ICIC Express Letters
Part B: Applications
Volume **12**, Number **2**, February **2021**

ICIC International ©2021 ISSN 2185-2766
pp. **133–141**

INTEGRATING AMR SEMANTIC GRAPHS TO CONVOLUTIONAL NEURAL MACHINE TRANSLATION

Long Nguyen[1,2,*], Viet Pham[1,2], Hung Minh[3], Dien Dinh[1,2] and Thanh Manh[3]

[1]Faculty of Information Technology
University of Science, Ho Chi Minh City
227 Nguyen Van Cu Street, Ward 4, District 5, Ho Chi Minh City, Vietnam

[2]Vietnam National University, Ho Chi Minh City
Linh Trung Ward, Thu Duc District, Ho Chi Minh City, Vietnam
[*]Corresponding author: long.hb.nguyen@gmail.com
viethungpham0304@gmail.com; ddien@fit.hcmus.edu.vn

[3]Computer Science Department
University of Sciences, Hue University
77 Nguyen Hue Street, Hue City, Vietnam
dmhung@hueuni.edu.vn; lmthanh1953@yahoo.com

Received July 2020; accepted October 2020

Abstract. *Semantic representations are essential to enforce meaning preservation and handle data sparsity of machine translation models. However, little work has been done on leveraging semantics for neural machine translation. There is an approach which tried to adapt AMR graphs to sequence to sequence learning using recurrent neural networks. This paper presents a novel method to integrate abstract meaning representation graphs as the semantic representations to a convolutional neural machine translation system via combining graph representations with the word representations as well as feeding the graph representations into the multi-head attention layer. Compared to recurrent models, computations over all elements can be fully parallelized during training to better exploit the GPU hardware and optimization is easier since the number of non-linearities is fixed and independent of the input length. Experiment results show that our proposed method significantly outperforms strong baselines on English-Vietnamese datasets in increasing the quality of translations.*

Keywords: Neural machine translation, Abstract meaning representation, Light convolutional neural networks

1. **Introduction.** Neural Machine Translation (NMT) models [1, 2, 3, 4] have been proven to be powerful and drawn much attention in recent years. In practical applications, NMT systems are often fed with a sentence-level input which requires solely word representations. Many researchers have proven that semantic information is essential to generate coherent and consistent translations for machine translations [5, 6, 7, 8]. Despite the great success of the above models, they are mostly designed for statistical machine translations. The task of exploring semantics for NMT has so far received relatively little attention.

Recent work has applied semantic representations to neural sequence modeling such as Marcheggiani et al. [9] who exploited Semantic Role Labeling (SRL) for NMT, showing that the predicate-argument information from SRL is able to improve the performance of an attention-based sequence-to-sequence model or Song et al. [10] who indicated that the structural semantic information from Abstract Meaning Representation (AMR) graphs [11] can be complementary to the source textual input by introducing a higher level of information abstraction. In this approach, a Graph Recurrent Network (GRN) was

DOI: 10.24507/icicelb.12.02.133

leveraged to encode AMR graphs without breaking the original graph structure, and a sequential Long-Short Term Memory (LSTM) [12] was used to encode the source input. The decoder was a doubly attentive LSTM, taking the encoding results of both the graph encoder and the sequential encoder as attention memories. Song et al. also proved that the integration of AMR is much better than the one of SRL alone because the AMR graphs not only contain the SRL but also have relations between nodes (i.e., words). However, the method is purely based on the LSTMs which maintain a hidden state of the entire past that prevents parallel computation within a sequence.

This paper presents how we integrate AMR graphs as additional semantic information into the NMT (Conv2seq), using light convolutional neural network [13] which is a lightweight version of original convolutional neural network [14]. AMR graphs are rooted, labeled, directed, acyclic graphs, comprising whole sentences. They are intended to abstract away from syntactic representations, in the sense that sentences which are similar in meaning should be assigned the same AMR, even if they are not identically worded. Figure 1 shows an AMR graph, in which the nodes (e.g., *start*-01, *university*) represent the concepts, and edges (e.g., :*ARG*0, :*ARG*1) represent the relations between concepts they connect. Comparing with semantic roles, AMRs capture more relations (e.g., between *exam* and *university*). Moreover, AMRs directly capture entity relations and remove inflections (i.e., using lemma) and function words. Consequently, they can adapt to the textual input to produce a better contextual representation. Furthermore, structural information from AMR graphs can help reduce data sparsity in low-resource settings. First, AMR graph representations are combined with word representations to establish a better contextual representation of a sentence. Second, the multi-head attention can attend to all the positions of the contextual features with outputs of the AMR graph representations.

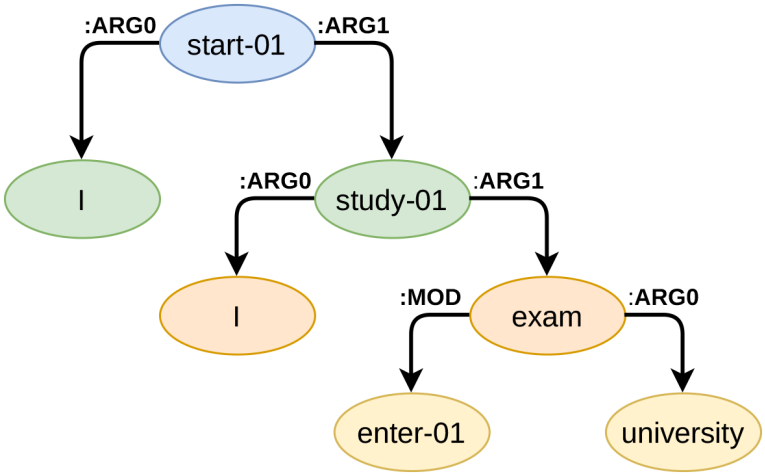


FIGURE 1. The AMR graph for the sentence: "**I started studying for the university entrance exam.**"

This method shows several advantages. First, it may be of great benefits to NMT models, potentially reducing data sparseness and semantic ambiguity problems. Second, the structural semantic information from AMRs can be complementary to the textual input by providing a higher level of information abstraction so the input word representation is better encoded. Finally, multi-head attention can also take advantages of the semantic information to improve the dependencies between words in a sentence.

Recently, several graph-to-sequence models or integrating semantic structures on NMT have been proposed. Semantic Role Labeling (SRL) was explored for NMT in [9]. A Graph2Seq model which is an extension of GraphSAGE [15] was proposed in [16], learning bi-directional node embeddings for directed and undirected graphs with node attributes

by employing various aggregator architectures and learning a graph-level embedding by exploiting two different graph embedding techniques.

Before us, only the method in [10] investigated the effectiveness of AMR on NMT based on their recent work on parsing AMR to text and on a Graph Recurrent Network (GRN) [17] for modelling AMRs. In this work, they adopted a doubly-attentive LSTM decoder, taking the encoding results of both the graph encoder and the sequential encoder as attention memories. In this paper, we also investigate the usefulness of AMR on NMT but leverage graph embedding algorithm of [16] to learn AMR representations.

There are several distinctions between the above works and ours. First, we extend the node embedding algorithm [16] to utilize edge information directly instead of adding a node representing an edge into a graph and assigning attributes of the edge as text attributes. Note that, adding nodes into graphs may not only change the inherent topology of graphs but also add noise information to graphs, leading to a significant drop in terms of performance. Second, we propose an architecture which adopted an inductive graph encoder rather than graph recurrent network in [10]. Finally, we apply AMRs on NMT systems with low-resource settings for English-Vietnamese translation via lightweight convolutional neural networks.

2. Lightweight Convolution to Sequence.

2.1. Lightweight convolution.

Depthwise Convolution performs a convolution independently over every channel in order to reduce the number of parameters from d^2k to dk where k is the kernel size. The output $O \in \mathbb{R}^{n \times d}$ of a depthwise convolution operation for the i-th element and each output dimension c, with $c < d$, is computed as:

$$O_{i,c} = DepthwiseConv(X, W_{c,:}, i, c) = \sum_{j=1}^{k} W_{c,j} \cdot X_{\left(i+j-\lceil \frac{k+1}{2} \rceil\right),c}$$

with $W \in \mathbb{R}^{d \times k}$ as model parameter.

Lightweight Convolution or **LightConv** [13] is a depthwise convolution which shares certain output channels and whose weights are normalized across the temporal dimension using softmax.

$$LC\left(X, W_{\lceil \frac{cH}{d} \rceil,:}, i, c\right) = DepthC\left(X, softmax\left(W_{\lceil \frac{cH}{d} \rceil,:}\right), i, c\right)$$

where LC and $DepthC$ stand for LightConv and DepthwiseConv operations, respectively. The LightConv is also known also a simplified version of the original CNN [14], which is optimized to take advantages of parallel processing to speed up the training process [18].

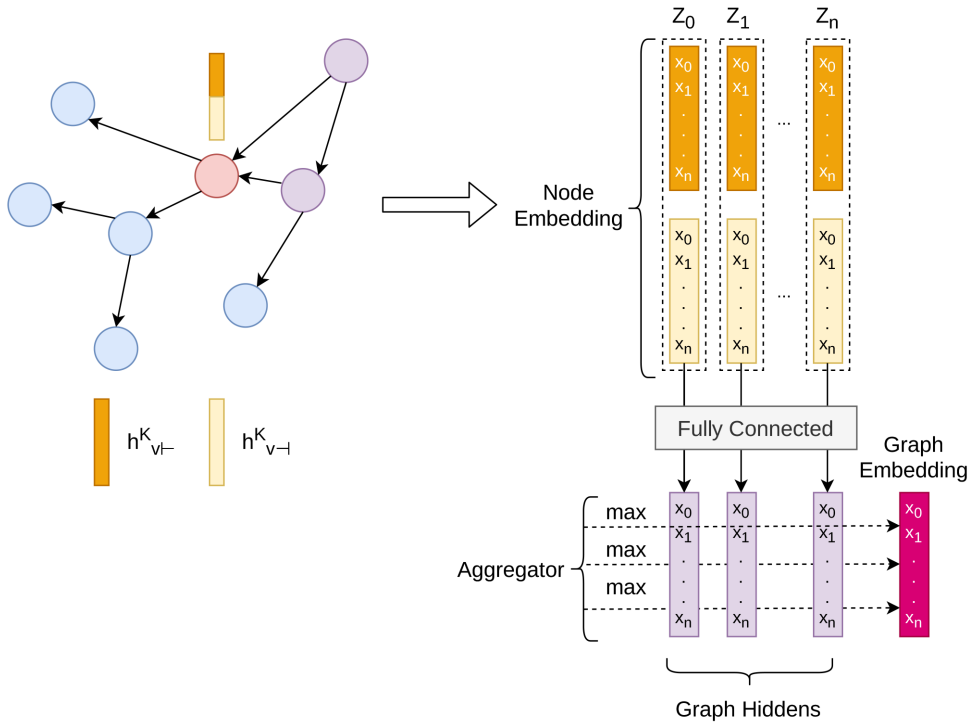
2.2. LightConv2Seq.

The LightConv2Seq follows the encoder-decoder architecture [2] as shown in Figure 2(b) except the graphical part on the right side. The encoder and decoder networks have N blocks each. The encoder blocks comprise two sub-blocks: the first is a LightConv module, and the second is a feed forward network followed by a $ReLU$ activation function. The sub-blocks are surrounded by a residual connections [19] and layer normalization [20].

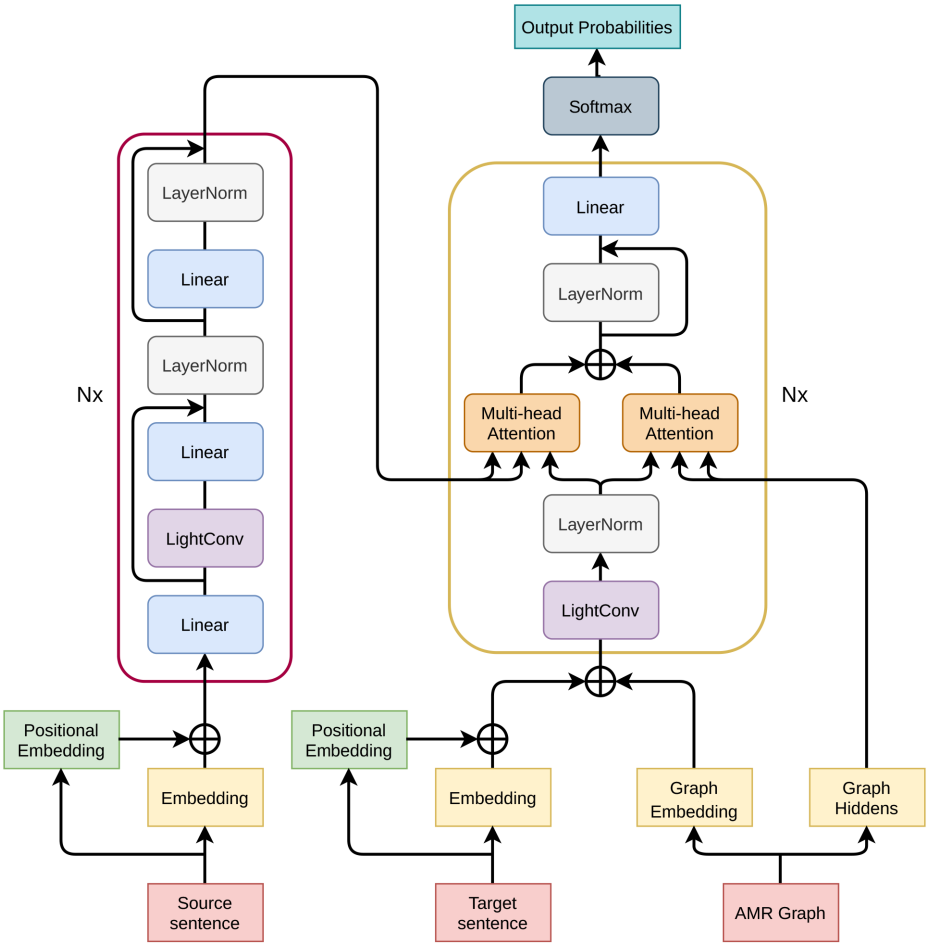
Decoder blocks share identical structure but they have an extra multi-head attention (orange block in Figure 2(b)) to learn alignment between source and target language. This attention projects the values and keys over the encoder output for each source word. Then the result is passed through a feed forward network and a softmax layer to produce output probabilities.

3. Proposed Method.

We propose a method to encode AMR graphs via graph embedding method and then adapt the graph embeddings to multi-head attention layer.



(a) Graph encoder architecture



(b) Model architecture

FIGURE 2. (color online) Graph encoder and model architecture

3.1. **Learning graph embedding.** Figure 2(a) shows the overall architecture of graph encoder which follows Xu et al.'s model [16]. However, we have some improvements to the architecture to adopt edge information and enhance performance of the model.

Firstly, we introduce our fined node embedding algorithm. Given an AMR graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we take the embedding process for node $v \in \mathcal{V}$ as follows:

1) We first transform the text attribute of node v and edge e into feature vectors: $\mathbf{a}_v$ and $\mathbf{a}_e$, respectively, by looking up the embedding matrix E. Node v is represented by a concatenation of $\mathbf{a}_v$ and $\mathbf{a}_e$.

$$\mathbf{a}^v = \texttt{CONCAT}(\mathbf{a}_v, \mathbf{a}_e)$$

2) Next, we categorize the neighbors of v into two subsets: forward neighbors, $\mathcal{N}_{\vdash}(v)$ and backward neighbors, $\mathcal{N}_{\dashv}(v)$. Particularly, $\mathcal{N}_{\vdash}(v)$ returns the nodes that v directs to and vice versa.
3) We aggregate the forward information of v's forward neighbors $\left\{\mathbf{h}_{u\vdash}^{k-1}, \forall u \in \mathcal{N}_{\vdash}(v)\right\}$ into a single vector, $\mathbf{h}_{\mathcal{N}_{\vdash}(v)}^{k}$, where $k \in \{1, \ldots, K\}$ is the iteration index. We do this by using one of three $\texttt{AGG}^{\vdash}$ mentioned below.
4) Then we concatenate v's current forward representation, $\mathbf{h}_{v\vdash}^{k-1}$, with the new neighborhood vector, $\mathbf{h}_{\mathcal{N}_{\vdash}(v)}^{k}$. The result is passed to a feed forward layer, followed by a non-linearity activation function σ, which updates the forward representation of v, to be used in the next iteration.
5) Update the backward representation of v, $\mathbf{h}_{v\dashv}^{k}$, using similar procedure in steps 3) and 4) but this time we use backward representations rather than the forward representations and use $\texttt{AGG}^{\dashv}$ to aggregate neighbor information.
6) Repeat steps 3)∼5) K times, the concatenation of the final forward and backward representation is used as the final bi-directional representation of v.

$$z_v = \texttt{CONCAT}\left(\mathbf{h}_{v\vdash}^{K}, \mathbf{h}_{v\dashv}^{K}\right), \quad \forall v \in \mathcal{V}$$

In steps 3) and 5), we aggregate v's representation by using one of these aggregator architectures:

Mean aggregator: This aggregator function takes the element-wise mean of the vectors in $\left\{\mathbf{h}_{u\vdash}^{k-1}, \forall u \in \mathcal{N}_{\vdash}(v)\right\}$ and $\left\{\mathbf{h}_{u\dashv}^{k-1}, \forall u \in \mathcal{N}_{\dashv}(v)\right\}$.

GCN aggregator: Similar to mean aggregator, but followed by a feed forward layer and a non-linearity activation function.

$$\texttt{AGG}_k^{\vdash} = \sigma\left(\mathbf{W}\texttt{MEAN}\left(\mathbf{h}_{u\vdash}^{k}\right) + \mathbf{b}\right), u \in \mathcal{N}_{\vdash}(v)$$
$$\texttt{AGG}_k^{\dashv} = \sigma\left(\mathbf{W}\texttt{MEAN}\left(\mathbf{h}_{u\dashv}^{k}\right) + \mathbf{b}\right), u \in \mathcal{N}_{\dashv}(v)$$

where MEAN denotes the element-wise average operator, and σ is a nonlinear activation function.

Pooling aggregator: In this aggregator, each neighbor's vector is fed through a fully-connected neural network, and an element-wise max-pooling operation is applied:

$$\texttt{AGG}_k^{\vdash} = \max\left(\left\{\sigma\left(\mathbf{W}_p\mathbf{h}_{u\vdash}^{k} + \mathbf{b}\right), u \in \mathcal{N}_{\vdash}(v)\right\}\right)$$
$$\texttt{AGG}_k^{\dashv} = \max\left(\left\{\sigma\left(\mathbf{W}_p\mathbf{h}_{u\dashv}^{k} + \mathbf{b}\right), u \in \mathcal{N}_{\dashv}(v)\right\}\right)$$

where max denotes the element-wise max operator, and σ is a nonlinear activation function. By applying max-pooling, the model can capture different information across the neighborhood set.

The graph embedding vector, Z, contains information in the entire graph, generated from node embeddings by applying one of three kinds of aggregators above across the node representations:

$$\texttt{Z} = \texttt{AGG}(z_v, \forall v \in \mathcal{V})$$

where AGG denotes the aggregator.

3.2. **Enhancing decoder.** Multi-head attention [4] performs linearly projections to obtain keys (K), queries (Q) and values (V) H times (H is the number of heads), learned the projections to d_k, d_q, d_v dimensions, respectively, then we do the attention function in parallel, resulting in the output values with d_v dimensions. These output values then are concatenated and once again projected, yielding the final values.

$$Attention = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$h_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

$$MultiHead(Q, K, V) = \texttt{CONCAT}(h_1, h_2, \ldots, h_h)W^O$$

where W_i^Q, W_i^K, W_i^V and W^O are model parameters.

Inspired by [10], we also use a doubly attentive decoder to incorporate external AMR knowledge (as shown in Figure 2(b) on the right side). Instead of using a single attention mechanism, we have found it beneficial to adopt multi-head attention mechanism which applies three projections to the input $X \in \mathbb{R}^{n\times d}$ to obtain key (K), query (Q) and value (V) representations, where n is the number of time steps, d is the input/output dimension. In case of number of heads $H > 1$, multi-head attention performs the attention mechanism in parallel, yielding a d_v-dimension output values. Then, these values are concatenated and pass through a linear layer, resulting in the final output values. Multi-head attention allows the model to jointly attend information from different representation sub-places at different positions which is unable to achieve when adopting a single attention head.

4. Experiments.

4.1. **Datasets.** We use the IWSLT 2015 English-Vietnamese dataset [21], which contains around 130 thousand sentence pairs for training and use tst2012 for tuning model parameters and early stopping. We evaluate on the official test set tst2013 and tst2015.

Table 1. Statistics of the English-Vietnamese datasets

Dataset	# tokens		# types		# sents
	en	vi	en	vi	
train	2,435,771	2,867,788	44,573	21,611	117,055
dev(tst2012)	27,988	34,298	3,518	2,170	1,553
test(tst2013)	26,729	33,683	3,676	2,332	1,268
test(tst2015)	20,850	26,235	3,127	2,059	1,080

For preprocessing phase, we use Byte-Pair Encoding (BPE)[1] [22] with 8,000 merge operations to deal with rare and compound words and apply to both English and Vietnamese sides.

For AMR parsing, we use NeuralAmr toolkit[2] [23] which implements the sequence-to-sequence models to the tasks of AMR parsing and AMR generation. Their model achieves competitive results of 62.1 SMATCH [24], the current best score reported without significant use of external semantic resources.

We measure the end translation quality with case-insensitive BLEU [25]. We also apply the bootstrap re-sampling method [26] to measuring the statistical significance ($p < 0.05$) of BLEU score differences between translation outputs of proposed models compared to the baseline.

[1]https://github.com/rsennrich/subword-nmt

[2]https://github.com/sinantie/NeuralAmr

4.2. **Experimental settings.** We train several models to study the effectiveness of AMR knowledge on NMT systems. First of all, we start from the baseline of using LightConv for translating from English to Vietnamese. We use 4 blocks with kernel size of 3, 7, 15, 31, respectively for each block for both encoder and decoder, H is set to 8. Embedding size is 512. In training phase, Adam optimizer [27] is adopted with a fixed learning rate (0.0002), max tokens per epoch is 3,500 and the number of epochs is 10. Finally, once the model is trained, using beam search with beam size of 5 to look for a translation which approximately maximizes the conditional probabilities.

For semantic-based models (i.e., using AMR graphs), we configure as the same with baselines and stack 2 graph encoder layers. LightConv2Seq-AMR-F takes information from outgoing (forward) neighbors, LightConv2Seq-AMR-B receives incoming (backward) neighbors, LightConv2Seq-AMR aggregates information from both directions (forward and backward neighbors). Both node and edge embedding dim are set to 128 and use max-pooling aggregator to aggregate neighbor information. Again, the number of heads in graph attention is set to 8; we will also give further investigation on the impact of this hyper-parameter.

4.3. **Results and analysis.** We explain our experiments and our analyses on the English-Vietnamese dataset.

Result. We show our experiments in Table 2 and output examples in Figure 3. For both datasets (tst2013 and tst2015), our approach substantially outperforms the baselines. Compared to Song et al.'s method who also used anonymized forms of AMRs to alleviate the data sparsity problem, LightConv2Seq-AMR achieves better performance in both tst2013 and tst2015 which is 28.46 (+2.34) and 25.67 (+2.09), respectively. Similarly, when aggregating information from only one direction (LightConv2Seq-F, LightConv2Seq-B), models also improve BLEU score (+1.0) over Song et al.'s model. This indicates that

Table 2. Experimental results

Models	BLEU	
	tst2013	tst2015
Song et al.'s method	26.12	23.58
LightConv2Seq	27.47	25.09
LightConv2Seq-AMR-F	27.71	25.05
LightConv2Seq-AMR-B	27.84	25.27
LightConv2Seq-AMR	**28.46**	**25.67**

AMR	die-01 :arg1 (person-name-0 :arg1-of (include-91 :arg2 (person :arg0-of manage-01 :mod (great :degree most))) :quant (multiple :op1 num-1)) :time history :time (before :op1 now :quant (few :op1 (temporal-quantity temporal-quantity-num-0 :unit year)))
Source	Peter Drucker , one of the greatest management consultants in history , died age 96 , a few years ago .
Reference	Peter Drucker , một trong những cố vấn về quản lí vĩ đại nhất trong lịch sử , qua đời khi 96 tuổi , một vài năm trước .
Song et al.'s method	Gần đây , một trong những nhà quản lý vĩ đại nhất trong lịch sử , chúng tôi đã chết , một người đã chết .
LightConv2Seq	Peter Drucker , một trong những tư vấn quản lý lớn nhất trong lịch sử , chết 96 , vài năm trước .
LightConv2Seq-AMR	Peter Drucker , một trong những nhà tư vấn quản lý vĩ đại nhất trong lịch sử , qua đời 96 , một vài năm trước .

Figure 3. Output examples

our system can leverage AMR knowledge to achieve better translation quality. Furthermore, our models spend much less time in training time (40 minutes) compared to the one of Song et al.'s model (roughly 6 hours).

Analysis. We have trained several models to investigate the impact of depth, N, and multi-head attention by adjusting the number of heads, H, in graph attention on model's performance. Figure 4 shows BLEU scores when stacking more encoder and decoder blocks and changing H. In this figure, left diagram shows test results when we stacked 2 graph encoder layers and 3 graph encoder layers on the right side. Overall, on both sides, performance is improved as H increased. In particular, left diagram shows that the number of blocks as $N = 4, 6$ can benefit the model as BLEU score raises significantly from 27.8 to 28.47 and 27.54 to 28.20 over investigated heads, respectively. The other settings witnessed a downward trend from the 2nd to 8th head. We also develop a "deep" model by stacking one more graph encoder layer to three layers. At the glimpse at the right diagram, we can see that the more encoder decoder blocks, the higher BLEU score that models can achieve, it is clearer when we set $H = 4$. This suggests that our models need more decoder layers in order to leverage AMR knowledge because when stacking more graph encoder layers, models are able to learn much more abstract representation.

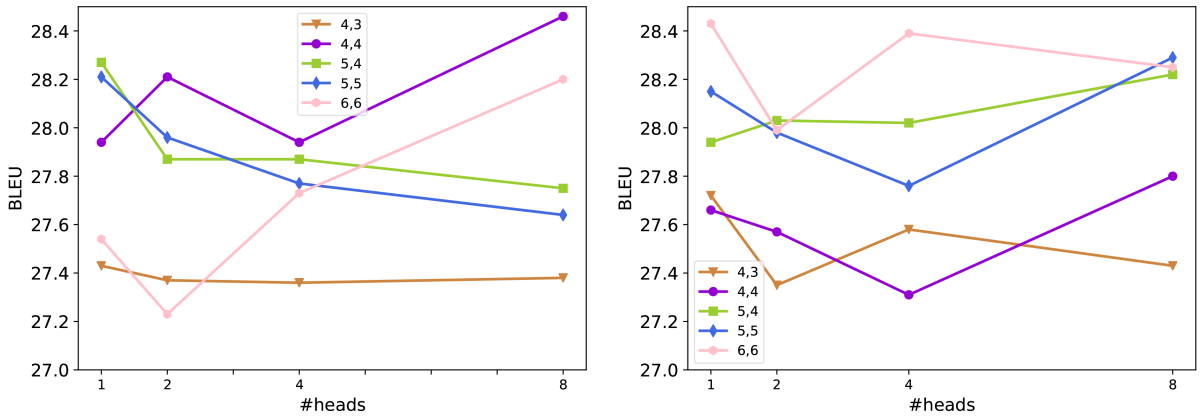


FIGURE 4. Test result with different encoder decoder blocks (4,3 denotes the model has 4 blocks in encoder and 3 blocks in encoder) and number of heads in graph attention, conducted with tst2013

5. **Conclusions.** In this paper, we propose a semantic NMT framework, which can successfully exploit the AMR graphs as the semantic representations. Extensive experimentation and analysis show that our model has indeed learned to leverage a better meaning context. We also propose the simple yet efficient approach for adapting lightweight convolutional neural networks in our NMT framework which makes the training/predicting process much faster.

For future work, we intend to investigate the possibilities to leverage the transformer framework with AMR graphs as well as exploring more semantic graphs such as semantic dependency parsing and elementary dependency structures.

REFERENCES

[1] I. Sutskever, O. Vinyals and Q. V. Le, Sequence to sequence learning with neural networks, *Proc. of the 27th International Conference on Neural Information Processing Systems*, Montreal, Canada, 2014.

[2] D. Bahdanau, K. Cho and Y. Bengio, Neural machine translation by jointly learning to align and translate, *Proc. of International Conference on Learning Representations*, San Diego, CA, USA, 2015.
[3] J. Gehring, M. Auli, D. Grangier, D. Yarats and Y. N. Dauphin, Convolutional sequence to sequence learning, *Proc. of the 34th International Conference on Machine Learning*, Sydney, NSW, Australia, 2017.
[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhi, Attention is all you need, *Conference on Neural Information Processing Systems*, 2017.
[5] D. Wu and P. Fung, Semantic roles for SMT: A hybrid two-pass model, *Proc. of the HLT-NAACL*, 2009.
[6] D. Liu and D. Gildea, Semantic role features for machine translation, *COLING-10*, 2010.
[7] D. Xiong, M. Zhang and H. Li, Modeling the translation of predicate-argument structure for SMT, *Proc. of the 50th Annual Meeting of the Association for Computational Linguistics*, 2012.
[8] M. Bazrafshan and D. Gildea, Semantic roles for string to tree machine translation, *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.
[9] D. Marcheggiani, J. Bastings and I. Titov, Exploiting semantics in neural machine translation with graph convolutional networks, *Proc. of the 2018 Meeting of the North American Chapter of the Association for Computational Linguistics*, 2018.
[10] L. Song, D. Gildea, Y. Zhang, Z. Wang and J. Su, Semantic neural machine translation using AMR, *Transactions of the Association for Computational Linguistics*, vol.7, pp.19-31, 2019.
[11] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer and N. Schneider, Abstract meaning representation for sembanking, *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pp.178-186, 2013.
[12] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.*, vol.9, pp.1735-1780, 1997.
[13] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin and M. Auli, Pay less attention with lightweight and dynamic convolutions, *arXiv preprint arXiv:1901.10430v2*, 2019.
[14] Y. LeCun and Y. Bengio, Convolutional networks for images, speech, and time series, *The Handbook of Brain Theory and Neural Networks*, 1995.
[15] W. Hamilton, Z. Ying and J. Leskovec, Inductive representation learning on large graphs, *Proc. of the 31st International Conference on Neural Information Processing Systems*, 2017.
[16] K. Xu, L. Wu, Z. Wang, Y. Feng, M. Witbrock and V. Sheinin, Graph2Seq: Graph to sequence learning with attention-based neural networks, *arXiv preprint arXiv:1804.00823*, 2018.
[17] L. Song, Y. Zhang, Z. Wang and D. Gildea, A graph-to-sequence model for AMR-to-text generation, *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia, vol.1, pp.1616-1626, 2018.
[18] J. Fu, Y. Huang, J. Xu and H. Wu, Optimization of distributed convolutional neural network for image labeling on asynchronous GPU model, *International Journal of Innovative Computing, Information and Control*, vol.15, no.3, pp.1145-1156, 2019.
[19] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, *Proc. of Conference on Computer Vision and Pattern Recognition*, 2015.
[20] J. L. Ba, J. R. Kiros and G. E. Hinton, Layer normalization, *arXiv,abs/1607.06450*, 2016.
[21] M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, R. Cattoni and M. Federico, *The IWSLT 2015 Evaluation Campaign*, 2015.
[22] R. Sennrich, B. Haddow and A. Birch, Neural machine translation of rare words with sub-word units, *Proc. of the 54th Annual Meeting of the Association for Computational Linguistics*, pp.1715-1725, 2016.
[23] I. Konstas, S. Iyer, M. Yatskar, Y. Choi and L. Zettlemoyer, Neural AMR: Sequence-to-sequence models for parsing and generation, *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.
[24] S. Cai and K. Knight, Smatch: An evaluation metric for semantic feature structures, *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.
[25] 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 of the Association for Computational Linguistics*, 2002.
[26] P. Koehn, Statistical significance tests for machine translation evaluation, *Proc. of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004.
[27] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, *International Conference on Learning Representations*, 2015.