Enhancing AMR-to-Text Generation with Dual Graph Representations

Leonardo F. R. Ribeiro†, Claire Gardent‡ and Iryna Gurevych†

†Research Training Group AIPHES and UKP Lab, Technische Universität Darmstadt
www.ukp.tu-darmstadt.de
‡CNRS/LORIA, Nancy, France
claire.gardent@loria.fr

Abstract

Generating text from graph-based data, such as Abstract Meaning Representation (AMR), is a challenging task due to the inherent difficulty in how to properly encode the structure of a graph with labeled edges. To address this difficulty, we propose a novel graph-to-sequence model that encodes different but complementary perspectives of the structural information contained in the AMR graph. The model learns parallel top-down and bottom-up representations of nodes capturing contrasting views of the graph. We also investigate the use of different node message passing strategies, employing different state-of-the-art graph encoders to compute node representations based on incoming and outgoing perspectives. In our experiments, we demonstrate that the dual graph representation leads to improvements in AMR-to-text generation, achieving state-of-the-art results on two AMR datasets†.

1 Introduction

Abstract Meaning Representation (AMR; Banarescu et al. (2013)) is a linguistically-grounded semantic formalism that represents the meaning of a sentence as a rooted directed graph, where nodes are concepts and edges are semantic relations. As AMR abstracts away from surface word strings and syntactic structure producing a language neutral representation of meaning, its usage is beneficial in many semantic related NLP tasks, including text summarization (Liao et al., 2018) and machine translation (Song et al., 2019).

The purpose of AMR-to-text generation is to produce a text which verbalises the meaning encoded by an input AMR graph. This is a challenging task as capturing the complex structural information stored in graph-based data is not trivial, as these are non-Euclidean structures, which implies that properties such as global parametrization, vector space structure, or shift-invariance do not hold (Bronstein et al., 2017). Recently, Graph Neural Networks (GNNs) have emerged as a powerful class of methods for learning effective graph latent representations (Xu et al., 2019) and graph-to-sequence models have been applied to the task of AMR-to-text generation (Song et al., 2018; Beck et al., 2018; Damonte and Cohen, 2019; Guo et al., 2019).

In this paper, we propose a novel graph-to-sequence approach to AMR-to-text generation, which is inspired by pre-neural generation algorithms. These approaches explored alternative (top-down, bottom-up and mixed) traversals of the input graph and showed that a hybrid traversal combining both top-down (TD) and bottom-up (BU) information was best as this permits integrating both global constraints top-down from the input and local constraints bottom-up from the semantic heads (Shieber et al., 1990; Narayan and Gardent, 2012).

Similarly, we present an approach where the input graph is represented by two separate structures, each representing a different view of the graph. The nodes of these two structures are encoded using separate graph encoders so that each concept and relation in the input graph is assigned both a TD and a BU representation.

Our approach markedly differs from existing graph-to-sequence models for MR-to-Text generation (Marcheggiani and Perez Beltrachini, 2018; Beck et al., 2018; Damonte and Cohen, 2019) in that these approaches aggregate all the immediate neighborhood information of a node in a single representation. By exploiting parallel and complementary vector representations of the AMR graph, our approach eases the burden on the neural model in encoding nodes (concepts) and edges (relations) in a single vector representation. It also elimi-
nates the need for additional positional information (Beck et al., 2018) which is required when the same graph is used to encode both TD and BU information, thereby making the edges undirected.

Our main contributions are the following:

- We present a novel architecture for AMR-to-text generation which explicitly encodes two separate TD and BU views of the input graph.

- We show that our approach outperforms recent AMR-to-text generation models on two datasets, including a model that leverages additional syntactic information (Cao and Clark, 2019).

- We compare the performance of three graph encoders, which have not been studied so far for AMR-to-text generation.

2 Related Work

Early works on AMR-to-text generation employ statistical methods (Flanigan et al., 2016; PoudRamgiani et al., 2016; Castro Ferreira et al., 2017) and apply linearization of the graph by means of a depth-first traversal.

Recent neural approaches have exhibited success by linearising the input graph and using a sequence-to-sequence architecture. Konstas et al. (2017) achieve promising results on this task. However, they strongly rely on named entities anonymisation. Anonymisation requires an ad hoc procedure for each new corpus. The matching procedure needs to match a rare input item correctly (e.g., “United States of America”) with the corresponding part in the output text (e.g., “USA”) which may be challenging and may result in incorrect or incomplete delexicalisations. In contrast, our approach omits anonymisation. Instead, we use a copy mechanism (See et al., 2017), a generic technique which is easy to integrate in the encoder-decoder framework and can be used independently of the particular domain and application. Our approach further differs from Konstas et al. (2017) in that we build a dual TD/BU graph representation and use graph encoders to represent nodes.

Cao and Clark (2019) factor the generation process leveraging syntactic information to improve the performance. However, they linearize both AMR and constituency graphs, which implies that important parts of the graphs cannot well be represented (e.g., coreference).

Several graph-to-sequence models have been proposed. Marcheggiani and Perez Beltrachini (2018) show that explicitly encoding the structure of the graph is beneficial with respect to sequential encoding. They evaluate their model on two tasks, WebNLG (Gardent et al., 2017) and SR11Deep (Belz et al., 2011), but do not apply it to AMR benchmarks. Song et al. (2018) and Beck et al. (2018) apply recurrent neural networks to directly encode AMR graphs. Song et al. (2018) and Beck et al. (2018) develop a model based on GRUs.

We go a step further in that direction by developing parallel encodings of graphs which are able to highlight different graph properties.

In a related task, Koncel-Kedziorski et al. (2019) propose an attention-based graph model that generates sentences from knowledge graphs. Schlichtkrull et al. (2018) use Graph Convolutional Networks (GCNs) to tackle the tasks of link prediction and entity classification on knowledge graphs.

Damonte and Cohen (2019) show that off-the-shelf GCNs cannot achieve good performance for AMR-to-text generation. To tackle this issue, Guo et al. (2019) introduce dense connectivity to GNNs in order to integrate both local and global features, achieving good results on the task. Our work is related to Damonte and Cohen (2019), that use stacking of GCN and LSTM layers to improve the model capacity and employ anonymization. However, our model is substantially different: (i) we learn dual representations capturing top-down and bottom-up adjuvant views of the graph, (ii) we employ more effective graph encoders (with different neighborhood aggregations) than GCNs and (iii) we employ copy and coverage mechanisms and do not resort to entity anonymization.

3 Graph-to-Sequence Model

In this section, we describe (i) the representations of the graph adopted as inputs, (ii) the model architecture, including the Dual Graph Encoder and (iii) the GNNs employed as graph encoders.

3.1 Graph Preparation

Let $G = (V, E, R)$ denote a rooted and directed AMR graph with nodes $v_i \in V$ and labeled edges $(v_i, r, v_j) \in E$, where $r \in R$ is a relation type.
Let $n = |V|$ and $m = |E|$ denote the numbers of nodes and edges, respectively.

We convert each AMR graph into an unlabeled and connected bipartite graph $G_t = (V_t, E_t)$, transforming each labeled edge $(v_i, r, v_j) \in E$ into two unlabeled edges $(v_i, r, v_j) \in E_t$, with $|V_t| = n + m$ and $|E_t| = 2m$. This process, called Levi Transformation (Beck et al., 2018), turns original edges into nodes creating an unlabeled graph. For instance, the edge between semester and that with label :mod in Figure 1(b) is replaced by two edges and one node in 1(c): an edge between semester, and the new node :mod and another one between :mod and that. The new graph allows us to directly represent the relationships between nodes using embeddings. This enables us to encode label edge information using distinct message passing schemes employing different GNNs.

$G_t$ captures a TD view of the graph. We also create a BU view of the graph $G_b = (V_b, E_b)$, where each directed edge $e_k = (v_i, v_j) \in E_t$ becomes $e_k = (v_j, v_i) \in E_b$, that is, we reverse the direction of original edges. An example of a sentence, its AMR graph and the two new graphs $G_t$ and $G_b$ is shown in Figure 1.

### 3.2 Dual Graph Encoder

We represent each node $v_i \in V_t$ with a node embedding $e_i \in \mathbb{R}^d$, generated from the node label. In order to explicitly encode structural information, our encoder starts with two graph encoders, denoted by $GE_t$ and $GE_b$, that compute representations for nodes in $G_t$ and $G_b$, respectively.

Each $GE$ learns node representations based on the specific view of its particular graph, $G_t$ or $G_b$. Since $G_t$ and $G_b$ capture distinct perspectives of the graph structure, the information flow is propagated throughout TD and BU directions, respectively. In particular, for each node $v_i$, the $GE$ receives the node embeddings of $v_i$ and its neighbors, and computes its node representation:

$$h_i^1 = GE_t(\{e_i, e_j : j \in N_t(i)\}),$$

$$h_i^b = GE_b(\{e_i, e_j : j \in N_b(i)\}),$$

where $N_t(i)$ and $N_b(i)$ are the immediate incoming neighborhoods of $v_i$ in $G_t$ and $G_b$, respectively.

Each node $v_i$ is represented by two different hidden states, $h_i^t$ and $h_i^b$. Note that we learn two representations per relation and node of the original AMR graph. The hidden states $h_i^t$ and $h_i^b$, and embedding $e_i$ contain different information regarding $v_i$. We concatenate them building a final node representation:

$$r_i = [h_i^t \| h_i^b \| e_i].$$

This approach is similar to bidirectional RNNs (Schuster and Paliwal, 1997). Bidirectional RNNs benefit from left-to-right and right-to-left propagation. They learn the hidden representations separately and concatenate them at the end. We perform a similar encoding: first we learn TD and BU representations independently, and lastly, we concatenate them.

The final representation $r_i$ is employed in a sequence input of a bidirectional LSTM. For each AMR graph, we generate a node sequence by depth-first traversal order. In particular, given a representation sequence from $r_1$ to $r_n$, the hidden forward and backward states of $r_i$ are defined as:

$$\overrightarrow{h}_i = LSTM_f(r_i, \overrightarrow{h}_{i-1}),$$

$$\overleftarrow{h}_i = LSTM_b(r_i, \overleftarrow{h}_{i-1}),$$

Figure 1: (a) an example sentence, (b) its original AMR graph ($G$) and different graph perspectives: (c) top-down ($G_t$) and (d) bottom-up ($G_b$).
different nuances of structural information. Xu et al. (2019) and Graph Isomorphic Networks (GIN, Xu et al. 2019)) employ different GNNs for both graph encoders (Section 3.3). Figure 2 shows the proposed encoder architecture.

3.3 Graph Neural Networks

The GE incorporates, in each node representation, structural information based on both views of the graph. We explore distinct strategies for neighborhood aggregation, adopting three GNNs: Gated Graph Neural Networks (GGNN, Li et al. (2016)), Graph Attention Networks (GAT, Veličković et al. (2018)) and Graph Isomorphic Networks (GIN, Xu et al. 2019)). Each GNN employs a specific message passing scheme which allows capturing different nuances of structural information.

Gated Graph Neural Networks

GGNNs employ gated recurrent units to encode node representations, reducing the recurrence to a fixed number of steps. In particular, the \( l \)-th layer of a GGNN is calculated as:

\[
h_i^{(l)} = GRU \left( h_i^{(l-1)}, \sum_{j \in \mathcal{N}(i)} W_1 h_j^{(l-1)} \right),
\]

where \( \mathcal{N}(i) \) is the immediate neighborhood of \( v_i \), \( W_1 \) is a parameter and GRU is a gated recurrent unit (Cho et al., 2014). Different from other GNNs, GGNNs use back-propagation through time (BPTT) to learn the parameters. GGNNs also do not require to constrain parameters to ensure convergence.

Graph Attention Networks

GATs apply attentive mechanisms to improve the exploitation of non-trivial graph structure. They encode node representations by attending over their neighbors, following a self-attention strategy:

\[
h_i^{(l)} = \alpha_{i,i} W_2 h_i^{(l-1)} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} W_2 h_j^{(l-1)},
\]

where attention coefficients \( \alpha_{i,j} \) are computed as:

\[
\alpha_{i,j} = \text{softmax} \left( \sigma \left( a^T \left(W_2 h_i^{(l-1)} || W_2 h_j^{(l-1)} \right) \right) \right),
\]

where \( \sigma \) is the activation function and \( || \) denotes concatenation. \( W_2 \) and \( a \) are model parameters. The virtue of the attention mechanism is its ability to focus on the most important parts of the node neighborhood. In order to learn attention weights in different perspectives, GATs can employ multi-head attentions.

Graph Isomorphic Networks

GIN is a GNN as powerful as the Weisfeiler-Lehman (WL) graph isomorphism test (Weisfeiler and Lehman, 1968) in representing isomorphic and non-isomorphic graphs with discrete attributes. Its \( l \)-th layer is defined as:

\[
h_i^{(l)} = h_W \left( h_i^{(l-1)} + \sum_{j \in \mathcal{N}(i)} h_j^{(l-1)} \right),
\]

where \( h_W \) is a multi-layer perceptron (MLP). In contrast to other GNNs, which combine node feature with its aggregated neighborhood feature, GINs do not apply the combination step and simply aggregate the node along with its neighbors.

Each of these GNNs applies different approaches to learn structural features from graph data and has achieved impressive results on many graph-based tasks (Li et al., 2016; Veličković et al., 2018; Xu et al., 2019).
3.4 Decoder

An attention-based unidirectional LSTM decoder is used to generate sentences, attending to the hidden representations of edges and nodes. In each step $t$, the decoder receives the word embedding of the previous word (during training, this is the previous word of the reference sentence; at test time it is the previously generated word), and has the decoder state $s_t$. The attention distribution $a_t$ is calculated as in See et al. (2017):

$$e_t = v \cdot \tanh(W_h h_t + W_s s_t + w_c c_t + b),$$

$$a_t = \text{softmax}(e_t),$$

where $s_c$ is the coverage vector and $v$, $W_h$, $W_s$, $w_c$ and $b$ are learnable parameters. The coverage vector is the accumulation of all attention distributions so far.

Copy and Coverage Mechanisms

Previous works (Damonte and Cohen, 2019; Cao and Clark, 2019) use anonymization to handle names and rare words, alleviating the data sparsity. In contrast, we employ copy and coverage mechanisms to address out-of-vocabulary issues for rare target words and to avoid repetition (See et al., 2017).

The model is trained to optimize the negative log-likelihood:

$$\mathcal{L} = -\sum_{t=1}^{|Y|} \log p(y_t|y_{1:t-1}, X; \theta),$$

where $Y = y_1, \ldots, y_{|Y|}$ is the sentence, $X$ is the AMR graph and $\theta$ represents the model parameters.

5 Experiments and Discussion

Implementation Details

We extract vocabularies (size of 20,000) from the training sets and initialize the node embeddings from GloVe word embeddings (Pennington et al., 2014) on Common Crawl. Hyperparameters are tuned on the development set of the LDC2015E86 dataset. For GIN, GAT, and GGNN graph encoders, we set the number of layers to 2, 5 and 5, respectively. To regu-

---

Table 1: Data statistics of LDC2015E86 and LDC2017T10 datasets. The values are calculated for all splits (train, development and test sets). DAG stands for directed acyclic graph.

<table>
<thead>
<tr>
<th></th>
<th>LDC2015E86</th>
<th>LDC2017T10</th>
</tr>
</thead>
<tbody>
<tr>
<td>training, dev and test instances</td>
<td>16,833</td>
<td>36,521</td>
</tr>
<tr>
<td>min, average and max graph diameter</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>min, average and max node degree</td>
<td>0.1</td>
<td>2.1</td>
</tr>
<tr>
<td>min, average and max number of nodes</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>min, average and max number of edges</td>
<td>0</td>
<td>18.6</td>
</tr>
<tr>
<td>number of DAG and non-DAG graphs</td>
<td>18,679</td>
<td>37,284</td>
</tr>
<tr>
<td>min, average and max length sentences</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3: Distribution of the AMR graph diameter (left) and node degree (right) in the training set for LDC2015E86 (red) and LDC2017T10 (blue) datasets.

---

2The datasets can be found at https://amr.isi.edu/download.html
We compare our models against several state-of-the-art results reported on the two datasets (Konstas et al., 2017; Song et al., 2018; Beck et al., 2018; Damonte and Cohen, 2019; Cao and Clark, 2019; Guo et al., 2019).

Results We call the models G2S-GIN (isomorphic encoder), G2S-GAT (graph-attention encoder), and G2S-GGNN (gated-graph encoder), according to the graph encoder utilized. As a baseline (S2S), we train an attention-based encoder-decoder model with copy and coverage mechanisms, and use a linearized version of the graph generated by depth-first traversal order as input. We compare our models against several state-of-the-art results reported on the two datasets (Konstas et al., 2017; Song et al., 2018; Beck et al., 2018; Damonte and Cohen, 2019; Cao and Clark, 2019; Guo et al., 2019).

For both datasets, our approach substantially outperforms the baselines. In LDC2015E86, G2S-GGNN achieves a BLEU score of 27.87, 4.46% higher than Song et al. (2018), who also use the copy mechanism. This indicates that our architecture can learn to generate better signals for text generation. On the same dataset, we have competitive results to Damonte and Cohen (2019). However, we do not rely on preprocessing anonymization not to lose semantic signals. In LDC2017T10, G2S-GGNN achieves a BLEU score of 27.87, which is 3.33 points higher than Damonte and Cohen (2019), a state-of-the-art model that does not employ external information. We also have competitive results to Guo et al. (2019), a very recent state-of-the-art model.

We also outperform Cao and Clark (2019) improving BLEU scores by 3.48% and 4.00%, in LDC2015E86 and LDC2017T10, respectively. In contrast to their work, we do not rely on (i) leveraging supplementary syntactic information and (ii) we do not require an anonymization preprocessing step. G2S-GIN and G2S-GAT have comparable performance on both datasets. Interestingly, G2S-GGNN has better performance among our models. This suggests that graph encoders based on gating mechanisms are very effective in text generation models. We hypothesize that the gating mechanism can better capture long-distance dependencies between nodes far apart in the graph.

We use both BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014) as evaluation metrics. In order to mitigate the effects of random seeds, we report the averages for 4 training runs of each model along with their standard deviation. Table 2 shows the comparison between the proposed models, the baseline and other neural models on the test set of the two datasets.

For both datasets, our approach substantially outperforms the baselines. In LDC2015E86, G2S-GGNN achieves a BLEU score of 27.87, 4.46% higher than Song et al. (2018), who also use the copy mechanism. This indicates that our architecture can learn to generate better signals for text generation. On the same dataset, we have competitive results to Damonte and Cohen (2019). However, we do not rely on preprocessing anonymization not to lose semantic signals. In LDC2017T10, G2S-GGNN achieves a BLEU score of 27.87, which is 3.33 points higher than Damonte and Cohen (2019), a state-of-the-art model that does not employ external information. We also have competitive results to Guo et al. (2019), a very recent state-of-the-art model.

We also outperform Cao and Clark (2019) improving BLEU scores by 3.48% and 4.00%, in LDC2015E86 and LDC2017T10, respectively. In contrast to their work, we do not rely on (i) leveraging supplementary syntactic information and (ii) we do not require an anonymization preprocessing step. G2S-GIN and G2S-GAT have comparable performance on both datasets. Interestingly, G2S-GGNN has better performance among our models. This suggests that graph encoders based on gating mechanisms are very effective in text generation models. We hypothesize that the gating mechanism can better capture long-distance dependencies between nodes far apart in the graph.

Table 2: BLEU and METEOR scores on the test set of LDC2015E86 and LDC2017T10 datasets.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>METEOR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LDC2015E86</td>
<td></td>
</tr>
<tr>
<td>Konstas et al. 2017</td>
<td>22.00</td>
<td>-</td>
</tr>
<tr>
<td>Song et al. 2018</td>
<td>23.28</td>
<td>30.10</td>
</tr>
<tr>
<td>Cao et al. 2019</td>
<td>23.50</td>
<td>-</td>
</tr>
<tr>
<td>Damonte et al. 2019</td>
<td>24.40</td>
<td>23.60</td>
</tr>
<tr>
<td>Guo et al. 2019</td>
<td>25.70</td>
<td>-</td>
</tr>
<tr>
<td>S2S</td>
<td>22.55 ± 0.17</td>
<td>29.90 ± 0.31</td>
</tr>
<tr>
<td>G2S-GIN</td>
<td>22.93 ± 0.20</td>
<td>29.72 ± 0.09</td>
</tr>
<tr>
<td>G2S-GAT</td>
<td>23.42 ± 0.16</td>
<td>29.87 ± 0.14</td>
</tr>
<tr>
<td>G2S-GGNN</td>
<td>24.32 ± 0.16</td>
<td>30.53 ± 0.30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LDC2017T10</td>
</tr>
<tr>
<td>Back et al. 2018</td>
<td>23.30</td>
</tr>
<tr>
<td>Song et al. 2018</td>
<td>24.86</td>
</tr>
<tr>
<td>Damonte et al. 2019</td>
<td>24.54</td>
</tr>
<tr>
<td>Cao et al. 2019</td>
<td>26.80</td>
</tr>
<tr>
<td>Guo et al. 2019</td>
<td>27.60</td>
</tr>
<tr>
<td>S2S</td>
<td>22.73 ± 0.18</td>
</tr>
<tr>
<td>G2S-GIN</td>
<td>26.90 ± 0.19</td>
</tr>
<tr>
<td>G2S-GAT</td>
<td>26.72 ± 0.20</td>
</tr>
<tr>
<td>G2S-GGNN</td>
<td>27.87 ± 0.15</td>
</tr>
</tbody>
</table>

Table 3: Results on LDC2015E86 test set when models are trained with additional Gigaword data.

<table>
<thead>
<tr>
<th>Model</th>
<th>External</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LDC2015E86</td>
</tr>
<tr>
<td>Konstas et al. 2017</td>
<td>200K</td>
<td>22.00</td>
</tr>
<tr>
<td>Song et al. 2018</td>
<td>200K</td>
<td>23.28</td>
</tr>
<tr>
<td>Guo et al. 2019</td>
<td>200K</td>
<td>25.70</td>
</tr>
<tr>
<td>G2S-GGNN</td>
<td>200K</td>
<td>27.87</td>
</tr>
</tbody>
</table>

For BLEU, we use the multi-BLEU script from the MOSES decoder suite (Koehn et al., 2007). For METEOR, we use the original meteor-1.5.jar script (https://github.com/cmu-mtlab/meteor).
the models handle graph and sentence features, we
different data properties. In order to evaluate how
coding the performance of the models concerning
encoder. However, we are also interested in esti-
coders and dual representations over the sequential
datasets shows the superiority of using graph en-
Length (57.6M vs 61.7M).
parameters than the model without graph encoders
MetEOR scores by 1.04 and 0.68 points, respec-
tively. The complete model has slightly more pa-
ters (including embeddings) used in each model.
length. The models have better re-
sults when handling sentences with 20 or fewer to-
sen. Longer sentences pose additional challenges
also investigate the performance with respect to
graph diameter, sentence length and max node out-degree.

Table 4: Results of the ablation study on the
LDC2017T10 development set.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>METEOR</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>biLSTM</td>
<td>22.50</td>
<td>30.42</td>
<td>57.6M</td>
</tr>
<tr>
<td>GE_t + biLSTM</td>
<td>26.33</td>
<td>32.62</td>
<td>59.6M</td>
</tr>
<tr>
<td>GE_b + biLSTM</td>
<td>26.12</td>
<td>32.49</td>
<td>59.6M</td>
</tr>
<tr>
<td>GE_t + GE_b + biLSTM</td>
<td>27.37</td>
<td>33.30</td>
<td>61.7M</td>
</tr>
</tbody>
</table>

Additional Training Data Following previous
works (Konstas et al., 2017; Song et al., 2018; Guo
et al., 2019), we also evaluate our models employ-
ing additional data from English Gigaword corpus
(Napoles et al., 2012). We sample 200K Gigaword
sentences and use JAMR\(^1\) (Flanigan et al., 2016a)
to parse them. We follow the method of
Konstas et al. (2017), which is fine-tuning the model on
the LDC2015E86 training set after every epoch of
pretraining on the Gigaword data. G2S–GGNN
outperforms others with the same amount of Giga-
word sentences (200K), achieving a 32.23 BLEU
score, as shown in Table 3. The results demonstra-
that pretraining on automatically generated
AMR graphs enhances the performance of our
model.

Ablation Study In Table 4, we report the results
of an ablation study on the impact of each com-
ponent of our model on the development set of
LDC2017T10 dataset by removing the graph en-
coders. We also report the number of param-
ters (including embeddings) used in each model.
The first thing we notice is the huge increase in
metric scores (17% in BLEU) when applying the
graph encoder layer, as the neural model receives
signals regarding the graph structure of the input.
The dual representation helps the model with a
different view of the graph, increasing BLEU and
MetEOR scores (17% in BLEU) when applying the
encoder.

Performing an inspection based on different sizes
of graph diameter, sentence length and max node out-degree.

Table 5: METEOR scores and differences to the
S2S, in the LDC2017T10 test set, with respect to the graph
diameter, sentence length and max node out-degree.

<table>
<thead>
<tr>
<th>Model</th>
<th>Graph Diameter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0-7</td>
</tr>
<tr>
<td>S2S</td>
<td>33.2</td>
</tr>
<tr>
<td>G2S–GIN</td>
<td>35.2 +6.0%</td>
</tr>
<tr>
<td>G2S–GAT</td>
<td>35.1 +5.9%</td>
</tr>
<tr>
<td>G2S–GGNN</td>
<td>36.2 +9.0%</td>
</tr>
</tbody>
</table>

Table 5: METEOR scores and differences to the
S2S, in the LDC2017T10 test set, with respect to the graph
diameter, sentence length and max node out-degree.

The performances of all models decrease as the
diameters of the graphs increase. G2S–GGNN has
a 17.9% higher METEOR score in graphs with a
diameter of at most 7 compared to graphs with di-
ameters higher than 13. This is expected as encod-
ing a bigger graph (containing more information)
is harder than encoding smaller graphs. Moreover,
71% of the graphs in the training set have a diam-
eter less than or equal to 7 and only 2% have a
diameter bigger than 13 (see Figure 3). Since the
models have fewer examples of bigger graphs to
learn from, this also leads to worse performance
when handling graphs with higher diameters. We
also investigate the performance with respect to the
sentence length. The models have better re-
sults when handling sentences with 20 or fewer to-
sen. Longer sentences pose additional challenges
to the models.

G2S–GIN has a better performance in han-
dling graphs with node out-degrees higher than 9.
This indicates that GINs can be employed in tasks
where the distribution of node degrees has a long

\(^1\)https://github.com/jflanigan/jamr

\(^5\)METEOR score is used as it is a sentence-level metric.
The premise and the hypothesis are the generated (GEN) and reference (REF) sentences, respectively. (Top) The hypothesis and the premise are the generated (GEN) and reference (REF) sentences, respectively. (Bottom) The premise and the hypothesis are the generated (GEN) and reference (REF) sentences, respectively.

Tail. Surprisingly, S2S has a better performance than G2S-GNN and G2S-GAT when handling graphs that contain high degree nodes.

**Semantic Equivalence** We perform an entailment experiment using BERT (Devlin et al., 2019) fine-tuned on the MultiNLI dataset (Williams et al., 2018) as a NLI model. We are interested in exploring whether a generated sentence (hypothesis) is semantically entailed by the reference sentence (premise). In a related text generation task, Falke et al. (2019) employ NLI models to rerank alternative predicted abstractive summaries.

Nevertheless, uniquely verifying whether the reference (REF) entails the generated sentence (GEN) or vice-versa (GEN entails REF) is not sufficient. For example, suppose that “Today Jon walks” is the REF and “Jon walks” is the GEN. Even though REF entails GEN, GEN does not entail REF, that is, GEN is too general (missing information). Furthermore, suppose that “Jon walks” is the REF and “Today Jon walks” is the GEN, GEN entails REF but REF does not entail GEN, that is, GEN is too specific (added information). Therefore, in addition to verify whether the reference entails the generated sentence, we also verify whether the generated sentence entails the reference.

Table 6 shows the average probabilities for entailment, contradiction and neutral classes on the LDC2017T10 test set. All G2S models have higher entailment compared to S2S. G2S-GNN has 33.5% and 5.2% better entailment performances than S2S, when REF entails GEN and GEN entails REF, respectively. G2S models also generate sentences that contradict the reference sentences less. This suggests that our models are capable of capturing better semantic information from the graph generating outputs semantically related to the reference sentences.

**Human Evaluation** To further assess the quality of the generated sentences, we conduct a human evaluation. We employ the Direct Assessment (DA) method (Graham et al., 2017) via Amazon Mechanical Turk. Using the DA method inspired by Mille et al. (2018), we assess two quality criteria: (i) meaning similarity: how close in meaning the generated text is to the gold sentence; and (ii) readability: how well the generated sentence reads (Is it good fluent English?).

We randomly select 100 sentences generated by S2S and G2S-GNN and randomly assign them to HITs (following Mechanical Turk terminology). Human workers rate the sentences according to meaning similarity and readability on a 0-100 rating scale. The tasks are executed separately and workers were first given brief instructions. For each sentence, we collect scores from 5 workers and average them. Models are ranked according to the mean of sentence-level scores. We apply a quality control step filtering workers who do not score some faked and known sentences properly.

Figure 4 shows the results. In both metrics, G2S-GNN has better human scores for meaning similarity and readability, suggesting a higher
China and Kyrgyzstan agreed in a joint communique that terrorism, separatism and extremism still pose major threats to regional security and stability.

In a joint communique, China and Kyrgyzstan have agreed to still be a major threat to regional security, and regional stability.

At a joint communique, China and Kyrgyzstan agreed that terrorism, separatism and extremism are still a major threat to region security and stability.

Table 7: An example of an AMR graph and generated sentences. GOLD refers to the reference sentence.

<table>
<thead>
<tr>
<th>Model</th>
<th>ADDED</th>
<th>MISS</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2S</td>
<td>47.34</td>
<td>37.14</td>
</tr>
<tr>
<td>G2S–GIN</td>
<td>48.67</td>
<td>33.64</td>
</tr>
<tr>
<td>G2S–GAT</td>
<td>48.24</td>
<td>33.73</td>
</tr>
<tr>
<td>G2S–GGNN</td>
<td>48.66</td>
<td>34.06</td>
</tr>
<tr>
<td>GOLD</td>
<td>50.77</td>
<td>28.35</td>
</tr>
</tbody>
</table>

Table 8: Fraction of elements in the output that are not present in the input (ADDED) and the fraction of elements in the input graph that are missing in the generated sentence (MISS), for the test set of LDC2017T10. The token lemmas are used in the comparison. GOLD refers to the reference sentences.

Semantic Adequacy We also evaluate the semantic adequacy of our model (how well does the generated output match the input?) by comparing the number of added and missing tokens that occur in the generated versus reference sentences (GOLD). An added token is one that appears in the generated sentence but not in the input graph. Conversely, a missing token is one that occurs in the input but not in the output. In GOLD, added tokens are mostly function words while missing tokens are typically input concepts that differ from the output lemma. For instance, in Figure 1, there and of are added tokens while person is a missing token. As shown in Table 8, G2S approaches outperform the S2S baseline. G2S–GIN is closest to GOLD with respect to both metrics suggesting that this model is better able to generate novel words to construct the sentence and captures a larger range of concepts from the input AMR graph, covering more information.

Manual Inspection Table 7 shows sentences generated by S2S, Song et al. (2018), G2S–GAT, and the reference sentence. The example shows that our approach correctly verbalises the subject of the embedded clause “China and ... agreed that terrorism, separatism and extremism ... pose major threats to ...”, while S2S and Song et al. (2018) are fooled by the fact that agree frequently takes an infinitival argument which shares its subject (“China ... agreed to threaten / have agreed to be a major threat”). While this is a single example, it suggests that dual encoding enhances the model ability to take into account the dependencies and the graph structure information, rather than the frequency of n-grams.

6 Conclusion We have studied the problem of generating text from AMR graphs. We introduced a novel architecture that explicitly encodes two parallel and adjuvant representations of the graph (top-down and bottom-up). We showed that our approach outperforms state-of-the-art results in AMR-to-text generation. We provided an extensive evaluation of our models and demonstrated that they are able to achieve the best performance. In the future, we will consider integrating deep generative graph models to express probabilistic dependencies among AMR nodes and edges.

Acknowledgments This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.
References


of the Association for Computational Linguistics, 7:297–312.


Boris Weisfeiler and A.A. Lehman. 1968. A reduction of a graph to a canonical form and an algebra arising during this reduction. Nauchno-Technicheskaya Informatsia, pages 12–16.
