ICIC Express Letters
Part B: Applications
Volume **10**, Number **12**, December **2019**

ICIC International ©2019 ISSN 2185-2766
pp. 1067–1075

INTEGRATING STRUCTURAL DEPENDENCIES IN NEURAL MACHINE TRANSLATION USING GRAPH CONVOLUTIONAL NETWORKS

Long Ly, Quang Nguyen, Long Hong Buu Nguyen* and Dinh Dien

Faculty of Information Technology
University of Science – VNU.HCMC.
No. 227, Nguyen Van Cu Street, Ward 4, District 5, Ho Chi Minh City, Vietnam
{ lklong; ntnquang }@apcs.vn; ddien@fit.hcmus.edu.vn
*Corresponding author: long.hb.nguyen@gmail.com

Received July 2019; accepted October 2019

Abstract. *We exploit an effective approach to incorporating structural dependencies into the attention-based encoder-decoder model for machine translation by using Graph Convolutional Networks (GCNs), a recent type of neural network architecture designed for working on graphs and leveraging their structural information. Our GCNs use the syntactic dependency trees of source sentences to produce word representations which are sensitive to their syntactic neighborhoods. As GCNs' input and output are both word representations, they can be easily integrated as extra layers into the standard encoders (e.g., on top of bidirectional Recurrent Neural Networks or Convolutional Neural Networks). We also evaluate their effectiveness for English-Vietnamese translation in the different types of encoders and observe improvements comparing to their versions with no linguistic information in all considered setups.*
Keywords: Graph convolutional network, Recurrent neural network, Convolutional neural network, Sequence to sequence model, Structural dependency, Word segmentation, Neural machine translation

1. Introduction.

Neural Machine Translation (NMT) is one of the successful areas in natural language processing which uses a neural network to predict the likelihood of a sequence of words. In recent years, NMT systems have outperformed their phrase-based statistical counterparts on many language pairs [1]. The current state of the art NMT systems are based on the sequence to sequence learning, (e.g., encoder-decoder model) [2, 3], and the lack of syntactic dependency or hierarchical structure of the language. One reason for this is due to the fact that there are both very simple and effective methods to integrate the syntactic dependencies into encoders. The common successful techniques have been used such as modeling the interface between syntax and the translation tasks [4], further extending the above model with tree-based coverage [5], or integrating syntactic information in NMT models [6, 12], but they are either restricted or integrated in an indirect approach. Therefore, we aim to provide the encoder with the ability to access rich syntactic information of source sentences and let it decide which aspects of syntax are beneficial for NMT, without placing the rigid constraints on the coordination between syntax and the translation task [7].

Attention-based NMT systems [2, 8, 9] were introduced to alleviate the problem of the encoder-decoder approach: encoding information of a source sentence into a fixed-length vector, which has been shown to give the ineffective performance when dealing with the long sentences. The attention mechanism represents the source sentence words as the latent-feature vectors in the encoder and uses these vectors to generate the translation. Our goal is to integrate the syntactic information about the neighborhoods of source words

DOI: 10.24507/icicelb.10.12.1067

into these feature vectors, and, thus, potentially enhance the quality of the translation output. We use the dependency syntax to achieve this, and the dependency tree represents the syntactic relationship between the words in a sentence, for example, *a* is the determiner of *community*, and *big* is an adjectival modifier of *community* as seen in Figure 1 below:

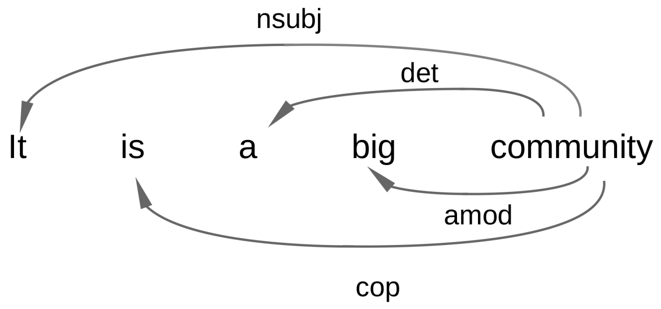

FIGURE 1. A dependency tree for the example sentence, "*It is a big community.*"

The first research on GCNs is presented in [10], which develops a variant of graph convolution based on the spectral graph theory. The basic principle in a graph is that each node is related to the neighbors via some complex linkage information, which is used to capture the interdependence among data [11]. We exploit Graph Convolutional Networks (GCNs) [7, 13, 14] to form the syntax-aware feature representations of words. We use Syntactic GCNs, a version of GCN, operating based on the syntactic dependency trees, as it is proven to be effective in the area of semantic role labeling [7, 15].

Though GCNs can be used separately in encoders (e.g., take word embeddings as input), [7, 16] have shown that they are more efficient when used as the layers on the top of Recurrent Neural Network (RNN) or Convolutional Neural Network (CNN) encoders, enriching their states with the syntactic information. It has also been shown that RNNs (e.g., LSTMs) are capable of capturing certain syntactic information (subject-verb agreement) themselves [17] and therefore, the comparison to RNNs is the most difficult challenge for GCNs. However, even when compared to RNNs, GCNs still prove to be beneficial: we obtain +0.15 BLEU score improvements in using syntactic GCNs on the top of bidirectional LSTMs for English-Vietnamese.

Our contributions can be summarized as follows:

- Exploiting the method proposed by [7] to incorporate structure into NMT using syntactic GCNs.
- Showing that CNN encoders can be improved with GCN layers on top to capture long distant dependencies.
- And proving that machine translation on English-Vietnamese can benefit from integrating syntactic information.

The rest of this paper is summarized as follows:

- Chapter 2 is the background to introduce the related concepts.
- We concentrate on how to integrate the structural dependencies in the encoder in Chapter 3.
- The experiments and report results are conducted in Chapter 4 to verify the effectiveness of the proposed method.
- The conclusions and future works are presented in Chapter 5.

2. Background.

2.1. **Sequence to sequence model.** The idea of sequence to sequence model is to transform the source sentence to the target sentence using 2 neural networks. It aims to map a fixed length input with a fixed length output where the length of those may differ. The model consists of three parts: encoder, context vector and decoder. The encoder summarizes the source sentence into a context vector and the decoder unfolds this vector into the target sequence.

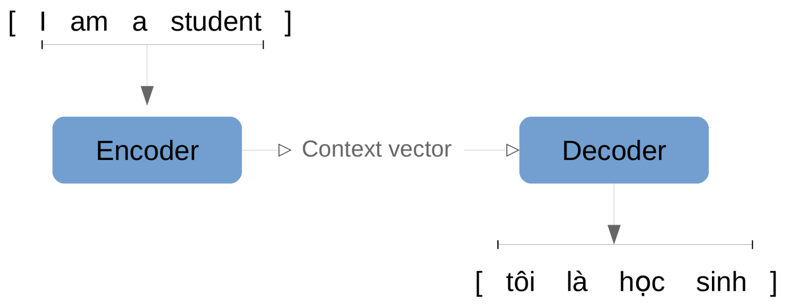

FIGURE 2. Encoder-decoder sequence to sequence model

2.1.1. *Recurrent neural network.* Recurrent neural network based encoder-decoder architectures has been widely adopted in sequence to sequence modelling [2]. The encoder RNN processes the input sequence x = $x_1, x_2, \ldots, x_m$ where m is the number of the elements then returns the context vector z. The decoder RNN takes z and generates the output sequence y = $y_1, y_2, \ldots, y_n$. To create the element y_{i+1}, the decoder computes the new hidden state h_{i+1} from the previous state h_i, the previous target language word y_i and a conditional input c_i derived from the encoder output z. This approach uses the past of a word to predict it; another approach called bidirectional RNN [28] can not only summarize the past of a word but also its future. A BiRNN reads the input sentence in the two directions and then concatenates the states at each time step. There are 2 famous versions of RNN: LSTM [18] and GRU [19].

2.1.2. *Convolutional neural network.* Convolutional neural network is less common than RNN for neural machine translation. Compared to the recurrent layers, the convolutions create representation for the fixed number of words, which allows the encoder to capture the local context within the sentences. The context size can be enlarged by stacking several convolutional layers on top of each other. One advantage of CNN over RNN is that it does not depend on the computations of the previous time step and therefore allows parallel training over every element in a sequence. RNN, on the other hand, needs the hidden state of the previous step to be calculated to continue training. Stacking multi-layer convolutional neural networks create a hierarchical representation over the input sequence where the higher the level and the wider the context. While the close elements interact with each other at the first few layers, the distant elements interact at the higher layers. (see Figure 3). The hierarchical structure provides a shorter path to capture the long-range dependencies compared to the linked list alike the structure of RNN, e.g., we can obtain a feature representation of a window of n words by applying only $O(\frac{n}{k})$ convolutional operation with the kernel width size of k, compared to the linear number $O(n)$ for RNN.

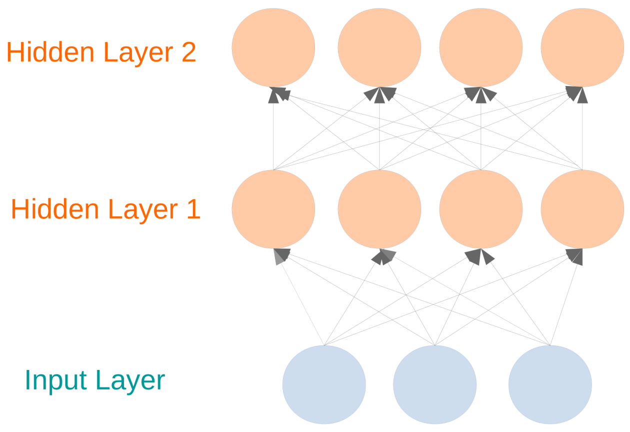

FIGURE 3. Convolutional layers

2.2. **Graph convolutional networks.** Graph Convolutional Network (GCN) proposed by [7] is a multi layer neural network that operates directly on graph and encodes the information of the neighbor of a node. In each GCN layer, the information transfers along the edge of a graph. We can get the information of the larger neighborhoods by stacking multiple layers of GCNs on top of each other.

Considering an undirected graph $G = (V, E)$ where V is the set of n nodes and E is the set of edges. According to the authors every node is assumed to be connected with itself. The new node representations are updated as follows:

$$h_v = \rho\left(\sum_{u \in N(v)} W x_u + b\right)$$

where $W \in R^{d \times d}$ is a weight vector matrix and $b \in R^d$ is a bias vector, ρ is an activation function such as ReLU. $N(v)$ is the set of the node that connects with the node v, which we assume here to include v itself all time. To allow the information to flow over the multiple hops, we need to stack the multiple GCN layers. For example, in the second layer, a node will receive the information from all of its immediate neighbors, but this information already includes the information from their respective neighbors. The recursive computation is computed as follows:

$$h_v^{(j+1)} = \rho\left(\sum_{u \in N(v)} W^{(j)} h_u^{(j)} + b^{(j)}\right)$$

2.3. **Syntactic GCNs.** Syntactic GCN is a version of GCN proposed by [20] that operates on directed and labeled graph. This makes it possible to integrate the linguistic structures that are long forgotten such as the dependency trees or in our case, the structural dependency, where direction and edge labels play an important role.

2.3.1. *Directions.* We use separate weight matrices for outgoing, incoming and looping the edges. We follow the convention in the lexical dependency that heads the point to their dependents. Hence, the outgoing edges are used for the head to the dependent connection, and the incoming edges are used for the dependent to the head connection. For example in Figure 1, the word *community* points toward the word *big*, so the convention *community* will be the head and the word *big* will be its dependent. We have the recursive computation for direction as follows:

$$h_v^{(j+1)} = \rho\left(\sum_{u \in N(v)} W_{dir(v,u)}^{(j)} h_u^{(j)} + b_{dir(v,u)}^{(j)}\right)$$

where $dir(v,u)$ chooses the weight matrix associated with the direction of the edge that connects v and u (W_{out} for v to u, W_{in} for u to v and W_{loop} for v to itself).

2.3.2. *Labels.* The typed dependency captures enormous linguistic information in the sentences, therefore using it as an additional input is straightforward. We use the type of dependency between each word to label the edges connecting them (see Figure 1). As seen in the paradigm of Direction, the separate weight matrices are used for each label, the result in the computation as follows:

$$h_v^{(j+1)} = \rho\left(\sum_{u \in N(v)} W_{lab(v,u)}^{(j)} h_u^{(j)} + b_{lab(v,u)}^{(j)}\right)$$

Finally, to prevent over-parameterization, we combine the two following formulas, where the weight matrices are made the direction-specific while the bias is made the label-specific. This modification reduces the number of the learnable parameter while still taking the two important factors into consideration. The syntactic GCNs also include the gates to control the contribution of the individual edges. This allows the model to deal with the noisy information such as the potentially erroneous directions and labels. For each edge, a scalar gate is computed as follows:

$$\delta_{u,v}^{(j)} = \sigma\left(h_u^{(j)} w_{dir(u,v)}^{(j)} + b_{lab(u,v)}^{(j)}\right)$$

where σ is the logistic sigmoid function, $w \in R^d$ and $b \in R$ are learned parameters for the gates. The final computation becomes:

$$h_v^{(j+1)} = \rho\left(\sum_{u \in N(v)} \delta_{u,v}^{(j)} W_{dir(v,u)}^{(j)} h_u^{(j)} + b_{lab(v,u)}^{(j)}\right)$$

3. **Enhancing Sequence-to-Sequence Models with GCNs.**

In this section, we concentrate on integrating the syntactic information in the encoder. We hypothesize that using an encoder with incorporating syntax will lead to the more informative word representations; hence, when these representations are used as the context vectors by the decoder, it will improve the translation accuracy. Therefore, in all of our sequence-to-sequence models implemented by the fairseq toolkit[1] [21], we keep the decoders part unchanged (the convolutional sequence-to-sequence model and the BiLSTM sequence-to-sequence model).

We define a series of encoders with increasing performance, where we exploit the power of GCNs to incorporate the syntactic-aware representations.

3.1. **BiRNN + GCN.**

In our first model, we use the bidirectional recurrent neural networks. We start with encoding the source sentence using a BiRNN (i.e., BiLSTM) and use the resulting hidden states as input for the GCN. Instead of only depending on the linear order as usual for RNNs, the GCN will allow the encoder to traverse along the dependency edges, connecting words that might be far apart in the sentence. The model may also benefit from the fact that GCNs are capable of exploiting the nature of the relations between words (i.e., dependency relation types) (see 2.3 for details). We assume this to be the most challenging set up for GCNs, as RNNs have been shown to be capable of capturing at least some degree of syntactic information [22], and therefore it should be hard for them to improve performance by incorporating syntactic dependency.

[1]https://github.com/pytorch/fairseq

3.2. **CNN + GCN.** In our second and most powerful model, we use CNNs to learn the word representations. [16] shows that CNNs can give the competitive results comparing to BiRNNs when the multiple layers of CNN are stacked, and they are also faster than RNNs. As seen in their approach, we use GCN to enrich the 4-layer CNN representations.

We expect that the models should be beneficial based on the further propagating information from the tree. We hypothesize that the first GCN layer is the most crucial one, as it captures most of the syntactic context, and the additional layers only modify the word representations modestly.

4. **Experiments.** We conduct the experiments using the fairseq toolkit [21], which implements various sequence-to-sequence models in Pytorch. We use the Adam optimizer [23, 24] with a learning rate of 0.001 (0.0002 for CNN models)[2], as well as the clip threshold of gradients, which is set to 0.1. The batch size is 40, and the maximum number of tokens in a batch is set to 4000. We apply the dropout with a probability of 0.2 between layers, and during the experiments with GCNs[3] we use the same value for the edge dropout. We also train for 60 epochs and evaluate the performance of the model every epoch on the validation set. We choose the model with the highest validation BLEU for testing purpose.

We will describe the details of the MT experiments below.

Data. For our experiments, we use the En-Vi data from the IWSLT 2015 dataset[4]. We train on the full data set. For our validation set and test set we use *tst2013* and *tst2012* respectively.

Preprocessing. Vietnamese words are not always separated as spaces as English words, so sometimes, we need to perform the word segmentation as a pre-processing task. For the experiments in Part 4.2, first, we do a word segmentation processing using the Underthesea-Vietnamese NLP Toolkit[5] for the Vietnamese sides. Then the following procedures are applied for the experiments of both Parts 4.1 and 4.2: both English sides and Vietnamese sides are tokenized, using the fairseq toolkit; and the English sides are parsed into the dependency trees using the Stanford parser [25]. The sentence pairs where either side is longer than 50 words are filtered after tokenization.

Vocabulary. For both the English and Vietnamese, we construct the vocabulary from all words from the training set. The dataset statistics are described in Table 1 and the vocabulary sizes are summarized in Table 2. The vocabulary size for the Vietnamese dataset after the word segmentation processed is larger than its non-processed counterpart.

TABLE 1. The number of sentences in our data sets

	Train	Val.	Test
English-Vietnamese	133317	1268	1553

TABLE 2. Vocabulary sizes

	Source	Target
English-Vietnamese	54176	25624
English-Vietnamese (segmented)	54176	44472

[2]As mentioned by [7], the learning rate of 0.001 for Adam is too aggressive for CNNs.
[3]GCN code is at https://github.com/bastings/neuralmonkey
[4]https://sites.google.com/site/iwsltevaluation2015/mt-track
[5]https://github.com/undertheseanlp/underthesea

Hyperparameters. For BiLSTMs, we use 128 units for the word embeddings, 512 units for the hidden size. For CNNs, we use 256 units for the word embeddings, 256 units for the convolutional layers and the kernels of width 3. The dimension of the GCN layers is equivalent to the dimension of their input. We report results for 1-layer GCNs, as when we stack more than 1 layer, the result does not improve much, but the training time takes much longer.

Evaluation. We report BLEU scores [26] using multi-bleu. We take the average results over five runs of each model, where each differs in the weight initialization. Translation is generated by a beam search with the width of 5.

4.1. **Data with no word segmentation processed.** We provide two baselines: a convolutional sequence-to-sequence model with window size $w = 3$ and a BiLSTM sequence-to-sequence model.

Table 3 shows the test results on English-Vietnamese. The BiRNN, the strongest baseline, reaches a BLEU_4 of 23.68. Interestingly, GCNs is still able to improve the result by +0.37 BLEU_1 and +0.16 BLEU_4 points. Although the CNN baseline is lower than the BiRNN models for both BLEU_1 and BLEU_4 scores, the CNN + GCN model improves over the CNN baseline by +3.38 and +1.64 for BLEU_1 and BLEU_4 respectively, which beats the BiRNN + GCN model.

Table 3. Results for English-Vietnamese

	BLEU_1	BLEU_4
BiRNN	54.75	23.68
+ GCN	55.12	23.84
CNN	52.5	22.53
+ GCN	55.88	24.17

Effect of GCN layers. How many GCN layers are efficient for training the models? Table 4 shows validation BLEU performance for 1-layer and 2-layer GCNs, together with the CNN baseline for English-Vietnamese. Using 1-layer GCN improves BLEU_1 and BLEU_4 by +3.38 and +1.64, respectively. Surprisingly, adding an additional GCN layer decreases performance by −0.35 and −0.153 for BLEU_1 and BLEU_4.

Table 4. Validation BLEU for 1-layer and 2-layer GCNs

	BLEU_1	BLEU_4
CNN	52.5	22.53
+ GCN (1L)	55.88	24.17
+ GCN (2L)	55.53	24.017

4.2. **Data with word segmentation processed.** We provide the same CNN baseline and CNN + 1-layer GCN baseline as those in 4.1 (non segmented).

Table 5 shows the test result on English-Vietnamese, with and without the word segmentation. Surprisingly, the CNN model shows a drop in performance with −8.27 BLEU_1 and −3.4 BLEU_4, comparing to the CNN baseline. Using 1-layer GCN still improves BLEU scores by +1.43 and +0.62, respectively, reaching a BLEU_4 of 19.75. However, it is still beaten by its counterpart – the CNN + GCN with non segmented data by −10.22 BLEU_1 and −4.42 BLEU_4.

TABLE 5. Test results for English-Vietnamese

	Segmented		Non segmented	
	$BLEU_1$	$BLEU_4$	$BLEU_1$	$BLEU_4$
CNN	44.23	19.13	52.5	22.53
+ GCN	45.66	19.75	55.88	24.17

5. Conclusions and Future Work.

The results depict that incorporating syntactic information using GCNs consistently leads to improvements in translation performance measured by $BLEU_1$ and $BLEU_4$. This gain in BLEU scores also shows that the improvements correlate with word order, a phenomenon that depends undeniably on syntax.

As for the results in Table 5, one explanation might be that because our data set is small (about 133000 sentences for the train set), and the vocabulary size for the Vietnamese is larger than its counterpart in 4.1. Moreover, as the vocabulary size increases, the number of words that appears only 1 time or too few increases, which makes it harder for the model to learn. Neural machine translation has its limitation in handling a larger vocabulary, as training complexity as well as decoding complexity increases proportionally to the number of target words [27].

The reason why stacking 2 layers of GCN decreases BLEU scores a bit may be due to the similar reason: our data set is small. As we stack more layers, the number of trainable parameters increases, and hence the model must be trained with more data in order to gain the better results.

This study, therefore, has shown that integrating the structural dependencies is beneficial for NMT models and improves BLEU performances for the English-Vietnamese language pairs.

In the future, we will train the model with more data in order to justify whether it performs better on data with Vietnamese with word segmented or not, and based on this we can find more effective ways to improve it.

Acknowledgments. This research is supported by research funding from Advanced Program in Computer Science, University of Science, Vietnam National University – Ho Chi Minh City.

REFERENCES

[1] R. Sennrich, B. Haddow and A. Birch, Edinburgh neural machine translation systems for WMT 16, *CoRR*, http://arxiv.org/abs/1606.02891, 2016.

[2] D. Bahdanau, K. Cho and Y. Bengio, Neural machine translation by jointly learning to align and translate, *https://arxiv.org/abs/1409.0473*, 2014.

[3] I. Sutskever, O. Vinyals and Q. V. Le, Sequence to sequence learning with neural networks, *Advances in Neural Information Processing Systems 27*, pp.3104-3112, 2014.

[4] A. Eriguchi, K. Hashimoto and Y. Tsuruoka, Tree-to-sequence attentional neural machine translation, *CoRR*, http://arxiv.org/abs/1603.06075, 2016.

[5] H. Chen, S. Huang, D. Chiang and J. Chen, Improved neural machine translation with a syntax-aware encoder and decoder, *CoRR*, http://arxiv.org/abs/1707.05436, 2017.

[6] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals and L. Kaiser, Multi-task sequence to sequence learning, *https://arxiv.org/abs/1511.06114*, 2015.

[7] T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, *CoRR*, http://arxiv.org/abs/1609.02907, 2016.

[8] M.-T. Luong, H. Pham and C. D. Manning, Effective approaches to attention-based neural machine translation, *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, pp.1412-1421, 2015.

[9] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiskỳ and P. Blunsom, Reasoning about entailment with neural attention, *https://arxiv.org/abs/1509.06664*, 2015.

[10] J. Bruna, W. Zaremba, A. Szlam and Y. LeCun, Spectral networks and locally connected networks on graphs, *https://arxiv.org/abs/1312.6203*, 2013.

[11] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu, A comprehensive survey on graph neural networks, *CoRR*, http://arxiv.org/abs/1901.00596, 2019.

[12] S. Wu, M. Zhou and D. Zhang, Improved neural machine translation with source syntax, *Proc. of the 26th International Joint Conference on Artificial Intelligence*, pp.4179-4185, 2017.

[13] S. Kearnes, K. McCloskey, M. Berndl, V. Pande and P. Riley, Molecular graph convolutions: Moving beyond fingerprints, *Journal of Computer-Aided Molecular Design*, vol.30, no.8, pp.595-608, 2016.

[14] M. Defferrard, X. Bresson and P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, *Advances in Neural Information Processing Systems*, pp.3844-3852, 2016.

[15] D. Marcheggiani and I. Titov, Encoding sentences with graph convolutional networks for semantic role labeling, *CoRR*, http://arxiv.org/abs/1703.04826, 2017.

[16] J. Gehring, M. Auli, D. Grangier and Y. N. Dauphin, A convolutional encoder model for neural machine translation, *https://arxiv.org/abs/1611.02344v3*, 2016.

[17] X. Shi, I. Padhi and K. Knight, Does string-based neural MT learn source syntax?, *Proc. of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, TX, pp.1526-1534, 2016.

[18] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Computation*, vol.9, no.8, pp.1735-1780, 1997.

[19] K. Cho, B. van Merrienboer, C. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, *CoRR*, http://arxiv.org/abs/1406.1078, 2014.

[20] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani and K. Sima'an, Graph convolutional encoders for syntax-aware neural machine translation, *CoRR*, http://arxiv.org/abs/1704.04675, 2017.

[21] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier and M. Auli, fairseq: A fast, extensible toolkit for sequence modeling, *Proc. of NAACL-HLT 2019: Demonstrations*, 2019.

[22] T. Linzen, E. Dupoux and Y. Goldberg, Assessing the ability of LSTMs to learn syntax-sensitive dependencies, *Transactions of the Association for Computational Linguistics*, vol.4, pp.521-535, 2016.

[23] I. Loshchilov and F. Hutter, Decoupled weight decay regularization, *CoRR*, http://arxiv.org/abs/1711.05101, 2018.

[24] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, *https://arxiv.org/abs/1412.6980*, 2014.

[25] D. Klein and C. D. Manning, Accurate unlexicalized parsing, *Proc. of the 41st Annual Meeting on Association for Computational Linguistics – Volume 1*, Sapporo, Japan, no.8, pp.423-430, 2003.

[26] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, BLEU: A method for automatic evaluation of machine translation, *Proc. of the 40th Annual Meeting on Association for Computational Linguistics*, Philadelphia, PA, pp.311-318, 2002.

[27] S. Jean, K. Cho, R. Memisevic and Y. Bengio, On using very large target vocabulary for neural machine translation, *CoRR*, http://arxiv.org/abs/1412.2007, 2014.

[28] M. Schuster and K. K. Paliwal, Bidirectional recurrent neural networks, *IEEE Trans. Signal Processing*, vol.45, no.11, pp.2673-2681, 1997.