INS	W-INS	SUB	W-SUB	MEAN
Train Noise	PER	–	**98.81**	82.55	79.61	92.21	92.37	71.39	69.88	85.70
	W-PER	**98.75**	–	81.31	78.3	91.32	91.25	69.55	67.91	84.64
	DEL	90.83	90.83	–	**86.02**	79.96	79.97	81.99	76.02	85.18
	W-DEL	86.75	86.75	**94.08**	–	78.83	78.87	80.35	79.07	84.74
	INS	94.79	94.79	77.3	74.81	–	**97.15**	82.86	80.42	87.41
	W-INS	95.67	95.67	78.34	75.95	**97.01**	–	82.96	80.78	**87.91**
	SUB	91.71	91.71	88.34	81.49	81.19	81.21	–	**83.65**	86.22
	W-SUB	87.05	87.05	83.42	82.42	79.27	79.17	**85.67**	–	83.65

Table 5: Data augmentation results. The values that are higher than these of Table 4 are bold.

		Test Noise								
		Per	W-PER	DEL	W-DEL	INS	W-INS	SUB	W-SUB	MEAN
Train Noise	W-ALL	96.45	96.45	**94.26**	**93.34**	95.3	95.28	**91.51**	**90.48**	**94.13**

Note that due to the constraint of space, we only show the results of the two types of noise: W-PER and W-INS. The example is shown in Table 6. We can see from the table that it is quite difficult for even humans to comprehend the noised sentence when first and last characters are also changed. However, MUDE can still recognize almost all of the words. In addition, for both cases, the MUDE has much less errors in the corrected sentence than scRNN, which is consistent with previous quantitative results.

Related Work

In this section, we briefly review the related literature that is grouped into two categories. The first category includes the exiting works on similar tasks and the second one contains previous works that have applied word recognition model to improve the robustness of other NLP systems.

Grammatical Error Correction

Since the CoNLL-2014 shared task (Ng et al. 2014), Grammatical Error Correction (GEC) has gained great attention from NLP communities (Zhao et al. 2019; Grundkiewicz and Junczys-Dowmunt 2018; Junczys-Dowmunt et al. 2018; Chollampatt and Ng 2017; Ji et al. 2017). Currently the most effective approaches regard GEC as machine translation problem that translates erroneous sentences to correct sentences. Thus, many methods that are based on statistical or neural machine translation architectures have been proposed. However, most of the existing GEC systems have focused on correction of grammar errors instead of noised spellings. For example, most of words in a wrong sentence in CoNLL-2014 shared task (Ng et al. 2014) are correct such as 'Nothing is absolute right or wrong', where the only error comes from the specific form 'absolute'. One of the existing works that are most similar to this paper is scRNN (Sakaguchi et al. 2017), where each word is represented in a fixed 'bag of characters' way. It only consists of a word-level RNN and focused on very easy noise. On the contrary, our proposed framework is more flexible and can obtain meaningful representations that incorporate both character and word-level dependencies. In addition, we have experimented on more difficult types of noise than these in (Sakaguchi et al. 2017) and achieved much better performance.

Denoising text for downstream tasks

Robust NLP systems are becoming increasingly important given the severe consequences adversarial samples can cause (Grosse et al. 2017; Iyyer et al. 2018; Xu et al. 2019c). However, previous works have shown that neural machine translation models can be easily broken with words whose characters are permuted (Belinkov and Bisk 2017). To solve this problem, researchers have found that misspelling correction models can play an extremely effective role (Pruthi, Dhingra, and Lipton 2019; Zhou et al. 2019) in improving the robustness of the systems. For example, Pruthi *et al* (Pruthi, Dhingra, and Lipton 2019) firstly applied the pretrained scRNN model to source sentence to remove noise and then the denoised source sentence was input into the neural translation model to obtain the correctly translated sentence. In addition, Zhou *et al* (Zhou et al. 2019) directly integrated such denoising models into the machine translation system that was trained in an end-to-end approach. In either way, these works suggest that the proposed framework which has demonstrated strong performance can have great potentials in improving the robustness of other NLP systems.

Conclusion

As most of the current NLP systems are very brittle, it is extremely important to develop robust neural models. In this paper, we have presented a word recognition frame-

Correct	According to a researcher at Cambridge University , it does n't matter in what order the letters in a word are , the only important thing is that the first and last letter be at the right place . The rest can be a total mess and you can still read it without problem . This is because the human mind does not read every letter by itself , but the word as a whole .
	W-PER
Noised	iodrcAngc ot a reeachsr at meigaCdbr srtiinUyve , it seod tn' amrtte in wtah rerdo het tserelt in a rdwo rae , the onyl onmtiaptr ingth si tath hte itfrs dan stla treelt be ta het tgrhi place . hTe rset nca be a aotlt mess dan ouy anc lsilt drae ti tthwuoi lorbmpe . hTsi is aubeecs the huamn dmni edos nto erad evrye lteter by etfisl , but het rdwo sa a eholw .
scRNN	According to a **research** at Cambridge University , it does n't matter in what order the letters in a word are , the only important thing is that the first and last letter be at the right place . The rest can be a total mess and you can still read it without problem . This is because the human mind does not read **very** letter by itself , but the word as a whole .
MUDE	According to a **research** at Cambridge University , it does n't matter in what order the letters in a word are , the only important thing is that the first and last letter be at the right place . The rest can be a total mess and you can still read it without problem . This is because the human mind does not read every letter by itself , but the word as a whole .
	W-INS
Noised	Acxcording to a reysearch at Cazmbridge Univversity , it doesw n't msatter in whmat orderh the letteros in a fword are , the oynly wimportant tghing is tyhat the fircst and ldast legtter be at the rightv placeu . The resty can be a totalp mesus and you can stillb rnead it withougt promblem . Txhis is bebcause the humgan minnd doess not reabd everyb lettfer by itslelf , but the whord as a whvole .
scRNN	according to a **research** at Cambridge University , it does n't matter in what order the letters in a word are , the only important thing is that the first and last **better** be at the right place . The rest can be a total **less** and you can still read it without problem . This is because the human mind does not **rated** every **better** by itself , but the word **a** a whole .
MUDE	According to a **research** at Cambridge University , it does n't matter in what order the letters in a word are , the only important thing is that the first and last letter be at the right place . The rest can be a total **uses** and you can still read it without problem . This is because the human mind does not **bear** every letter by itself , but the word as a whole .

Table 6: An illustrative example of spelling correction outputs for the *Cambridge* sentence. Words that the models fail to correct are underlined and bold.

work, MUDE, that achieves very strong and robust performance with different types of noise presenting. The proposed framework is able to capture both character and word-level dependencies to obtain effective word representations. Extensive experiments on datasets with various types of noise have demonstrated its superior performance over the exiting popular models.

There are several meaningful future research directions that are worthy exploring. The first is to extend MUDE to deal with sentences where word-level noise presents. For example, in the noised sentences, some of the words might be swapped, dropped, inserted or replaced, etc. In addition, it is also meaningful to improve the generality of MUDE such that it can achieve strong performance with the presence of various types of noise not seen in the training dataset. Another possible future direction is to utilize MUDE to improve the robustness other NLP systems including machine translation, reading comprehension, text classification, etc. Lastly, as this work primarily focuses on English, it would be very meaningful to experiment the proposed framework on other languages.

Acknowledgements

Zhiwei Wang and Jiliang Tang are supported by the National Science Foundation (NSF) under grant numbers IIS-1714741, IIS-1715940, IIS-1845081 and CNS-1815636.

References

Andermatt, S.; Pezold, S.; and Cattin, P. 2016. Multi-dimensional gated recurrent units for the segmentation of biomedical 3d-data. In *Deep Learning and Data Labeling for Medical Applications*. Springer. 142–151.

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

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*.

Chollampatt, S., and Ng, H. T. 2017. Connecting the dots: Towards human-level grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, 327–333.

Davis, M. 2012. Psycholinguistic evidence on scrambled letters in reading.

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

Dinan, E.; Humeau, S.; Chintagunta, B.; and Weston, J. 2019. Build it break it fix it for dialogue safety: Robustness from adversarial human attack. *arXiv:1908.06083*.

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

Fumera, G.; Pillai, I.; and Roli, F. 2006. Spam filtering based on the analysis of text information embedded into images. *Journal of Machine Learning Research* 7(Dec):2699–2720.

Grosse, K.; Papernot, N.; Manoharan, P.; Backes, M.; and McDaniel, P. 2017. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*, 62–79. Springer.

Grundkiewicz, R., and Junczys-Dowmunt, M. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. *arXiv:1804.05945*.

Iyyer, M.; Wieting, J.; Gimpel, K.; and Zettlemoyer, L. 2018. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv:1804.06059*.

Ji, J.; Wang, Q.; Toutanova, K.; Gong, Y.; Truong, S.; and Gao, J. 2017. A nested attention neural hybrid model for grammatical error correction. *arXiv:1707.02026*.

Junczys-Dowmunt, M.; Grundkiewicz, R.; Guha, S.; and Heafield, K. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. *arXiv:1804.05940*.

Liu, H.; Derr, T.; Liu, Z.; and Tang, J. 2019. Say what i want: Towards the dark side of neural dialogue models. *arXiv:1909.06044*.

Marcus, M.; Santorini, B.; and Marcinkiewicz, M. A. 1993. Building a large annotated corpus of english: The penn treebank.

Ng, H. T.; Wu, S. M.; Briscoe, T.; Hadiwinoto, C.; Susanto, R. H.; and Bryant, C. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, 1–14.

Perea, M.; Jiménez, M.; Talero, F.; and López-Cañada, S. 2015. Letter-case information and the identification of brand names. *British Journal of Psychology* 106(1):162–173.

Pruthi, D.; Dhingra, B.; and Lipton, Z. C. 2019. Combating adversarial misspellings with robust word recognition. *arXiv:1905.11268*.

Rayner, K.; White, S. J.; and Liversedge, S. 2006. Raeding wrods with jubmled lettres: There is a cost.

Sakaguchi, K.; Duh, K.; Post, M.; and Van Durme, B. 2017. Robsut wrod reocginiton via semi-character recurrent neural network. In *AAAI2017*.

Sutskever, I.; Martens, J.; and Hinton, G. E. 2011. Generating text with recurrent neural networks. In *ICML-11*, 1017–1024.

Tieleman, T., and Hinton, G. 2012. Rmsprop. *COURSERA: Lecture* 7017.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wolf, M. J.; Miller, K.; and Grodzinsky, F. S. 2017. Why we should have seen that coming: comments on microsoft's tay experiment, and wider implications. *ACM SIGCAS Computers and Society* 47(3):54–64.

Xu, D.; Cheng, W.; Luo, D.; Gu, Y.; Liu, X.; Ni, J.; Zong, B.; Chen, H.; and Zhang, X. 2019a. Adaptive neural network for node classification in dynamic networks. In *ICDM*. IEEE.

Xu, D.; Cheng, W.; Luo, D.; Liu, X.; and Zhang, X. 2019b. Spatio-temporal attentive rnn for node classification in temporal attributed graphs. In *IJCAI*, 3947–3953.

Xu, H.; Ma, Y.; Liu, H.; Deb, D.; Liu, H.; Tang, J.; and Jain, A. 2019c. Adversarial attacks and defenses in images, graphs and text: A review. *arXiv:1909.08072*.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; and Le, Q. V. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Yu, A. W.; Dohan, D.; Luong, M.-T.; Zhao, R.; Chen, K.; Norouzi, M.; and Le, Q. V. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv:1804.09541*.

Zhao, W.; Wang, L.; Shen, K.; Jia, R.; and Liu, J. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. *arXiv:1903.00138*.

Zhao, Z.; Dua, D.; and Singh, S. 2017. Generating natural adversarial examples. *arXiv:1710.11342*.

Zhou, S.; Zeng, X.; Zhou, Y.; Anastasopoulos, A.; and Neubig, G. 2019. Improving robustness of neural machine translation with multi-task learning. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, 565–571.