Neural Language Correction with Character-Based Attention

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, Andrew Y. Ng
Computer Science Department, Stanford University
{zxie,avati,naveen67,ang}@cs.stanford.edu, jurafsky@stanford.edu

Abstract

Natural language correction has the potential to help language learners improve their writing skills. While approaches with separate classifiers for different error types have high precision, they do not flexibly handle errors such as redundancy or non-idiomatic phrasing. On the other hand, word and phrase-based machine translation methods are not designed to cope with orthographic errors, and have recently been outpaced by neural models. Motivated by these issues, we present a neural network-based approach to language correction. The core component of our method is an encoder-decoder recurrent neural network with an attention mechanism. By operating at the character level, the network avoids the problem of out-of-vocabulary words. We illustrate the flexibility of our approach on dataset of noisy, user-generated text collected from an English learner forum. When combined with a language model, our method achieves a state-of-the-art $F_{0.5}$-score on the CoNLL 2014 Shared Task. We further illustrate that training the network on additional data with synthesized errors can improve performance.

1 Introduction

Systems that provide writing feedback have great potential to assist language learners as well as native writers. Although tools such as spell checkers have been useful, detecting and fixing errors in natural language, even at the sentence level, remains far from solved.

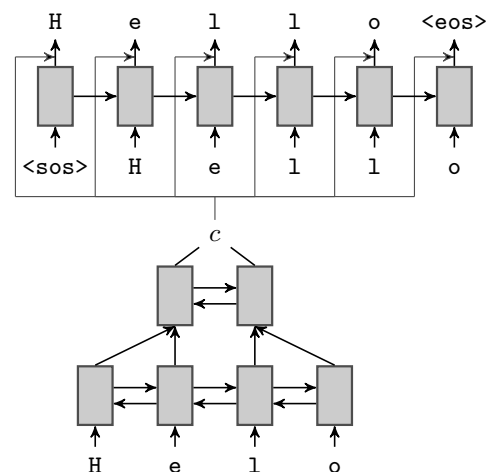Much of the prior research focuses solely on training classifiers for a small number of error

Figure 1: Illustration of the encoder-decoder neural network model with two encoder hidden layers and one decoder hidden layer. Character-level reasoning allows handling of misspellings and OOVs.

types, such as article or preposition errors (Han et al., 2006; Rozovskaya and Roth, 2010). More recent methods that consider a broader range of error classes often rely on language models to score n-grams or statistical machine translation approaches (Ng et al., 2014). These methods, however, do not flexibly handle orthographic errors in spelling, capitalization, and punctuation.

As a motivating example, consider the following incorrect sentence: "*I visitted Tokyo on Nov 2003. :)*". Several errors in this sentence illustrate the difficulties in the language correction setting. First, the sentence contains a misspelling, *visitted*, an issue for systems with fixed vocabularies. Second, the sentence contains rare words such as *2003* as well as punctuation forming an emoticon *:)*, issues that may require special handling. Finally, the use of the preposition *on* instead of *in* when not referring to a specific day is non-idiomatic, demonstrating the complex patterns that must be

captured to suggest good corrections. In hopes of capturing such complex phenomena, we use a neural network-based method.

Building on recent work in language modeling and machine translation, we propose an approach to natural language error correction based on an encoder-decoder recurrent neural network trained on a parallel corpus containing "good" and "bad" sentences (Figure 1). When combined with a language model, our system obtains state-of-the-art results on the CoNLL 2014 Shared Task, beating systems using statistical machine translation systems, rule-based methods, and task-specific features. Our system naturally handles orthographic errors and rare words, and can flexibly correct a variety of error types. We further find that augmenting the network training data with sentences containing synthesized errors can result in significant gains in performance.

2 Model Architecture

Given an input sentence x that we wish to map to an output sentence y, we seek to model $P(y|x)$. Our model consists of an encoder and a decoder (Sutskever et al., 2014; Cho et al., 2014). The encoder maps the input sentence to a higher-level representation with a pyramidal bidirectional RNN architecture similar to that of Chan et al. (2015). The decoder is also a recurrent neural network that uses a content-based attention mechanism (Bahdanau et al., 2014) to attend to the encoded representation and generate the output sentence one character at a time.

2.1 Character-Level Reasoning

Our neural network model operates at the character level, in the encoder as well as the decoder. This is for two reasons, as illustrated by our motivating example. First, we do not assume that the inputs are spell-checked and often find spelling errors in the sentences written by English learners in the datasets we consider. Second, word-level neural MT models with a fixed vocabulary are poorly suited to handle OOVs such as multi-digit numbers, emoticons, and web addresses (Graves, 2013), though recent work has proposed workarounds for this problem (Luong et al., 2014). Despite longer sequences in the character-based model, optimization does not seem to be a significant issue, since the network often only needs to copy characters from source to target.

2.2 Encoder Network

Given the input vector x_t, the forward, backward, and combined activations of the jth hidden layer are computed as:

$$f_t^{(j)} = \text{GRU}(f_{t-1}^{(j)}, c_t^{(j-1)}),$$
$$b_t^{(j)} = \text{GRU}(b_{t+1}^{(j)}, c_t^{(j-1)}),$$
$$h_t^{(j)} = f_t^{(j)} + b_t^{(j)}$$

where GRU denotes the gated recurrent unit function, which, similar to long short-term memory units (LSTMs), have shown to improve the performance of RNNs (Cho et al., 2014; Hochreiter and Schmidhuber, 1997).

The input from the previous layer input $c_t^{(0)} = x_t$ and

$$c_t^{(j)} = \tanh\left(W_{\text{pyr}}^{(j)} \left[h_{2t}^{(j-1)}, h_{2t+1}^{(j-1)} \right]^\top + b_{\text{pyr}}^{(j)} \right)$$

for $j > 0$. The weight matrix W_{pyr} thus reduces the number of hidden states for each additional hidden layer by half, and hence the encoder has a pyramid structure. At the final hidden layer we obtain the encoded representation c consisting of $\lceil T/2^{N-1} \rceil$ hidden states, where N denotes the number of hidden layers.

2.3 Decoder Network

The decoder network is recurrent neural network using gated recurrent units with M hidden layers. After the final hidden layer the network also conditions on the encoded representation c using an attention mechanism.

At the jth decoder layer the hidden activations are computed as

$$d_t^{(j)} = \text{GRU}(d_{t-1}^{(j)}, d_t^{(j-1)}),$$

with the output of the final hidden layer $d_t^{(M)}$ then being used as part of the content-based attention mechanism similar to that proposed by Bahdanau et al. (2014):

$$u_{tk} = \phi_1(d_t^{(M)})^\top \phi_2(c_k)$$
$$\alpha_{tk} = \frac{u_{tk}}{\sum_j u_{tj}}$$
$$a_t = \sum_j \alpha_{tj} c_j$$

where ϕ_1 and ϕ_2 represent feedforward affine transforms followed by a $\tanh$ nonlinearity. The

weighted sum of the encoded hidden states a_t is then concatenated with $d_t^{(M)}$, and passed through another affine transform followed by a ReLU nonlinearity before the final softmax output layer.

The loss function is the cross-entropy loss per time step summed over the output sequence y:

$$L(x, y) = -\sum_{t=1}^{T} \log P(y_t|x, y_{<t}).$$

Note that during training the ground truth y_{t-1} is fed into the network to predict y_t, while at test time the most probable $\hat{y}_{t-1}$ is used. Figure 1 illustrates the model architecture.

2.4 Attention and Pyramid Structure

In preliminary experiments, we found that having an attention mechanism was crucial for the model to be able to generate outputs character-by-character that did not diverge from the input. While character-based approaches have not attained state-of-the-art performance on large scale translation and language modeling tasks, in this setting the decoder network simply needs to copy input tokens during the majority of time steps.

Although character-level models reduce the softmax over the vocabulary at each time step over word-level models, they also increase the total number of time-steps of the RNN. The content-based attention mechanism must then consider all the encoder hidden states $c_{1:T}$ at every step of the decoder. Thus we use a pyramid architecture, which reduces computational complexity (as shown by Chan et al. (2015)). For longer batches, we observe over a $2\times$ speedup for the same number of parameters when using a 400 hidden unit per layer model with 3 hidden layers ($4\times$ reduction of steps in c).

3 Decoding

While it is simpler to integrate a language model by using it as a re-ranker, here the language model probabilities are combined with the encoder-decoder network through beam search. This is possible because the attention mechanism in the decoder network prevents the decoded output from straying too far from the source sentence.

3.1 Language Model

To model the distribution

$$P_{\text{LM}}(y_{1:T}) = \prod_{t=1}^{T} P(y_t|y_{<t}) \tag{1}$$

we build a Kneser-Ney smoothed 5-gram language model on a subset of the Common Crawl Repository[1] collected during 2012 and 2013. After pruning, we obtain 2.2 billion n-grams. To build and query the model, we use the KenLM toolkit (Heafield et al., 2013).

3.2 Beam Search

For inference we use a beam search decoder combining the neural network and the language model likelihood. Similar to Hannun et al. (2014), at step k, we rank the hypotheses on the beam using the score

$$s_k(y_{1:k}|x) = \log P_{\text{NN}}(y_{1:k}|x) + \lambda \log P_{\text{LM}}(y_{1:k})$$

where the hyper-parameter λ determines how much the language model is weighted. To avoid penalizing longer hypotheses, we additionally normalize scores by the number of words in the hypothesis $|y|$. Since decoding is done at the character level, the language model probability $P_{\text{LM}}(\cdot)$ is only incorporated after a space or end-of-sentence symbol is encountered.

3.3 Controlling Precision

For many error correction tasks, precision is emphasized more than recall; for users, an incorrect suggestion is worse than a missed mistake.

In order to filter spurious edits, we train an edit classifier to classify edits as correct or not. We run our decoder on uncorrected sentences from our training data to generate candidate corrected sentences. We then align the candidate sentences to the uncorrected sentences by minimizing the word-level Levenshtein distance between each candidate and uncorrected sentence. Contiguous segments that do not match are extracted as proposed edits[2]. We repeat this alignment and edit extraction process for the gold corrected sentences and the uncorrected sentences to obtain the gold edits. "Good" edits are defined as the intersection of the proposed and gold edits and "bad" edits are defined as the proposed edits not contained in the gold edits. We compute edit features and train a multilayer perceptron binary classifier on the extracted edits to predict the probability of an edit being correct. The features computed on an edit $s \rightarrow t$ are:

---

[1] http://commoncrawl.org
[2] Note this is an approximation and cannot distinguish side-by-side edits as separate edits.

- **edit distance features**: normalized word and character lengths of s and t, normalized word and character insertions, deletions, and substitutions between s and t.

- **embedding features**: sum of 100 dimensional GloVe (Pennington et al., 2014) vectors of words in s and t, GloVe vectors of left and right context words in s.

In order to filter incorrect edits, we only accept edits whose predicted probability exceeds a threshold $p_{\min}$. This assumes that classifier probabilities are reasonably calibrated (Niculescu-Mizil and Caruana, 2005). Edit classification improves precision with a small drop in recall; most importantly, it helps filter edits where the decoder network misbehaves and t deviates wildly from s.

4 Experiments

We perform experiments using two datasets of corrected sentences written by English learners. The first is the Lang-8 Corpus, which contains erroneous sentences and their corrected versions collected from a social language learner forum (Tajiri et al., 2012). Due to the online user-generated setting, the Lang-8 data is noisy, with sentences often containing misspellings, emoticons, and other loose punctuation. Sample sentences are show in Table 4.

The other dataset we consider comes from the CoNLL 2013 and 2014 Shared Tasks, which contain about 60K sentences from essays written by English learners with corrections and error type annotations. We use the larger Lang-8 Corpus primarily to train our network, then evaluate on the CoNLL Shared Tasks.

4.1 Training and Decoding Details

Our pyramidal encoder has 3 layers, resulting in a factor 4 reduction in the sequence length at its output, and our decoder RNN has 3 layers as well. Both the encoder and decoder use a hidden size of 400 and gated recurrent units (GRUs), which along with LSTMs (Hochreiter and Schmidhuber, 1997) have been shown to be easier to optimize and preserve information over many time steps better than vanilla recurrent networks.

Our vocabulary includes 98 characters: the printing ASCII character set and special $\langle sos \rangle$, $\langle eos \rangle$, and $\langle unk \rangle$ symbols indicating the start-of-sentence, end-of-sentence, and unknown symbols, respectively.

Method	Test BLEU
No edits	59.54
Spell check	58.91
RNN	61.63
RNN + LM	**61.70**

Table 1: Performance on Lang-8 test set. Adding the language model results in a negligible increase in performance, illustrating the difficulty of the user-generated forum setting.

To train the encoder-decoder network we use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0003, default decay rates β_1 and β_2, and a minibatch size of 128. We train for up to 40 epochs, selecting the model with the lowest perplexity on the Lang-8 development set. We found that using dropout (Srivastava et al., 2014) at a rate of 0.15 on the non-recurrent connections (Pham et al., 2014) helped reduce perplexity. We use uniform initialization of the weight matrices in the range $[-0.1, 0.1]$ and zero initialization of biases.

Decoding parameter λ and edit classifier threshold $p_{\min}$ were chosen to maximize performance on the development sets of the datasets described. All results were obtained using a beam width of 64, which seemed to provide a good trade-off between speed and performance.

4.2 Noisy Data: Lang-8 Corpus

We use the train-test split provided by the Lang-8 Corpus of Learner English (Tajiri et al., 2012), which contains 100K and 1K entries with about 550K and 5K parallel sentences, respectively. We also split 5K sentences from the training set to use as a separate development set for model and parameter selection.

Since we do not have gold annotations that distinguish side-by-side edits as separate edits, we report BLEU score[3] using just the encoder-decoder network as well as when combined with the n-gram language model (Table 1). Note that since there may be multiple ways to correct an error and some errors are left uncorrected, the baseline of using uncorrected sentences is more difficult to improve upon than it may initially appear. As another baseline we apply the top suggestions from

---

[3]Using case-sensitive `multi-bleu.perl` from Moses.

Method	P	R	$F_{0.5}$
RNN	42.96	6.27	19.81
RNN aug	49.30	10.10	27.75
RNN + LM	43.27	15.14	31.55
RNN aug + LM	46.94	**17.11**	34.81
RNN aug + LM + EC	**51.38**	15.83	**35.45**

Table 2: Development set performance. EC denotes edit classification (Section 3.3), and "aug" indicates data augmentation was used.

Method	P	R	$F_{0.5}$
AMU	41.62	21.40	35.01
CUUI	41.78	24.88	36.79
CAMB	39.71	30.10	37.33
Susanto (2014)	53.55	19.14	39.39
Ours (no EC)	45.86	26.40	39.97
Ours (+ EC)	49.24	23.77	**40.56**
Ours (A1)	32.56	14.76	26.23
Ours (A2)	44.04	14.83	31.59
A1 (A2)	50.47	32.29	**45.36**
A2 (A1)	37.14	45.38	**38.54**

Table 3: CoNLL 2014 test set performance. We compare to the 3 best CoNLL 2014 submissions which used combinations of MT, LM ranking, and error type-specific classifiers. We report F-score against both and single annotators, as well as each annotator scored against the other as a human ceiling. A1 and A2 denote Annotators 1 and 2.

a spell checker with default configurations[4]. We suspect due to proper nouns, acronyms, and inconsistent capitalization conventions in Lang-8, however, this actually decreased BLEU slightly. To the best of our knowledge, no other work has reported results on this challenging task.

4.3 Main Results: CoNLL Shared Tasks

Description For our second set of experiments we evaluate on the CoNLL 2014 Shared Task on Grammatical Error Correction (Ng et al., 2013; Ng et al., 2014). We use the revised CoNLL 2013 test data with all error types as a development set for parameter and model selection with the 2014 test data as our test set. The 2013 test data contains 1381 sentences with 3470 errors in total, and the

2014 test data contains 1312 sentences with 3331 errors. The CoNLL 2014 training set contains 57K sentences with the corresponding gold edits by a single annotator. The 2013 test set is also only labeled by a single annotator, while the 2014 test set has two separate annotators.

We use the NUS MaxMatch scorer (Dahlmeier and Ng, 2012) v3.2 in order to compute the precision (P), recall (R), and F-score for our corrected sentences. Since precision is considered more important than recall for the error correction task, $F_{0.5}$ score is reported as in the CoNLL 2014 Challenge. We compare to the top submissions in the 2014 Challenge as well as the method by Susanto (2014), which combines 3 of the weaker systems to achieve the state-of-the-art result. All results reported on the 2014 test set exclude alternative corrections submitted by the participants.

Synthesizing Errors In addition to the Lang-8 training data, we include the CoNLL 2014 training data in order to train the encoder-decoder network. Following prior work, we additionally explore synthesizing additional sentences containing errors using the CoNLL 2014 training data (Felice and Yuan, 2014; Rozovskaya et al., 2012). Our data augmentation procedure generates synthetic errors for two of the most common error types in the development set: article or determiner errors (ArtOrDet) and noun number errors (Nn). Similar to Felice and Yuan (2014), we first collect error distribution statistics from the CoNLL 2014 training data. For ArtOrDet errors, we estimate the probability that an article or determiner is deleted, replaced with another determiner, or inserted before the start of a noun phrase. For Nn errors, we estimate the probability that it is replaced with its singular or plural form. To obtain sentence parses we use the Stanford CoreNLP Toolkit (Manning et al., 2014). Example synthesized errors:

- **ArtOrDet**: They will generate and brainstorm _the_ innovative ideas.

- **Nn**: Identification is becoming more important in our _society_ $\rightarrow$ _societies_.

Errors are introduced independently according to their estimated probabilities by iterating over the words in the training sentences, and we produce two corrupted versions of each training sentence whenever possible. The original Lang-8 training data contains 550K sentence pairs. Adding the

CoNLL 2014 training data results in about 610K sentence pairs, and after data augmentation we obtain a total of 720K sentence pairs. We examine the benefits of synthesizing errors in Section 5.

Results   Results for the development set are shown in Table 2, and results for the CoNLL 2014 test set in Table 3. On the CoNLL 2014 test set, which contains the full set of 28 error types, our method achieves a state-of-the-art result, beating all systems from the 2014 Challenge as well as a system combination method (Susanto, 2014). Methods from the 2014 Challenge used statistical machine translation, language model ranking, rule-based approaches, and error type-specific features and classifiers, often in combination. System descriptions for participating teams are given in Ng et al. (2014).

5   Discussion

Qualitative Analysis   We present examples of correct and incorrect edits on Lang-8 development set in Table 4 and Table 5. Despite operating at the character level, the network is occasionally able to perform rearrangements of words to form common phrases (e.g. *I and my roommate → my roommate and I*) and insert and delete words where appropriate. On the other hand, the network can also sometimes mangle rare words (*Moodysson →*
Moodysnot) and fail to split common words missing a separating space (*withthe → withth*), suggesting that while common patterns are captured, the network lacks semantic understanding.
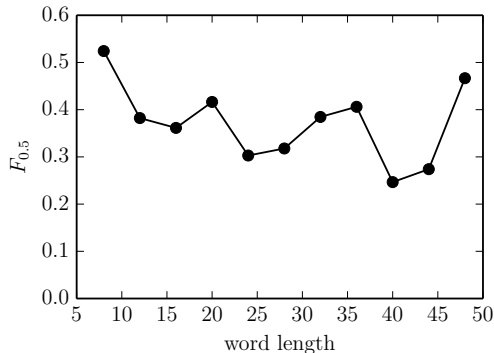

Figure 2: F-score vs. length of input sentence on development set. Only bins with 10 or more sentences included.

Performance Breakdown   While the encoder-decoder network can itself produce modifications, on less noisy datasets such as the CoNLL

Type	Count	R no aug	R aug
ArtOrDet	717	20.08	29.14
Wci	434	2.30	1.61
Nn	400	31.50	51.00
Preposition	315	13.01	7.93
Word form	223	26.90	19.73

Table 6: CoNLL development set recall for 5 most frequent error categories with and without training on data with synthesized article/determiner and noun number errors. Wci denotes wrong collocation/idiom errors.

Challenge datasets a language model can greatly improve performance. Increasing the language model weight λ tends to improves recall at the expense of precision. On the other hand, using edit classification to filter spurious edits increases precision, often with smaller drops in recall. We do not observe a trend of decreasing F-score for a wide range of sentence lengths (Figure 2), likely due to the attention mechanism, which helps to prevent the decoded output from diverging from the input sentence.

We report the inter-annotator agreement in Table 3, which gives a possible bound on the F-score for this task.

Effects of Data Augmentation   We obtain promising improvements using data augmentation, boosting $F_{0.5}$-score on the development set from 31.55 to 34.81. For the two error types where we synthesize data (article or determiner and noun number) we observe significant increases in recall, as shown in Table 6. The same phenomenon has been observed by Rozovskaya et al. (2012). Interestingly, the recall of other error types (see Ng et al. (2014) for descriptions) decreases. We surmise this is because the additional training data contains only ArtOrDet and Nn errors, and hence the network is encouraged to simply copy the output when those error types are not present. We hope synthesizing data with a variety of other error types may fix this issue and improve performance.

Challenging Error Types   We now examine a few illustrative error types from the CoNLL Challenges that originally motivated our approach: orthographic (Mec), redundancy (Rloc-), and idiomatic errors (Wci). Since the 2013 Challenge

	Original	Proposed
1	It 's *heavy rain* today	It 's *raining heavily* today
2	Everyone wants to be *success* .	Everyone wants to be *successful* .
3	*On the* 3 weeks , I learned many things .	*In the last* 3 weeks , I learned many things .
4	*this is the* first entry ! : D	*This is my* first entry ! : D
5	Help me *getting English skill* , please .	Help me *improve my English skills* , please .
6	*At* last night , the 24th of June 2010 was big night for the Japanese national team and heaps of fans .	Last night , the 24th of June 2010 was *a* big night for the Japanese national team and heaps of fans .
7	I *start to learning* English again .	I *am starting to learn* English again .
8	I went to *Beijin* in China *for* four days *in* this week .	I went to *Beijing* in China four days this week .
9	After a long day , *I and my roommate* usually sit down , drink coffee and listen to music .	After a long day , *my roommate and I* usually sit down , drink coffee and listen to music .
10	Do you know *a Toeic* ?	Do you know about *TOEIC* ?

Table 4: Sample edits from our Lang-8 development set using only the character-based encoder-decoder network. Note that the model is able to handle misspellings (*Beijin*) as well as rare words (*TOEIC*) and emoticons (*: D*).

	Original	Proposed
1	Broke my heart	*I* broke my heart
2	I want to big size bag	I want to *be* a big size bag
3	This is typical Japanese male *hobit*	This is *a* typical Japanese male *hobby*
4	I 'm so sorry to miss Lukas *Moodysson 's* Lijia 4-ever .	I 'm so sorry to miss Lukas *Moodysnot* Lijia 4-ever .
5	The match is the Rockets *withthe* Bulls .	The match is the Rockets *withth* Bulls .

Table 5: Sample incorrect and ambiguous edits, again using just the encoder-decoder network.

did not score these error types, we compare our recall to those of participants in the 2014 Challenge (Ng et al., 2014).[5] Note that systems only predict corrected sentences and not error types, and hence precision is not compared. We use the results from our final system, including both data augmentation and edit classification. Some examples of these error types are shown in Table 7.

- **Mec**: We obtain a recall of 37.17 on the Mec error type, higher than all the 2014 Challenge teams besides one team (RAC) that used rule-based methods to attain 43.51 recall. The word/phrase-based translation and language modeling approaches do not seem to perform as well for fixing orthographic errors.

- **Rloc-**: Redundancy is difficult to capture

using just rule-based approaches and classifiers; our approach obtains 17.47 recall which places second among the 12 teams. The top system obtains 20.16 recall using a combination MT, LM, and rule-based method.

- **Wci**: Although there are 340 collocation errors, all teams performed poorly on this category. Our recall places 3rd behind two teams (AMU and CAMB) whose methods both used an MT system. Again, this demonstrates the difficulty of capturing whether a sentence is idiomatic through only classifiers and rule-based methods.

We note that our system obtains significantly higher precision than any of the top 10 teams in the 2014 Challenge (49.24 vs. 41.78), which comes at the expense of recall.

Type	Description	Original	Proposed
Mec	Spelling, punctuation, capitalization, etc.	Another identification is *Implanting* RFID chips ...	Another identification is *implanting* RFID chips ...
Rloc-	Redundancy	... it seems that our freedom *of doing things* is being invaded.	... it seems that our freedom is being invaded.
Wci	Wrong collocation/idiom	Every coin has *its* two sides.	Every coin has two sides.

Table 7: Examples of the aforementioned challenging error types that our system fixed.

Limitations A key limitation of our method as well as most other translation-based methods is that it is trained on just parallel sentences, despite some errors requiring information about the surrounding text to make the proper correction. Even within individual sentences, when longer context is needed to make a correction (for example in many subject-verb agreement errors), the performance is hit-and-miss. The edits introduced by the system tend to be fairly local.

Other errors illustrate the need for natural language understanding, for example in Table 5 the correction *Broke my heart → I broke my heart* and *I want to big size bag → I want to be a big size bag*. Finally, although end-to-end approaches have the potential to fix a wide variety of errors, it is not straightforward to then classify the types of errors being made. Thus the system cannot easily provide error-specific feedback.

6 Related Work

Our work primarily builds on prior work on training encoder-decoder RNNs for machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014). The attention mechanism, which allows the decoder network to copy parts of the source sentence and cope with long inputs, is based on the content-based attention mechanism introduced by Bahdanau et al. (2014), and the overall network architecture is based on that described by Chan et al. (2015). Our model is also inspired by character-level models as proposed by Graves (2013). More recent work has applied character-level models to machine translation and speech recognition as well, suggesting that it may be applicable to many other tasks that involve the problem of OOVs (Ling et al., 2015; Maas et al., 2015; Chan et al., 2015).

Treating grammatical error correction as a statistical machine translation problem is an old idea; the method of mapping "bad" to "good" sentences was used by many of the teams in the CoNLL 2014 Challenge (Felice et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2014). The work of Felice et al. (2014) achieved the best $F_{0.5}$-score of 37.33 in that year's challenge using a combination of rule-based, language-model ranking, and statistical machine translation techniques. Many other teams used a language model for re-ranking hypotheses as well. Other teams participating in the CoNLL 2014 Challenge used techniques ranging from rule-based systems to type-specific classifiers, as well as combinations of the two (Rozovskaya et al., 2014; Lee and Lee, 2014). The rule-based systems often focus on only a subset of the error types. The previous state of the art was achieved by Susanto (2014) using the system combination method proposed by Heafield and Lavie (2010) to combine three weaker systems.

Finally, our work uses data collected and shared through the generous efforts of the teams behind the CoNLL and Lang-8 datasets (Mizumoto et al., 2011; Mizumoto et al., 2012; Ng et al., 2013; Ng et al., 2014). Prior work has also proposed data augmentation for the language correction task (Felice and Yuan, 2014; Rozovskaya et al., 2012).

Conclusion

We present a neural network-based model for performing language correction. Our system is able correct errors on noisy data collected from an English learner forum and attains state-of-the-art performance on the CoNLL 2014 Challenge dataset of annotated essays. Key to our approach is the use of a character-based model with an attention mechanism, which allows for orthographic errors to be captured and avoids the OOV problem suffered by word-based neural machine translation methods. We hope the generality of this approach will also allow it to be applied to other tasks that must deal with noisy text, such as in the online user-generated setting.

References

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bergstra et al.2010] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

[Chan et al.2015] William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. 2015. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*.

[Cho et al.2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Dahlmeier and Ng2012] Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

[Felice and Yuan2014] Mariano Felice and Zheng Yuan. 2014. Generating artificial errors for grammatical error correction. In *EACL*.

[Felice et al.2014] Mariano Felice, Zheng Yuan, istein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *In Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

[Graves2013] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

[Han et al.2006] Na-Rae Han, Martin Chodorow, and Claudia Leakcock. 2006. Detecting errors in english article usage by non-native speakers. *Natural Language Engineering*.

[Hannun et al.2014] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

[Heafield and Lavie2010] Kenneth Heafield and Alon Lavie. 2010. Cmu multi-engine machine translation for wmt 2010. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10.

[Heafield et al.2013] K. Heafield, I. Pouzyrevsky, J. H. Clark, and P. Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *ACL-HLT*, pages 690–696, Sofia, Bulgaria.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

[Junczys-Dowmunt and Grundkiewicz2014] Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The amu system in the conll-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

[Kalchbrenner and Blunsom2013] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[Kingma and Ba2014] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Lee and Lee2014] Kyusong Lee and Gary Geunbae Lee. 2014. Postech grammatical error correction system in the conll-2014 shared task. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

[Ling et al.2015] Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

[Luong et al.2014] Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

[Maas et al.2015] Andrew L. Maas, Ziang Xie, Dan Jurafsky, and Andrew Y. Ng. 2015. Lexicon-free conversational speech recognition with neural networks. In *Proceedings the North American Chapter of the Association for Computational Linguistics (NAACL)*.

[Manning et al.2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit.

In *Association for Computational Linguistics (ACL) System Demonstrations*.

[Mizumoto et al.2011] Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning sns for automated japanese error correction of second language learners. In *International Joint Conference on Natural Language Processing (IJCNLP)*.

[Mizumoto et al.2012] Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yuji Matsuomto. 2012. The effect of learner corpus size in grammatical error correction of ESL writings. In *International Conference on Computational Linguistics*.

[Ng et al.2013] Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction.

[Ng et al.2014] Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction.

[Niculescu-Mizil and Caruana2005] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *International Conference on Machine learning (ICML)*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[Pham et al.2014] Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*.

[Rozovskaya and Roth2010] Alla Rozovskaya and Dan Roth. 2010. Generating confusion sets for context-sensitive error correction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language (EMNLP)*.

[Rozovskaya et al.2012] Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. The ui system in the hoo 2012 shared task on error correction. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*.

[Rozovskaya et al.2014] Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. 2014. The illinois-columbia system in the conll-2014 shared task. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*.

[Susanto2014] Raymond Hendy Susanto. 2014. Systems combination for grammatical error correction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems (NIPS)*.

[Tajiri et al.2012] Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for esl learners using global context. In *Association for Computational Linguistics: Short Papers*.