INCREMENTAL SENTENCE COMPRESSION USING LSTM RECURRENT NETWORKS

Sakriani Sakti[1], *Faiz Ilham*[1,2], *Graham Neubig*[1], *Tomoki Toda*[1*],
Ayu Purwarianti[2], *Satoshi Nakamura*[1]

[1]Graduate School of Information Science, Nara Institute of Science and Technology, Japan
[2]School of Electrical Engineering and Informatics, Bandung Institute of Technology, Indonesia
{ssakti,neubig,tomoki,s-nakamura}@is.naist.jp,{13511080@std.,ayu@}stei.itb.ac.id

ABSTRACT

Many of the current sentence compression techniques attempt to produce a shortened form of a sentence by relying on syntactic structure such as dependency tree representations. While the performance of sentence compression has been improving, these approaches require a full parse of the sentence before performing sentence compression, making it difficult to perform compression in real time. In this paper, we examine the possibilities of performing incremental sentence compression using long short-term memory (LSTM) recurrent neural networks (RNN). The decision of whether to remove a word is done at each time step, without waiting for the end of the sentence. Various RNN parameters are investigated, including the number of layers and network connections. Furthermore, we also propose using a pretraining method in which the network is pretrained as an autoencoder. Experimental results reveal that our method obtains compression rates similar to human references and a better accuracy than the state-of-the-art tree transduction models.

Index Terms— Sentence compression, recurrent neural network, long short term memory

1. INTRODUCTION

Today, there is increasing demand for real-time broadcast closed captioning services that can convey spoken utterances using text on a television, video screen, or other visual display with minimum delay. These closed captions are mostly created for deaf and hard-of-hearing individuals to assist in comprehension [1]. Since the text information needs to reach the audience within the time the relevant video information is displayed on the screen, the closed caption captioning process has to be done simultaneously with the visual and audio display. However, it is often the case that the number of spoken words that need to be transcribed exceeds the maximum number of words that people can read in real time on a TV screen. Therefore, the development of an automatic closed captioning system requires not only high-speed speech recognition but also real-time sentence compression.

Sentence compression is a text-to-text rewriting method that produces a shortened form of a sentence while retaining the most important information [2]. Research on sentence compression has been extensively pursued across different modeling paradigms. Specifically, *extractive* sentence compression focuses on word deletion; a target compressed sentence is formed by dropping any subset of words from the input sentence [3]. Many solutions to the compression problem have been cast as a translation task between two sentences in the same language based on a parallel corpus of original and compressed sentences.

To date, the work by Knight and Marcu [3] proposed a noisy-channel formulation using a Synchronous Context-Free Grammar (SCFG) [4, 5], and a variety of improvements exist including those capable of handling syntactically complex expressions [6] and head-driven Markovization of SCFG deletion rules [7]. A number of other approaches also exist in the speech domain; Clarke and Lapata [8] use integer linear programming (ILP) to infer globally optimal compressions in broadcast news, and Liu and Liu [9] model the speech utterance compression task as a sequence labeling problem using conditional random fields (CRF). However, one notable method in extractive sentence compression is the tree transduction formulation by Cohn and Lapata [10, 11, 12]. In this method, sentence compression is seen as rewriting a parse tree into a smaller one. Specifically, the model adopts the synchronous tree substitution grammar (STSG) formalism [6], which can construct non-isomorphic tree structures while using efficient inference algorithms.

While the performance of sentence compression has been improving, these approaches require a full parse of the sentence before performing sentence compression, making it difficult to perform compression as part of a real-time closed-captioning process. The motivation in the current study is to examine the possibilities of performing incremental sentence compression using recurrent neural networks (RNN). The decision of whether to remove a word is done at each time step, without waiting for the end of the sentence. In addition, because the available parallel training data for the sentence compression task is relatively small, we also propose a method to

[*]He is now at Information Technology Center, Nagoya University, Japan (email: tomoki@icts.nagoya-u.ac.jp).

use pretraining to learn representations of the sentence using large unsupervised data before fine-tuning the sentence compression data.

2. RELATED WORKS

An RNN is a type of neural network that has feedback connections, such that in each time-step layers in the net will also receive inputs by using the output values at the previous time step. Consequently RNNs can be considered deep neural networks (DNNs) with the number of layers being proportional to the number of time steps. These additional feedback connections allow RNNs to remember and process information from previous time steps, making them a powerful and expressive model for sequential tasks.

To date, RNNs have provided advantages in various speech and language processing tasks. One of the first application of RNNs in speech processing was to use it as phone modeling and phone probability estimation [13, 14]. RNNs have also been widely applied for text classification [15]. Perhaps the most successful application of RNNs in recent years has been their use in RNN language models (LMs) [16], which have demonstrated outstanding performance in a variety of tasks, including speech recognition [16, 17], machine translation [18], and learning word embeddings [19, 20]. More recently, many research works have focused on utilizing RNNs for spoken language understanding [21, 22].

However, to the authors' knowledge, there has been no attempt to apply RNNs to solving the problem of sentence compression. In this study, we utilize RNNs for incremental sentence compression. Specifically, we apply long short-term memory (LSTM) RNNs to predict whether to remove a word at each time step. Furthermore, we explore various RNN structures and propose modifications to the standard RNN layers as well as the use of pretraining methods.

3. COMPRESSION MODELING WITH LSTM-RNN

3.1. Basic LSTM-RNN

In this sentence compression task, we utilize the classic Elman RNN architecture [23] shown in Fig. 1. Each layer represents a set of neurons, and the layers are connected with weights. The input layer x_t represents the input word at time t, and the hidden layer h_t is activated by the current input x_t and the previous hidden activation h_{t-1}:

$$h_t = \text{sigmoid}(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \quad (1)$$

where W_{xh} and W_{hh} are the weight matrices between the input and hidden layers, and between the hidden and previous hidden layers, respectively. The output determines the probability distribution over whether the word is important or unimportant:

$$o_t = \text{softmax}(W_{hy}h_t + b_y). \quad (2)$$

Parameters b_h and b_y are bias vectors connected to hidden and output units, respectively. Here, h_0 as a base for recursion is assumed to be 0.

Training the RNN is done using the back-propagation through time (BPTT) algorithm [24, 25], which is a modification of the conventional back-propagation algorithm used in feed-forward neural network training, while sharing the connection weights across different time steps. Here, three time-steps are used in the unfolding process.

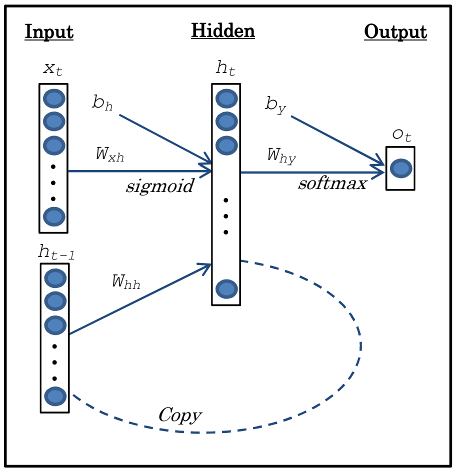

Fig. 1. *Overview of RNNs.*

In order to avoid the problem of vanishing gradients that plague standard RNNs, we utilize LSTM cells [26], which are memory cells modulated with four gates: previous value input, input gate, output gate and forget gate. Because LSTM cells have linear feedback loops between the states of multiple time steps, the error remains constant, allowing for more effectively capturing long-distance dependencies.

3.2. Various Network Structures

We explore various network structures and propose modifications to the basic LSTM-RNN layers as shown in Fig. 2:

- **Basic LSTM (denoted as "LSTM-basic")**
  We used a basic LSTM with one input layer, one output layer, and one hidden layer (including the recurrent layer).

- **LSTM with a continuous representation layer (denoted as "LSTM-basic+cont")**
  To handle the sparseness of the input representation, we extend the above LSTM structure to include a continuous representation layer placed between the input and hidden layers, with the same size as the hidden layer. The motivation is to learn a better word representation in a smooth continuous space. In this layer, the hyperbolic tangent $tanh(W_{xh}x_t + b_h)$ is applied.

- **LSTM with a continuous representation layer and extra connection (denoted as "LSTM-basic+cont+ext")** To support the decision at the output layer, we further extend the LSTM-basic+cont structure with an extra connection from the continuous representation layer to the output layer.

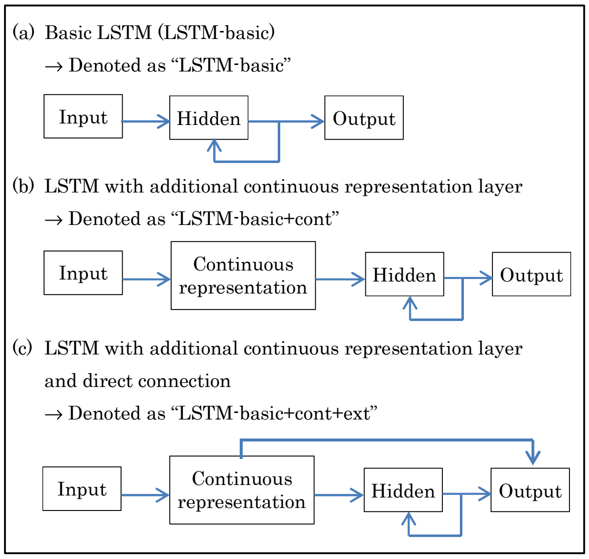
(a) Basic LSTM (LSTM-basic)
→ Denoted as "LSTM-basic"
Input
Hidden
Output
(b) LSTM with additional continuous representation layer
→ Denoted as "LSTM-basic+cont"
Input
Continuous representation
Hidden
Output
(c) LSTM with additional continuous representation layer and direct connection
→ Denoted as "LSTM-basic+cont+ext"
Input
Continuous representation
Hidden
Output

Fig. 2. *Various RNN structures.*

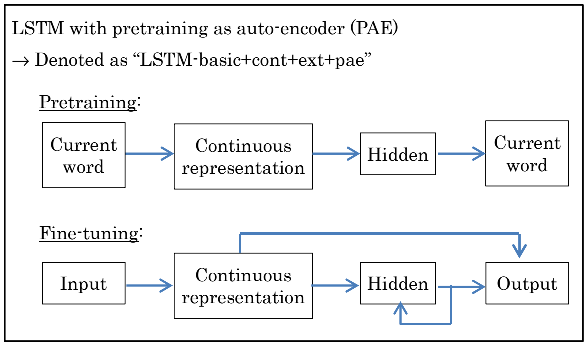
LSTM with pretraining as auto-encoder (PAE)
→ Denoted as "LSTM-basic+cont+ext+pae"
Pretraining:
Current word
Continuous representation
Hidden
Current word
Fine-tuning:
Input
Continuous representation
Hidden
Output

Fig. 3. *LSTM with pretraining method as an autoencoder.*

3.3. Pretraining Method

Because the available parallel training data for the sentence compression task is relatively small, we also propose a method to use pretraining to learn representations of a sentence using large unsupervised data before fine-tuning the sentence compression data.

First, we modified the "LSTM-basic+cont+ext" design so that the output layer had the same size as the input layer, without recurrent and extra connections. Then, we pretrained it as an autoencoder that outputs exactly the same value as the input. After pretraining, we changed the output layer back to "LSTM-basic+cont+ext" and performed fine-tuning. Fig. 3 illustrates the structure, denoted as "LSTM-basic+cont+ext+pae."

We also attempted other methods where we pretrained the model as a language model that predicts the next word. But the performance is below the performance obtained with autoencoder pretraining. This might be because the method tuned the network for the next word but not for the current word. Therefore, we decided to only use pretraining as an autoencoder.

4. EXPERIMENTAL SETUP

4.1. Corpora

The experiments were conducted on CLwritten and CLspoken corpora[1], which are widely used for evaluating sentence compression techniques [8]. Samples of both CLWritten and CLSpoken are shown in Table 1. The former was created by sampling from written sources including the British National Corpus (BNC) and the American News text corpus, while the latter was created by manually transcribed speech of broadcast news stories. Since CLSpoken is created from a speech corpus, it often contains incomplete and ungrammatical utterances and speech artifacts such as disfluencies, false starts and hesitations.

Table 1. *CLWritten and CLSpoken sample data.*

Data Set	Original	Compressed
CLWritten	The powerful balance of these figure compositions is highlighted when they are transposed into tubes and sheets of metal.	The balance is highlighted when they are transposed into tubes and sheets of metal.
	Laurie had a warmth of personality in all his doings with people and politics.	Laurie had a warmth with people and politics.
CLSpoken	All those things suggest to most voters that the White House is hiding something.	Those things suggest that the White House is hiding something.
	Most international airports that service large planes like this are 4,000 meters, double in length, and it may be a contributing factor.	Most international airports that service large planes are 4,000 meters, and it may be a contributing factor.

The corresponding shortened/compressed form of all sentences were created manually. Here, the annotators were asked to produce the smallest possible target compression by deleting extraneous words from the source, without changing the word order and the meaning of the sentences.

[1]CLWritten and CLSpoken: http://homepages.inf.ed.ac.uk/s0460084/data/

The total number of sentences and the corresponding shortened/compressed pairs in CLWritten is 1433, which were taken from 82 articles, while CLSpoken consist of 1370 pairs taken from 50 articles. After some preprocessing, we had 1,400 pairs from CLWritten and 1,280 pairs from CLSpoken. To ensure sufficient training data for the LSTM, we used 1000 pairs of our data for training and the remainder to test the model. This was done for both CLWritten and CLSpoken. To allow pretraining to learn representations of the sentences, we used 50,000 sentences of web-crawled news data from the IWSLT 2011 Evaluation Campaign [27].

4.2. T3 Baseline Model

As a baseline, we used state-of-the-art tree transduction models [12]. These models use a tree-rewriting process that changes the tree from the original parse tree to a compressed parse tree. Specifically, the model adopts weighted STSGs [6].

Here, two things are learned from the training data: the rewrite rules and their weights. The original and compressed sentences are first aligned word by word, and then the rewriting rules are extracted and their frequency is counted. Based on the rules and their frequency, the model is then trained using support vector machines (SVM) [28]. In these experiments, the tree transduction models are implemented using the Tree Transduction Toolkit (T3)[2], which includes training using SVMstruct[3].

4.3. RNN Parameters

Each hidden layer in the RNN designs uses 100 neurons. During training, the optimum result is obtained with threshold $t = 0.5$, where $t >= 0.5$ is regarded as important and $t < 0.5$ is regarded as unimportant.

In the input layer, the standard 1-of-$|V|$ word representation is used, in which the i-th word of the vocabulary V is encoded with a $|V|$-dimensional vector, where the i-th element is set as 1 and all others are 0. We performed some preliminary experiments where we used three different varieties of input context when feeding the words into the neural net, including unigram, bigram and trigram. Here, the n-gram input vector is a concatenation of n word vectors. For example, the bigram input vector would be encoded as a $2V$-dimensional vector, in which the vector is constructed by concatenating unigram word vectors for the current and previous words. However, the compression rate using unigram is already quite high, and the use of bigram and trigram inputs further increases the compression rate. This might due to data sparseness problems. Therefore, we simplified the problems by only using the unigram type of input.

For every experiment, we use the PyBrain library[4] for training and testing the RNN model.

[2]T3: http://staffwww.dcs.shef.ac.uk/people/T.Cohn/t3/

[3]http://svmlight.joachims.org/svm_struct.html

[4]PyBrain: http://pybrain.org/

4.4. Evaluation Method

Following the evaluation criteria defined in [29], we investigated the experiments' results based on two factors:

- **Compression rate: How much the original sentence remains after compression**
  Compression rate is defined as the length of the compressed sentence divided by the original length, where lower values indicate shorter sentences.

- **Accuracy or importance factor: How much of the important information is retained from the original.**
  Accuracy is evaluated using simple string accuracy (SSA) [30], which is based on string edit distance between the compression output generated by the system and the reference string.

5. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we evaluate the accuracy of the proposed compression method. First, however, we examine the level of compression achieved by each method for the various architectures[5]. The key solution is to remove as many words as possible while retaining the important information. Here, the compression rate of a human annotator is 72%, and we treat this as the optimum solution.

5.1. Examination of Compression Rate

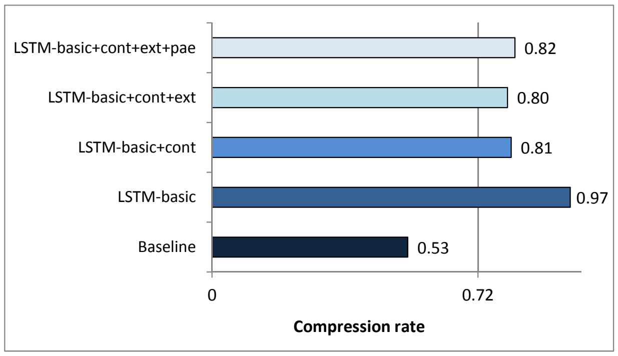

Fig. 4. *Compression rate of various NN structures: "LSTM-basic," "LSTM-basic+cont," "LSTM-basic+cont+ext," and "LSTM-basic+cont+ext+pae," in comparison with tree transduction baseline on CLWritten data (note that human annotator compression rate is 72%).*

[5]For the neural models, adjusting the threshold parameter can make an arbitrarily short or long compression, with a threshold of 0 resulting in a compression rate of 1 or a threshold of 1 resulting in a compression rate of 0, but tuning this parameter in detail is beyond the scope of this work

Here, we examine the compression rate of networks with various structures. Four different structures were evaluated: "LSTM-basic," "LSTM-basic+cont," "LSTM-basic+cont+ext," and "LSTM-basic+cont+ext+pae" as described in Section 3. The neural networks were also trained and tested with CLWritten data. Fig. 4 shows the compression rate of these various RNN structures. For comparison, we also include the baseline result.

The results show that the T3 baseline tends to over-delete content, resulting in a quite low compression rate compared to the compression rate achieved by human annotators. It should be emphasized that, unfortunately, T3 does not provide any flexibility for user control over varying compression rates, so the results presented here reflect whatever output T3 generated for a given sentence. The tendency of T3 to produce a low compression rate has also been discussed by others when using T3 for sentence compression [31, 32, 33].

On the other hand, "LSTM-basic" produced a rather high compression rate, very close to 1. This indicates that the proposed systems provide almost no compression at all. This might be due to data sparseness problems. However, the results reveal that the proposed system incorporating a continuous representation layer seems to help handle the sparseness of the 1-of-$|V|$ representation. The compression rate reaches 81%, which is close to the human annotator's compression rate. The results by "LSTM-basic+cont+ext" and "LSTM-basic+cont+ext+pae" seem to yield a similar compression rate to that of "LSTM-basic+cont." This shows that the main element influencing the compression rate is the addition of the continuous word representation, as opposed to the extra layers or pretraining.

5.2. Evaluation based on Accuracy

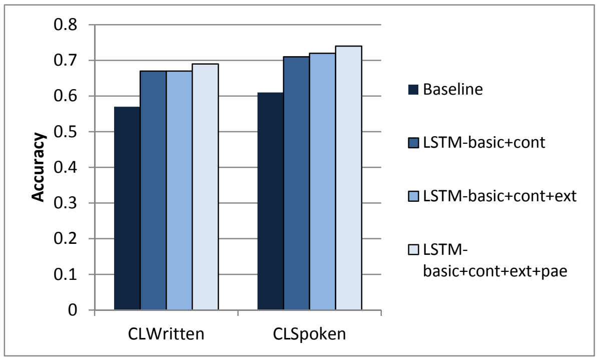

Fig. 5. *Accuracy score of various NN structures: "LSTM-basic," "LSTM-basic+cont," "LSTM-basic+cont+ext," and "LSTM-basic+cont+ext+pae," in comparison with tree transduction baseline on both CLWritten and CLSpoken data.*

In this section, we investigate the performance of various RNN structures based on accuracy. Here, we use only our proposed models that provide a compression rate close to human annotators, which are "LSTM-basic+cont," "LSTM-basic+cont+ext," and "LSTM-basic+cont+ext+pae." The neural networks are trained and tested with CLWritten and CLSpoken. For comparison, we also include the baseline result.

Fig. 5 shows accuracy scores using the various RNN methods. The results reveal that all proposed systems provide much better accuracy than the baseline; the best one is given by "LSTM-basic+cont+ext+pae." This indicates that although the pretraining method does not significantly affect the compression rate, it still contributes to increased performance.

6. CONCLUSIONS

In this study, we investigated the use of LSTM recurrent networks in performing incremental sentence compression. In addition, we also analyzed various input layers, RNN structures, and pretraining methods. The results reveal that RNN-based sentence compression could obtain compression output comparable or superior to tree transduction models. The optimal performance was provided by the "LSTM-basic+cont+ext+pae" structure, since it gives a similar compression rate to that of a human annotator and thus improves the accuracy over the baseline.

At this point, we do not yet take into account the sentence's grammatical information. In the future, we will incorporate grammatical information and analyze the results with evaluation criteria based on grammaticality. In this way, we will investigate other structures in order to further compress sentences while improving accuracy. Furthermore, since our current investigations are based on the assumption that ASR is correct, in the future we will investigate the robustness of our model to ASR errors.

7. ACKNOWLEDGEMENT

Part of this work was supported by JSPS KAKENHI Grant Number 26870371.

8. REFERENCES

[1] N. STRL, "Speech recognition for real-time closed captioning," Broadcast Technology, 2012.

[2] J. Yao, X. Wan, and J. Xiao, "Joint decoding of tree transduction models for sentence compression," in *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, pp. 1828–1833.

[3] K. Knight and D. Marcu, "Summarization beyond sentence extraction: a probabilistic approach to sentence compression," *Journal of Artifficial Intelligence*, vol. 139, no. 1, pp. 91–107, 2002.

[4] P. Lewis and R. Stearns, “Syntax-directed transduction,” *Journal of the Association for Computing Machinery*, vol. 15, pp. 465–488, 1968.

[5] S. B. Davis and P. Mermelstein, “Syntax directed translations and the pushdown assembler,” *Journal of Computer and System Science*, vol. 3, pp. 37–56, 1969.

[6] J. Turner and E. Charniak, “Supervised and unsupervised learning for sentence compression,” in *Proc. of 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, Ann Arbor, MI, USA, 2005, pp. 290–297.

[7] M. Galley and K. McKeown, “Lexicalized Markov grammars for sentence compression,” in *Proc. of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, Rochester, NY, 2007, pp. 180–187.

[8] J. Clarke and M. Lapata, “Global inference for sentence compression: An integer linear programming approach,” *Journal of Artificial Intelligence Research*, vol. 31, pp. 399–429, 2008.

[9] F. Liu and Y. Liu, “Using spoken utterance compression for meeting summarization: a pilot study,” in *Proc. of SLT*, Berkeley, California, USA, 2010.

[10] T. Cohn and M. Lapata, “Large margin synchronous generation and its application to sentence compression,” in *Proc. of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic, 2007, pp. 73–82.

[11] ——, “Sentence compression beyond word deletion,” in *Proc. of International Conference on Computational Linguistics*, Haifa, Israel, 2008, pp. 137–144.

[12] ——, “Sentence compression as tree transduction,” *Journal of Artificial Intelligence Research*, vol. 34, pp. 637–674, 2009.

[13] M. H. T. Robinson and S. Renals, “IPA: Improvedphone modelling with recurrent neural networks,” in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Adelaide, SA, USA, 1994, pp. 37–40.

[14] A. Robinson, “An application of recurrent nets to phone probability estimation,” *Journal of IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 298–305, 2002.

[15] G. Arevian, “Recurrent neural networks for robust real-world text classification,” in *Proc. of International Conference on Web Intelligence (IEEE/WIC/ACM)*, Fremont, CA, USA, 2007, pp. 326–329.

[16] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, “Recurrent neural network based language model,” in *Proc. of INTERSPEECH*, Makuhari, Japan, 2010, pp. 1045–1048.

[17] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, “Extensions of recurrent neural network language model,” in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 5528–5531.

[18] M. Auli, M. Galley, C. Quirk, and G. Zweig, “Joint language and translation modeling with recurrent neural networks,” in *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*, Seattle, Washington, USA, 2013, pp. 1044–1054.

[19] G. Mesnil, X. He, L. Deng, and Y. Bengio, “Investigation of recurrent-neural-network architectures and learning methods for language understanding,” in *Proc. of INTERSPEECH*, Lyon, France, 2013.

[20] K. Yao, G. Zweig, M. Hwang, Y. Shi, and D. Yu, “Recurrent neural networks for language understanding,” in *Proc. of INTERSPEECH*, Lyon, France, 2013.

[21] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, and G. Zweig, “Syntax-directed transduction,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3.

[22] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, “Spoken language understanding using long short - term memory neural networks,” in *Proc. of IEEE SLT*, South Lake Tahoe, Nevada, USA, 2014.

[23] J. Elman, “Finding structure in time,” *Journal of Cognitive Science*, vol. 14, pp. 179–211, 1990.

[24] M. Mozer, *A Focused Backpropagation Algorithm*, 1995, pp. 137–169.

[25] P. J. Werbos, “Generalization of backpropagation with application to a recurrent gas market model,” *Journal of Neural Networks*, vol. 1, no. 4.

[26] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1438, 1997.

[27] M. Federico, L. Bentivogli, M. Paul, and S. Stüker, “Overview of the IWSLT 2011 evaluation campaign,” in *Proc. of International Workshop on Spoken Language Translation*, San Fransisco, USA, 2011, pp. 11–27.

[28] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, “Large margin methods for structured and interdependent output variables,” *Journal of Machine Learning Research*, vol. 6, pp. 1453–1484, 2005.

[29] R. McDonald, “Discriminative sentence compression with soft syntactic evidence,” in *Proc. of Annual Meeting of The European Chapter of The Association of Computational Linguistics (EACL)*, Lyon, France, 2006.

[30] S. Bangalore, O. Rambow, and S. Whittaker, “Evaluation metrics for generation,” in *Proc. of 1st INLG*, Mitzpe Ramon, Israel, 2000, pp. 1–8.

[31] T. Nomoto, “A comparison of model free versus model intensive approaches to sentence compression,” in *Proc. of the 2009 Conference on Empirical Methods in Natural Language Processing*, Singapore, 2009, pp. 391–399.

[32] E. Marsi, E. Krahmer, I. Hendrickx, and W. Daelemans, “On the limits of sentence compression by deletion,” *Empirical Methods in Natural Language Generation*, pp. 45–66, 2010.

[33] D. Feblowitz and D. Kauchak, “Sentence simplification as tree transduction,” in *Proceedings of the 2nd Workshop on Predicting and Improving Text Readability for Target Reader Populations*, Sofia, Bulgaria, 2013, pp. 1–10.