Hypothesis Correction Based on Semi-character Recurrent Neural Network for End-to-end Speech Recognition

Yuuki Tachioka
* Denso IT Laboratory, Tokyo, Japan
E-mail: ytachioka@d-itlab.co.jp

Abstract—**End-to-end automatic speech recognition (ASR) has become popular because of its simple modeling, but it encounters out-of-vocabulary words more frequently than the conventional hybrid approaches that jointly use acoustic and language models. In particular, word-based end-to-end systems cannot output any words unseen in training data. To address this problem, character-based end-to-end systems have been proposed; however, they are susceptible to noise, and their output words are not necessarily correct in terms of language. This is because language constraints, such as lexicons and language models, are lacking in the decoding process. Thus, errors like misspellings occur frequently. In the field of natural language processing, to correct spelling errors, a semi-character recurrent neural network (scRNN) was proposed whose inputs are the counts of characters in a word and outputs are word ids. To apply scRNN to ASR, extensions are needed because scRNN focuses only on substitution errors. Here, to consider insertion and deletion errors, we introduce blank word symbols, similar to blank symbols in connectionist temporal classification, and word concatenation. Two different ASR tasks, a noisy ASR task and an ASR task with a large vocabulary, showed that scRNN with the proposed extension improved the word error rate.**

I. Introduction

Conventional automatic speech recognition (ASR) has used hybrid approaches that combine multiple models such as acoustic or language models with a pronunciation dictionary to achieve high accuracy [1]. The advancement of research on deep neural networks simplifies these approaches, and various end-to-end (E2E) types of systems have been proposed [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. The advantages of E2E systems are easy modeling, fast decoding, and no need for dictionaries. They directly convert acoustic features to symbols such as phonemes [3], [4], [5], [8], characters [2], [3], [6], [7], [11], [12], and words [9], [10].

Character-based or word-based E2E systems are simpler because there is no need for additional decoding with language models [2], [3], which has advantages such as a lower cost for model construction and easier applicability for different languages [13]. However, the data sparsity problem makes E2E infeasible for out-of-vocabulary (OOV) words unseen in training data because the amount of speech data is much less than that of written documents and OOV words appear more frequently than with conventional hybrid approaches. Word-based E2E systems output ASR hypotheses faster than others but cannot output OOV words without the joint use of character-based E2E systems [14]. Character-based E2E systems can avoid this problem because they can represent OOV words; however, they are infeasible for noise. In addition, errors like misspellings occur frequently, as shown in Section V, because words are produced on the basis of weaker language constraints.

In the field of natural language processing (NLP), neural machine translation (NMT) is widely used [15]. If a parallel corpus can be prepared, a neural network can learn any correspondence (translation). When ASR hypotheses are inputs and references are outputs, NMT models can be trained with sentence-to-sentence conversion. This translation globally changes word sequences but most ASR errors are local. Experiments in Section V show that NMT does not work well. In comparison, to correct spelling errors, the semi-character recurrent neural network (scRNN) was proposed [16]. It focuses on jumble errors and can correct local errors at the word level. Their experiments show that it achieved the highest performance in correcting spelling errors among state-of-the-art methods including commercial products.

To apply scRNN to ASR hypothesis correction, modifications are necessary because scRNN performs word-to-word conversion, which only deals with substitution errors, but it cannot deal with insertion and deletion errors where one-by-one word correspondences are not obtained. The experiments in [16] assumed that there are no insertion and deletion errors, but ASR causes many such errors. To extend their approach in order to address this problem, we introduce blank word symbols and word concatenation. From a different perspective, because they correct ASR errors, scRNN can produce a discriminative training like effect [17].

This paper is organized as follows. In Section II, we show the character-based E2E ASR approach we take, and we describe scRNN in Section III. To apply scRNN to ASR, in Section IV, we describe the drawbacks of scRNN and introduce both blank word symbols and word concatenation to scRNN. Experiments in Section V show the effectiveness of our proposed method on two different ASR datasets. The first one is the fourth CHiME challenge, which is a noisy ASR task. The second one is a TED-LIUM dataset, which is a large vocabulary ASR task. These experiments on different datasets show that, although NMT does not work at all, scRNN with our extension improves the word correct and error rate.

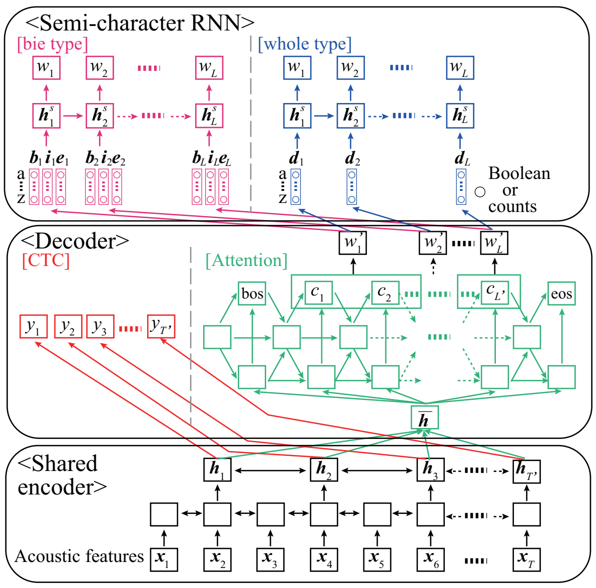

Fig. 1. Structure of proposed E2E system with scRNN based correction on top of ASR system.

II. End-to-end speech recognition

We use one of the state-of-the-art character-based E2E ASR systems as a baseline. E2E systems are mainly classified into two types: connectionist temporal classification (CTC) [2], [3], [5], [9], [10] and attention mechanism (ATT) [4], [6], [7], [8]. The lower two parts of Fig. 1, the shared encoder and decoder, show the E2E system we use, which jointly uses CTC and ATT [11], [12]. Inputs are acoustic features in T frames, $\boldsymbol{x}_1, ..., \boldsymbol{x}_T$. The encoder receives $\boldsymbol{x}_{1:T}$ and converts T'-length internal representations $\boldsymbol{h}_{1:T'}$, where T' does not necessarily match with T due to thinning out.

A. CTC model

Outputs of CTC are T'-length label sequences $y_1, ..., y_{T'}$. CTC allows blank symbols and repeated characters in labels to compensate for the difference between character length and label length T'. The probability of CTC for a character sequence $\boldsymbol{c}$ is independent between labels y as

$$P_{CTC}(\boldsymbol{c}|\boldsymbol{x}) = \sum_{\boldsymbol{y}\in\Psi(\boldsymbol{c})} \prod_{t'=1}^{T'} \sigma_{\boldsymbol{h}_{t'}(\boldsymbol{x})}(y_{t'}), \quad (1)$$

where Ψ is a set of all T'-length possible label sequences, $\boldsymbol{y} = y_1, ..., y_{t'}, ..., y_{T'}$, realizing a character sequence $\boldsymbol{c}$ and $\sigma_{\boldsymbol{h}_{t'}(\boldsymbol{x})}$ is a softmax output for the label $y_{t'}$ conditioned on the encoder output $\boldsymbol{h}_{t'}(\boldsymbol{x})$. Finally, CTC outputs characters after deleting blank symbols and repeated characters.

B. ATT model

Outputs of ATT are L'-length character sequences $c_1, ..., c_{L'}$ with the beginning of a sentence token (bos) and the ending of sentence token (eos). The probability of ATT is represented as a recursive form starting from $c_0 = \text{bos}$:

$$P_{ATT}(\boldsymbol{c}|\boldsymbol{x}) = \prod_{l'=1}^{L'} P(c_{l'}|\bar{\boldsymbol{h}}(\boldsymbol{x}), c_{0:(l'-1)}), \quad (2)$$

where $\bar{\boldsymbol{h}}$ is a bunch of encoder outputs $\boldsymbol{h}_1, ..., \boldsymbol{h}_{T'}$. These output characters are separated by spaces and converted to L-length word sequences $w_1, ..., w_L$. The relationship between L' and L is $L' = \sum_{l=1}^{L} |w_l|$, where $|w_l|$ is the number of characters in a word w_l. Output words are not necessarily correct in terms of language because this system does not use explicit language constraints. In fact, for E2E systems, performance has been frequently evaluated in terms of character error rates (CERs) rather than word error rates (WERs) because WERs are likely to be worse than the conventional systems that use language models even though the CERs are almost the same.

C. Multi-task learning of CTC and ATT

To exploit the advantages of CTC and ATT, a combination of these two models was proposed [11]. Multi-task loss for reference character sequences $\boldsymbol{c}^*$

$$\mathcal{L}(\boldsymbol{c}^*|\boldsymbol{x}) = -\lambda \ln P_{CTC}(\boldsymbol{c}^*|\boldsymbol{x}) - (1-\lambda) \ln P_{ATT}(\boldsymbol{c}^*|\boldsymbol{x}), \quad (3)$$

is to be minimized. CTC can take a monotonic alignment by using a forward-backward algorithm but cannot consider inter-label dependencies because of an assumed independence between labels. ATT can consider inter-label dependencies but is susceptible to noise because an ATT model uses weaker constraints than a CTC model. By using CTC-model training as an auxiliary task, CTC alignment makes convergence faster in multi-task learning.

III. Semi-character recurrent neural network (scRNN)

In the field of NLP, to correct spelling errors, scRNN was proposed with a focus mainly on jumble errors. These are permutation errors in internal characters with the beginning and ending characters being constant, e.g., "characters" $\rightarrow$ "chraatcres", because it is relatively easy to recognize an original form when the beginning and ending characters are the same [16]. To model this, three types of characters, i.e., beginning, ending, and internal, are separately dealt with. We validate original count type with separate modeling in addition to two different variations: boolean type and whole modeling.

A. Count type

The upper part of Fig. 1 shows the scRNN, which receives the counts of characters in a word converted from the words of ASR hypotheses, $w'_1, ..., w'_L$, and outputs the corrected word sequences $w_1, ..., w_L$. w_l is selected from the words, W_{tr}, appeared in the training data. The word w'_l is composed of $|w'_l|$ characters, $c^l_1, ...c^l_{|w'_l|}$, which is one of the prepared C types of characters. The dimension of input vectors is three times the number of characters including alphabets and tokens, because scRNN deals with three types of characters separately. Inputs $\boldsymbol{b}_l$ and $\boldsymbol{e}_l$ are the C-dimensional one-hot vector, and $\boldsymbol{i}_l$

is a sparse vector corresponding to the word w'_l. These vectors are represented by

$$\boldsymbol{b}_l = \boldsymbol{o}(c^l_1),\ \boldsymbol{e}_l = \boldsymbol{o}(c^l_{|w'_l|}),\ \boldsymbol{i}_l = \sum_{j=2}^{|w'_l|-1} \boldsymbol{o}(c^l_j). \tag{4}$$

Here, we denote $\boldsymbol{o}(c)$ as a one-hot vector whose element corresponding to the character c is unity.

For example, when the input word w is "speech," the respective vectors are $\boldsymbol{b} = \boldsymbol{o}(\text{‘s’})$, $\boldsymbol{i} = \boldsymbol{o}(\text{‘p’}) + \boldsymbol{o}(\text{‘e’}) + \boldsymbol{o}(\text{‘e’}) + \boldsymbol{o}(\text{‘c’}) = \boldsymbol{o}(\text{‘c’}) + 2\boldsymbol{o}(\text{‘e’}) + \boldsymbol{o}(\text{‘p’})$, and $\boldsymbol{e} = \boldsymbol{o}(\text{‘h’})$, The input to the scRNN is their concatenated vector and the output is a corrected word w_l as

$$\begin{aligned} \boldsymbol{h}^s_l &= LSTM\left(\boldsymbol{W}_1[\boldsymbol{b}_l^\top \boldsymbol{i}_l^\top \boldsymbol{e}_l^\top]^\top + \boldsymbol{W}_2\boldsymbol{h}^s_{l-1}\right), \\ w_l &= \underset{w\in W_{tr}}{\arg\max}\, \sigma_{\boldsymbol{W}_3\boldsymbol{h}^s_l}(w), \end{aligned} \tag{5}$$

where $\top$ denotes the transpose. Here, a long short-term memory (LSTM) model [18] with weight matrices ($\boldsymbol{W}_1$, $\boldsymbol{W}_2$, and $\boldsymbol{W}_3$) and hidden layer $\boldsymbol{h}^s_l$ is used to obtain word embeddings.

B. Boolean type

The counts of characters are susceptible to noise. Boolean variables may be more effective than counts. Although ambiguity increases and discrimination performance decreases, noise robustness might improve especially when misspellings like repeated characters or a drop of characters frequently appear. The experiments in Section V were done to compare the performance of count types with that of Boolean types. The Boolean counterparts of $\boldsymbol{i}_l$ in Eq. (4) are

$$\boldsymbol{i}_l = \boldsymbol{o}(c^l_2) \vee \boldsymbol{o}(c^l_3) \ldots \vee \boldsymbol{o}(c^l_{|w'_l|-1}), \tag{6}$$

where $\vee$ is a boolean disjunction and $\boldsymbol{b}$ and $\boldsymbol{e}$ are the same. For the above example, $\boldsymbol{i}$ is $\boldsymbol{o}(\text{‘p’}) \vee \boldsymbol{o}(\text{‘e’}) \vee \boldsymbol{o}(\text{‘e’}) \vee \boldsymbol{o}(\text{‘c’}) = \boldsymbol{o}(\text{‘c’}) + \boldsymbol{o}(\text{‘e’}) + \boldsymbol{o}(\text{‘p’})$.

C. Whole modeling instead of separate modeling

In addition, compared with separate modeling, the whole model uses the counts of whole characters as an input vector:

$$\boldsymbol{d}_l = \sum_{j=1}^{|w'_l|} \boldsymbol{o}(c^l_j). \tag{7}$$

For the above example, $\boldsymbol{d}$ is $\boldsymbol{o}(\text{‘c’}) + 2\boldsymbol{o}(\text{‘e’}) + \boldsymbol{o}(\text{‘h’}) + \boldsymbol{o}(\text{‘p’}) + \boldsymbol{o}(\text{‘s’})$ in count type and $\boldsymbol{o}(\text{‘c’}) + \boldsymbol{o}(\text{‘e’}) + \boldsymbol{o}(\text{‘h’}) + \boldsymbol{o}(\text{‘p’}) + \boldsymbol{o}(\text{‘s’})$ in Boolean type.

IV. Application of scRNN to ASR

A. Drawbacks of scRNN

To correct ASR hypotheses, it is necessary to deal with four types of errors consisting of substitution, insertion, and two types of deletion as shown in Fig. 2. Here, its input is the ASR hypothesis, and its output is the reference. scRNN can only be used for substitution errors in Fig. 2 (substitution). Original form of scRNN cannot deal with other types of errors. For insertion and deletion errors, it is necessary to compensate for the mismatch of word length between hypotheses and references, because one-by-one word correspondence is needed by scRNN. The simplest solution is to ignore insertion and deletion errors (ign), but this interrupts the context of words. We propose to use blank word symbols and word concatenation to compensate for these mismatches.

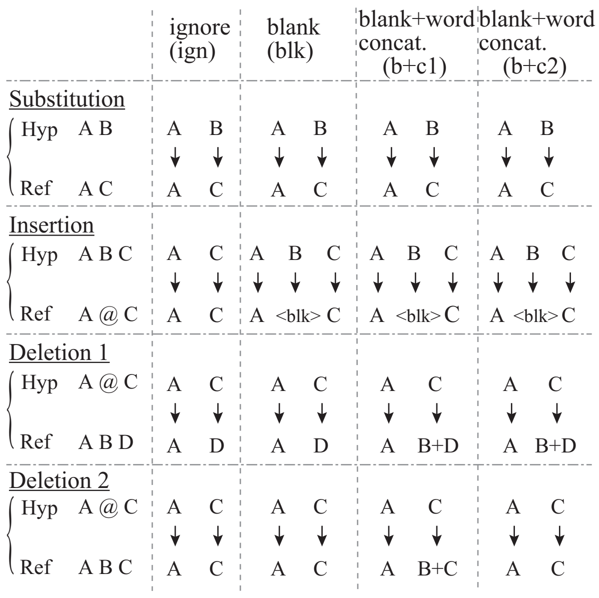

Fig. 2. Four types of blank word symbols (blk) and word concatenation in training stage, where hypotheses (Hyp) and references (Ref) are aligned and @ is null symbol. Input and output to scRNN are Hyp and Ref, respectively.

B. Blank word symbols

The insertion errors in Fig. 2 (insertion) are easier to handle than deletion errors. For insertion errors, we propose to introduce blank word symbols (blk), similar to the blank symbols in CTC. The symbols make a one-by-one word correspondence in insertion and deletion errors. An input word in a hypothesis can be related to a blank word symbol in the reference. Blank word symbols (blk) only focus on insertion errors and ignores deletion errors.

C. Word concatenation

For deletion errors, blank word symbols cannot be used because it is impossible to detect the existence of deletion in the hypotheses during testing time. To make one-by-one word correspondences between a reference and hypothesis, we use word concatenation where multiple words are concatenated and dealt with as if they are one word. Here, we propose two types of word concatenations: b+c1 and b+c2. In Fig. 2 (deletion 1), where a word after a deletion does not match, "C" in the hypotheses are related to "B D" in the reference, which are common to b+c1 and b+c2. If there are many deletions in a sentence, the length of concatenated words is long, and long concatenated words scarcely appear in the dataset and thus degrade the performance. In this paper, the maximum concatenation of words was set to be two. In Fig. 2 (deletion 2), where a word after a deletion matches that in the reference, "B" may be dropped and hard to recognize. Thus, ignoring "B" can be a better option, which is b+c2. This method can also be applied to word-based E2E combined with character-based E2E [14]. In addition, because this corrects ASR errors, it can have a discriminative training like effect [17].

V. Experiments

A. Experimental setups

We evaluated the proposed method on two datasets. The first one was a 1ch track of the fourth CHiME challenge (CHiME 4), which is a noisy ASR task without speech enhancement [19]. We expected that the proposed method can be more effective for noisy ASR tasks because E2E systems are susceptible to noisy data. The second one was a TED-LIUM dataset, which is a large vocabulary ASR task [20]. Training, development, and evaluation sets were prepared for both tasks.

Hypotheses were obtained by the espnet toolkit[1] [11], [12] with attached scripts. This E2E system did not use any language models or lexicons. Detailed setups are shown in [11]. Location-based ATT was used. The number of units was 320, and λ was 0.5. For decoding, the beam size was 20. In the shared encoder, the top two LSTM layers picked up every second hidden state of the lower layers, i.e., $T' = T/4$.

The vocabulary of scRNN, W_{tr} was constructed with words appearing in the respective training set. For development and evaluation sets, OOV words appeared. Special OOV word symbols (unk) were added to the scRNN. The scRNN was trained on the training set with a minibatch size of 256. LSTM models with 650 units were used with a dropout of 0.01. To adjust learning rates, Adam [21] was used. The number of epochs was 15. We modified the scripts[2] written by the authors of [16]. We compared two input vector types (Boolean and counts) and two types of modeling (three separated types of counts (bie) and the counts in whole words (w)). The dimension of the input character types C was 50, which consisted of alphabets (a–z) and tokens (hyphen, comma, period, etc.). There are four types of introduced blank word symbols and word concatenations in Fig. 2. "unk" and blank word symbols were deleted before evaluation.

In addition, the proposed method was compared with NMT by using the seq2seq toolkit[3] [22]. This model was trained as if it were a translation from ASR hypotheses as an original language to references as a target language. NMT was trained in terms of accuracy criterion (acc) and bleu [23] criterion.

B. Fourth CHiME challenge (Noisy ASR)

Table I shows the WERs [%] on the CHiME 4 development set. NMT performed the worst, although accuracy criterion was slightly better than the bleu criterion. Table II shows the detailed comparison in terms of the word correct (C), substitution (S), deletion (D), insertion (I), and error (E) rates [%]. Except for the word correct (C) rates, a lower rate indicates a better performance. The number of substitution errors increased significantly, whereas the number of deletions and insertions was almost the same as that of the baseline. NMT created likely sentences that did not hold the original meanings. The average number of words per sentence was close between the baseline (16.3 words/sent) and NMT (15.8 words/sent). scRNN (groups 2–4 in Table I) performed better than the baseline. In all cases, the Boolean type (B) was inferior to the count type (C). The influence of noise on the character counts was limited. The scRNN with blank word symbols (blk) outperformed the scRNN ignoring deletion and substitution errors (ign) because ign basically does not reduce the number of deletion and insertion errors except outputting "unk". Word concatenation (b+c1) achieved the best correct rate, but, in total, blk achieved the best WER. Unfortunately, b+c2 was worse than blk. A relative reduction of 15.7% in insertion error, 9.3% in substitution error, and 4.4% in word

TABLE I
WER[%] on CHiME 4 challenge development (dev) and evaluation (eval) set. Inputs were whole count (w) or separate count (bie). Input vector type was Boolean (B) or counts (C). scRNN has three types of inputs and outputs in Fig. 2. CERs of baseline on development set were 33.5% (real) and 33.7% (simu), and those on evaluation set were 44.1% (real) and 41.7% (simu), respectively.

			dev set		eval set	
type	model	B/C	real	simu	real	simu
baseline			64.7	64.0	78.1	75.2
ign	w	B	64.7	64.3	78.2	75.1
		C	63.3	62.5	77.2	74.1
	bie	B	63.1	62.4	77.1	73.8
		C	62.8	62.3	76.7	73.6
blk	w	B	64.5	63.8	77.4	74.4
		C	62.6	61.8	76.2	73.0
	bie	B	62.3	61.8	75.9	73.0
		C	**62.0**	**61.3**	**75.3**	**72.4**
b+c1	bie	B	62.8	62.1	76.5	73.6
		C	62.4	61.8	76.2	73.2
b+c2		B	62.7	62.0	76.4	73.4
		C	62.2	61.6	76.0	72.9
NMT (acc)			96.5	96.3	99.3	99.4
NMT (bleu)			97.2	97.2	100.0	100.0

TABLE II
Word correct (C), substitution (S), deletion (D), insertion (I), and error (E) rates [%] on CHiME 4 challenge. Baseline system was compared with ign (bie,C), blk (bie,C), b+c1 (bie,C), and NMT (acc) systems.

	real					simu				
type	C	S	D	I	E	C	S	D	I	E
dev set										
baseline	41.9	49.6	**8.5**	6.6	64.7	44.0	48.1	**7.9**	8.1	64.0
ign	43.9	47.5	8.6	6.6	62.8	45.8	46.2	8.0	8.2	62.3
blk	43.7	**45.4**	10.9	**5.7**	**62.0**	45.8	**44.0**	10.3	**7.0**	**61.3**
b+c1	**44.0**	45.9	10.1	6.4	62.4	**46.0**	44.7	9.3	7.7	61.8
NMT	9.6	80.3	10.1	6.2	96.5	11.1	79.0	9.8	7.4	96.3
eval set										
baseline	31.5	59.2	**9.2**	9.6	78.1	34.8	57.1	**8.1**	10.0	75.2
ign	**33.0**	57.6	9.3	9.7	76.7	**36.5**	55.4	8.2	10.1	73.6
blk	32.9	**55.4**	11.7	8.3	**75.3**	36.2	**53.0**	10.8	**8.6**	**72.4**
b+c1	**33.0**	56.1	11.0	9.2	76.2	36.4	53.7	9.9	9.6	73.2
NMT	8.8	80.8	10.4	**8.0**	99.3	9.6	80.9	9.5	9.0	99.4

[1]Available at https://github.com/espnet/espnet
[2]Available at https://github.com/keisks/robsut-wrod-reocginiton
[3]Available at https://github.com/google/seq2seq

REF: You know I was already bargaining as a five year old child with doctor P to try to get out of doing these exercises unsuccessfully of course
e2e: You know I was already barganing at the five year old child with darker pretty to talk to get out of doing these ext eyes on successfully of course
scRNN: You know I was already bargaining at the five year old child with doctor pretty to talk to get out of doing these ex eyes on successfully of course
NMT: You know I was already married at the old number of my family dropped to me was going to get to some sense with these genomes of course on

Fig. 3. Samples of error corrections on TED-LIUM evaluation set. Two substitution errors (barganing → bargaining and darker → doctor) removed.

TABLE III
WER [%] ON TED-LIUM DEVELOPMENT (DEV) AND EVALUATION (EVAL) SET. CERS OF BASELINE WERE 12.8% (DEV) AND 12.4% (EVAL), RESPECTIVELY.

type	model	B/C	dev set	eval set
baseline			25.8	24.5
ign	bie	B	26.0	24.6
		C	25.7	24.3
blk	w	B	29.4	27.6
		C	26.4	25.2
	bie	B	25.6	24.5
		C	**25.4**	**24.2**
b+c1	bie	B	25.9	24.7
		C	25.5	24.4
b+c2		B	25.8	24.7
		C	25.6	24.4
NMT (acc)			77.0	70.4
NMT (bleu)			77.9	71.4

error was obtained.

Table I also shows the WERs on the CHiME 4 evaluation set. The tendencies were the same. NMT was the worst and scRNN was effective, especially with blank word symbols. ign and b+c1 achieved the best correct rate, and blk achieved the best WER. A relative 3.7–3.9% WER improvement was observed.

C. TED-LIUM (Large-vocabulary ASR)

Table III shows the WERs [%] on the TED-LIUM dataset. The performance of NMT was much worse than that of the baseline. In this case, the number of substitutions was three times larger, and the number of deletion and insertions was twice or three times larger than those of the baseline.

The other tendencies were similar to those observed in the CHiME 4. scRNN improved the WER by a relative 1.2–1.5%. Although TED-LIUM was more difficult for scRNN because the vocabulary was large and the number of OOV words for scRNN was greater than that of the CHiME 4, scRNN was still effective. Fig. 3 shows sample results, where “barganing” was an invalid word and this was corrected by scRNN.

VI. Conclusion

E2E ASR systems are susceptible to noise. In particular, character-based E2E outputs errors like misspellings because it does not use explicit language constraints. To correct these errors, we propose applying scRNN, whose aim is spelling correction, to ASR problems, but direct application is impossible because scRNN focuses only on substitution errors. To deal with insertion and deletion errors, blank word symbols and word concatenation are introduced. Experiments on two different ASR tasks show that scRNN with our extension improved the baseline on both tasks, whereas NMT did not at all. In particular, for a noisy ASR task, the proposed scRNN reduced WER relatively by 4%.

References

[1] K. Kita, T. Kawabata, and T. Hanazawa, “HMM continuous speech recognition using stochastic language models,” in *Proceedings of ICASSP*, vol. 1, 1990, pp. 581–584.

[2] A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 1764–1772.

[3] Y. Miao, M. Gowayyed, and F. Metze, “EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding,” in *Proceedings of ASRU*, 2015, pp. 167–174.

[4] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” in *Proceedings of NIPS*, 2015, pp. 577–585.

[5] A. Senior, H. Sak, F. de Chaumont Quitry, T. Sainath, and K. Rao, “Acoustic modeling with CD-CTC-SMBR LSTM RNNs,” in *Proceedings of ASRU*, 2015, pp. 604–609.

[6] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, “End-to-end attention-based large vocabulary speech recognition,” in *Proceedings of ICASSP*, 2016, pp. 4945–4949.

[7] L. Lu, X. Zhang, and S. Renals, “On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition,” in *Proceedings of ICASSP*, 2016, pp. 5060–5064.

[8] R. Prabhavalkar, T. Sainath, B. Li, K. Rao, and N. Jaitly, “An analysis of “attention” in sequence-to-sequence models,” in *Proceedings of INTERSPEECH*, 2017, pp. 3702–3706.

[9] K. Audhkhasi, B. Ramabhadran, G. Saon, M. Picheny, and D. Nahamoo, “Direct acoustics-to-word models for English conversational speech recognition,” in *Proceedings of INTERSPEECH*, 2017, pp. 959–963.

[10] H. Soltau, H. Liao, and H. Sak, “Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition,” in *Proceedings of INTERSPEECH*, 2017, pp. 3707–3711.

[11] S. Kim, T. Hori, and S. Watanabe, “Joint CTC-attention based end-to-end speech recognition using multi-task learning,” in *Proceedings of ICASSP*, 2017, pp. 4835–4839.

[12] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, “Hybrid CTC/attention architecture for end-to-end speech recognition,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[13] S. Watanabe, T. Hori, and J. R. Hershey, “Language independent end-to-end architecture for joint language and speech recognition,” in *Proceedings of ASRU*, 2017, pp. 265–271.

[14] J. Li, G. Ye, R. Zhao, J. Droppo, and Y. Gong, “Acoustic-to-word model without OOV,” in *Proceedings of ASRU*, 2017, pp. 111–117.

[15] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *Proceedings of EMNLP*, 2015, pp. 1412–1421.

[16] K. Sakaguchi, K. Duh, M. Post, and B. Van Durme, “Robsut wrod reocginiton via semi-character recurrent neural network,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 3281–3287.

[17] Y. Tachioka and S. Watanabe, "Discriminative method for recurrent neural network language models," in *Proceedings of ICASSP*, 2015, pp. 5386–5390.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[19] E. Vincent, S. Watanabe, A. A. Nugraha, J. Barker, and R. Marxer, "An analysis of environment, microphone and data simulation mismatches in robust speech recognition," *Computer Speech and Language*, vol. 46, pp. 535–557, 2016.

[20] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 2014.

[21] D. Kingma and L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of ICLR*, 2015.

[22] D. Britz, A. Goldie, T. Luong, and Q. Le, "Massive exploration of neural machine translation architectures," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1442–1451.

[23] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of Association for Computational Linguistics (ACL)*, 2002, pp. 311–318.