Improving Neural Machine Translation with the Abstract Meaning Representation by Combining Graph and Sequence Transformers

Changmao Li and **Jeffrey Flanigan**
University of California, Santa Cruz
{changmao.li,jmflanig}@ucsc.edu

Abstract

Previous studies have shown that the Abstract Meaning Representation (AMR) can improve Neural Machine Translation (NMT). However, there has been little work investigating incorporating AMR graphs into Transformer models. In this work, we propose a novel encoder-decoder architecture which augments the Transformer model with a Heterogeneous Graph Transformer (Yao et al., 2020) which encodes source sentence AMR graphs. Experimental results demonstrate the proposed model outperforms the Transformer model and previous non-Transformer based models on two different language pairs in both the high resource setting and low resource setting. Our source code, training corpus and released models are available at **https://github.com/jlab-nlp/amr-nmt**.

1 Introduction

Neural Machine Translation (NMT, Bahdanau et al. 2015; Vaswani et al. 2017) has proven to be an effective approach, and is the dominant method for machine translation in recent years. However, state-of-the-art NMT methods sometimes repeat words, leave out important pieces of the translation, and hallucinate information not contained in the source, or in other words, fail to accurately capture the semantics of the source in some cases.

To address this problem, researchers have explored incorporating syntactic and semantic information into NMT systems. While most of previous NMT studies incorporating extra information are focused on syntax-based NMT (Stahlberg et al., 2016; Aharoni and Goldberg, 2017; Li et al., 2017; Chen et al., 2017; Bastings et al., 2017; Wu et al., 2017; Chen et al., 2018; Currey and Heafield, 2019; Zhang et al., 2019; Eriguchi et al., 2019; Sundararaman et al., 2019; Zhang et al., 2021; Ni et al., 2021), there has recently been interest in incorporating semantic information into NMT. Marcheggiani et al. (2018) shows that incorporating Semantic Role Labeling (SRL) information can alleviating the "argument switching" problem for NMT. Song et al. (2019) shows that Abstract Meaning Representation (AMR, Banarescu et al. 2013) graphs can be helpful for NMT for the Bi-LSTM with attention. AMR (Banarescu et al., 2013) is a semantic formalism that encodes the meaning of a sentence as a rooted, directed graph. Figure 1 shows an AMR graph, in which the nodes (eg. end-01) represent the concepts, and edges (eg. AGR0) represent the relations between concepts they connect.

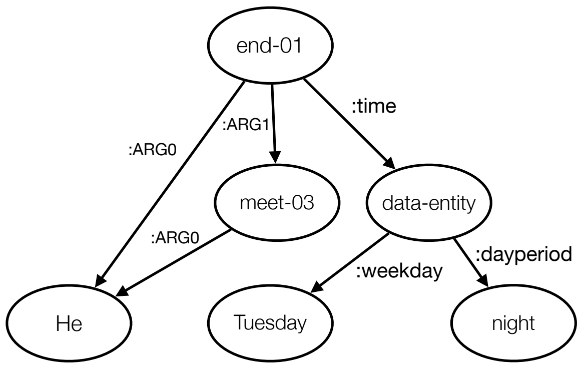


Figure 1: The AMR graph for sentence "He ended his meeting on Tuesday night.".

In prior work, Nguyen et al. (2021) examined the effect of AMRs in different NMT models, proposing a method for incorporating AMR into NMT. However, the method Nguyen et al. (2021) proposed for incorporating AMR into the Transformer showed limited success, as their performance with the Transformer with AMR was less than their Bi-LSTM with AMR.

In this work, we re-examine methods for incorporating AMR graphs into Transformer models. The Transformer (Vaswani et al., 2017) architecture has been the state-of-the-art for NMT for several years. We propose to improve upon the Transformer model by incorporating AMR graphs with a graph Transformer in a novel manner. In particular, we observe the best performance gains when integrating the semantic information contained in the AMR graphs into *both* the encoder and decoder modules of the Transformer.

While much research on Transformers is for text, many researchers have also investigated Transformer-like architectures for the encoding of graph structures. Yao et al. (2020) proposed the Heterogeneous Graph Transformer which independently models the different relations in the individual subgraphs of the original graph, including direct relations, indirect relations and other possible relations between nodes.

We improve the performance of the Transformer by employing a vanilla Transformer to encode and decode the source sentence and a Heterogeneous Graph Transformer to encode an AMR graph of the source sentence. We use a novel integration model to combine the graph representations (§3) into the encoder and decoder. We show that our method improves upon the Transformer, and improves upon the best previous method for incorporating AMR graphs into NMT.

Experiments on the WMT16 English to German dataset and IWSLT15 English to Vietnamese show that incorporating AMR into Transformer models with proper encoding representation combination models can robustly improve the vanilla sequence-to-sequence Transformer baseline and outperforms all previous approaches when incorporating AMR in both low data setting and large data setting.

In summary, our contributions are the following:

- We propose a novel integration encoder-decoder model which combines the sentence representations from the vanilla sequence Transformer and graph representations from Heterogeneous Graph Transformer to better incorporate AMR into machine translation purely using Transformers.

- We introduce two encoder integration methods and two decoder integration methods to combine the two Transformers which enforces the model to combine information from both representations independently and coherently.

- We perform several comparison experiments and results show that our proposed models robustly performs better than both vanilla sequence Transformer and previous baselines which shows that including AMR into machine translation can be more effective by only using Transformer-based models.

2 Background

In this section, we review the original Transformer architecture for sequences as well as the Heterogeneous Graph Transformer, and introduce notation we will use in later sections.

2.1 Transformer

The Transformer (Vaswani et al., 2017) contains several layers, which has a multi-head self-attention layer (Bahdanau et al. 2015; Graves et al. 2014; Weston et al. 2015) followed by a feedforward layer, along with residual connections (He et al., 2016) and layer normalization (Ba et al., 2016) .

Let the input sequence be $S = [s_1, ..., s_L] \in \mathbb{R}^{L\times e}$, where L is the sequence length and e is the hidden size of the attention layer. Queries Q, keys K, and values V used in the self-attention computation are obtained by linearly projecting the input, or the output of the previous layer, X:

$$Q = SW^Q, K = SW^K, V = SW^V, \quad (1)$$

While $W^Q, W^K, W^V \in \mathbb{R}^{e\times e}$ are learnable projection matrices. To perform multi-head self attention, Q, K, and V are split into heads $Q_h, K_h, V_h \in \mathbb{R}^{L\times d}$ for h in $1, ..., H$ where H is the number of heads and $d = e/H$. Then, the context representation $E_h \in \mathbb{R}^{L\times d}$, that corresponds to each attention head h, is obtained by:

$$E_h = \text{softmax}(\frac{Q_h K_h^T}{\sqrt{d}})V_h, \quad (2)$$

Where d is the hidden size dimension of each K_h and the softmax is performed row-wise. The head context representations are concatenated to obtain the final context representation $E_S \in \mathbb{R}^{L\times e}$:

$$E_S = [E_1, ..., E_H]W^R, \quad (3)$$

where $W^R \in \mathbb{R}^{e\times e}$ is another projection matrix that aggregates all head's representations.

2.2 Heterogeneous Graph Transformer

A Heterogeneous Graph Transformer (Yao et al., 2020) is a Transformer-based graph encoder and decoder model. Yao et al. (2020) extends the input transformed Levi graph (Beck et al., 2018) into multiple types of subgraphs (i.e.fully-connected, reverse, etc.) according to its heterogeneity then updating the node representation in different subgraphs based on its neighbor nodes in the current subgraph and finally combining all the representations of this node in different subgraphs to get the graph final representation.

Let the input graph nodes be $G = [g_1, ..., g_N] \in \mathbb{R}^{N \times e}$, where N is the number of nodes and e is the hidden size of the attention layer. Then the output representation of node i in each attention head Z_i is obtained by:

$$Z_i = \sum_{j \in N_i} \alpha_{ij}(g_j W^V) \tag{4}$$

$$\alpha_{ij} = \text{softmax}(\frac{(g_i W^Q)(g_j W^K)^T}{\sqrt{d}}) \tag{5}$$

where $W^V, W^Q, W^K \in \mathbb{R}^{e \times e}$ are layer-specific learnable parameter matrices and α_{ij} represents the attention score of node j to i and $d = e/H$ where H is the number of attention heads. Then the output Z in each encoder layer is obtained by:

$$Z = [Z^{G_1^{\text{sub}}}, ..., Z^{G_M^{\text{sub}}}]W^R \tag{6}$$

$$Z_i^{G_m^{\text{sub}}} = \sum_{j \in N_i^{G_m^{\text{sub}}}} \alpha_{ij}(g_j W^V), m \in [1, M] \tag{7}$$

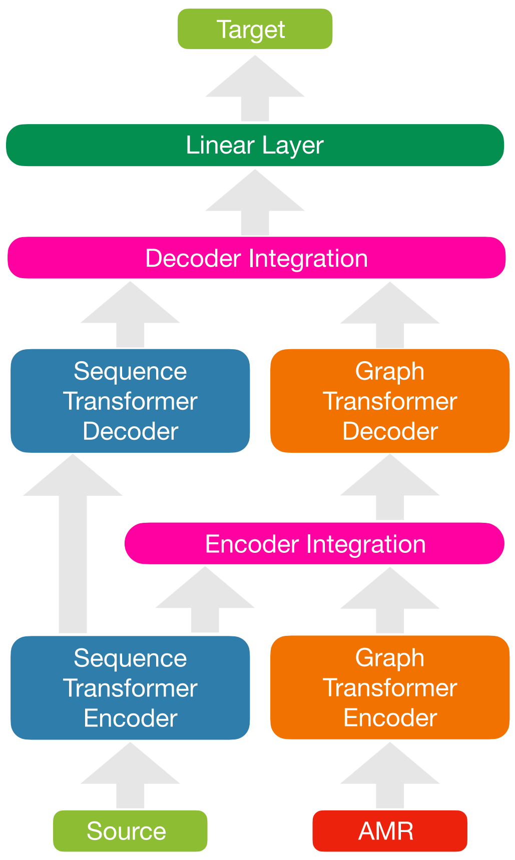
where M is the number of subgraphs, $W^R \in \mathbb{R}^{Me \times e}$, G_m^{sub} is the set of subgraphs including M elements (i.e. $G^{\text{sub}} = \{$*fullyconnected, connected, default, reverse*$\}$) and $N_i^{G_m^{\text{sub}}}$ is the set of neighbors in the m-th subgraph of node i. Finally there is a layer aggregation strategy from Xu et al. (2018) using Jumping Knowledge architecture (Xu et al., 2018), so the final output of the graph representation $E_G \in \mathbb{R}^{N \times e}$ is:

$$E_G = [Z^1, , ..., Z^T]W_{\text{jump}} \tag{8}$$

where $W_{\text{jump}} \in \mathbb{R}^{Le \times e}$ and T is the number of layers including the embedding layer.

3 Our AMR-Transformer Model

Figure 2 shows the overview of our proposed model architecture. To encode and decode both source sentences and source AMR graphs to target sentences, our model consists of two parallel stacked encoder and decoder layers, one for sequence encoding and decoding from the neural sequence to sequence model, and the other for graph encoding and decoding from the neural graph to sequence model. Given the encoded sequence representation from the sequence encoder and the encoded graph representation from the graph encoder, the sequence to sequence decoder only receives the sequence representation while the graph to sequence decoder receives the combination of the sequence representation and the graph representation. The specific combination approaches are discussed in §3.2 and §3.3. Finally, two decoder representations are concatenated and fed into the final linear layer to generate target sequence representation. In this way, the model can combine the advantage of the traditional sequence to sequence model which does translation based on source sentence encodings and the graph to sequence model which incorporates AMR graphs into the translation. The combination of source sentence representation and the graph representation into the graph to sequence decoder can lead the graph to sequence decoder to decoding towards good translation quality since using only AMR graphs representation can lead to poor translation quality compared to the vanilla sequence to sequence model using source sentences.


Target
Linear Layer
Decoder Integration
Sequence Transformer Decoder
Graph Transformer Decoder
Encoder Integration
Sequence Transformer Encoder
Graph Transformer Encoder
Source
AMR
Figure 2: Overview of our AMR-Transformer model.

3.1 Sequence and Graph Encodings

Here we describe our sentence and graph encodings. Let $S = [s_1, ..., s_{L_s}] \in \mathbb{R}^{L_s \times e}$ be the source sentence where s_i is the ith token in S, L_s is the length of the source sentence and e is the hidden size of the encoder. Let $G = [g_1, ...g_N] \in \mathbb{R}^{N \times e}$ be the AMR graph of the source sentence where g_j is the j'th node in G and N is the number of nodes. The source sequence encoding representation E_S is computed by Eq. 3 and the AMR graph encoding representation E_G is computed by Eq. 8.

3.2 Encoder Integration: Multi-head Attention Integration

To integrate the encoder representations for the sequence encode and graph encoder, we employ a multi-head attention mechanism. Figure 3 shows an overview of the multi-head attention (MHA) (Vaswani et al., 2017) integration of the two encoder representations. At a high level, we compute MHA between the source sequence encoding representation E_S and the AMR graph encoding representation E_G, which allows the model to learn correlations between individual tokens and nodes in S and G, s_* and g_*.

Each row in E_S is the representation $E_i^S \in \mathbb{R}^{1 \times e}$ of the corresponding token s_i. Each row in E_G is the representation $E_j^G \in \mathbb{R}^{1 \times e}$ of the corresponding node g_j. These two matrices, E_S and E_G, are fed into two types of multi-head attention (MHA) layers, one finding correlations from S to G (S2G) and the other from G to S (G2S), which generate two attention matrices, $\mathcal{A}^{s2g} \in \mathbb{R}^{L_s \times e}$ and $\mathcal{A}^{g2s} \in \mathbb{R}^{N \times e}$.

$$\mathcal{A}^{s2g} = [h_1^{s2g}, ..., h_H^{s2g}] W^{O_{s2g}} \tag{9}$$

$$h_i^{s2g} = \sigma\left(\frac{E_S W_i^{Q_{s2g}} (E_G W_i^{K_{s2g}})^T}{\sqrt{d}}\right) E_G W_i^{V_{s2g}} \tag{10}$$

H is the number of heads and $d = e/H$. $W_i^{Q_{s2g}}, W_i^{K_{s2g}}, W_i^{V_{s2g}} \in \mathbb{R}^{e \times d}$, $W^{O_{s2g}} \in \mathbb{R}^{N \times e}$ are learned parameters and σ represents softmax.

$$\mathcal{A}^{g2s} = [h_1^{g2s}, ..., h_H^{g2s}] W^{O_{g2s}} \tag{11}$$

$$h_i^{g2s} = \sigma\left(\frac{E_G W_i^{Q_{g2s}} (E_S W_i^{K_{g2s}})^T}{\sqrt{d}}\right) E_S W_i^{V_{g2s}} \tag{12}$$

$W^{O_{g2s}} \in \mathbb{R}^{e \times e}$ and $W_i^{Q_{g2s}}, W_i^{K_{g2s}}, W_i^{V_{g2s}} \in \mathbb{R}^{e \times d}$ are learned parameters and σ is softmax.

Then the graph to sequence decoder input representation $D_{\text{in}}^g \in \mathbb{R}^{(L_s+N) \times e}$ is computed by:

$$D_{\text{in}}^g = [\mathcal{A}^{s2g}, \mathcal{A}^{g2s}] \tag{13}$$

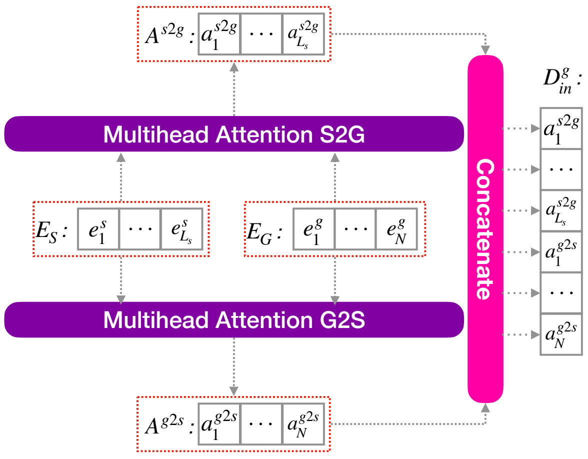

A^{s2g} : a_1^{s2g} ⋯ $a_{L_s}^{s2g}$
Multihead Attention S2G
E_S : e_1^s ⋯ $e_{L_s}^s$
E_G : e_1^g ⋯ e_N^g
Multihead Attention G2S
A^{g2s} : a_1^{g2s} ⋯ a_N^{g2s}
Concatenate
D_{in}^g :
a_1^{s2g} ⋯ $a_{L_s}^{s2g}$ a_1^{g2s} ⋯ a_N^{g2s}

Figure 3: The multi-head attention integration.

3.2.1 Direct Integration

As a baseline, we also experiment with a simpler method of integrating the two encoders, which we call direct integration. Given the previous obtained source sequence encoding representation E_S and AMR graph encoding representation E_G, the graph to sequence decoder input representation $D_{\text{in}}^g \in \mathbb{R}^{(L_s+N) \times e}$ is computed using concatenation:

$$D_{\text{in}}^g = [E_S, E_G] \tag{14}$$

3.3 Decoder Integration

To keep the advantages of the vanilla sequence Transformer, the sequence to sequence decoder input representation D_{in}^s is identical to E_S , then D_{in}^s is fed into the sequence to sequence decoder to obtain the target sentence representation $D_{\text{out}}^s \in \mathbb{R}^{L_t * e}$, where L_t is the length of the target sentence. The previous obtained D_{in}^g which is the graph to sequence decoder input representation is fed into the graph to sequence decoder to obtain the target sentence representation $D_{\text{out}}^g \in \mathbb{R}^{L_t * e}$. Then the final target sentence representation $Z^{target} \in \mathbb{R}^{L_t \times e}$ is obtained by:

$$Z^{target} = (D_{\text{out}}^s + D_{\text{out}}^g) W_e^T + B_e \tag{15}$$

Where $W_e \in \mathbb{R}^{v * e}$ is the embedding weight matrix, $B_e \in \mathbb{R}^{L_t * v}$ is the bias and v is the vocabulary size.

Dataset	Train	Dev	Test	
WMT16 EN-DE NC-V11	238K	3000	2999	
WMT16 EN-DE Full	4.5M	3000	2999	
IWST15 EN-VI	133K	1553	1268	1080

Table 1: The statistics of datasets. EN-DE: English to German; EN-VI: English to Vietnamese. For IWST15 English-Vietnamese, there are two test sets, the left cell in the Test column represents the tst2013 and the right cell in the Test column represents the tst2015.

4 Experiments

4.1 Data and Preprocessing

Following Song et al. (2019), we use the WMT16 English to German dataset[1] in both the news commentary setting (News Commentary v11, NC-V11) and the full data scenario. For all experiments we use newstest2013 and newstest2016 respectively as the development and test sets. To evaluate the model performance on low-resource languages, we also include experiments on IWST15 English to Vietnamese dataset[2]and follow the preprocessing steps described below. For this dataset, we use tst2012 as development set and use tst2013 and tst2015 as test sets following Nguyen et al. (2021). Table 1 shows the number of sentences for training, development and testing splits.

To preprocess the data, we use `Moses`[3] data cleaning and tokenization tools to clean and tokenize all data for both sides. We used Google `sentencepiece`[4] in BPE mode to deal with rare and compound words for both sides and conducted 4000 BPE merges for English-Vietnamese data, 8000 BPE merges for the English-German News Commentary V11 data and 16000 BPE merges for the English-German full data. For the AMR parsing, instead of `JAMR` (Flanigan et al., 2016) used by Song et al. (2019), we employed a recent AMR parser, AMR-gs[5] (Cai and Lam, 2020) to obtain better AMR parsing quality. However we also conducted an AMR parsing ablation experiment using `JAMR` in §5.2 to show comparison of the effect of AMR parsing quality.

4.2 Models

We trained and evaluated the following models on WMT2016 English-German in both subset data setting and full data setting and one real low resource languages and IWST15 English-Vietnamese. Following Nguyen et al. (2021) we also carefully reimplemented and ran their best system which is a non-Transformer based model with our settings to show a fair comparison. We use AMR-Transformer to refer to our proposed model. The models we compare are:

- Vanilla sequence Transformer (Baseline, §2.1)
- AMR-Transformer-DI: Ours with direct integration (§3.2.1)
- AMR-Transformer: Ours with `MHA` integration (§3.2)

We also compared to other Non-Transformer baselines including Dual2seq ((Song et al., 2019)) which leverages the BiLSTM to encode sequences and graph recurrent network (GRN) to encode AMR graphs and an improved version proposed by ((Ni et al., 2021)) which also applies the BiLSTM to encode sequences but employs the graph attention network (GAN) to encode AMR graphs.

4.3 Hyperparameters

We use the Adam optimizer (Kingma and Ba, 2015). The batch size on tokens is set to 4096 with gradient accumulation size 2. Between layers, we apply dropout with a probability of 0.1 for the vanilla sequence Transformer. The best model is selected based on the word accuracy on the development set. BLEU (Papineni et al., 2002), TER (Snover et al., 2006) and Meteor (Denkowski and Lavie, 2014) are used as the metrics on cased and tokenized results. For experiments with WMT16 English-German, both Sequence Transformer and Heterogeneous Graph Transformer word embedding size are 512 and hidden size are 2048, the dropout for the Heterogeneous Graph Transformer part is 0.3 and the models are trained for at most 300000 steps with early stopping and 16000 warm up steps. For experiments with IWST15 English-Vietnamese, the Sequence Transformer word embedding size is 256 and hidden size is 1024, Heterogeneous Graph Transformer embedding size is 256 and hidden size is 512, the dropout for the Heterogeneous Graph Transformer part is 0.8 and the models are trained for at most 120000 steps with early stopping and 2000 warm up steps. All models were trained on either one A40 or A100 GPU.

[1]http://www.statmt.org/wmt16/translation-task.html
[2]https://wit3.fbk.eu/2015-01
[3]http://www.statmt.org/moses/
[4]https://github.com/google/sentencepiece
[5]https://github.com/jcyk/AMR-gs

System on WMT16 English-German	NC-v11					Full				
	BLEU	TER↓	Meteor	PS	GH	BLEU	TER↓	Meteor	PS	GH
Dual2seq (Song et al., 2019)	19.2	63.1	38.4	-	-	25.5	54.8	43.8	-	-
Bi-LSTM -AMR (Nguyen et al. 2021, reimplement)	19.0	66.4	37.5	62M	7h	24.8	58.9	43.1	72M	15h
Vanilla sequence Transformer (§2.1)	20.3	66.3.	39.4	52M	12h	26.0	58.5	45.2	61M	20h
Vanilla sequence Transformer (Double Parameters)	20.9	62.1	40.3	138M	12h	26.2	57.6	45.2	151M	20h
AMR-Transformer-DI (§3.2.1)	21.5	62.7	40.4	117M	16h	26.4	56.7	44.9	132M	28h
AMR-Transformer (§3.2)	**22.1***	**62.0***	**41.1***	117M	16h	**26.5****	**56.4***	**45.2**	133M	28h

Table 2: TEST performance on WMT16 English-German. NC-v11 represents training only with the NC-v11 data, while Full means using the full training data. * represents significant (Koehn, 2004) result ($p < 0.001$) over vanilla sequence Transformer. ** represents significant result ($p < 0.05$) over vanilla sequence Transformer. ↓ indicates lower is better. PS: approximate parameter size. GH: approximate GPU training hours with early stopping.

System on IWST15 English-Vietnamese	PS	GH	tst2013			tst2015		
			BLEU	TER↓	Meteor	BLEU	TER↓	Meteor
Bi-LSTM -AMR (Nguyen et al., 2021)	-	-	29.3	-	-	26.4	-	-
Bi-LSTM -AMR (Nguyen et al. 2021, reimplement)	17M	9h	26.4	56.4	44.1	25.2	60.5	42.1
Vanilla sequence Transformer (§2.1)	13M	5h	30.0	52.1	48.2	27.6	57.6	45.4
Vanilla sequence Transformer (Double parameters)	36M	5h	28.3	54.4	46.4	26.8	59.2	44.2
AMR-Transformer-DI (§3.2.1)	20M	7h	30.2	52.4	48.2	28.2	57.3	45.5
AMR-Transformer (§3.2)	20M	7h	**30.6***	**52.1**	**48.5**	**28.2****	**57.1**	**45.9**

Table 3: TEST performance on IWST15 English-Vietnamese. tst2013 represents the results evaluated on tst2013 and tst2015 represents the results evaluated on tst2015. * represents $p < 0.05$ over vanilla sequence Transformer. ** represents $p < 0.11$ over vanilla sequence Transformer. ↓ indicates lower is better. PS: approximate parameter size. GH: approximate GPU training hours with early stopping.

4.4 Main Results

4.4.1 Results on WMT16 English-German

Table 2 shows the test BLEU, TER and Meteor scores of all systems trained on the small scale News Commentary v11 subset or the large scale full set. The result shows that our Transformer baseline already outperforms all previous non-Transformer based results. Our system using AMR-Transformer whether it is DI or MI are all consistently better than the other systems under all three metrics, showing the effectiveness of the semantic information provided by AMR with Transformers. Particularly, AMR-Transformer is the best performing model for both settings and significantly better than vanilla sequence Transformer baselines under all three metrics. In terms of different settings, our best model shows 1.8 BLEU points improvement over the vanilla sequence Transformer baseline and at least 2.9 BLEU points improvement over the non-Transformer baselines on News Commentary V11 data. For the Full data, the improvement is smaller but our best model is still significantly better than vanilla sequence Transformer baseline in terms of BLEU points and at least 1.0 BLEU points improvement over the non-Transformer baselines. The results show the same conclusion as Song et al. (2019) that AMR graphs helps more on a low resource setting. Our AMR-Transformer model has roughly double the parameters as the baseline Transformer model due to the graph encoder. To show the effectiveness of our approach is not from increasing the parameter size, we conduct experiments on Transformer baselines with doubled parameters. Our approach still shows better performance.

4.4.2 Results on IWST15 English-Vietnamese

Table 3 shows the results of all systems trained on the IWST15 English to Vietnamese data. Our best AMR-Transformer is significantly better than vanilla sequence Transformer on tst2013 and also better than the previous non-Transformer based model. However, the model is not significantly better on tst 2015, which is due to the different data distribution between tst2013 and tst2015. Our experiments also show that adjusting the model dropout rate of Heterogeneous Graph Transformer side when using fixed hyperparameter of the Sequence Transformer side during training can improve the performance since the model dropout rate can control how much AMR information is used to contribute to the final predictions. Our experiments inidcate that a high dropout rate for Heterogeneous Graph Transformer side during low resource settings can enable AMR information help sequence to sequence model better than a low dropout rate.

Ablation on WMT16 English-German	NC-v11			Full		
	BLEU	TER↓	Meteor	BLEU	TER↓	Meteor
AMR-Transformer, No Encoder Integration	20.9	64.1	39.5	26.4	56.5	45.4
AMR-Transformer-DI, Single Decoder	19.9	65.8	36.6	25.5	60.8	42.6
AMR-Transformer, Single Decoder	16.1	72.7	32.3	21.6	65.4	38.7
AMR-Transformer-DI	21.5	62.7	40.4	26.4	56.7	44.9
AMR-Transformer	22.1	62.0	41.1	26.5	56.4	45.2

Table 4: Model ablations TEST performance comparasion on WMT16 English-German. NC-v11 represents training only with the NC-v11 data, while Full means using the full training data.). ↓ indicates lower is better.

The improvement gap between our best model and vanillas Transformer is smaller than the model trained on English to German News commentary V11 data which indicates that the size of the training data in low resource settings takes an effect on how much AMR information can help when incorporating into the sequence to sequence translation models. With more training data when it is in low resource setting, the help of AMR information increases but during high resource setting the help of AMR information decreases. Our AMR-Transformer model has roughly double the parameters as the baseline Transformer model due to the graph encoder. To show the effectiveness of our approach is not because of enlarging the parameter size in this dataset, we also double the parameters of Transformer baselines, and the performance is even lower than the smaller parameters baseline due to the possible over-fitting.

5 Analysis and ablation studies

5.1 Model ablations

To verify the effectiveness of our encoder integration and decoder integration we conduct ablation experiments on WMT16 English-German data. Table 4 shows model ablations test performance. we can see that compared to the best model, the performance drops largely on the both data setting without decoder integration , at least 2.2 BLEU points drop on News Commentary V11 data and at least 2.7 BLEU drop on the full data which indicates the decoder integration have a large contribution to the performance improvement in both data settings. For the encoder integration part, it shows different situations on the two data setttings. On the News Commentary V11 training data, without encoder integration, the BLEU drops 1.2 points while on the full training data, however, the BLEU score does not drop too much which indicates that encoder integration is more helpful in low resource settings. Generally, the drop gap between the best model and without decoder integration is larger than the drop gap between the best model and without encoder integration which indicates that decoder integration is more helpful for the performance improvement than the encoder integration in both data settings.

5.2 Influence of AMR parsing accuracy

To verify the influence of AMR parsing quality we also conduct an experiment on News Commentary V11 dataset using a previous JAMR paser (Flanigan et al., 2016) with the best model. Table 6 shows the result. We can see that with a lower quality AMR parser the BLEU score drops 0.9 points but it is still better than the vanilla sequence Transformer baseline and previous non-Transformer based models, which indicates that the quality of the AMR parser influences the performance of the model. However, even with lower quality AMR parses, our approach can still improve upon the Transformer baseline.

5.3 Case study

We conduct case studies for a better understanding of the model performance. We compare the outputs of the vanilla Transformer baseline and our AMR-Transformer model with multihead attention integration trained on News commentary V11 data. Tables 5 presents these examples. In the first example, the source sentence is in the syntax of "someone said something" and the vanilla Transformer baseline model completely misses this syntax which causes the incorrect translation while our model perfectly kept the original sentence syntax and meaning. In the second example, the vanilla Transformer baseline model incorrectly translate the verb "hold up" into "verteilt" which means "distributed" in German which causes meaning of the sentence entirely different from the source sentence, while our model perfectly translate it the same as the reference sentence which indicates that our model with AMR graphs is helpful for keeping

AMR: (c0 / say-01 :ARG0 (c2 / we) :ARG1 (c1 / take-01 :ARG0 c2 :ARG1 (c4 / they) :ARG3 (c5 / city :name (c7 / name :op1 "passau")) :manner (c6 / car)) :time (c3 / once))
Src: We said at once that we would take them to Passau by car .
Ref: Wir haben gleich gesagt , wir bringen sie mit dem Auto nach Passau .
Vanilla Transformer: Sobald wir sie zu einem Auto nach Passau nehmen würden .
AMR-Transformer: Wir sagten einmal darauf , dass wir sie mit dem Auto Passau nehmen würden .

AMR: (c0 / and :op2 (c1 / hold-up-11 :ARG1 (c2 / number :mod (c4 / this)) :location (c3 / state :mod (c5 / early))))
Src: And these numbers hold up in early states .
Ref: Und diese Zahlen halten sich in frühen Staaten .
Vanilla Transformer: Und diese Zahlen sind in frühen Bundesstaaten verteilt .
AMR-Transformer: Und diese Zahlen halten an frühe Staaten fest .

AMR: (c0 / note-01 :ARG1 (c1 / it) :ARG1-of (c3 / cause-01 :ARG0 (c4 / reason :ARG1-of (c5 / personal-02))) :mod (c2 / too))
Src: It was noteworthy because of personal reasons , too .
Ref: Sie war auch aus persönlichen Gründen bemerkenswert .
Vanilla Transformer: Auch weil es aus persönlichen Gründen bemerkenswert war , war sie beachtenswert .
AMR-Transformer: Sie war auch aufgrund von persönlichen Gründen bemerkenswert .

AMR: (c0 / contrast-01 :ARG1 (c1 / hard-02 :ARG1 (c3 / find-01 :ARG1 (c7 / keep-01)) :mod (c4 / usual) :polarity -) :ARG2 (c2 / possible-01 :ARG1 (c6 / get-03 :ARG1 (c8 / store) :ARG2 (c9 / busy-01 :ARG1 c8))))
Src: While the store can get busy , parking is usually not hard to find .
Ref: Auch wenn der Laden gut besucht ist , ist es nicht schwer , einen Parkplatz zu finden .
Vanilla Transformer: Während sich der Laden mit dem Glücksfall beschäftigt , ist es normalerweise nicht schwer, einen Parking zu finden .
AMR-Transformer: Während der Stur Busy bekommen kann , ist Parking normalerweise nicht schwer zu finden .

Table 5: Sample system outputs

Ablation with JAMR	NC-v11		
	BLEU	TER↓	Meteor
AMR-Transformer w/ JAMR	21.2	64.4	40.3
AMR-Transformer w/ AMR-gs	22.1	62.0	41.1

Table 6: TEST performance on WMT16 English to German NC-v11 using two different AMR pasers with the best model. ↓ indicates lower is better.

the meaning of the verbs. In the third example, the vanilla sequence Transformer repeatedly translates the same meaning twice while our model correctly translate it only once which indicates our model can avoid the repetition of tokens in the same meaning. However, there are situations that our model performs badly. In the forth example, our model incorrectly translates the noun word "store" while the vanilla Transformer baseline translate it correctly which indicates that AMRs may not be helpful when translating nouns.

Generally from our observations, with the AMR incorporated with our proposed model, although there may be a problem for translation of nouns, our system can more correctly translate the key verbs, more easily keep the same sentence syntax with the source sentence and avoid repetitions which are enable the NMT system more easily to keep the source sentence meaning and generate a better translation quality.

6 Related Work

Several recent studies have investigated on how to incorporate semantic information into neural machine translation (NMT) models. Marcheggiani et al. (2018) studied the semantic role labeling (SRL) information for NMT, which used graph convolutional network (GCN) layers to encode the predicate-argument structure from SRL to improve the translation performance of the NMT model. In line with their work, Song et al. (2019) was the first to exploit the AMR information on NMT, which used a graph recurrent network to encode the AMR graph and found that AMR information can improve attention-based sequence to sequence neural translation model and they only evaluated their model on WMT16 English to German dataset. Nguyen et al. (2021) then examine the effect of AMR in different sequence to sequence machine translation models, however, they found that their proposed single decoder Transformer model to incorporate the AMR information performs worse than the Bi-LSTM model with simple graph attention network. In this paper, we focus on improving the performance of incorporating AMR information purely with Transformers. Our proposed method of integrating vanilla sequence Transformer and Heterogeneous Graph Transformer model with encoder integration and decoder integration provides a better way to incorporate the AMR information into NMT.

7 Conclusion

We combine the Transformer and the Heterogeneous Graph Transformer to incorporate semantics captured in AMR graphs into neural machine translation. Experimental results show that our proposed AMR-Transformer model robustly outperforms the vanilla sequence Transformer baseline and previous non-Transformer based models across two different language pairs in both high resource setting and low resource setting.

Acknowledgements

J.F. was supported in part by the NSF National AI Institute for Student-AI Teaming (iSAT) under grant DRL 2019805. The opinions expressed are those of the authors and do not represent views of the NSF. We are thankful for the computing resources provided by the Pacific Research Platform's Nautilus cluster, supported by the NSF under Award Numbers CNS-1730158, ACI-1540112, ACI1541349, OAC-1826967, the University of California Office of the President, and the University of California San Diego's California Institute for Telecommunications and Information Technology/Qualcomm Institute.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140, Vancouver, Canada. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1936–1945, Vancouver, Canada. Association for Computational Linguistics.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2018. Syntax-directed attention for neural machine translation. In *AAAI*.

Anna Currey and Kenneth Heafield. 2019. Incorporating source syntax into transformer-based neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 24–33, Florence, Italy. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2019. Incorporating Source-Side Phrase Structures into Neural Machine Translation. *Computational Linguistics*, 45(2):267–292.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206, San Diego, California. Association for Computational Linguistics.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR*, abs/1410.5401.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–697, Vancouver, Canada. Association for Computational Linguistics.

Diego Marcheggiani, Jasmijn Bastings, and Ivan Titov. 2018. 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)*, pages 486–492, New Orleans, Louisiana. Association for Computational Linguistics.

Long H. B. Nguyen, Viet H. Pham, and Dien Dinh. 2021. Improving neural machine translation with amr semantic graphs. *Mathematical Problems in Engineering*, 2021:1–12.

Bin Ni, Xiaolei Lu, and Yiqi Tong. 2021. Synxlm-r: Syntax-enhanced xlm-r in translation quality estimation. In *Natural Language Processing and Chinese Computing*, pages 27–40, Cham. Springer International Publishing.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. *Transactions of the Association for Computational Linguistics*, 7:19–31.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically guided neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 299–305, Berlin, Germany. Association for Computational Linguistics.

Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. 2019. Syntax-infused transformer and BERT models for machine translation and natural language understanding. *CoRR*, abs/1911.06156.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Shuangzhi Wu, Ming Zhou, and Dongdong Zhang. 2017. Improved neural machine translation with source syntax. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4179–4185.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *ICML*.

Shaowei Yao, Tianming Wang, and Xiaojun Wan. 2020. Heterogeneous graph transformer for graph-to-sequence learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7145–7154, Online. Association for Computational Linguistics.

Meishan Zhang, Zhenghua Li, Guohong Fu, and Min Zhang. 2019. Syntax-enhanced neural machine translation with syntax-aware word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1151–1161, Minneapolis, Minnesota. Association for Computational Linguistics.

Tianfu Zhang, Heyan Huang, Chong Feng, and Longbing Cao. 2021. Self-supervised bilingual syntactic alignment for neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14454–14462.