Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 9939389, 12 pages
https://doi.org/10.1155/2021/9939389

Research Article

Improving Neural Machine Translation with AMR Semantic Graphs

Long H. B. Nguyen,[1,2] **Viet H. Pham**,[1,2] **and Dien Dinh**[1,2]

[1]*Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam*
[2]*Vietnam National University, Ho Chi Minh City, Vietnam*

Correspondence should be addressed to Long H. B. Nguyen; long.hb.nguyen@gmail.com

Received 15 March 2021; Revised 18 June 2021; Accepted 23 June 2021; Published 8 July 2021

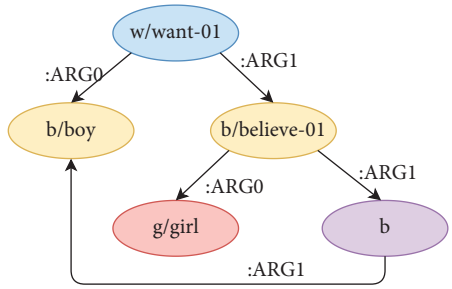
Academic Editor: Ali Ahmad

The Seq2Seq model and its variants (ConvSeq2Seq and Transformer) emerge as a promising novel solution to the machine translation problem. However, these models only focus on exploiting knowledge from bilingual sentences without paying much attention to utilizing external linguistic knowledge sources such as semantic representations. Not only do semantic representations can help preserve meaning but they also minimize the data sparsity problem. However, to date, semantic information remains rarely integrated into machine translation models. In this study, we examine the effect of abstract meaning representation (AMR) semantic graphs in different machine translation models. Experimental results on the IWSLT15 English-Vietnamese dataset have proven the efficiency of the proposed model, expanding the use of external language knowledge sources to significantly improve the performance of machine translation models, especially in the application of low-resource language pairs.

1. Introduction

Neural machine translation (NMT) [1–4] has proven its effectiveness and thus has gained researchers' attention in recent years. In practical applications, the typical inputs to NMT systems are sentences in which words are represented as individual vectors in a word embedding space. This word embedding space does not show any connection among words within a sentence such as dependency or semantic role relationships. Recent studies [5–8] found that semantic information is essential to generate concise and appropriate translations in machine translation. Although these models have made a significant progress, their design and functions are limited to statistical machine translation systems only. Consequently, the tasks of surveying, analyzing, and applying additional semantic information to NMT systems have not received comprehensive attention.

In this study, we present the method of integrating abstract meaning representation (AMR) graphs (https://amr.isi.edu) as additional semantic information into the current popular NMT systems such as Seq2Seq, ConvSeq2Seq, and Transformer. AMR graphs are rooted, labeled, directed, and acyclical graphs representing the entire content of a sentence. They are also abstracted from related syntactic representations in the sense that sentences with similar meanings will have the same AMR graph, even if the words used in these sentences are different. Figure 1 illustrates an AMR graph in which the nodes (e.g., want-01, girl) symbolize concepts, while the edges (e.g., ARG0 and ARG1) represent the relationship between the concepts that they connect. Compared to semantic role graphs, AMR graphs contain more relationships (e.g., between boy and girl). Besides, AMR graphs directly hold entity relations while excluding the alternating variables (i.e., using lemma) and the function words. Therefore, AMR graphs can be combined with the input text to generate better contextual representations. Moreover, the structured information from AMR graphs can help minimize the problem of data sparsity in resource-poor settings. First, the AMR graph representations are combined with the word embedding to create a better context representation for a sentence. Then, multihead attention can focus on all positions of



The boy *desires* the girl to believe him.
The boy *wants* the girl to believe him.
The boy has a *desire* to be believed by the girl.
The boy *is desirous of* the girl believing him.

Figure 1: The AMR graph for the multiple sentences.

contextual features with the outputs of the AMR graph representations.

Integrating AMR graphs into NMT yields several benefits. First, this addresses the problems of data sparsity and semantic ambiguity. Second, structured semantic information constructed from AMR graphs could help complement the input text by providing high-level abstract information, thereby improving the encoding of the input word embedding. Last, multihead attention can also take advantage of semantic information to improve the dependency among words within a sentence.

Recent studies have applied semantic representation to NMT models. For instance, Marcheggiani et al. [9] exploited the semantic role labeling (SRL) information for NMT, indicating that the predicate-argument structure from SRL can help increase the quality of an attention-based sequence-to-sequence model. Meanwhile, Song et al. [10] proved that semantic information structured from AMR graphs can complement input text by incorporating high-level abstract information. In this approach, the graph recurrent network (GRN) was utilized to encode AMR graphs without breaking the original graph structure, and a sequential long short-term memory (LSTM) 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. had also argued that the results of an AMR integration is significantly greater than those of a sole SRL integration because AMR graphs include both SRL and the relationships between the nodes (i.e., words). However, Song's approach has encountered some drawbacks such as failed to address the problem of the correlation between nodes in AMR graphs and investigated only on the machine translation system using the recurrent neural network (RNN).

The contributions of our work are as follows:

(i) First, instead of adding a node to represent an edge in the graph and assigning properties of the edge as those of the documents, we extend the node embedding algorithm [11] to use direct edge information

(ii) Second, instead of using the graph recurrent network in [10], we propose an architecture that binds an inductive graph encoder

(iii) Finally, we examined and analyzed the results on the English-Vietnamese bilingual set, which is considered a low-resource language pair. Through experiments, we demonstrate the effectiveness of integrating AMR into neural network machine translation and draw insightful conclusions for future studies.

The organization for the remaining of the article is as follows. Section 2 introduces current popular machine translation architectures such as Seq2Seq, ConvSeq2Seq, and Transformer. Next, Section 3 presents the method of representing AMR graphs in the vector form as well as proposing a method to integrate AMR graphs into different NMT models. Then, Sections 4 and 5 discuss the corpus used in the experiment and the experimental configuration for the model, respectively. Afterward, Section 6 presents the experimental results of the machine translation model with an integrated AMR and analyzes the effect of an AMR on the model along with some translation errors generated by the model. Section 7 summarizes our work.

2. Neural Machine Translation

In this section, we provide a brief introduction about the Seq2Seq model and its variants such as ConvSeq2Seq and Transformer.

2.1. Seq2Seq. We take the attention-based sequence-to-sequence model of [1] as the baseline model, but we use LSTM [12] in both encoder and decoder.

2.1.1. Encoder. Given a sentence, $x = (x_1, x_2, \ldots, x_n)$.

(i) Uni-LSTM. As usual, the RNN reads an input sequence x in order starting from the first token x_1 to x_n and computes a sequence of hidden state $[\overrightarrow{h}_1, \overrightarrow{h}_2, \ldots, \overrightarrow{h}_n]$ to generate input representation from left to right.

$$H = \left[\overrightarrow{h}_1, \overrightarrow{h}_2, \ldots, \overrightarrow{h}_n\right]. \quad (1)$$

(ii) Bi-LSTM. Consists of forward and backward LSTM's. The forward LSTM works similar to Uni-LSTM and the backward LSTM reads the sequence in the reverse order from the last token x_n to x_1, resulting a sequence of backward hidden states $[\overleftarrow{h}_1, \overleftarrow{h}_2, \ldots, \overleftarrow{h}_n]$. We obtain the word embedding x_i by concatenating the forward $\overrightarrow{h}_i$ and backward $\overleftarrow{h}_i$ hidden state, $h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$.

$$H = [h_1, h_2, \ldots, h_n]. \tag{2}$$

2.1.2. Decoder. The decoder predicts the next word y_t, given the context vector c and all previously predicted words $(y_0, y_1, \ldots, y_t - 1)$. We used an attention-based LSTM decoder [1], with attention memory as the concatenation of the attention vectors among all source tokens.

For each decoding step t, the decoder feeds the concatenation of the embedding of current input e_{y_t} and previous context vector c_{t-1} into LSTM to update the hidden state:

$$s_t = \mathrm{LSTM}\left(s_{t-1}, \left[e_{y_t}, c_{t-1}\right]\right). \tag{3}$$

Then, the new context vector is computed as

$$\begin{aligned} \epsilon_{t,i} &= a\left(s_t, h_i\right), \\ \alpha_{ti} &= \frac{\exp\left(\epsilon_{t,i}\right)}{\sum_{k=1}^{\mathscr{V}} \exp\left(\epsilon_{t,k}\right)}, \\ c_i &= \sum_{i=1}^{n} \alpha_{t,i} h_i, \end{aligned} \tag{4}$$

where a is the alignment model which is a feed forward network, scores how well the inputs surround position i, and the input at position t match.

The output probability over target vocabulary is calculated:

$$P_{\mathrm{vocab}} = \mathrm{soft\,max}\left(W_o\left[s_t; c_i\right] + b_o\right), \tag{5}$$

where W_o and b_o are the model parameters.

2.2. ConvSeq2Seq

2.2.1. ConvSeq2Seq. This architecture is proposed by Gehring et al. [2] to completely replace the RNN with the CNN with the following components:

The ConvS2S model followed the encoder-decoder architecture. Both encoder and decoder blocks share an identical structure that computes hidden states based on a fixed number of input elements. To enlarge the context size, we stack several blocks over each other. Each block comprises a one-dimensional convolution and a nonlinearity. In each convolution kernel, parameters are $W \in \mathscr{R}^{2\,d \times k\,d}$ and $b_w \in \mathscr{R}^{2\,d}$. The input is represented as $\in\in\mathscr{R}^{k\,d}$, which is a concatenation of k input elements with dimension of d and maps them to get the single output $Y \in \mathscr{R}^{2\,d}$ with dimension twice of that of the input. Then, the k output elements will be fed into subsequent layers. We leverage the gated linear unit (GLU) as nonlinearity which applied on the output of the convolution $Y = [AB] \in \mathscr{R}^{2\,d}$:

$$v([AB]) = A \otimes \sigma(B), \tag{6}$$

where $A, B \in \mathscr{R}^d$ are the inputs to the nonlinearity, $\otimes$ denotes the element-wise multiplication, the output $Y = [AB] \in \mathscr{R}^{2\,d}$ has a half of size compared to Y, and σB is the gate that control which inputs A of the current contexts are relevant.

In order to enable deep convolutional blocks, we adopt the residual connections which connect the input of each convolutional layer with the output:

$$h_i^l = v\left(W^l\left[h_{i-(k/2)}^{l-1}, \ldots, h_{i+(k/2)}^{l-1}\right] + B_w^l\right) + h_i^{l-1}, \tag{7}$$

where h^l is the hidden state of l^{th} layer.

2.2.2. LightConvSeq2Seq. As a variant of a CNN called lightweight convolution [13] which allows computation with linear complexity, $O(n)$, with n being the length of the input string.

The structure of LightConvSeq2Seq consists of the elements similar to Conv2Seq but using lightweight convolution operation rather than convolution operation.

Depthwise Convolution (DConv). Perform a convolution operation independently over every channel; thereby, the number of parameters reduce significantly from d^2k to dk, where k is the kernel width. In general, at position i and direction c, the output $O_{i,c}$ is calculated as follows:

$$O_{i,c} = \sum_{j=1}^{k} W_{c,j} \cdot X_{\left(x+j-\left\lceil\frac{k+1}{2}\right\rceil\right),c}. \tag{8}$$

2.3. Transformer.

Transformer [4] also includes an encoder and a decoder. The encoder generates a vector representation of the input sentence. Assuming an input of the form $x = (x_1, x_2, \ldots, x_n)$ and a representation of x of the form $z = (z_1, z_2, \ldots, z_n)$, the decoder produces sequentially for a translation of $y = (y_1, y_2, \ldots, y_m)$ based on z and the previous outputs.

2.3.1. The Encoder. There are N stacked similar blocks. Each of these blocks consists of 2 subblocks: a self-attention mechanism and a feed forward network. A residual connection surrounds each subblock, followed by layer normalization. The general representation formula for the encoder is as follows:

$$\begin{aligned} \widehat{z} &= \mathrm{LayerNorm}\left(x + \mathrm{self} - \mathrm{attention}\left(x\right)\right), \\ z &= \mathrm{LayerNorm}\left(\widehat{z} + \mathrm{feed\,forward}\left(\widehat{z}\right)\right). \end{aligned} \tag{9}$$

2.3.2. The Decoder. There are also N blocks. However, each block consists of 3 subblocks: a self-attention block, a feed forward block, and an encoder-decoder attention block inserted between them. The residual connection and layer normalization are used similarly to the encoder. The encoder generates outputs step by step. The self-attention block only pays attention to the positions generated in the previous steps by using a mask. The mask prevents the decoder from paying attention to locations that have not been generated,

so outputs can only be predicted based on the result z of the encoder and previous outputs.

2.3.3. Self-Attention. There are 3 components as follows: query (Q), key (K), and value (V), defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}V\right), \tag{10}$$

where Q, K, and V are the parameters with the number of dimensions d_k, d_k, and d_v respectively.

3. The Proposed Method

In this section, we present the graph embedding algorithm and propose our method to integrate the AMR graph embedding representation to various well-known NMT systems such as Seq2Seq, ConvSeq2Seq, and Transformer.

3.1. Graph-Level Information Representation. Figure 2 depicts the graph encoder architecture based on the model of Xu et al. [11], with some enhancements to integrate more information about the edge of the graph.

The directional graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the label on the edge $e_{u,v} \in \mathcal{E}$ presents the relationship between the nodes u and v to which it connects. The process of learning to represent the node $v \in \mathcal{V}$ is as follows:

(1) We first transform the text attribute of node v into a feature vector $\mathbf{a}_v$ by looking up the embedding matrix W_E

(2) Next, we categorize the neighbors of v into two subsets: forward neighbors, $\mathcal{N}_{\longrightarrow}(v)$ and backward neighbors, $\mathcal{N}_{\leftarrow}(v)$. Particularly, $\mathcal{N}_{\longrightarrow}(v)$ returns the nodes that v directs to and vice versa. The information about the edge $e_{u,v}$ associated between the node v and the adjacent node u is combined as follows:

$$\begin{aligned} u_{\leftarrow} &= u_{\leftarrow} + e_{u,v}, \forall u \in \mathcal{N}_{\leftarrow}(v), \\ u_{\longrightarrow} &= u_{\longrightarrow} + e_{u,v}, \quad \forall u \in \mathcal{N}_{\longrightarrow}(v). \end{aligned} \tag{11}$$

(3) We aggregate the forward information of v's forward neighbors $\left\{\mathbf{h}_{u\longrightarrow}^{k-1}, \forall u \in \mathcal{N}_{\longrightarrow}(v)\right\}$ into a single vector, $\mathbf{h}_{\mathcal{N}_{\longrightarrow}(v)}^{k}$, where $k \in \{1, \ldots, K\}$ is the iteration index. We do this by using one of three $\texttt{AGG}^{\longrightarrow}$ mentioned.

(4) Then, we concatenate v's current forward representation, $\mathbf{h}_{v\longrightarrow}^{k-1}$, with the new neighborhood vector, $\mathbf{h}_{\mathcal{N}_{\longrightarrow}(v)}^{k}$. The result is passed to a feed forward layer, followed by a nonlinearity activation function σ, which updates the forward representation of v, to be used in the next iteration.

(5) Update the backward representation of v, $\mathbf{h}_{v\leftarrow}^{k}$, using similar procedure in steps (3) and (4), but this time, we utilize backward representations rather than the forward representations and use $\texttt{AGG}^{\leftarrow}$ to aggregate neighbor information.

(6) Repeat steps (3) ~ (5) K times, and the concatenation of the final forward and backward representation is used as the final bidirectional representation of v.

$$z_v = \texttt{CONCAT}\left(\mathbf{h}_{v\longrightarrow}^{K}, \mathbf{h}_{v\leftarrow}^{K}\right), \quad \forall v \in \mathcal{V}. \tag{12}$$

As mentioned in steps (3) and (5), the representation association operation of node v is performed with one of the following aggregation functions:

(i) Mean aggregator: performs the average calculation on each element of $\left\{\mathbf{h}_{u\longrightarrow}^{k-1}, \forall u \in \mathcal{N}_{\longrightarrow}(v)\right\}$ và $\left\{\mathbf{h}_{u\leftarrow}^{k-1}, \forall u \in \mathcal{N}_{\leftarrow}(v)\right\}$

(ii) GCN aggregator: it is quite similar to mean aggregator, except that the result is fed into a fully connected layer and a nonlinear activation function [14].

$$\begin{aligned} \texttt{AGG}_k^{\longrightarrow} &= \sigma\left(\mathbf{W}\texttt{MEAN}\left(\mathbf{h}_{u\longrightarrow}^{k}\right) + \mathbf{b}\right), u \in \mathcal{N}_{\longrightarrow}(v), \\ \texttt{AGG}_k^{\leftarrow} &= \sigma\left(\mathbf{W}\texttt{MEAN}\left(\mathbf{h}_{u\leftarrow}^{k}\right) + \mathbf{b}\right), \quad u \in \mathcal{N}_{\leftarrow}(v), \end{aligned} \tag{13}$$

with MEAN as the function returning the average value, and σ as the nonlinear activation function.

(iii) Pooling aggregator: each node embedding vector is passed through a feed forward layer followed by the pooling operation (which can be max, min, and average):

$$\begin{aligned} \texttt{AGG}_k^{\longrightarrow} &= \max\left(\left\{\sigma\left(\mathbf{W}_p\mathbf{h}_{u\longrightarrow}^{k} + \mathbf{b}\right), \quad u \in \mathcal{N}_{\longrightarrow}(v)\right\}\right), \\ \texttt{AGG}_k^{\leftarrow} &= \max\left(\left\{\sigma\left(\mathbf{W}_p\mathbf{h}_{u\leftarrow}^{k} + \mathbf{b}\right), \quad u \in \mathcal{N}_{\leftarrow}(v)\right\}\right), \end{aligned} \tag{14}$$

with max as the maximum operation, and σ as the nonlinear activation function.

3.1.1. Graph Embedding. Graph embedding Z contains all the information on the graph and is calculated by one of the following two methods:

(i) Pooling based: the node embeddings z_v, $v \in \mathcal{V}$ are passed through a linear transform network and performs pooling.

$$\mathbf{Z} = \text{pooling}\left(\{z_v, \quad \forall v \in \mathcal{V}\}\right). \tag{15}$$

(ii) Adding a super node: node v_s is pointed by all nodes in the graph. Using the algorithm in Section 3.1, the representation of v_s is z_{v_s}. The representation of v_s contains all information of the nodes that should be considered as representations of the graph or graph embedding.

3.2. Dual Attention Mechanism. The architecture of an integrated AMR machine translation model is illustrated in

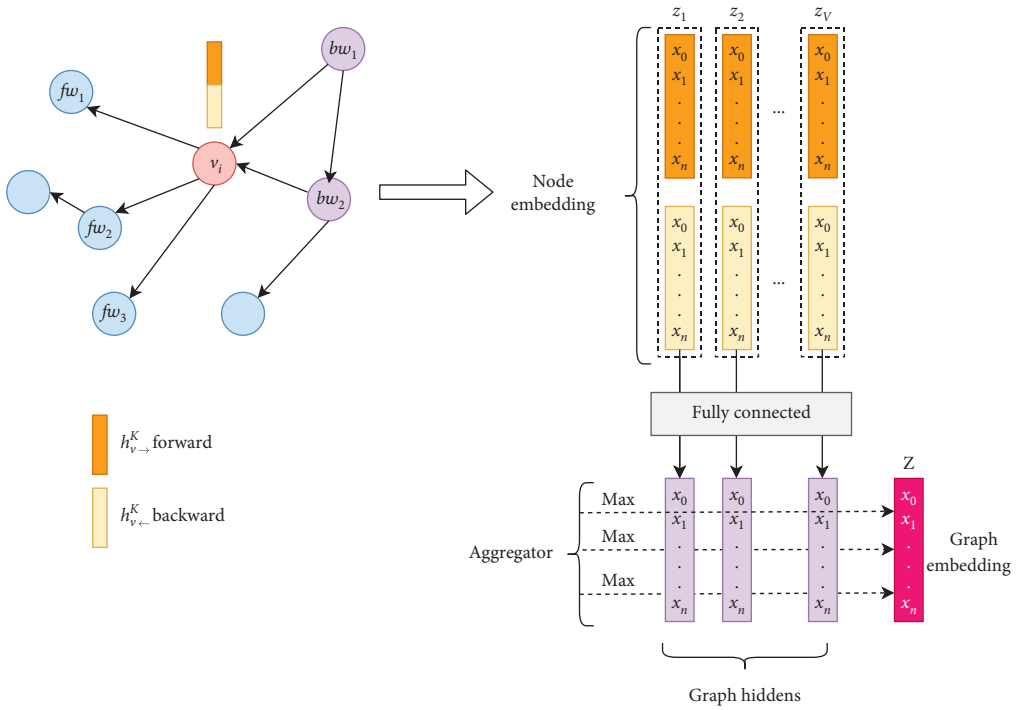


Figure 2: The graph encoder architecture.

Figure 3 with an English sentence input and a corresponding AMR graph. The proposed architecture consists of an encoder for the input sentence and a decoder with the input value resulting from the encoder. The main difference from the traditional decoder-encoder model is that there is an additional graph encoder to process information on graphs and to represent this information in a vector format. This vector is then combined with the hidden states of the encoder and fed into the decoder to find the corresponding representation in Vietnamese.

We propose a specific integration method for the Seq2Seq model with sequential processing in Section 3.2.1 and focus on models with parallel processing such as ConvSeq2Seq, LightConvSeq2Seq, and Transformer in Section 3.2.2.

3.2.1. Seq2Seq Model with the Sequential Processing Mechanism. The model (Figure 4(a)) consists of two attention mechanisms operating independently: the original attention (left) learns the alignment between the result y_{i-1} and the hidden states h_j, $j \in [1, n]$ of the encoder and the graph attention learns to align between the output and the nodes in the AMR graph, yielding a context vector $\hat{c}_{i-1}$. In particular, the computation of $\hat{c}_{i-1}$ is as follows:

$$\hat{e}_{ij} = a\left(s_{i-1}, z_j\right),$$

$$\hat{\alpha}_{ij} = \frac{\exp\left(\hat{e}_{ij}\right)}{\sum_{k=1}^{\mathscr{V}} \exp\left(\hat{e}_{ik}\right)} \tag{16}$$

$$\hat{c}_i = \sum_{j=1}^{\mathscr{V}} \hat{\alpha}_{ij} z_j,$$

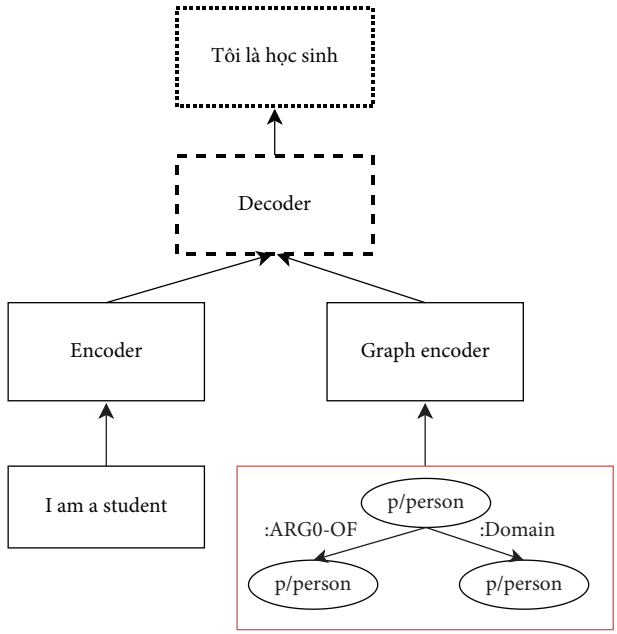


Figure 3: Recommended architecture for AMR integration.

where a is a feed forward network, evaluating the matching between the nodes surrounding the position j and the input i.

These two context vectors are then combined with the decoder's state s_i and the embedding vector of y_{i-1} to calculate a probability distribution that determines y_i.

$$P_{\text{vocab}} = \text{Softmax}\left(W_o\left[s_i, y_{i-1}, c_i, \hat{c}_i\right] + b_o\right). \tag{17}$$

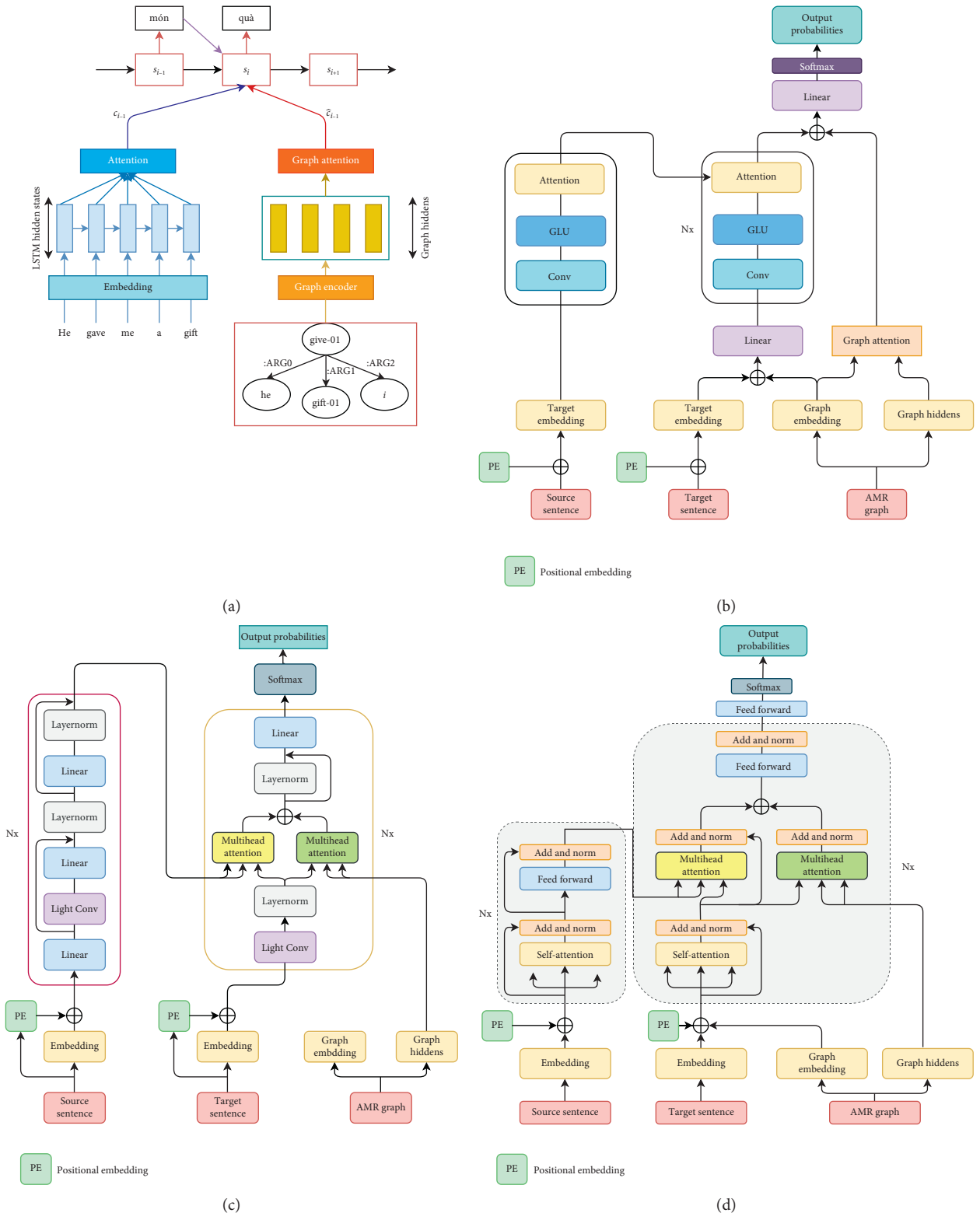


FIGURE 4: Integrating AMR to models with a parallel processing mechanism. (a) Seq2Seq (LSTM) model with the AMR. (b) ConvSeq2Seq-AMR model. (c) LightConvSeq2Seq-AMR model. (d) Transformer-AMR model.

3.2.2. Models with a Parallel Processing Mechanism. On the contrary, with parallel processing, the model has no information about the state s_{i-1} of the decoder. In other words, except z_v, no information about the graph is included in the calculation of attention. Besides, using only the states z_v, $v \in \mathcal{V}$ along with the parallel computation leaves the model with no information about the association between the output and the AMR graph in step $i-1$. Consequently,

the model cannot effectively learn the connection between the input sentence, the output sentence, and AMR graph, with a small increase of about 0.2 (experiments with LightConvSeq2Seq and Transformer). Therefore, the use of the graph embedding of Z should help the model obtain more information about the graph before the attention calculation. This has been proven with experimental results, which show an increase of the BLUE score by 0.6.

Figures 4(b)–4(d) describe the proposed model that integrates AMR with a dual attention mechanism. Regarding the LightConvSeq2Seq-AMR and Transformer-AMR models, the self-attention mechanism for the graph is similar to the description of the self-attention mechanism in Section 2.3 with the input being representations of nodes z_v, $\forall v \in \mathscr{V}$ instead of the state h_i, $\forall i \in n$. Regarding the ConvSeq2Seq-AMR model, experimental results show that utilizing Luong's attention mechanism to learn the alignment between the graph and the output produced better results than the multistep attention.

4. The Corpus

The corpus used to evaluate the model is IWSLT15 [15], which includes approximately 130, 000 English-Vietnamese bilingual sentences taken from TED Talks presentations for the training set. For fine-tuning, we use the set called tst2012, which includes 1553 parallel pairs language. Besides, the test sets consist of tst2013 and tst2015, which include 1268 and 1080 English-Vietnamese bilingual pairs, respectively. The statistical information is given in Table 1.

For the preprocessing phase, byte-pair encoding (BPE) (https://github.com/rsennrich/subword-nmt) [16] with 8000 operations is utilized to deal with rare words and compound words for both English and Vietnamese, thereby significantly reducing the vocabulary size in English from 54111 to 5208 and in Vietnamese from 25335 to 3336.

For AMR parsing, we use NeuralAmr toolkit (https://github.com/sinantie/NeuralAmr) [17] 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 [18], the current best score (at the time doing this work, Jan 2020) reported without the significant use of external semantic resources. This tool produces AMR graphs represented in the PENMAN notation (https://www.isi.edu/natural-language/penman/penman.html) and in a linear form, as demonstrated in the AMR preprocessing example.

5. Experimental Configuration

The models are implemented in Python 3 and use the library Fairseq (https://fairseq.readthedocs.io/en/latest/#) [19].

The configuration of the base models is as follows:

(i) Seq2Seq: we investigate the MT model with two types of LSTM which are uni-LSTM (one-directional) and bi-LSTM (two-directional). There are 512-word embedding dimensions, which utilize 512 LSTM hidden units in both the encoder and the decoder.

(ii) ConvSeq2Seq: it comprises 4 convolutional blocks and 512 hidden units for both the encoder and the decoder. The kernel size is 3.

(iii) LightConvSeq2Seq: it consists of 4 convolutional blocks with the kernel size of 3, 7, 15, and 31 for each block and applies to both the encoder and the decoder. Self-attention is adopted with $H = 8$ heads.

(iv) Transformer: it has $N = 6$ blocks for both the encoder and the decoder. The word embedding dim is set to 512 and 2048 for the feed forward network. Self-attention used with the number of heads was 8.

The proposed models have the same configuration as the base model. Besides, the graph encoder used 128-dimensional embedding for the representation of both edge and node. We stacked 2 layers of the graph encoder and aggregating information from neighboring nodes with the mean aggregator for LSTM and max pooling with the rest of the models.

During training, Adam optimizer [20] is used with a fixed learning rate of 0.001 for LSTM and ConvSeq2Seq, 0.0002 for LightConvSeq2Seq, and 0.0005 for Transformer.

Besides the basic models presented above, the results of the proposed model are also compared with the method of Song et al. [10]. To make a fair comparison, we have retrained Song's model with the same preprocessed dataset and tuned hyperparameters.

After the models are trained, the BLEU score [21] was used to evaluate the translation quality. We also apply the bootstrap resampling method [22] to measure the statistical significance ($p < 0.05$) of BLEU score differences between translation outputs of proposed models compared to the baseline.

6. Results and Discussion

In this section, we present our experimental results and our analyzes on the results.

6.1. Results. Once the models have been trained, a beam search with the size of 5 is utilized to find a translation that maximizes the conditional probabilities.

With both the test sets tst2013 and tst2015, the proposed models are proven to be superior to the corresponding base model. In particular, as given in Table 2, with uni-LSTM-AMR-F and bi-LSTM-AMR, the BLEU scores are 27.21 and 29.29, respectively, which are 1.09 and 3.17 higher than Song's method [10]. Similarly, with the set tst2015, bi-LSTM-AMR improved BLEU by 2.83, compared to Song's method. This shows that despite using the double attention mechanism, bi-LSTM-AMR and uni-LSTM-AMR can integrate the information from AMR more effectively, thereby producing better translation results.

Meanwhile, when LightConvSeq2Seq is run on tst2013 and tst 2015, the BLEU scores are only 27.47 and 25.09, respectively. However, when integrating the AMR into the system, the BLEU score increased significantly by 1.0 and 0.58 on tst2013 and tst2015, respectively. Besides, LightConvSeq2Seq-AMR-F and LightConvSeq2Seq-AMR-B,

Table 1: Statistics on the corpus.

Corpus	#Sentences	#Tokens (English)	#Tokens (Vietnamese)
Training	133K	2.44 M	2.87 M
Fine tuning (tst2012)	1553	28 K	34 K
Test 1 (tst2013)	1268	26.7 K	33.6 K
Test 2 (tst2015)	1080	21 K	26.2 K

Table 2: Experimental results on Seq2Seq using one- and two-directional LSTMs.

Model	BLEU	
	tst2013	tst2015
Song's method	26.12	23.58
Uni-LSTM-AMR	26.97	24.80
Uni-LSTM-AMR-F	**27.21**	**24.86**
Uni-LSTM-AMR-B	26.61	24.66
Bi-LSTM-AMR	**29.29**	**26.41**
Bi-LSTM-AMR-F	28.67	26.20
Bi-LSTM-AMR-B	28.36	26.04

The bold values are the highest results for each group of models.

Table 3: Experimental results on LightConvSeq2Seq.

Model	BLEU	
	tst2013	tst2015
LightConvSeq2Seq	27.47	25.09
LightConvSeq2Seq-AMR-F	27.71	25.05
LightConvSeq2Seq-AMR-B	27.84	25.27
LightConvSeq2Seq-AMR	**28.46**	**25.67**

The bold values are the highest results when evaluating each model for the "tst2013" and "tst2015" testsets.

Table 4: Experimental results on ConvSeq2Seq and Transformer.

Model	BLEU	
	tst2013	tst2015
ConvSeq2Seq	26.98	24.78
ConvSeq2Seq-AMR	27.30	24.70
ConvSeq2Seq-AMR-F	**27.40**	**25.20**
ConvSeq2Seq-AMR-B	26.73	24.53
Transformer	28.60	25.76
Transformer-AMR	28.69	25.91
Transformer-AMR-F	**28.88**	**26.28**
Transformer-AMR-B	28.69	26.01

The bold values are the highest results for each group of models.

which were integrated graph information from one direction, also outperform LightConvSeq2Seq, as given in Table 3.

As given in Table 4, ConvSeq2Seq also shows an improvement in machine translation quality with an increase in the BLEU score to about 0.3 for ConvSeq2Seq-AMR with tst2013. However, there is a BLUE decrease of 0.08 with tst2015. However, the ConvSeq2Seq-AMR-F model achieves the best results when integrating information from the forward neighbors. An increase of 0.1 in BLEU is observed with tst2013 and 0.5 with tst2015. Similar to Transformer, integrating information from the forward and backward neighbors in Transformer-AMR is not effective, with only an increase of 0.09 over the base model with tst2013. Only combining information from the forward neighbors in Transformer-AMR-F achieves a noticeable BLEU score of 28.88 and 26.28 with tst2013 and tst2015, respectively, which signal an increase of 0.28 and 0.52 compared to Transformer.

6.2. The Effect of AMR on the NMT Model. According to the results presented in Section 6.1, the bi-LSTM-AMR and LightConvSeq2Seq-AMR models improve BLEU more than the other two models, ConvSeq2Seq and Transformer. Therefore, to analyze the impact of AMR on the machine translation system, bi-LSTM-AMR and LightConvSeq2Seq-AMR models are selected for further training to examine graph elements such as information integration directions, graph encoding layers, and aggregators.

6.2.1. Bi-LSTM-AMR

(i). Direction and Depth. Figure 5 depicts the change in performance when adjusting the number of graph encoding layers. The mean aggregator is used to combine information from neighbors. In general, bi-LSTM-AMR and uni-LSTM-AMR-B show the highest translation quality throughout the 30 examined layers. However, an increase in the number of layers does not always help the model achieve a higher BLEU. A decrease in BLEU scores is also observed. The more stacked layers there are, the greater the amount of information the model could learn, which ultimately leads to the overfitting problem due to saturated information. All models obtain the best results with only 2 or 3 graph coding layers. As the number of layers increases, the BLEU scores decrease. Nevertheless, the results seem more consistent and less fluctuating with bi-LSTM than with uni-LSTM.

There are three aggregators used for aggregating information from neighboring nodes: mean aggregator (MA), max-pooling (MP) aggregator, and GCN aggregator (GCN-A). The strategy of using information from one direction (forward or backward) is also considered to make more accurate statements about the effect of the aggregator on the effectiveness of the model. The results in Table 5 show that Bi-LSTM-AMR-MA achieved the highest result on the two test sets with the BLEU scores of 29.29 and 26.41, respectively. Meanwhile, uni-LSTM-AMR-MA, which uses information from both sides, achieved lower BLEU scores than the variants uni-LSTM-AMR-F and uni-LSTM-AMR-B, which only combines information from the forward and the backward neighbors, respectively. Moreover, bi-LSTM-AMR-MA outperforms bi-LSTM-AMR-F and bi-LSTM-AMR-B due to its ability to capture information from two directions during the node embedding learning and combine with information

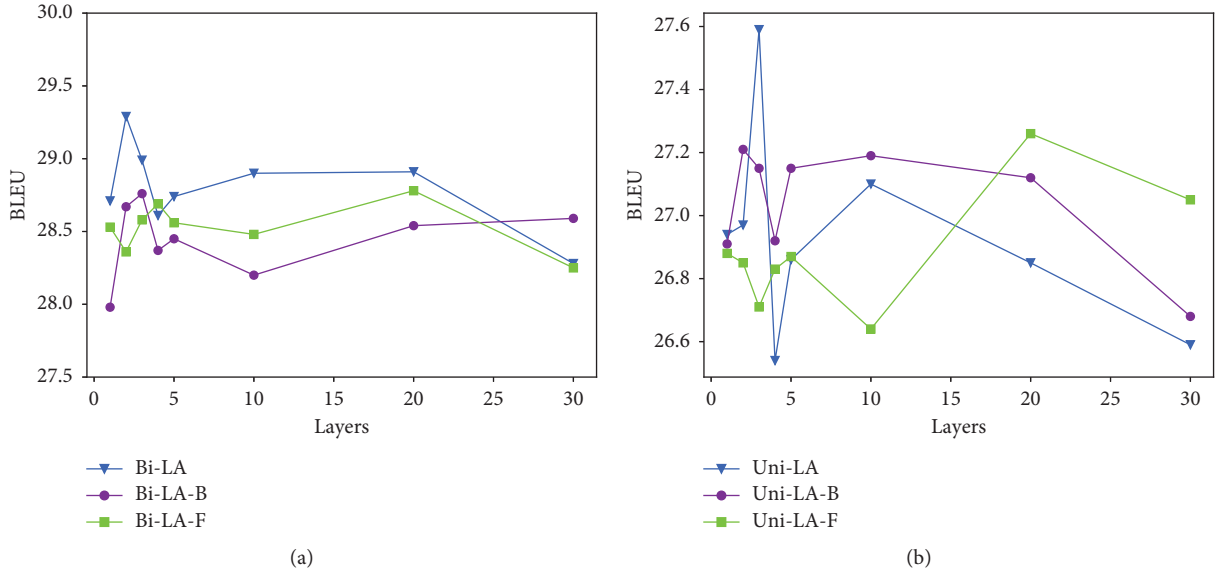


Figure 5: Experimental results on tst2013 with a change in the number of graph encoding layers.

Table 5: The effect of aggregators on the machine translation quality.

Model	tst2013			tst2015		
	MA	MP	GCN-A	MA	MP	GCN-A
Uni-LSTM-AMR	26.97	26.54	26.55	24.81	24.22	24.47
Uni-LSTM-AMR-F	26.85	27.21	26.76	24.59	**24.86**	24.48
Uni-LSTM-AMR-B	26.61	**27.22**	26.42	24.66	24.85	24.47
Bi-LSTM-AMR	**29.29**	28.94	28.38	**26.41**	26.24	25.59
Bi-LSTM-AMR-F	28.67	28.99	28.43	26.20	25.81	25.70
Bi-LSTM-AMR-B	28.36	28.40	28.41	26.04	25.90	25.81

Table 6: Experimental results of aggregators on LightConvSeq2Seq-AMR.

Model	tst2013			tst2015		
	MA	MP	GCN-A	MA	MP	GCN-A
LightConvSeq2Seq-AMR	28.20	**28.46**	27.76	**25.49**	25.05	25.33
LightConvSeq2Seq-AMR-F	**27.82**	27.71	27.59	**25.96**	25.27	25.39
LightConvSeq2Seq-AMR-B	**28.25**	27.84	28.27	25.66	**25.67**	25.52

The bold values are the highest results when evaluating each model on aggregators (i.e., MA, MP, and GCN-A) for testset (i.e., tst2013 or tst2015).

from the bi-LSTM encoder. Therefore, the LSTM decoder can leverage information from the graph more efficiently to improve the machine translation quality. This shows that bidirectional aggregation is more useful when combined with a bidirectional LSTM encoder. Accordingly, uni-LSTM-AMR-F-MP and uni-LSTM-AMR-B-MP, which only combine information from one direction, achieve good results when used with a unidirectional LSTM encoder.

6.2.2. LightConvSeq2Seq-AMR. Similar to bi-LSTM-AMR, the LightConvSeq2Seq-AMR model is also affected by different aggregators. In particular, as given in Table 6, the mean aggregator (MA) yields better results on average values than the rest. The results with tst2015 show that all the three modes with MA both achieve much higher results than the rest of models. On the contrary, the GCN-A results are the lowest, similar to Seq2Seq. This proves that the information combination of GCN-A is not as efficient as those of MA and MP.

Figure 6 shows the change of BLEU when stacking convolutional blocks in the encoder and the decoder and the effect of heads H in self-attention. On both sides, the BLEU scores increase when the number of heads increases. In particular, the figure on the left shows the LightConvSeq2Seq-AMR model with the configuration $(4, 4)$, which stacked 4 convolutional blocks at the encoder and 4 convolutional blocks at the decoder, and $(6, 6)$ yields the best results. The BLEU scores are approximately 28 and 27.6 with just 1 head and then increases to 28.46 and 28.2 when $H = 8$. However, with an additional graph encoding layer, the $(4, 4)$ configuration is inferior to the $(6, 6)$ configuration. This configuration yields the highest

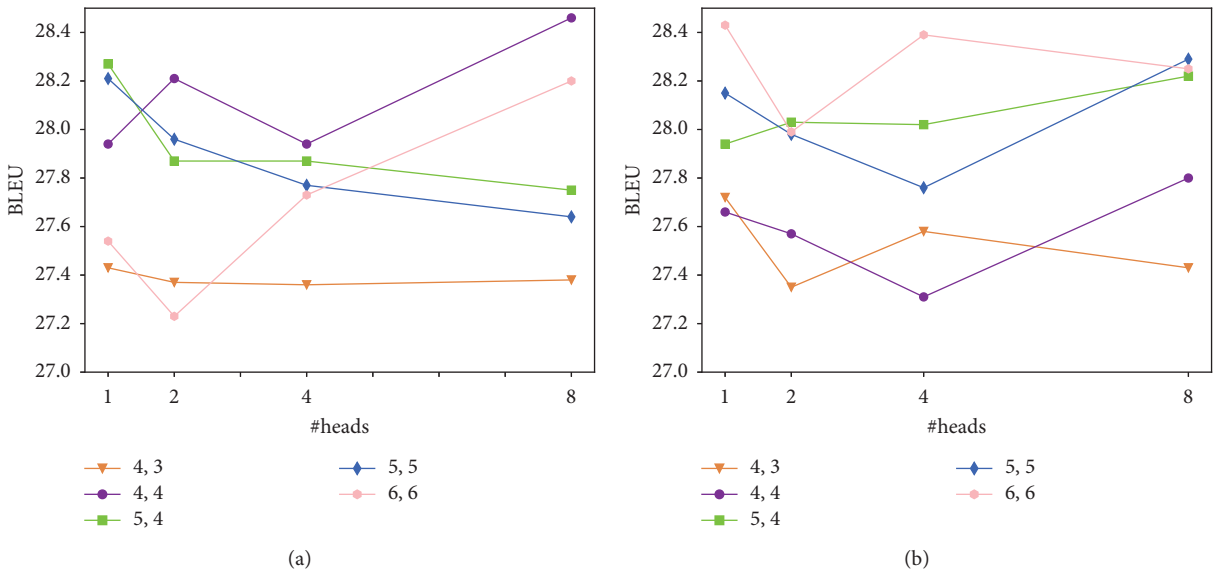


Figure 6: Experimental results on the effect of heads H of the graph attention and the depth of the LightConvSeq2Seq-AMR model. (a) LightConvSeq2Seq-AMR with two graph coding layers. (b) The model with three layers.

Table 7: Example 1.

SRC	Here is me on the soccer team and in V Magazine.
REF	Đây là tôi với đội bóng đá trong tạp chí V.
Bi-LSTM-AMR	Đây là tôi trên đội bóng đá và V là Magazine.
ConvSeq2Seq-AMR	Đây là tôi trên đội bóng đá, và trong V.
LightConvSeq2Seq-AMR	Đây là tôi đang ở trong đội bóng đá và V Magazine.
Transformer-AMR	Đây là tôi trong đội bóng đá và ở Magazine.

Table 8: Example 2.

SRC	The internal combustion engine is not sustainable.
REF	Động cơ đốt trong không bền vững.
Bi-LSTM-AMR	Động cơ nội tạng không bền vững.
ConvSeq2Seq-AMR	Loại động cơ bên trong không bền vững.
LightConvSeq2Seq-AMR	Động cơ đốt nội không bền vững.
Transformer-AMR	Động cơ đốt trong không bền vững.

Table 9: Example 3.

SRC	But it is not only about me.
REF	Nhưng những thông tin đó không chỉ nói về tôi.
Bi-LSTM-AMR	Nhưng nó không chỉ là về tôi.
ConvSeq2Seq-AMR	Nhưng đó không chỉ là tôi.
LightConvSeq2Seq-AMR	Nhưng nó không chỉ là tôi.
Transformer-AMR	Nhưng nó không chỉ là tôi.

results at $H = 1$ with BLEU approximately 28.4 and observes a slight decrease as H approaches 8. Meanwhile, $(5, 4)$ and $(5, 5)$ configurations tend to decline sharply as H increases from 1 to 2 (≈ -0.3) and continues to decline slightly until $H = 8$. Meanwhile, the two reconfigurations tend to be the opposite when adding a graph encoding layer, as shown on the right figure in Figure 6. The remaining $(4, 3)$ configuration yields the lowest

Table 10: More example translation outputs.

AMR	':arg we:arg (work:arg we:arg office:degree total)
SRC	We do not work from offices.
REF	Chúng tôi không làm việc từ những văn phòng.
Song's method	Chúng tôi không làm việc trong văn phòng.
Bi-LSTM	Chúng tôi không làm việc.
Bi-LSTM-AMR	Chúng tôi không làm việc từ các văn phòng.
AMR	eat:arg they:arg tomatoes:condition (grow:arg they)
SRC	If they grow tomatoes, they eat tomatoes.
REF	Nếu chúng trồng cà chua, chúng sẽ ăn cà chua.
Song's method	Nếu họ phát triển, họ sẽ ăn.
Bi-LSTM	Nếu họ trồng cà chua, họ ăn cà chua.
Bi-LSTM-AMR	Nếu chúng trồng cà chua, chúng ăn cà chua.
AMR	assure:arg i:arg i:arg (thing:arg-of (think:arg you:arg i:duration forever)): degree total:mod just
SRC	I just totally transformed what you thought of me in six seconds.
REF	Tôi vừa mới thay đổi hoàn toàn những gì bạn nghĩ về tôi trong vòng 6 giây.
Song's method	Tôi không chỉ là những gì bạn nghĩ về tôi trong vòng 6 giây.
Bi-LSTM	Tôi hoàn toàn thay đổi những gì bạn nghĩ trong vòng sáu giây.
Bi-LSTM-AMR	Tôi hoàn toàn thay đổi những gì bạn nghĩ về tôi trong 6 giây.

results for the 2 graph encoding layer options. The results also fluctuate more with 3 layers, as opposed to being nearly constant at 2 layers.

7. Conclusions

We proposed a method to integrate the AMR graphs into popular machine translation architectures such as Seq2-Seq, ConvSeq2Seq, and Transformer. Structured semantic information from AMR graphs can supplement the context information in the translation model for a better representation of abstract information. Experimental results show that AMR graphs yield better results than other representations such as dependency trees or semantic roles.

For future studies, we plan to examine other methods to integrate more complex semantic graphs, such as Prague Semantic Dependencies, Elementary Dependency Structures, and Universal Conceptual Cognitive Annotation, and investigate different encoding methods suitable for a range of semantic graphs.

Appendix

A. Error Analysis

This section presents some translation errors of the proposed model.

In the first example in Table 7, with bi-LSTM-AMR, the model incorrectly predicts the phrase "and in V Magazine" to be "và V là Magazine." Although the translation is incorrect, the model still recognizes "V Magazine" as a proper noun and that V is a magazine ("V là Magazine"). Meanwhile, both ConvSeq2Seq-AMR and Transformer-AMR cannot recognize this pattern and omit the word "Magazine" when translating. Light-ConvSeq2Seq-AMR is the only model that provides a relatively complete translation.

Example 2 in Table 8 illustrates the case in which the model still understands the meaning but selects the wrong representation. The English word "internal" is meant to complement the phrase "combustion engine," which already entailed the meaning of "động cơ đốt trong." In this case, ConvSeq2Seq-AMR and bi-LSTM -AMR has taken "internal" to mean "inside" as an adjective that modifies the location information of the engine and ignores the word "combustion" when translated into Vietnamese. Meanwhile, Light-ConvSeq2Seq-AMR and Transformer-AMR prove a better performance in capturing information, as they produce accurate translations.

Table 9 describes the case in which the model retains the meaning correctly, but the reference data are incorrect. The word "it" is translated to "những thông tin đó" in the data. This is an inaccurate translation because the word "it" refers to a singular entity, while the translation is in the plural form. Besides, there is only one sentence and no information about the surrounding context, so the results obtained from the proposed models are similar to one another. The Vietnamese word "nó" can be used to refer to previously mentioned things or events. It is thus highly ambiguous, causing difficulty in interpreting even for humans.

B. More Illustrative Results

Table 10 illustrates some sample translations of the models: Song's method, bi-LSTM (base model), and bi-LSTM-AMR (proposed model).

Data Availability

The datasets used to support the findings of this study are from https://wit3.fbk.eu/.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Long H. B. Nguyen and Viet H. Pham contributed equally to this work.

Acknowledgments

This research is funded by University of Science, VNU-HCM under grant number CNTT 2020-06.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Conference Track Proceedings 3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 2015.

[2] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *ICML'17*, Sydney, Australia, August 2017.

[3] I. Sutskever, O. Vinyals, and V. L. Quoc, "Sequence to sequence learning with neural networks," in *NIPS'14*, pp. 3104–3112, MIT Press, Cambridge, MA, USA, March 2014.

[4] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*, pp. 6000–6010, Curran Associates Inc., Long Beach, California, USA, May 2017.

[5] M. Bazrafshan and D. Gildea, "Semantic roles for string to tree machine translation," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 419–423, Association for Computational Linguistics, Sofia, Bulgaria, July 2013, https://www.aclweb.org/anthology/P13-2074.

[6] L. Ding and D. Gildea, "Semantic role features for machine translation," in *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pp. 716–724, Coling 2010 Organizing Committee, Beijing, China, August 2010, https://www.aclweb.org/anthology/C10-1081.

[7] D. Wu and P. Fung, "Semantic roles for SMT: a hybrid two-pass model," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, Association for Computational Linguistics, Boulder, Colorado, June 2009, https://www.aclweb.org/anthology/N09-2004.

[8] D. Xiong, M. Zhang, and H. Li, "Modeling the translation of predicate-argument structure for SMT," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics*, pp. 902–911, July 2012, https://www.aclweb.org/anthology/P12-1095.

[9] D. Marcheggiani, J. Bastings, and I. Titov, "Exploiting semantics in neural machine translation with graph convolutional networks," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 486–492, Association for Computational Linguistics, New Orleans, Louisiana, 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] K. Xu, L. Wu, Z. Wang, Y. Feng, and V. Sheinin, "Graph2Seq: graph to sequence learning with attention-based neural networks," 2018, https://arxiv.org/abs/1804.00823.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, 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," 2019, https://arxiv.org/abs/1901.10430.

[14] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS'17*, Curran Associates Inc., Long Beach, CA, USA, May 2017.

[15] M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, R. Cattoni, and M. Federico, "The IWSLT 2015 evaluation campaign," 2015.

[16] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Association for Computational Linguistics, Berlin, Germany, June 2016.

[17] I. Konstas, S. Iyer, M. Yatskar, Y. Choi, and L. Zettlemoyer, "Neural AMR: sequence-to-sequence models for parsing and generation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 146–157, Association for Computational Linguistics, Vancouver, Canada, July 2017.

[18] S. Cai and K. Knight, "Smatch: an evaluation metric for semantic feature structures," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 748–752, Association for Computational Linguistics, Sofia, Bulgaria, July 2013, https://www.aclweb.org/anthology/P13-2131.

[19] M. Ott, S. Edunov, A. Baevski et al., "Fairseq: a fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, Minneapolis, MN, USA, February 2019.

[20] D. P. Kingma and Ba Jimmy, "Adam: a method for stochastic optimization," in *Conference Track Proceedings 3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 2015.

[21] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Association for Computational Linguistics, Philadelphia, PA, USA, July 2001.

[22] P. Koehn, "Statistical significance tests for machine translation evaluation," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 388–395, Association for Computational Linguistics, Barcelona, Spain, July 2004, https://www.aclweb.org/anthology/W04-3250.