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 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

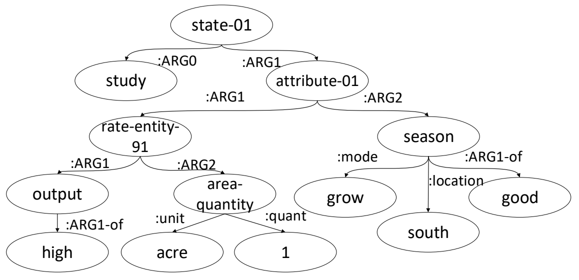

Figure 1: An example of AMR graph meaning “The study stated that the high output per acre was attributed to a good growing season in the south.”. The current graph2seq model outputs “The study stated that the high output had been attributed to high output at a good growth season in the south.”, which misses the information of concept “acre” and mistranslates the semantic relation between “attritube” and “season”.

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 predict 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 adopt 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 sentence 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 $\mathcal{N}_i^{in}$ and the neighbor nodes reached by outgoing edges as $\mathcal{N}_i^{out}$. 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).

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

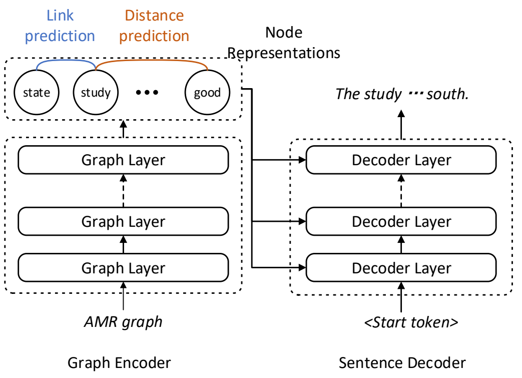

Figure 2: The overall architecture of our proposed model.

For convenience, we denote queries, keys and values for attention in Transformer as Q, K, and V. Let MHAtt(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 $\mathbf{h}^l = (h_1^l, h_2^l, ..., h_N^l) \in \mathbb{R}^{d_{model} \times N}$ be the node representations learned at layer l, where d_{model} is the dimension size of the model. In particular, $\mathbf{h}^0$ are the linearly transformed input embeddings.

$$h_i^0 = W_e x_i^n \tag{1}$$

where $W_e \in \mathbb{R}^{d_{model} \times d_{emb}}$ is the transformation matrix and $x_i^n \in \mathbb{R}^{d_{emb}}$ 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:

$$\overrightarrow{e}_{i,j}^{\,l} = W_{ou_2}[\text{LeakyReLu}(W_{ou_1}[h_i^{l-1}; h_j^{l-1}; x_{i,j}^r])] \tag{2}$$

where $[;;]$ is the concatenation operator, $x_{i,j}^r \in \mathbb{R}^{d_{model}}$ is the embedding of edge label $r_{i,j}$, and $W_{ou_1} \in \mathbb{R}^{d_{model} \times 3d_{model}}$ and $W_{ou_2} \in \mathbb{R}^{1 \times d_{model}}$ are the parameters. Similarly, the attention score of incoming edge $(j, i, r_{j,i})$ is computed as follows:

$$\overleftarrow{e}_{i,j}^{\,l} = W_{in_2}[\text{LeakyReLu}(W_{in_1}[h_j^{l-1}; h_i^{l-1}; x_{j,i}^r])] \tag{3}$$

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

$$\begin{aligned}
\overrightarrow{\alpha}^{l}_{i,j} &= \frac{\exp(\overrightarrow{e}^{l}_{i,j})}{\sum_{k\in\mathcal{N}_i^{out}}\exp(\overrightarrow{e}^{l}_{i,k})} \\
\overleftarrow{\alpha}^{l}_{i,j} &= \frac{\exp(\overleftarrow{e}^{l}_{i,j})}{\sum_{k\in\mathcal{N}_i^{in}}\exp(\overleftarrow{e}^{l}_{i,k})} \\
\overrightarrow{g}^{l}_{i} &= \sum_{j\in\mathcal{N}_i^{out}} \overrightarrow{\alpha}^{l}_{i,j} W_{ou_3}[h_j; x^r_{i,j}] \\
\overleftarrow{g}^{l}_{i} &= \sum_{j\in\mathcal{N}_i^{in}} \overleftarrow{\alpha}^{l}_{i,j} W_{in_3}[h_j; x^r_{j,i}]
\end{aligned} \tag{4}$$

where $W_{in_3}, W_{ou_3} \in \mathbb{R}^{d_{model}\times 2d_{model}}$.

Following Veličković *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.

$$\overrightarrow{g}^{l}_{i} = \Big\Vert_{k=1}^{K} \left(\sum_{j\in\mathcal{N}_i^{out}} \overrightarrow{\alpha}^{l,k}_{i,j} W^k_{ou_3}[h_j; x^r_{i,j}] \right) \tag{5}$$

where $||$ is the concatenation operator. $\overleftarrow{g}^{l}_{i}$ 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.

$$g^l_i = W_c[h^{l-1}_i; \overrightarrow{g}^{l}_{i}; \overleftarrow{g}^{l}_{i}] + b_c \tag{6}$$

where $W_c \in \mathbb{R}^{d_{model}\times 3d_{model}}$ and $b_c \in \mathbb{R}^{d_{model}}$ 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.

$$\mathbf{h}^l = \text{LN}(\text{FFN}(\mathbf{g}^l) + \mathbf{h}^{l-1}) \tag{7}$$

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

$$h_i = W_l[\Big\Vert_{l=0}^{L_1} h^l_i] + b_l \tag{8}$$

where the final representation for node v_i is denoted as h_i for convenience, and $W_l \in \mathbb{R}^{d_{model}\times(L_1+1)d_{model}}$ and $b_l \in \mathbb{R}^{d_{model}}$ are the parameters.

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 $\hat{\mathbf{h}} = (\hat{h}^l_1, \hat{h}^l_2, ..., \hat{h}^l_T) \in \mathbb{R}^{d_{model}\times T}$ represent the token representations at layer l. In particular, $\hat{\mathbf{h}}^0$ are sum of the linearly transformed input embeddings of tokens x^t and position encoding pe.

$$\hat{h}^0_i = W_e x^t_i + pe_i \tag{9}$$

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.

$$\mathbf{a}^l = \text{LN}(\text{MHAtt}(\hat{\mathbf{h}}^{l-1}, \hat{\mathbf{h}}^{l-1}, \hat{\mathbf{h}}^{l-1}) + \hat{\mathbf{h}}^{l-1}) \tag{10}$$

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

$$\begin{aligned}
\mathbf{c}^l &= \text{LN}(\text{MHAtt}(\mathbf{a}^l, \mathbf{h}, \mathbf{h}) + \mathbf{a}^l) \\
\hat{\mathbf{h}}^l &= \text{LN}(\text{FFN}(\mathbf{c}^l) + \mathbf{c}^l)
\end{aligned} \tag{11}$$

For convenience, we denote the final representations of the tokens in the decoder as $\hat{\mathbf{h}}$. The final output is transformed and passed through a softmax layer to generate the probability p^g_i of next word over the vocabulary

$$p^g_i = \text{softmax}(W_g\hat{h}_i + b_g) \tag{12}$$

where $W_g \in \mathbb{R}^{d_{vocab}\times d_{model}}$, $b_g \in \mathbb{R}^{d_{vocab}}$, and d_{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.

$$\eta_i = \text{sigmoid}(W_{\eta_2}[\tanh(W_{\eta_1}[\hat{h}_i; x_i] + b_{\eta_1})] + b_{\eta_2}) \tag{13}$$

where $W_{\eta_1} \in \mathbb{R}^{d_{model}\times 2d_{model}}$, $W_{\eta_2} \in \mathbb{R}^{1\times d_{model}}$, $b_{\eta_1} \in \mathbb{R}^{d_{model}}$ and $b_{\eta_2} \in \mathbb{R}^1$ are the parameters. We use the average of the K independent attention probabilities $\alpha^{l,k}_{i,j}$ of the last multi-head attention sub-layer as the copy probabilities p^c_i.

$$p^c_i = \sum_{j=1}^{N} \left(\frac{1}{K} \sum_{k=1}^{K} \alpha^{l,k}_{i,j} \right) z_j \tag{14}$$

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 θ_i.

$$p_i = \eta_i * p^g_i + (1 - \eta_i) * p^c_i \tag{15}$$

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.

$$\hat{r}_{i,j} = \text{softmax}(W_r[\text{MLP}([h_i; h_j])] + b_r) \tag{16}$$

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.

$$\begin{aligned} \hat{e}_{i,k} &= \frac{W_q h_i \left(W_k [h_k; x^r_{i,k}] \right)^\top}{\sqrt{d_{model}}} \\ \hat{\alpha}_{i,k} &= \frac{\exp(\hat{e}_{i,k})}{\sum_{\hat{k}=1}^{N} \exp(\hat{e}_{i,\hat{k}})} \\ k^* &= \arg\max_k(\hat{\alpha}_{i,k}) \end{aligned} \tag{17}$$

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.

$$\hat{d}_{i,j} = \text{softmax}(W_d[\text{MLP}([h_i; h_j])] + b_d) \tag{18}$$

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, given the input graph G.

$$\mathcal{L}_g = -\sum_{i=1}^{T} \log P(s_i | s_{1:i-1}, G, \theta) \tag{19}$$

where s_i is the gold answer for i-th token, θ 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:

$$\begin{aligned} \mathcal{L}^1_l = &- \sum_{(i,j,r_{i,j})\in E} \log P(r_{i,j}|i,j,G,\theta) - \\ &\frac{1}{N}\lambda_n \sum_{(i,j,*)\notin E} \log P(r_{i,j}|i,j,G,\theta) \end{aligned} \tag{20}$$

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), λ_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.

$$\mathcal{L}^2_l = -\sum_{(i,j,r_{i,j})\in E} \log P(j|i, r_{i,j}, G, \theta) \tag{21}$$

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

$$\mathcal{L}_d = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} \log P(d_{i,j}|G,\theta) \tag{22}$$

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.

$$\mathcal{L} = \mathcal{L}_g + \lambda_l * \mathcal{L}^{1/2}_l + \lambda_d * \mathcal{L}_d \tag{23}$$

where $\mathcal{L}^{1/2}_l$ represents one form of the link prediction objective and λ_l and λ_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. λ_n is set to 0.1, λ_l is set to 0.4 and λ_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

Model	LDC2015E86			LDC2017T10		
	BLEU	Meteor	CHRF++	BLEU	Meteor	CHRF++
S2S [Konstas *et al.*, 2017]	21.7	-	-	-	-	-
Transformer [Zhu *et al.*, 2019]	25.5	33.1	59.9	27.3	34.6	61.9
GGNN [Beck *et al.*, 2018]	-	-	-	23.3	-	50.4
GGNN* [Beck *et al.*, 2018]	-	-	-	27.5	-	53.5
GraphLSTM [Song *et al.*, 2018]	23.3	-	-	-	-	-
GCNSEQ [Damonte and Cohen, 2019]	24.4	23.6	-	24.5	24.1	-
DenselyGCN [Guo *et al.*, 2019]	25.7	-	-	27.6	-	57.3
DenselyGCN* [Guo *et al.*, 2019]	28.2	-	-	30.4	-	59.6
G2S-GGNN [Ribeiro *et al.*, 2019]	24.3	30.5	-	27.9	33.2	-
StructuralTransformer-SA [Zhu *et al.*, 2019]	29.7	35.5	63.0	31.5	36.0	63.8
StructuralTransformer-CNN [Zhu *et al.*, 2019]	29.1	35.0	62.1	31.8	36.4	64.1
GraphTransformer [Cai and Lam, 2019]	27.4	32.9	56.4	29.8	35.1	59.4
Ours (w/o graph reconstruction)	30.5	35.5	63.2	32.7	36.5	64.9
Ours	**32.1**	**36.1**	**64.0**	**33.9**	**37.1**	**65.8**

Table 1: Comparison results on the test set of LDC2015E86 and LDC2017T10. * denotes the ensemble model.

Objective function	BLEU
only generation	29.8
generation + link (first form)	31.0
generation + link (second form)	30.9
generation + distance	30.5
generation + link (first form) + distance	31.3
generation + link (second form) + distance	31.4

Table 2: Ablation results on the LDC2015E86 development set

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 ($\mathcal{L}_g$); use auxiliary link prediction objective in the first form ($\mathcal{L}_g + \lambda_l * \mathcal{L}_l^1$); use link prediction objective in the second form ($\mathcal{L}_g + \lambda_l * \mathcal{L}_l^2$); use distance prediction objective ($\mathcal{L}_g + \lambda_d * \mathcal{L}_d$); use link prediction objective in the first form and distance prediction objective ($\mathcal{L}_g + \lambda_l * \mathcal{L}_l^1 + \lambda_d * \mathcal{L}_d$); use link prediction objective in the second form and distance prediction objective ($\mathcal{L}_g + \lambda_l * \mathcal{L}_l^2 + \lambda_d * \mathcal{L}_d$). 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

Metric	DGCN	GT	Ours(base)	Ours	Human
SMATCH	68.4	68.1	69.9	**70.8**	**76.3**
Unlabeled	71.9	71.3	73.6	**74.2**	**79.0**
Concepts	77.1	78.3	**80.9**	80.8	**84.8**
Reentrancies	51.7	49.8	54.9	**55.3**	**59.8**
SRL	62.0	60.8	64.8	**65.7**	**69.6**

Table 3: SMATCH and fine-grained F1 scores on the test set of LDC2015E86. DGCN denotes DenselyGCN and GT denotes GraphTransformer.

graph reconstruction objectives at the same time. These results verify the usefulness of graph structure reconstruction.

3.5 Semantic Error Analysis

We further analyze the semantic error types in the outputs of different AMR-to-text models. Similar to Konstas *et al.* [2017], we observe that there are three types of common semantic errors: 1) missing information; 2) generating words inconsistent with the given concept; 3) mixing the given semantic relations. To quantitatively compare different models, suitable metrics are needed. Considering that human evaluation is very time-consuming and requires expertise, we use an SOTA AMR parser [Zhang *et al.*, 2019] to automatically parse the outputs into graphs and compare them with the given input graphs. SMATCH [Cai and Knight, 2013] and a set of fine-grained metrics [Damonte *et al.*, 2016] are used for evaluation. These scores directly measure the degree of semantic overlap between two semantic structures, and indirectly reflect the semantic consistency between the generated sentence and the input graph. We compare the baseline DenselyGCN (DGCN), GraphTransformer (GT), our base model without graph reconstruction, our proposed model and the gold answer.

The results are listed in Table 3. Because the text and the AMR graph are not strictly one-to-one mapping and the parser is not perfect, the scores of gold answer are not 100 points. The SMATCH score reflects the overall semantic overlap and our model outperforms baselines. The Concepts score reflects the translation quality of the concepts and higher Concept score means that less concept information might be missing or mistranslated. Our model performs closely to our base model and outperforms GT and DGCN. The Unlabeled and SRL scores reflect the translation quality of semantic relations, i.e., the edges in the AMR graph. Our model achieves higher scores than baselines on these metrics but is still much lower than the gold answer, which indicates that our proposed reconstruction objectives can reduce the mixing of semantic relations to some extent. Higher Reentrancies score shows that our model can learn better representations for reentrant structure. In general, the above semantic errors occur in outputs of all compared models and our proposed model have less errors in the outputs than baselines.

4 Related Works

Most studies on AMR-to-text generation regard it as a distinct machine translation task. Early works focus on statistical methods. Flanigan *et al.* [2016] apply tree-to-str transducers to generate texts after transforming AMR graphs to trees. Pourdamghani *et al.* [2016] use a phrase-based machine translation model on the input of linearized graphs. Song *et al.* [2017] adopt a synchronous node replacement grammar to generate texts. Moving to neural machine translation methods, Konstas *et al.* [2017] achieve promising results by named entity anonymization and applying sequence-to-sequence model on the linearized graphs. Zhu *et al.* [2019] adopt the Transformer architecture and introduce Byte Pair Encoding to solve the data sparseness problem.

Recent works treat the task as a graph-to-sequence learning problem and propose various graph neural networks to tackle it. Beck *et al.* [2018] use Graph Gated Neural Network (GGNN) to directly encode the graph and an attentive decoder to generate texts. Song *et al.* propose a graph state LSTM as the encoder. Damonte and Cohen [2019] develop a hybrid neural model by stacking a BiLSTM on the output of a Graph Convolution Network (GCN) encoder. Guo *et al.* [2019] also use GCN and propose densely connected GCN to capture both local and non-local semantic relations in the graph. These methods all focus on aggregating information from one-hop neighborhoods and propagate information from distant nodes by stacking aggregation layers.

The current state-of-the-art approaches propose using relation encoders to model relation of indirectly connected concepts by encoding the shortest path between these nodes. Zhu *et al.* [2019] propose five different methods to encode the relation path and further extend the conventional self-attention architecture to explicitly utilize the encoded relation between concept pairs. Cai and Lam [2019] use bi-directional GRU encoder to get the representation for the shortest relation path.

5 Conclusion

In this paper, we propose a novel approach that utilize graph structure reconstruction for the AMR-to-text generation problem. Two simple but effective reconstruction objectives, i.e, link prediction objective and distance prediction objective, are 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

[Banerjee and Lavie, 2005] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

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

[Cai and Knight, 2013] Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, 2013.

[Cai and Lam, 2019] Deng Cai and Wai Lam. Graph transformer for graph-to-sequence learning. *arXiv preprint arXiv:1911.07470*, 2019.

[Damonte and Cohen, 2019] Marco Damonte and Shay B. Cohen. Structural neural encoders for amr-to-text generation. *arXiv preprint arXiv:1903.11410v1*, 2019.

[Damonte *et al.*, 2016] Marco Damonte, Shay B Cohen, and Giorgio Satta. An incremental parser for abstract meaning representation. *arXiv preprint arXiv:1608.06111*, 2016.

[Flanigan *et al.*, 2016] Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 731–739, 2016.

[Guo *et al.*, 2019] Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*, 7:297–312, 2019.

[Kingma and Ba, 2015] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *international conference on learning representations*, 2015.

[Konstas *et al.*, 2017] Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 146–157, 2017.

[Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[Popović, 2017] Maja Popović. chrf++: words helping character n-grams. In *Proceedings of the second conference on machine translation*, pages 612–618, 2017.

[Pourdamghani *et al.*, 2016] Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. Generating english from abstract meaning representations. In *Proceedings of the 9th international natural language generation conference*, pages 21–25, 2016.

[Ribeiro *et al.*, 2019] Leonardo FR Ribeiro, Claire Gardent, and Iryna Gurevych. Enhancing amr-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3174–3185, 2019.

[Song *et al.*, 2017] Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. Amr-to-text generation with synchronous node replacement grammar. *meeting of the association for computational linguistics*, 2:7–13, 2017.

[Song *et al.*, 2018] Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1616–1626, 2018.

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

[Veličković *et al.*, 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Zhang *et al.*, 2019] Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. Amr parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, 2019.

[Zhu *et al.*, 2019] Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. Modeling graph structure in transformer for better amr-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5462–5471, 2019.