Semantic Neural Machine Translation Using AMR

Linfeng Song,[1] **Daniel Gildea,**[1] **Yue Zhang,**[2] **Zhiguo Wang,**[3] and **Jinsong Su**[4]

[1]Department of Computer Science, University of Rochester, Rochester, NY 14627
[2]School of Engineering, Westlake University, China
[3]IBM T.J. Watson Research Center, Yorktown Heights, NY 10598
[4]Xiamen University, Xiamen, China
[1]{lsong10,gildea}@cs.rochester.edu [2]yue.zhang@wias.org.cn
[3]zgw.tomorrow@gmail.com [4]jssu@xmu.edu.cn

Abstract

It is intuitive that semantic representations can be useful for machine translation, mainly because they can help in enforcing meaning preservation and handling data sparsity (many sentences correspond to one meaning) of machine translation models. On the other hand, little work has been done on leveraging semantics for neural machine translation (NMT). In this work, we study the usefulness of AMR (abstract meaning representation) on NMT. Experiments on a standard English-to-German dataset show that incorporating AMR as additional knowledge can significantly improve a strong attention-based sequence-to-sequence neural translation model.

1 Introduction

It is intuitive that semantic representations ought to be relevant to machine translation, given that the task is to produce a target language sentence with the same meaning as the source language input. Semantic representations formed the core of the earliest symbolic machine translation systems, and have been applied to statistical but non-neural systems as well.

Leveraging syntax for neural machine translation (NMT) has been an active research topic (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). On the other hand, exploring semantics for NMT has so far received relatively little attention. Recently, Marcheggiani et al. (2018) exploited semantic role labeling (SRL) for NMT, showing that the predicate–argument information from SRL can improve the performance of an attention-based sequence-to-sequence model by alleviating the "argument switching" problem,[1] one frequent and severe issue faced by NMT systems (Isabelle et al., 2017). Figure 1(a) shows one example of semantic role information, which only captures the relations between a predicate (*gave*) and its arguments (*John*, *wife*, and *present*). Other important information, such as the relation between *John* and *wife*, cannot be incorporated.

In this paper, we explore the usefulness of abstract meaning representation (AMR) (Banarescu et al., 2013) as a semantic representation for NMT. AMR is a semantic formalism that encodes the meaning of a sentence as a rooted, directed graph. Figure 1(b) shows an AMR graph, in which the nodes (such as *give-01* and *John*) represent the concepts and edges (such as *:ARG0* and *:ARG1*) represent the relations between concepts they connect. Comparing with semantic roles, AMRs capture more relations, such as the relation between *John* and *wife* (represented by the subgraph within dotted lines). In addition, AMRs directly capture entity relations and abstract away inflections and function words. As a result, they can serve as a source of knowledge for machine translation that is orthogonal to the textual input. Furthermore, structural information from AMR graphs can help reduce data sparsity when training data is not sufficient for large-scale training.

Recent advances in AMR parsing keep pushing the boundary of state-of-the-art performance (Flanigan et al., 2014; Artzi et al., 2015; Pust et al., 2015; Peng et al., 2015; Flanigan et al., 2016; Buys and Blunsom, 2017; Konstas et al., 2017; Wang and Xue, 2017; Lyu and Titov, 2018;

[1]That is, flipping arguments corresponding to different roles.

Transactions of the Association for Computational Linguistics, vol. 7, pp. 19–31, 2019. Action Editor: Philipp Koehn.
Submission batch: 6/2018; Revision batch: 10/2018; Published 3/2019.

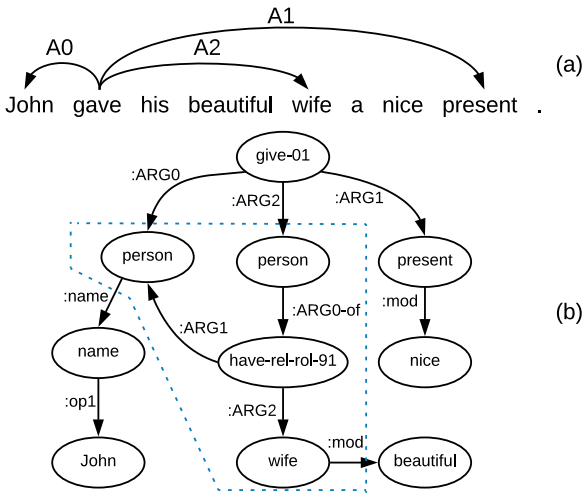

Figure 1: (a) A sentence with semantic roles annotations; (b) the corresponding AMR graph of that sentence.

Peng et al., 2018; Groschwitz et al., 2018; Guo and Lu, 2018), and have made it possible for automatically generated AMRs to benefit downstream tasks, such as question answering (Mitra and Baral, 2015), summarization (Takase et al., 2016), and event detection (Li et al., 2015a). However, to our knowledge, no existing work has exploited AMR for enhancing NMT.

We fill in this gap, taking an attention-based sequence-to-sequence system as our baseline, which is similar to Bahdanau et al. (2015). To leverage knowledge within an AMR graph, we adopt a graph recurrent network (GRN) (Song et al., 2018; Zhang et al., 2018) as the AMR encoder. In particular, a full AMR graph is considered as a single state, with nodes in the graph being its substates. State transitions are performed on the graph recurrently, allowing substates to exchange information through edges. At each recurrent step, each node advances its current state by receiving information from the current states of its adjacent nodes. Thus, with increasing numbers of recurrent steps, each word receives information from a larger context. Figure 3 shows the recurrent transition, where each node works simultaneously. Compared with other methods for encoding AMRs (Konstas et al., 2017), GRN keeps the original graph structure, and thus no information is lost (Song et al., 2018). For the decoding stage, two separate attention mechanisms are adopted in the AMR encoder and sequential encoder, respectively.

Experiments on WMT16 English-to-German data (4.17M) show that adopting AMR significantly improves a strong attention-based sequence-to-sequence baseline (25.5 vs 23.7 BLEU). When trained with small-scale (226K) data, the improvement increases (19.2 vs 16.0 BLEU), which shows that the structural information from AMR can alleviate data sparsity when training data are not sufficient. To our knowledge, we are the first to investigate AMR for NMT.

Our code and parallel data (training/dev/test) with automatically parsed AMRs are available at https://github.com/freesunshine0316/semantic-nmt.

2 Related Work

Most previous work on exploring semantics for statistical machine translation (SMT) studies the usefulness of predicate–argument structure from semantic role labeling (Wong and Mooney, 2006; Wu and Fung, 2009; Liu and Gildea, 2010; Baker et al., 2012). Jones et al. (2012) first convert Prolog expressions into graphical meaning representations, leveraging synchronous hyperedge replacement grammar to parse the input graphs while generating the outputs. Their graphical meaning representation is different from AMR under a strict definition, and their experimental data are limited to 880 sentences. We are the first to investigate AMR on a large-scale machine translation task.

Recently, Marcheggiani et al. (2018) investigated SRL on NMT. The predicate–argument structures are encoded via graph convolutional network (GCN) layers (Kipf and Welling, 2017), which are laid on top of regular BiRNN or CNN layers. Our work is in line with exploring semantic information, but different in exploiting AMR rather than SRL for NMT. In addition, we leverage a GRN (Song et al., 2018; Zhang et al., 2018) for modeling AMRs rather than GCN, which is formally consistent with the RNN sentence encoder. Since there is no one-to-one correspondence between AMR nodes and source words, we adopt a doubly attentive LSTM decoder, which is another major difference from Marcheggiani et al. (2018).

GRNs have recently been used to model graph structures in NLP tasks. In particular, Zhang et al. (2018) use a GRN model to represent raw sentences by building a graph structure of neighboring words and a sentence-level node, showing that the encoder outperforms BiLSTMs

and Transformer (Vaswani et al., 2017) on classification and sequence labeling tasks; Song et al. (2018) build a GRN for encoding AMR graphs for text generation, showing that the representation is superior compared to BiLSTM on serialized AMR. We extend Song et al. (2018) by investigating the usefulness of AMR for neural machine translation. To our knowledge, we are the first to use GRN for machine translation.

In addition to GRNs and GCNs, there have been other graph neural networks, such as graph gated neural network (GGNN) (Li et al., 2015b; Beck et al., 2018). Because our main concern is to empirically investigate the effectiveness of AMR for NMT, we leave it to future work to compare GCN, GGNN, and GRN for our task.

3 Baseline: Attention-Based BiLSTM

We take the attention-based sequence-to-sequence model of Bahdanau et al. (2015) as the baseline, but use LSTM cells (Hochreiter and Schmidhuber, 1997) instead of GRU cells (Cho et al., 2014).

3.1 BiLSTM Encoder

The encoder is a bidirectional LSTM on the source side. Given a sentence, two sequences of states $[\overleftarrow{\boldsymbol{h}}_1, \overleftarrow{\boldsymbol{h}}_2, \ldots, \overleftarrow{\boldsymbol{h}}_N]$ and $[\overrightarrow{\boldsymbol{h}}_1, \overrightarrow{\boldsymbol{h}}_2, \ldots \overrightarrow{\boldsymbol{h}}_N]$ are generated for representing the input word sequence $x_1, x_2, \ldots, x_N$ in the right-to-left and left-to-right directions, respectively, where for each word x_i,

$$\overleftarrow{\boldsymbol{h}}_i = \text{LSTM}(\overleftarrow{\boldsymbol{h}}_{i+1}, \boldsymbol{e}_{x_i})$$
$$\overrightarrow{\boldsymbol{h}}_i = \text{LSTM}(\overrightarrow{\boldsymbol{h}}_{i-1}, \boldsymbol{e}_{x_i})$$

$\boldsymbol{e}_{x_i}$ is the embedding of word x_i.

3.2 Attention-Based Decoder

The decoder yields a word sequence in the target language $y_1, y_2, \ldots, y_M$ by calculating a sequence of hidden states $\boldsymbol{s}_1, \boldsymbol{s}_2 \ldots, \boldsymbol{s}_M$ recurrently. We use an attention-based LSTM decoder (Bahdanau et al., 2015), where the attention memory ($\boldsymbol{H}$) is the concatenation of the attention vectors among all source words. Each attention vector $\boldsymbol{h}_i$ is the concatenation of the encoder states of an input token in both directions ($\overleftarrow{\boldsymbol{h}}_i$ and $\overrightarrow{\boldsymbol{h}}_i$):

$$\boldsymbol{h}_i = [\overleftarrow{\boldsymbol{h}}_i; \overrightarrow{\boldsymbol{h}}_i]$$
$$\boldsymbol{H} = [\boldsymbol{h}_1; \boldsymbol{h}_2; \ldots; \boldsymbol{h}_N].$$

N is the number of source words.

While generating the m-th word, the decoder considers four factors: (1) the attention memory $\boldsymbol{H}$; (2) the previous hidden state of the LSTM model $\boldsymbol{s}_{m-1}$; (3) the embedding of the current input (previously generated word) $\boldsymbol{e}_{y_m}$; and (4) the previous context vector $\boldsymbol{\zeta}_{m-1}$ from attention memory $\boldsymbol{H}$. When $m = 1$, we initialize $\boldsymbol{\zeta}_0$ as a zero vector, set $\boldsymbol{e}_{y_1}$ to the embedding of sentence start token "<s>", and calculate $\boldsymbol{s}_0$ from the last step of the encoder states via a dense layer:

$$\boldsymbol{s}_0 = \boldsymbol{W}_1[\overleftarrow{\boldsymbol{h}}_0; \overrightarrow{\boldsymbol{h}}_N] + \boldsymbol{b}_1,$$

where $\boldsymbol{W}_1$ and $\boldsymbol{b}_1$ are model parameters.

For each decoding step m, the decoder feeds the concatenation of the embedding of the current input $\boldsymbol{e}_{y_m}$ and the previous context vector $\boldsymbol{\zeta}_{m-1}$ into the LSTM model to update its hidden state:

$$\boldsymbol{s}_m = \text{LSTM}(\boldsymbol{s}_{m-1}, [\boldsymbol{e}_{y_m}; \boldsymbol{\zeta}_{m-1}]).$$

Then the attention probability $\alpha_{m,i}$ on the attention vector $\boldsymbol{h}_i \in \boldsymbol{H}$ for the current decode step is calculated as:

$$\epsilon_{m,i} = \boldsymbol{v}_2^\intercal \tanh(\boldsymbol{W}_h \boldsymbol{h}_i + \boldsymbol{W}_s \boldsymbol{s}_m + \boldsymbol{b}_2)$$
$$\alpha_{m,i} = \frac{\exp(\epsilon_{m,i})}{\sum_{j=1}^{N} \exp(\epsilon_{m,j})}.$$

$\boldsymbol{W}_h$, $\boldsymbol{W}_s$, $\boldsymbol{v}_2$, and $\boldsymbol{b}_2$ are model parameters. The new context vector $\boldsymbol{\zeta}_m$ is calculated via

$$\boldsymbol{\zeta}_m = \sum_{i=1}^{N} \alpha_{m,i} \boldsymbol{h}_i.$$

The output probability distribution over the target vocabulary at the current state is calculated by

$$\boldsymbol{P}_{vocab} = \text{softmax}(\boldsymbol{V}_3[\boldsymbol{s}_m, \boldsymbol{\zeta}_m] + \boldsymbol{b}_3), \quad (1)$$

where $\boldsymbol{V}_3$ and $\boldsymbol{b}_3$ are learnable parameters.

4 Incorporating AMR

Figure 2 shows the overall architecture of our model, which adopts a BiLSTM (bottom left) and our graph recurrent network (GRN)[2] (bottom right) for encoding the source sentence and AMR, respectively. An attention-based LSTM decoder is used to generate the output sequence in the target language, with attention models over both the sequential encoder and the graph encoder. The

[2]We show the advantage of our graph encoder by comparing with another popular method for encoding AMRs in Section 6.3.

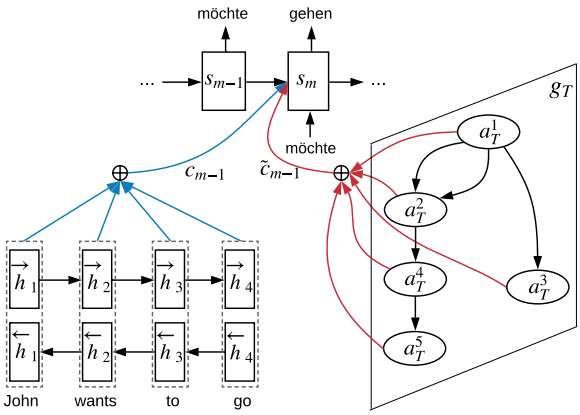

Figure 2: Overall architecture of our model.

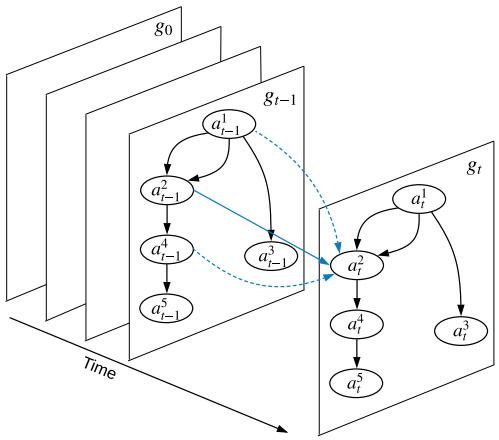

Figure 3: Architecture of the graph recurrent network.

attention memory for the graph encoder is from the last step of the graph state transition process, which is shown in Figure 3.

4.1 Encoding AMR with GRN

Figure 3 shows the overall structure of our graph recurrent network for encoding AMR graphs, which follows Song et al. (2018). Formally, given an AMR graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, we use a hidden state vector $\boldsymbol{a}^j$ to represent each node $v_j \in \boldsymbol{V}$. The state of the graph can thus be represented as:

$$\boldsymbol{g} = \{\boldsymbol{a}^j\}|_{v_j \in \boldsymbol{V}}.$$

In order to capture non-local interaction between nodes, information exchange between nodes is executed through a sequence of state transitions, leading to a sequence of states $\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_T$, where $\boldsymbol{g}_t = \{\boldsymbol{a}_t^j\}|_{v_j \in \boldsymbol{V}}$, and T is the number of state transitions, which is a hyperparameter. The initial state $\boldsymbol{g}_0$ consists of a set of initial node states $\boldsymbol{a}_0^j = \boldsymbol{a}_0$, where $\boldsymbol{a}_0$ is a vector of all zeros.

A recurrent neural network is used to model the state transition process. In particular, the transition from $\boldsymbol{g}_{t-1}$ to $\boldsymbol{g}_t$ consists of a hidden state transition for each node (such as from $\boldsymbol{a}_{t-1}^j$ to $\boldsymbol{a}_t^j$), as shown in Figure 3. At each state transition step t, our model conducts direct communication between a node and all nodes that are directly connected to the node. To avoid gradient diminishing or bursting, LSTM (Hochreiter and Schmidhuber, 1997) is adopted, where a cell $\boldsymbol{c}_t^j$ is taken to record memory for $\boldsymbol{a}_t^j$. We use an input gate $\boldsymbol{i}_t^j$, an output gate $\boldsymbol{o}_t^j$, and a forget gate $\boldsymbol{f}_t^j$ to control information flow from the inputs and to the output $\boldsymbol{a}_t^j$.

The inputs include representations of edges that are connected to v_j, where v_j can be either the source or the target of the edge. We define each edge as a triple (i, j, l), where i and j are indices of the source and target nodes, respectively, and l is the edge label. $\boldsymbol{x}_{i,j}^l$ is the representation of edge (i, j, l), detailed in Section 4.1.1. The inputs for v_j are grouped into incoming and outgoing edges before being summed up:

$$\boldsymbol{\phi}_j = \sum_{(i,j,l) \in \boldsymbol{E}_{in}(j)} \boldsymbol{x}_{i,j}^l$$

$$\hat{\boldsymbol{\phi}}_j = \sum_{(j,k,l) \in \boldsymbol{E}_{out}(j)} \boldsymbol{x}_{j,k}^l$$

where $\boldsymbol{E}_{in}(j)$ and $\boldsymbol{E}_{out}(j)$ are the sets of incoming and outgoing edges of v_j, respectively.

In addition to edge inputs, our model also takes the hidden states of the incoming and outgoing neighbors of each node during a state transition. Taking v_j as an example, the states of its incoming and outgoing neighbors are summed up before being passed to the cell and gate nodes:

$$\boldsymbol{\psi}_j = \sum_{(i,j,l) \in \boldsymbol{E}_{in}(j)} \boldsymbol{a}_{t-1}^i$$

$$\hat{\boldsymbol{\psi}}_j = \sum_{(j,k,l) \in \boldsymbol{E}_{out}(j)} \boldsymbol{a}_{t-1}^k.$$

Based on the above definitions of $\boldsymbol{\phi}_j$, $\hat{\boldsymbol{\phi}}_j$, $\boldsymbol{\psi}_j$, and $\hat{\boldsymbol{\psi}}_j$, the state transition from $\boldsymbol{g}_{t-1}$ to $\boldsymbol{g}_t$, as represented by $\boldsymbol{a}_t^j$, can be defined as:

$$\boldsymbol{i}_t^j = \sigma(\boldsymbol{W}_i \boldsymbol{\phi}_j + \hat{\boldsymbol{W}}_i \hat{\boldsymbol{\phi}}_j + \boldsymbol{U}_i \boldsymbol{\psi}_j + \hat{\boldsymbol{U}}_i \hat{\boldsymbol{\psi}}_j + \boldsymbol{b}_i)$$
$$\boldsymbol{o}_t^j = \sigma(\boldsymbol{W}_o \boldsymbol{\phi}_j + \hat{\boldsymbol{W}}_o \hat{\boldsymbol{\phi}}_j + \boldsymbol{U}_o \boldsymbol{\psi}_j + \hat{\boldsymbol{U}}_o \hat{\boldsymbol{\psi}}_j + \boldsymbol{b}_o)$$
$$\boldsymbol{f}_t^j = \sigma(\boldsymbol{W}_f \boldsymbol{\phi}_j + \hat{\boldsymbol{W}}_f \hat{\boldsymbol{\phi}}_j + \boldsymbol{U}_f \boldsymbol{\psi}_j + \hat{\boldsymbol{U}}_f \hat{\boldsymbol{\psi}}_j + \boldsymbol{b}_f)$$
$$\boldsymbol{u}_t^j = \sigma(\boldsymbol{W}_u \boldsymbol{\phi}_j + \hat{\boldsymbol{W}}_u \hat{\boldsymbol{\phi}}_j + \boldsymbol{U}_u \boldsymbol{\psi}_j + \hat{\boldsymbol{U}}_u \hat{\boldsymbol{\psi}}_j + \boldsymbol{b}_u)$$
$$\boldsymbol{c}_t^j = \boldsymbol{f}_t^j \odot \boldsymbol{c}_{t-1}^j + \boldsymbol{i}_t^j \odot \boldsymbol{u}_t^j$$
$$\boldsymbol{a}_t^j = \boldsymbol{o}_t^j \odot \tanh(\boldsymbol{c}_t^j),$$

where $\boldsymbol{i}_t^j$, $\boldsymbol{o}_t^j$, and $\boldsymbol{f}_t^j$ are the input, output, and forget gates mentioned earlier. $\boldsymbol{W}_x$, $\hat{\boldsymbol{W}}_x$, $\boldsymbol{U}_x$, $\hat{\boldsymbol{U}}_x$, $\boldsymbol{b}_x$, where $x \in \{i, o, f, u\}$, are model parameters.

With this state transition mechanism, information of each node is propagated to all its neighboring nodes after each step. So after several transition steps, each node state contains the information of a large context, including its ancestors, descendants, and siblings. For the worst case where the input graph is a chain of nodes, the maximum number of steps necessary for information from one arbitrary node to reach another is equal to the size of the graph. We experiment with different numbers of transition steps to study the effectiveness of global encoding.

4.1.1 Input Representation

The edges of an AMR graph contain labels, which represent relations between the nodes they connect, and are thus important for modeling the graphs. The representation for each edge (i, j, l) is defined as:

$$\boldsymbol{x}_{i,j}^l = \boldsymbol{W}_4\Big([\boldsymbol{e}_l; \boldsymbol{e}_{v_i}]\Big) + \boldsymbol{b}_4,$$

where $\boldsymbol{e}_l$ and $\boldsymbol{e}_i$ are the embeddings of edge label l and source node v_i, and $\boldsymbol{W}_4$ and $\boldsymbol{b}_4$ are model parameters.

4.2 Incorporating AMR Information with a Doubly Attentive Decoder

There is no one-to-one correspondence between AMR nodes and source words. To incorporate additional knowledge from an AMR graph, an external attention model is adopted over the baseline model. In particular, the attention memory from the AMR graph is the last graph state $\boldsymbol{g}_T = \{\boldsymbol{a}_T^j\}|_{v_j \in \boldsymbol{V}}$. In addition, the contextual vector based on the graph state is calculated as:

$$\tilde{\epsilon}_{m,i} = \tilde{\boldsymbol{v}}_2^\intercal \tanh(\boldsymbol{W}_a \boldsymbol{a}_T^i + \tilde{\boldsymbol{W}}_s \boldsymbol{s}_m + \tilde{\boldsymbol{b}}_2)$$
$$\tilde{\alpha}_{m,i} = \frac{\exp(\tilde{\epsilon}_{m,i})}{\sum_{j=1}^N \exp(\tilde{\epsilon}_{m,j})}.$$

$\boldsymbol{W}_a$, $\tilde{\boldsymbol{W}}_s$, $\tilde{\boldsymbol{v}}_2$, and $\tilde{\boldsymbol{b}}_2$ are model parameters. The new context vector $\tilde{\boldsymbol{\zeta}}_m$ is calculated via $\sum_{i=1}^N \tilde{\alpha}_{m,i} \boldsymbol{a}_T^i$. Finally, $\tilde{\boldsymbol{\zeta}}_m$ is incorporated into the calculation of the output probability distribution over the target vocabulary (previously defined in Equation 1):

$$\boldsymbol{P}_{vocab} = \text{softmax}(\boldsymbol{V}_3[\boldsymbol{s}_m, \boldsymbol{\zeta}_m, \tilde{\boldsymbol{\zeta}}_m] + \boldsymbol{b}_3). \quad (2)$$

5 Training

Given a set of training instances $\{(\boldsymbol{X}^{(1)}, \boldsymbol{Y}^{(1)}), (\boldsymbol{X}^{(2)}, \boldsymbol{Y}^{(2)}), \dots\}$, we train our models using the cross-entropy loss over each gold-standard target sequence $\boldsymbol{Y}^{(j)} = y_1^{(j)}, y_2^{(j)}, \dots, y_M^{(j)}$:

$$l = -\sum_{m=1}^{M} \log p(y_m^{(j)} | y_{m-1}^{(j)}, \dots, y_1^{(j)}, \boldsymbol{X}^{(j)}; \boldsymbol{\theta}).$$

$\boldsymbol{X}^{(j)}$ represents the inputs for the jth instance, which is a source sentence for our baseline, or a source sentence paired with an automatically parsed AMR graph for our model. $\boldsymbol{\theta}$ represents the model parameters.

6 Experiments

We empirically investigate the effectiveness of AMR for English-to-German translation.

6.1 Setup

Data We use the WMT16[3] English-to-German dataset, which contains around 4.5 million sentence pairs for training. In addition, we use a subset of the full dataset (News Commentary v11 [NC-v11], containing around 243,000 sentence pairs) for development and additional experiments. For all experiments, we use newstest2013 and newstest2016 as the development and test sets, respectively.

To preprocess the data, the tokenizer from Moses[4] is used to tokenize both the English and German sides. The training sentence pairs where either side is longer than 50 words are filtered out after tokenization. To deal with rare and compound words, byte-pair encoding (BPE)[5] (Sennrich et al., 2016) is applied to both sides. In particular, 8,000 and 16,000 BPE merges are used on the News Commentary v11 subset and the full training set, respectively. On the other hand, JAMR[6] (Flanigan et al., 2016) is adopted to parse the English sentences into AMRs before BPE is applied. The statistics of the training data and vocabularies after preprocessing are shown in Tables 1 and 2, respectively. For the experiments with the full training set, we used the top 40K

[3] http://www.statmt.org/wmt16/translation-task.html.
[4] http://www.statmt.org/moses/.
[5] https://github.com/rsennrich/subword-nmt.
[6] https://github.com/jflanigan/jamr.

Dataset	#Sent.	#Tok. (EN)	#Tok. (DE)
NC-v11	226K	6.4M	7.3M
Full	4.17M	109M	118M
News2013	3000	84.7K	95.6K
News2016	2999	88.1K	98.8K

Table 1: Statistics of the dataset. Numbers of tokens are after BPE processing.

Dataset	EN-ori	EN	AMR	DE
NC-v11	79.8K	8.4K	36.6K	8.3K
Full	874K	19.3K	403K	19.1K

Table 2: Sizes of vocabularies. *EN-ori* represents original English sentences without BPE.

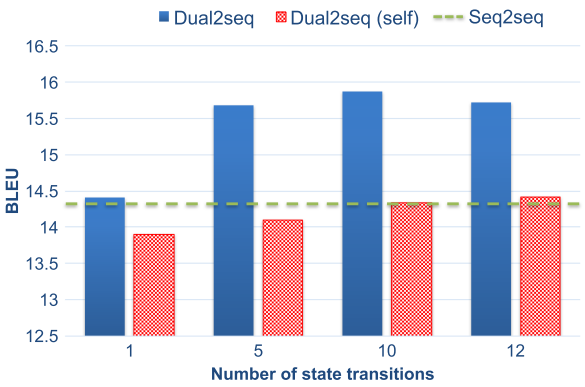

Figure 4: DEV BLEU scores against transition steps for the graph encoders. The state transition is not applicable to *Seq2seq*, so we draw a dashed line to represent its performance.

of the AMR vocabulary, which covers more than 99.6% of the training set.

For our dependency-based and SRL-based baselines (which will be introduced in **Baseline Systems**), we choose Stanford CoreNLP Manning et al. (2014) and IBM SIRE to generate dependency trees and semantic roles, respectively. Since both dependency trees and semantic roles are based on the original English sentences without BPE, we used the top 100K frequent English words, which cover roughly 99.0% of the training set.

Hyperparameters We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0005. The batch size is set to 128. Between layers, we apply dropout with a probability of 0.2. The best model is picked based on the cross-entropy loss on the development set. For model hyperparameters, we set the graph state transition number to 10 according to development experiments. Each node takes information from at most six neighbors. 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 the NC-v11 subset, both word embedding and hidden vector sizes are set to 500, and the models are trained for at most 30 epochs. For experiments with full training set, the word embedding and hidden state sizes are set to 800, and our models are trained for at most 10 epochs. For all systems, the word embeddings are randomly initialized and updated during training.

Baseline Systems We compare our model with the following systems. *Seq2seq* represents our attention-based LSTM baseline (Section 3), and *Dual2seq* is our model, which takes both a sequential and a graph encoder and adopts a doubly attentive decoder (Section 4). To show the merit of AMR, we further contrast our model with the following baselines, all of which adopt the same doubly attentive framework with a BiLSTM for encoding BPE-segmented source sentences: *Dual2seq-LinAMR* uses another BiLSTM for encoding linearized AMRs. *Dual2seq-Dep* and *Dual2seq-SRL* adopt our graph recurrent network to encode original source sentences with dependency and semantic role annotations, respectively. The three baselines are useful for contrasting different methods of encoding AMRs and for comparing AMRs with other popular structural information for NMT.

We also compare with Transformer (Vaswani et al., 2017) and OpenNMT (Klein et al., 2017), trained on the same dataset and with the same set of hyperparameters as our systems. In particular, we compare with *Transformer-tf*, one popular implementation[7] of Transformer based on TensorFlow, and we choose *OpenNMT-tf*, an official release[8] of OpenNMT implemented with TensorFlow. For a fair comparison, *OpenNMT-tf* has one layer for both the encoder and the decoder, and *Transformer-tf* has the default configuration ($N = 6$), but with parameters being shared among different blocks.

6.2 Development Experiments

Figure 4 shows the system performances as a function of the number of graph state transitions

[7]https://github.com/Kyubyong/transformer.
[8]https://github.com/OpenNMT/OpenNMT-tf.

System	NC-v11			Full		
	BLEU	TER↓	Meteor	BLEU	TER↓	Meteor
OpenNMT-tf	15.1	0.6902	0.3040	24.3	0.5567	0.4225
Transformer-tf	17.1	0.6647	0.3578	25.1	0.5537	0.4344
Seq2seq	16.0	0.6695	0.3379	23.7	0.5590	0.4258
Dual2seq-LinAMR	17.3	0.6530	0.3612	24.0	0.5643	0.4246
Duel2seq-SRL	17.2	0.6591	0.3644	23.8	0.5626	0.4223
Dual2seq-Dep	17.8	0.6516	0.3673	25.0	0.5538	0.4328
Dual2seq	**19.2***	**0.6305**	**0.3840**	**25.5***	**0.5480**	**0.4376**

Table 3: Test performance. *NC-v11* represents training only with the NC-v11 data, while *Full* means using the full training data. * represents significant (Koehn, 2004) result ($p < 0.01$) over *Seq2seq*. ↓ indicates the lower the better.

on the development set. *Dual2seq (self)* represents our dual-attentive model, but its graph encoder encodes the source sentence, which is treated as a chain graph instead of an AMR graph. Compared with *Dual2seq*, *Dual2seq (self)* has the same number of parameters, but without semantic information from AMR. Due to hardware limitations, we do not perform an exhaustive search by evaluating every possible state transition number, but only transition numbers of 1, 5, 10, and 12.

Our *Dual2seq* shows consistent performance improvement by increasing the transition number both from 1 to 5 (roughly +1.3 BLEU points) and from 5 to 10 (roughly 0.2 BLEU points). The former shows greater improvement than the latter, showing that the performance starts to converge after five transition steps. Further increasing transition steps from 10 to 12 gives a slight performance drop. We set the number of state transition steps to 10 for all experiments according to these observations.

On the other hand, *Dual2seq (self)* shows only small improvements by increasing the state transition number, and it does not perform better than *Seq2seq*. Both results show that the performance gains of *Dual2seq* are not due to an increased number of parameters.

6.3 Main Results

Table 3 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. *Dual2seq* is consistently better than the other systems under all three metrics, showing the effectiveness of the semantic information provided by AMR. Especially, *Dual2seq* is better than both *OpenNMT-tf* and *Transformer-tf*. The recurrent graph state transition of *Dual2seq* is similar to Transformer in that it iteratively incorporates global information. The improvement of *Dual2seq* over *Transformer-tf* undoubtedly comes from the use of AMRs, which provide complementary information to the textual inputs of the source language.

In terms of BLEU score, *Dual2seq* is significantly better than *Seq2seq* in both settings, which shows the effectiveness of incorporating AMR information. In particular, the improvement is much larger under the small-scale setting (+3.2 BLEU) than that under the large-scale setting (+1.7 BLEU). This is an evidence that structural and coarse-grained semantic information encoded in AMRs can be more helpful when training data are limited.

When trained on the NC-v11 subset, the gap between *Seq2seq* and *Dual2seq* under Meteor (around 5 points) is greater than that under BLEU (around 3 points). Since Meteor gives partial credit to outputs that are synonyms to the reference or share identical stems, one possible explanation is that the structural information within AMRs helps to better translate the concepts from the source language, which may be synonyms or paronyms of reference words.

As shown in the second group of Table 3, we further compare our model with other methods of leveraging syntactic or semantic information. *Dual2seq-LinAMR* shows much worse performance than our model and only slightly outperforms the *Seq2seq* baseline. Both results show that simply taking advantage of the AMR concepts without their relations does not help very much. One reason may be that AMR concepts, such as *John* and *Mary*, also appear in the textual input, and thus are also encoded by the other

AMR Anno.	BLEU
Automatic	16.8
Gold	**17.5***

Table 4: BLEU scores of *Dual2seq* on *The Little Prince* data, when gold or automatic AMRs are available.

(sequential) encoder.[9] The gap between *Dual2seq* and *Dual2seq-LinAMR* comes from modeling the relations between concepts, which can be helpful for deciding target word order by enhancing the relations in source sentences. We conclude that properly encoding AMRs is necessary to make them useful.

Encoding dependency trees instead of AMRs, *Dual2seq-Dep* shows a larger performance gap with our model (17.8 vs 19.2) on small-scale training data than on large-scale training data (25.0 vs 25.5). It is likely because AMRs are more useful on alleviating data sparsity than dependency trees, since words are lemmatized into unified concepts when parsing sentences into AMRs. For modeling long-range dependencies, AMRs have one crucial advantage over dependency trees by modeling concept-concept relations more directly. It is because AMRs drop function words; thus the distances between concepts are generally closer in AMRs than in dependency trees. Finally, *Dual2seq-SRL* is less effective than our model, because the annotations labeled by SRL are a subset of AMRs.

We outperform Marcheggiani et al. (2018) on the same datasets, although our systems vary in a number of respects. When trained on the *NC-v11* data, they show BLEU scores of 14.9 only with their BiLSTM baseline, 16.1 using additional dependency information, 15.6 using additional semantic roles, and 15.8 taking both as additional knowledge. Using *Full* as the training data, the scores become 23.3, 23.9, 24.5, and 24.9, respectively. In addition to the different semantic representation being used (AMR vs SRL), Marcheggiani et al. (2018) laid GCN (Kipf and Welling, 2017) layers on top of a bidirectional LSTM (BiLSTM) layer, and then concatenated layer outputs as the attention memory. GCN layers encode the semantic role information, while BiLSTM layers encode the input sentence in the source language, and the concatenated hidden

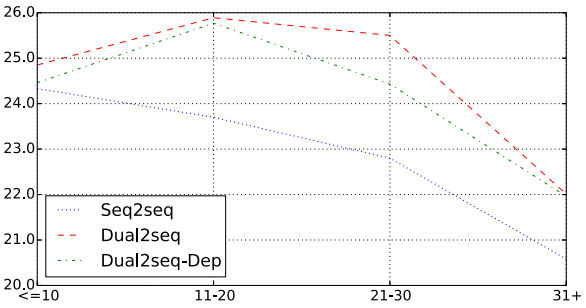

Figure 5: Test BLEU score of various sentence lengths.

states of both layers contain information from both semantic role and source sentence. For incorporating AMR, because there is no one-to-one word-to-node correspondence between a sentence and the corresponding AMR graph, we adopt separate attention models. Our BLEU scores are higher than theirs, but we cannot conclude that the advantage primarily comes from AMR.

6.4 Analysis

Influence of AMR Parsing Accuracy To analyze the influence of AMR parsing on our model performance, we further evaluate on a test set where the gold AMRs for the English side are available. In particular, we choose *The Little Prince* corpus, which contains 1,562 sentences with gold AMR annotations.[10] Since there are no parallel German sentences, we take a German-version *The Little Prince* novel, and then perform manual sentence alignment. Taking the whole *The Little Prince* corpus as the test set, we measure the influence of AMR parsing accuracy by evaluating on the test set when gold or automatically parsed AMRs are available. The automatic AMRs are generated by parsing the English sentences with JAMR.

Table 4 shows the BLEU scores of our *Dual2seq* model taking gold or automatic AMRs as inputs. Not listed in Table 4, *Seq2seq* achieves a BLEU score of 15.6, which is 1.2 BLEU points lower than using automatic AMR information. The improvement from automatic AMR to gold AMR (+0.7 BLEU) is significant, which shows that the translation quality of our model can be further improved with an increase of AMR parsing accuracy. However, the BLEU score with gold AMR does not indicate the potentially best

[9] AMRs can contain multi-word concepts, such as *New York City*, but they are in the textual input.

[10] https://amr.isi.edu/download.html.

AMR: (s2 / say-01 :ARG0 (p3 / person :ARG1-of (h / have-rel-role-91 :ARG0 (p / person :ARG1-of (m2 / meet-03 :ARG0 (t / they) :ARG2 15) :mod (m / mutual)) :ARG2 (f / friend)) :name (n2 / name :op1 “Carla” :op2 “Hairston”)) :ARG1 (a / and :op1 (p2 / person :name (n / name :op1 “Lamb”))) :ARG2 (s / she) :time 20)
Src: Carla Hairston said she was 15 and Lamb was 20 when they met through mutual friends .
Ref: Carla Hairston sagte , sie war 15 und Lamm war 20 , als sie sich durch gemeinsame Freunde trafen .
Dual2seq: Carla Hairston sagte , sie war 15 und Lamm war 20 , als sie sich durch gegenseitige Freunde trafen .
Seq2seq: Carla Hirston sagte , sie sei 15 und Lamb 20 , als sie durch gegenseitige Freunde trafen .

AMR: (s / say-01 :ARG0 (m / media :ARG1-of (l / local-02)) :ARG1 (c2 / come-01 :ARG1 (v / vehicle :mod (p / police)) :manner (c3 / constant) :path (a / across :op1 (r / refugee :mod (n2 / new))) :time (s2 / since :op1 (t3 / then)) :topic (t / thing :name (n / name :op1 (c / Croatian) :op2 (t2 / Tavarnik)))))
Src: Since then , according to local media , police vehicles are constantly coming across new refugees in Croatian Tavarnik .
Ref: Laut lokalen Medien treffen seitdem im kroatischen Tovarnik ständig Polizeifahrzeuge mit neuen Flüchtlingen ein .
Dual2seq: Seither kommen die Polizeifahrzeuge nach den örtlichen Medien ständig über neue Flüchtlinge in Kroatische Tavarnik .
Seq2seq: Seitdem sind die Polizeiautos nach den lokalen Medien ständig neue Flüchtlinge in Kroatien Tavarnik .

AMR: (b2 / breed-01 :ARG0 (p2 / person :ARG0-of (h / have-org-role-91 :ARG2 (s3 / scientist))) :ARG1 (w2 / worm) :ARG2 (s2 / system :ARG1-of (c / control-01 :ARG0 (b / burst-01 :ARG1 (w / wave :mod (s / sound))) :ARG1-of (p / possible-01)) :ARG1-of (n / nervous-01) :mod (m / modify-01 :ARG1 (g / genetics))))
Src: Scientists have bred worms with genetically modified nervous systems that can be controlled by bursts of sound waves .
Ref: Wissenschaftler haben Würmer mit genetisch veränderten Nervensystemen gezüchtet , die von Ausbrüchen von Schallwellen gesteuert werden können .
Dual2seq: Die Wissenschaftler haben die Würmer mit genetisch veränderten Nervensystemen gezüchtet, die durch Verbrennungen von Schallwellen kontrolliert werden können .
Seq2seq: Wissenschaftler haben sich mit genetisch modifiziertem Nervensystem gezüchtet , die durch Verbrennungen von Klangwellen gesteuert werden können .

Figure 6: Sample system outputs.

performance that our model can achieve. The primary reason is that even though the test set is coupled with gold AMRs, the training set is not. Trained with automatic AMRs, our model can learn to selectively trust the AMR structure. An additional reason is the domain difference: *The Little Prince* data are in the literary domain while our training data are in the news domain. There can be a further performance gain if the accuracy of the automatic AMRs on the training set is improved.

Performance Based on Sentence Length We hypothesize that AMRs should be more beneficial for longer sentences: Those are likely to contain long-distance dependencies (such as discourse information and predicate–argument structures), which may not be adequately captured by linear chain RNNs but are directly encoded in AMRs. To test this, we partition the test data into four buckets by length and calculate BLEU for each of them. Figure 5 shows the performances of our model along with *Dual2seq-Dep* and *Seq2seq*. Our model outperforms the *Seq2seq* baseline rather uniformly across all buckets, except for the first one, where they are roughly equal. This may be surprising. On the one hand, *Seq2seq* fails to capture some dependencies for medium-length instances; on the other hand, AMR parses are more noisy for longer sentences, which prevents us from obtaining extra improvements with AMRs.

Dependency trees have been proved useful in capturing long-range dependencies. Figure 5 shows that AMRs are comparatively better than dependency trees, especially on medium-length (21–30) sentences. The reason may be that the AMRs of medium-length sentences are much more accurate than longer sentences, and thus are better at capturing the relations between concepts. On the other hand, even though dependency trees are more accurate than AMRs, they still fail to represent relations for long sentences. It is likely because relations for longer sentences are more difficult to detect. Another possible reason is that dependency trees do not incorporate coreferences, which AMRs consider.

Human Evaluation We further study the translation quality of predicate–argument structures by conducting a human evaluation on 100 instances from the test set. In the evaluation, translations of both *Dual2seq* and *Seq2seq*, together with the source English sentence, the German reference, and an AMR are provided to a German-speaking annotator to decide which translation better captures the predicate–argument structures in the source sentence. To avoid annotation bias, translation results of both models are swapped for some instances, and the German annotator does not know which model each translation belongs to. The annotator either selects a "winner" or makes a "tie" decision, meaning that both results are equally good.

Out of the 100 instances, *Dual2seq* wins on 46, *Seq2seq* wins on 23, and there is a tie on the remaining 31. *Dual2seq* wins on almost half of the instances, about twice as often as *Seq2seq* wins, indicating that AMRs help in translating the predicate–argument structures on the source side.

Case Study The outputs of the baseline system (*Seq2seq*) and our final system (*Dual2seq*) are shown in Figure 6. In the first sentence, the AMR-based Dual2seq system correctly produces the reflexive pronoun *sich* as an argument of the verb *trafen* (*meet*), despite the distance between the words in the system output, and despite the fact that the equivalent English words *each other* do not appear in the system output. This is facilitated by the argument structure in the AMR analysis.

In the second sentence, the AMR-based Dual2seq system produces an overly literal translation for the English phrasal verb *come across*. The Seq2seq translation, however, incorrectly states that the police vehicles *are* refugees. The difficulty for the Seq2seq probably derives in part from the fact that *are* and *coming* are separated by the word *constantly* in the input, while the main predicate is clear in the AMR representation.

In the third sentence, the Dual2seq system correctly translates the object of *breed* as *worms*, while the Seq2seq translation incorrectly states that the scientists breed *themselves*. Here the difficulty is likely the distance between the object and the verb in the German output, which causes the Seq2seq system to lose track of the correct input position to translate.

7 Conclusion

We showed that AMRs can improve neural machine translation. In particular, the structural semantic information from AMRs can be complementary to the source textual input by introducing a higher level of information abstraction. A graph recurrent network (GRN) is leveraged to encode AMR graphs without breaking the original graph structure, and a sequential LSTM is used to encode the source input. The decoder is a doubly attentive LSTM, taking the encoding results of both the graph encoder and the sequential encoder as attention memories. Experiments on a standard benchmark showed that AMRs are helpful regardless of the sentence length and are more effective than other more popular choices, such as dependency trees and semantic roles.

Acknowledgments

We would like to thank the action editor and the anonymous reviewers for their insightful comments. We also thank Kai Song from Alibaba for suggestions on large-scale training, Parker Riley for comments on the draft, and Rochester's CIRC for computational resources.

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 (ACL-17)*, pages 132–140.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-15)*, pages 1699–1710.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Kathryn Baker, Michael Bloodgood, Bonnie J Dorr, Chris Callison-Burch, Nathaniel W Filardo, Christine Piatko, Lori Levin, and Scott Miller. 2012. Modality and negation in SIMT use of modality and negation in semantically-informed syntactic MT. *Computational Linguistics*, 38(2):411–438.

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.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-17)*, pages 1957–1967.

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 (ACL-18)*, pages 273–283.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL-17)*, pages 1215–1226.

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 (ACL-17)*, pages 1936–1945.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2018. Syntax-directed attention for neural machine translation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-18)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*, pages 1724–1734.

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.

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.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*, pages 1426–1436.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. AMR dependency parsing with a typed semantic algebra. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL-18)*, pages 1831–1841.

Zhijiang Guo and Wei Lu. 2018. Better transition-based AMR parsing with a refined search space. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL-18)*, pages 1712–1722.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Pierre Isabelle, Colin Cherry, and George Foster. 2017. A challenge set approach to evaluating machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-17)*, pages 2486–2496.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of the International Conference on Computational Linguistics (COLING-12)*, pages 1359–1376.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *arXiv preprint arXiv:1701.02810*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, pages 388–395.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL-17)*, pages 146–157.

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 (ACL-17)*, pages 688–697.

Xiang Li, Thien Huu Nguyen, Kai Cao, and Ralph Grishman. 2015a. Improving event detection with abstract meaning representation. In *Proceedings of the First Workshop on Computing News Storylines*, pages 11–15.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015b. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-10)*, pages 716–724.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL-18)*, pages 397–407.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Diego Marcheggiani, Joost Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-18)*, pages 486–492.

Arindam Mitra and Chitta Baral. 2015. Locationing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-16)*.

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 (ACL-02)*, pages 311–318.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 32–41.

Xiaochang Peng, Linfeng Song, Daniel Gildea, and Giorgio Satta. 2018. Sequence-to-sequence models for cache transition systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL-18)*, pages 1842–1852.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-15)*, pages 1143–1154.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings*

of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16), pages 1715–1725.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL-18)*, pages 1842–1852.

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 (ACL-16)*, pages 299–305.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*, pages 1054–1059.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems, 30*, pages 5998–6008.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of AMR parsing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-17)*, pages 1257–1268.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the 2006 Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-06)*, pages 439–446.

Dekai Wu and Pascale Fung. 2009. Semantic roles for SMT: A hybrid two-pass model. In *Proceedings of the 2009 Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-09)*, pages 13–16.

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.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state LSTM for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL-18)*, pages 317–327.