Better AMR-To-Text Generation with Graph Structure Reconstruction

Tianming Wang, Xiaojun Wan and Shaowei Yao
Wangxuan Institute of Computer Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{wangtm, wanxiaojun, yaosw}@pku.edu.cn

Abstract
AMR-to-text generation is a challenging task of generating texts from graph-based semantic representations. Recent studies formalize this task as a graph-to-sequence learning problem and use various graph neural networks to model graph structure. In this paper, we propose a novel approach that generates texts from AMR graphs while reconstructing the input graph structures. Our model employs graph attention mechanism to aggregate information for encoding the inputs. Moreover, better node representations are learned by optimizing two simple but effective auxiliary reconstruction objectives: link prediction objective which requires predicting the semantic relationship between nodes, and distance prediction objective which requires predicting the distance between nodes. Experimental results on two benchmark datasets show that our proposed model improves considerably over strong baselines and achieves new state-of-the-art.

1 Introduction
Abstract Meaning Representation (AMR) is a popular semantic formalism in representing the meaning of natural language text. AMR abstracts away from the surface form of a sentence and encodes its meaning as a rooted and direct graph, where nodes denote the concepts and edges denote the relations between the concepts. AMR-to-text generation is the task of recovering a text representing the same meaning as a given graph and it has attracted lots of attention in recent years. Because the function words and syntactic realizations are abstracted away and numerous details including tense, number, and definiteness in AMR graph are underspecified, this task is very challenging.

Off-the-shelf approaches for neural machine translation have been explored for AMR-to-text generation. Konstas et al. [2017] first transform the graph into sequence and apply sequence-to-sequence model to solve the task. Such a method may lose the information of reentrant structure. Recent works regard this task as a graph-to-sequence learning problem [Beck et al., 2018; Song et al., 2018; Damonte and Cohen, 2019]. These studies propose various graph neural networks to encode the graph, which focus on aggregating information from one-hop neighbors. The current state-of-the-art methods propose using relation encoders to model relation of indirectly connected concepts by encoding the shortest path between nodes [Zhu et al., 2019; Cai and Lam, 2019]. They extend the encoder in the Transformer and use relative position encoding to utilize the information captured by the relation encoder. However, these graph-to-sequence models bring errors like missing information from the input graph and mistranslation of the semantic relations between the concepts [Konstas et al., 2017], which indicates that some graph structure information is not captured in the node representations. An example is shown in Figure 1.

To enhance graph structure learning, we propose a novel approach that generates natural language texts from AMR graphs while reconstructing the input graph structure. We adopt the Transformer encoder-decoder architecture and propose a variant of Transformer, which employs graph attention mechanism to aggregate information from one-hop neighborhoods. To learn better node representations, we propose to optimize two simple but effective auxiliary reconstruction objectives, i.e. link prediction objective and distance prediction objective. Link prediction requires the model to predict the relation between two given nodes or predict the target node (or source node) given a source node (or target node) and a labeled edge. Distance prediction requires the model to pre-
dict the distance between two given nodes. All the information to be predicted is explicitly or implicitly provided in the input graph and we require the model to reconstruct the graph structure based on the learned node representations. The former encourages the model to encode the neighbor relation information as much as possible in the node representations, and the latter helps the model to distinguish information from neighboring nodes and distant nodes. The learned node representations with the aid of the two objectives can reduce the errors of mistranslation of semantic relations and missing information in the output texts.

Experimental results on two benchmark datasets show that our model substantially outperforms the prior methods and achieves a new state-of-the-art performance. Our model improves the BLEU scores by 2.4 points on LDC2015E86 and 2.1 points on LDC2017T10, respectively. In all, our contributions can be summarized as follows:

- We propose a novel approach that generates texts from AMR graphs while reconstructing graph structures. Two simple but effective reconstruction objectives are optimized during training, which help better capture the information provided in the graph. 1
- Our proposed variant of Transformer shows its effectiveness on AMR-to-text generation.
- Empirical studies on two benchmark datasets exhibit that our model advances the state of the art for the AMR-to-text generation task.

2 Our Approach

Our model adopts the Transformer encoder-decoder architecture, which can generate natural language texts from AMR graphs while reconstructing the input structure. In this section, we begin by providing the notations we use, followed by describing our variant of the Transformer model, including a graph encoder which employs graph attention mechanism to aggregating information of incoming neighbors and outgoing neighbors, respectively, and a sentence decoder which generates sentences with copy mechanism. Then we introduce the proposed graph structure reconstruction objectives. Finally, the objective functions will be detailed. The overall architecture of our model is shown in Figure 2.

2.1 Notations

Let $G = \{V, E, R\}$ denote an AMR graph, where $V$ is a set of $N$ nodes, $E$ is a set of $M$ edges, and $R$ is a set of $L$ edge label types. $N$, $M$, and $L$ are the number of nodes, edges and label types, respectively. Each edge $e$ in $E$ is denoted as $(i, j, r_{i,j})$, where $i$ and $j$ are the indices of source node and target node, respectively, and $r_{i,j} \in R$ is the edge label. In addition, we denote the neighbor nodes reached by incoming edges of node $v_i$ as $N^{in}_i$ and the neighbor nodes reached by outgoing edges as $N^{out}_i$. The distance $d_{i,j}$ of a given node pair is defined as the length of the shortest path from $v_i$ to $v_j$ (regardless of the direction of the edges).

1The code is available at https://github.com/sodawater/graph-reconstruction-amr2text.

![Figure 2: The overall architecture of our proposed model.](image)

For convenience, we denote queries, keys and values for attention in Transformer as $Q$, $K$, and $V$. Let $MHA_{ij}(Q, K, V)$ denote the multi-head attention, $FFN(x)$ denote the feed-forward network, and $LN(x)$ denote the layer normalization.

2.2 Graph Encoder

The purpose of the graph encoder is to directly encode the input graph and learn the representation for each node. It is composed of a stack of $L_1$ identical graph layers, where different parameters are used from layer to layer. Each layer has two sub-layers: a graph attention mechanism and a feed-forward network.

At each layer, we first update the node representations by aggregating a weighted average from neighbors by using learned attention weights. Let $h^i_l = (h^i_{1,l}, h^i_{2,l}, ..., h^i_{N,l}) \in \mathbb{R}^{d_{model} \times N}$ be the node representations learned at layer $l$, where $d_{model}$ is the dimension size of the model. In particular, $h^i_0$ are the linearly transformed input embeddings.

$$h^i_0 = W_c x^i_0$$  \hspace{1cm} (1)

where $W_c \in \mathbb{R}^{d_{model} \times d_{emb}}$ is the transformation matrix and $x^i_0 \in \mathbb{R}^{d_{emb}}$ is the word embedding of node $v_i$, and $d_{emb}$ is the dimension size of the embedding. Considering that AMR is a directed graph, neighbor nodes reached by incoming edges and nodes reached by outgoing edges play different roles and contribute different information to the central node. We perform graph attention mechanisms over incoming neighborhoods and outgoing neighborhoods, respectively. We use an additive form of attention in our model. The attention score of outgoing edge $(i, j, r_{i,j})$ for node $v_i$ is computed as follows:

$$\overrightarrow{c}^l_{i,j} = W_{ou2} \text{LeakyReLU}(W_{ou1}[h^i_{l-1}; h^j_{l-1}; x^r_{i,j}])$$  \hspace{1cm} (2)

where $[; ; ]$ is the concatenation operator, $x^r_{i,j} \in \mathbb{R}^{d_{model}}$ is the embedding of edge label $r_{i,j}$, and $W_{ou1} \in \mathbb{R}^{d_{model} \times 3d_{model}}$ and $W_{ou2} \in \mathbb{R}^{1 \times d_{model}}$ are the parameters. Similarly, the attention score of incoming edge $(j, i, r_{j,i})$ is computed as follows:

$$\overleftarrow{c}^l_{i,j} = W_{in2} \text{LeakyReLU}(W_{in1}[h^j_{l-1}; h^i_{l-1}; x^r_{j,i}])$$  \hspace{1cm} (3)

Then the attention probabilities $\overrightarrow{a}^l_{i,j}$ and $\overleftarrow{a}^l_{i,j}$ are calculated as a softmax over the scores $\overrightarrow{c}^l_{i,j}$ and $\overleftarrow{c}^l_{i,j}$, respectively. We
further aggregate the information of neighbor nodes and corresponding edges using these attention probabilities.

\[
\bar{a}_{i,j}^l = \frac{\exp(\bar{e}_{i,j}^l)}{\sum_{k \in \mathcal{N}_{i}^{\text{out}}} \exp(\bar{e}_{i,k}^l)}
\]

\[
\bar{g}_{i}^l = \sum_{j \in \mathcal{N}_{i}^{\text{out}}} \bar{a}_{i,j}^l W_{\text{ins}}[h_j; x_{i,j}^r]
\]

where \(W_{\text{ins}}, W_{\text{ou}_3} \in \mathbb{R}^{d_{\text{model}} \times 2d_{\text{model}}}\).

Following Velčičković et al. [2017], we also extend the graph attention mechanism to employ multi-head attention, which is beneficial for stabilizing the learning process. We split the attention into \(K\) heads and perform \(K\) independent attention mechanisms to execute the computation of the weighted average, which are further concatenated to get the final representation.

\[
\tilde{g}_{i}^l = \left[ \sum_{k=1}^{K} \alpha_{i,j}^{l,k} W_{\text{ou}_3}[h_j; x_{i,j}^r] \right]
\]

where \(\|\) is the concatenation operator. \(\tilde{g}_{i}^l\) is computed in the same way.

Since the above graph attention mechanism does not aggregate the information of the central node, we use a linear transformation layer to combine the information of the central node and neighborhoods.

\[
g_{i}^l = W_c[h_{i}^{l-1}; \tilde{g}^l_i; \tilde{g}^l_i] + b_c
\]

where \(W_c \in \mathbb{R}^{d_{\text{model}} \times 3d_{\text{model}}\times 3d_{\text{model}}}\) and \(b_c \in \mathbb{R}^{d_{\text{model}}}\) are the parameters.

The second sub-layer is a fully connected feed-forward network. Residual connection and layer normalization are employed for connecting the adjacent layers.

\[
h^l = \text{LN}(\text{FFN}(g^l_i) + h^{l-1})
\]

A linear transformation layer is employed upon the encoder stack for aggregating the outputs of different layers.

\[
h_i = W_l[\| h_i^l ] + b_l
\]

where the final representation for node \(v_i\) is denoted as \(h_i\) for convenience, and \(W_l \in \mathbb{R}^{d_{\text{model}} \times (L_l+1)d_{\text{model}}}\) and \(b_l \in \mathbb{R}^{d_{\text{model}}\text{size of vocabulary.}}\) are the parameters.

\subsection{2.3 Sentence Decoder}

The sentence decoder in our model has an architecture similar to the decoder in Transformer. It is composed of \(L_2\) identical layers, where parameters are different from layer to layer. Each layer has three sub-layers: a multi-head self-attention mechanism, a multi-head attention mechanism over the node representations, and a feed-forward network.

At each layer, we first update the token representations by a self-attention mechanism. Let \(h = (h_1^l, h_2^l, ..., h_T^l) \in \mathbb{R}^{d_{\text{model}} \times T}\) represent the token representations at layer \(l\). In particular, \(h_i\) are the sum of the linearly transformed embeddings of tokens \(x^r\) and position encoding \(p_c\).

\[
\hat{h}_i^0 = W_c x_i^r + p_c
\]

Note that a masking is used for ensuring that the attention and prediction for position \(i\) depend only on the known words at position preceding \(i\).

\[
a^l = \text{LN}(\text{MHAtt}(h_i^{l-1}, h_{i-1}^{l-1}, h_{i+1}^{l-1}) + h_i^{l-1})
\]

Following the self-attention, we employ a multi-head attention over the output of the encoder and a feed-forward network.

\[
c^l = \text{LN}(\text{MHAtt}(a^l, h, h) + a^l)
\]

\[
\hat{h}^l = \text{LN}(\text{FFN}(c^l) + c^l)
\]

For convenience, we denote the final representations of the tokens in the decoder as \(h\). The final output is transformed and passed through a softmax layer to generate the probability \(p^l_i\) of next word over the vocabulary

\[
p^l_i = \text{softmax}(W_g \hat{h}_i + b_g)
\]

where \(W_g \in \mathbb{R}^{d_{\text{vocab}} \times d_{\text{model}}}\), \(b_g \in \mathbb{R}^{d_{\text{vocab}}}\), and \(d_{\text{vocab}}\) is the size of vocabulary.

We apply a copy mechanism to tackle the data sparseness problem. A gate for controlling generating words from vocabularies or copying words from inputs is used.

\[
\eta_i = \text{sigmoid}(W_{\eta_2} \text{tanh}(W_{\eta_1}[h_i; x_i] + b_{\eta_1}) + b_{\eta_2})
\]

where \(W_{\eta_1} \in \mathbb{R}^{d_{\text{model}} \times 2d_{\text{model}}}\), \(W_{\eta_2} \in \mathbb{R}^{1 \times d_{\text{model}}}\), \(b_{\eta_1} \in \mathbb{R}^{d_{\text{model}}}\), and \(b_{\eta_2} \in \mathbb{R}^1\) are the parameters. We use the average of the \(K\) independent attention probabilities \(\alpha_{i,j}^{l,k}\) of the last multi-head attention sub-layer as the copy probabilities \(p_{c,i}^l\).

\[
p_{c,i}^l = \sum_{j=1}^{N} \left( \frac{1}{K} \sum_{k=1}^{K} \alpha_{i,j}^{l,k} \right) z_j
\]

where \(z_j\) is the one-hot indicator vector for the node \(v_j\). The final distribution \(p_i\) is the weighted average of the two probabilities with gate \(\theta_i\).

\[
p_i = \eta_i * p_{c,i}^l + (1 - \eta_i) * p_i^l
\]

\subsection{2.4 Graph Structure Reconstruction}

To learn better node representations from the graph structure, we propose to optimize two simple but effective auxiliary reconstruction objectives.

The first objective is based on link prediction, which requires the model to predict the semantic relations for the given node pairs. For a given pair of nodes \((v_i, v_j)\), we concatenate the representations of these two nodes and employ a multi-layer perceptron to predict the corresponding semantic relation.

\[
\hat{r}_{i,j} = \text{softmax}(W_r[\text{MLP}([h_i; h_j])] + b_r)
\]
where $W_r \in \mathbb{R}^{(L+1) \times d_{model}}, b_r \in \mathbb{R}^{L+1}$, and $L$ is the number of semantic label types in the AMR graph. For the pair of nodes that are adjacent in the graph, i.e., connected with an labeled edge, the gold relation label is exactly the given semantic relation $r_{i,j}$. For the pair of nodes that are not adjacent, the gold label is non-adjacent.

The link prediction can be expressed in another way, which requires the model to predict the target node (or source node) given a source node (or target node) and a semantic relation type. In other words, we aim to predict the node $v_j$ in relation $(v_i, v_j, r_{i,j})$ when $v_i$ and $r_{i,j}$ is given. We employ a pointer module to predict the node:

$$
\hat{e}_{i,k} = \frac{W_p h_i \left(W_k [h_k; x_{i,k}^T] \right)^T}{\sqrt{d_{model}}}
$$

$$
\hat{\alpha}_{i,k} = \frac{\exp(\hat{e}_{i,k})}{\sum_{k=1}^{N} \exp(\hat{e}_{i,k})}
$$

$$
k^* = \arg \max_{k} (\hat{\alpha}_{i,k})
$$

Obviously $j$ is the gold answer for $k^*$. Either of the above two forms of link prediction encourages the model to exactly encode the neighbor relations as much as possible in the node representations, so less information will be missing and less semantic relation will be mistranslated during decoding.

The second objective is based on distance prediction, which requires the model to predict the distance between a pair of nodes in the graph. As mentioned before, the distance $d_{i,j}$ of a pair of nodes is defined as the length of the shortest path from $v_i$ to $v_j$, regardless of the edge direction. In the graph encoder, non-local information is captured by using multiple aggregation layers. This objective helps the model to distinguish nodes whether they are adjacent or distant, so direct relations of neighbor nodes and indirect relations of distant nodes can be encoded with distinction and mixing of semantic relation during generation can be reduced. We employ a multi-layer perceptron to predict the distance between two nodes,

$$
\hat{d}_{i,j} = \text{softmax}(W_d[\text{MLP}(h_i; h_j)] + b_d)
$$

where $W_d \in \mathbb{R}^{(D+1) \times d_{model}}, b_d \in \mathbb{R}^{D+1}$, and $D$ is the maximum diameter of the AMR graphs in the dataset. Obviously $d_{i,j}$ is the gold answer.

### 2.5 Objective Function

We aim to optimize the negative log-likelihood of each gold-standard output sentence, $s_i$, given the input graph $G$.

$$
L_g = -\sum_{i=1}^{T} \log P(s_i | s_{1:i-1}, G, \theta)
$$

where $s_i$ is the gold answer for $i$-th token, $\theta$ represents the model parameters, and $P(s_i | s_{1:i-1}, G, \theta)$ is computed in Eq.(15).

To learn better node representations and generate texts of better quality, we also optimize the two proposed graph reconstruction objectives. As described before, link prediction objective has two forms. The first form is to predict the relation given the node pair, which aims to optimize the following negative log-likelihood:

$$
L_1 = - \sum_{(i,j,r_{i,j}) \in E} \log P(r_{i,j} | i, j, G, \theta) - \frac{1}{N} \sum_{(i,j,r_{i,j}) \in E} \log P(r_{i,j} | i, j, G, \theta)
$$

where $r_{i,j}$ is the gold answer for the relation of nodes $v_i$ and $v_j$ (note that the gold label is set to non-adjacent when the two nodes are not adjacent), $\lambda_n$ and $\frac{1}{N}$ are used for balancing the weight of negative samples and $P(r_{i,j} | i, j, G, \theta)$ can be computed from the predicted probability $\hat{r}_{i,j}$ in Eq.(16).

The second form of link prediction objective is to predict the target node given a source node and a labeled edge.

$$
L_2 = - \sum_{(i,j,r_{i,j}) \in E} \log P(j | i, r_{i,j}, G, \theta)
$$

where $P(j | i, r_{i,j}, G, \theta)$ can be computed from the results in Eq.(17).

Distance prediction objective is defined as follows

$$
L_d = - \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \log P(d_{i,j} | G, \theta)
$$

where $P(d_{i,j} | G, \theta)$ can be computed from the results in Eq.(18).

Our model is trained by optimizing the weighted sum of the generation objective and graph reconstruction objectives.

$$
L = L_g + \lambda_l * L_1^{1/2} + \lambda_d * L_d
$$

where $L_1^{1/2}$ represents one form of the link prediction objective and $\lambda_l$ and $\lambda_d$ are the hyper-parameters.

### 3 Experiment

#### 3.1 Data

Two standard English AMR corpora (LDC2015E86 and LDC2017T10) are used as our evaluation datasets. The LDC2015E86 dataset contains 16833 training instances, 1368 development instances, and 1371 test instances. The LDC2017T10 contains 36521 training instances, and the same instances for the development and test as LDC2015E86.

#### 3.2 Setup

We set the model parameters based on preliminary experiments on the development set. $d_{model}$ is set to 512. The numbers $L_1, L_2$ of layers of the encoder and decoder are both set to 6. The head number $K$ is set to 2. The batch size is set to 64. $\lambda_n$ is set to 0.1, $\lambda_l$ is set to 0.4 and $\lambda_d$ is set to 0.1. We share the vocabulary of the encoder and decoder, and use Glove vectors [Pennington et al., 2014] to initialize the word embeddings and $d_{emb}$ is set to 300. We apply dropout and use a rate of 0.2. Label smoothing is employed and the rate is set to 0.1. We use the Adam optimizer [Kingma and Ba, 2015] with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. The same learning
rate schedule of Vaswani et al. [2017] is adopted and the maximum learning rate is set to 0.0005. During training, we filter out instances with more than 50 nodes in graph or 50 words in sentence for speeding up. During inference, beam search with size 5 is used.

Following prior works, we use BLEU [Papineni et al., 2002], Meteor [Banerjee and Lavie, 2005], and CHRF++ [Popović, 2017] as automatic metrics for evaluation.

### 3.3 Comparison Results

We compare our model with several strong baselines, including sequence-to-sequence models and graph-to-sequence models. We use the first form of link prediction objective in this experiment. Models trained with the second form of link prediction objective or other combination of objectives will be detailed in the ablation study. Our base model without graph reconstruction objectives is also compared. Note that some compared baselines are ensemble model but our models are both single model.

Table 1 summarizes the results of these models on the benchmarks. Our model substantially outperforms previous models and achieves the new state-of-the-art performances. Our base model (i.e., without graph reconstruction) also outperforms baselines, which shows the effectiveness of our proposed variant of Transformer. With graph reconstruction objectives, the performance of our model improves by 1.6 BLEU points and 1.2 BLEU points on two datasets, respectively. Among all the baselines, StructuralTransformer-SA achieves the best score on LDC2015E86. Our model improves the BLEU score by 2.4 points, Meteor score by 0.6 points, and CHRF++ score by 1.0 points. On LDC2017T10, our proposed model also outperforms StructuralTransformer-CNN by more than 2 BLEU points. Comparing the two sequence-to-sequence neural models, Transformer is much better than the RNN-based model S2S. Similar phenomenon is observed in the graph-to-sequence models. This is the reason we adopt the Transformer encoder-decoder architecture in our model. We can also see that ensemble models achieve better performance than single models with the same architecture, which indicates that ensemble learning is beneficial in this task. However, our single model still strongly outperforms these ensemble models.

### 3.4 Ablation Study

We further perform an ablation study on the LDC2015E86 development dataset to investigate the influence of the proposed auxiliary objectives. We vary the overall objective function in the following ways: only use the generation objective ($L_g$); use auxiliary link prediction objective in the first form ($L_g + \lambda_l * L^1_l$); use link prediction objective in the second form ($L_g + \lambda_l * L^2_l$); use distance prediction objective ($L_g + \lambda_d * L_d$); use link prediction objective in the first form and distance prediction objective ($L_g + \lambda_l * L^1_l + \lambda_d * L_d$); use link prediction objective in the second form and distance prediction objective ($L_g + \lambda_l * L^2_l + \lambda_d * L_d$). Note that we only report the BLEU score in this experiment.

Table 2 presents the results. We can see that using either the link prediction objective or the distance prediction objective could improve the performance. Both two forms of the link prediction objective result in an improvement of about 1 BLEU points. With the distance prediction objective, the result is 0.7 BLEU points higher. Our model achieves the best performance by optimizing the generation objective and two

<table>
<thead>
<tr>
<th>Objective function</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>only generation</td>
<td>29.8</td>
</tr>
<tr>
<td>generation + link (first form)</td>
<td>31.0</td>
</tr>
<tr>
<td>generation + link (second form)</td>
<td>30.9</td>
</tr>
<tr>
<td>generation + distance</td>
<td>30.5</td>
</tr>
<tr>
<td>generation + link (first form) + distance</td>
<td>31.3</td>
</tr>
<tr>
<td>generation + link (second form) + distance</td>
<td>31.4</td>
</tr>
</tbody>
</table>

Table 2: Ablation results on the LDC2015E86 development set.
of different AMR-to-text models. Similar to Konstas et al., we further analyze the semantic error types in the outputs. The results verify the usefulness of graph structure reconstruction. These re-

lationships are also proposed for enhancing the capturing of structure information and semantic relation in the node representations. We perform experiments on two English benchmarks and the results show that our model achieves the new state-of-the-art performance. The result of ablation study indicates the effectiveness of our base model and the graph reconstruction objectives. In addition, we analyze the semantic errors in the outputs by using automatic metrics.

In future work, we will apply our model to other graph-to-sequence problems. We will also incorporate more well-designed and effective graph reconstruction objectives for better node representation learning.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). Xiaojun Wan is the corresponding author.
References


