Graph-to-Text Generation with Dynamic Structure Pruning

Liang Li[1,2], **Ruiying Geng**[3], **Bowen Li**[3], **Can Ma**[1]*, **Yinliang Yue**[1], **Binhua Li**[3], **Yongbin Li**[3]*
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]DAMO Academy, Alibaba Group
`{liliang, macan, yueyinliang}@iie.ac.cn`
`{ruiying.gry, binhua.lbh, shuide.lyb}@alibaba-inc.com`

Abstract

Most graph-to-text works are built on the encoder-decoder framework with cross-attention mechanism. Recent studies have shown that explicitly modeling the input graph structure can significantly improve the performance. However, the vanilla structural encoder cannot capture all specialized information in a single forward pass for all decoding steps, resulting in inaccurate semantic representations. Meanwhile, the input graph is flatted as an unordered sequence in the cross attention, ignoring the original graph structure. As a result, the obtained input graph context vector in the decoder may be flawed. To address these issues, we propose a Structure-Aware Cross-Attention (SACA) mechanism to re-encode the input graph representation conditioning on the newly generated context at each decoding step in a structure aware manner. We further adapt SACA and introduce its variant Dynamic Graph Pruning (DGP) mechanism to dynamically drop irrelevant nodes in the decoding process. We achieve new state-of-the-art results on two graph-to-text datasets, LDC2020T02 and ENT-DESC, with only minor increase on computational cost.

1 Introduction

Data-to-text task aims to generate a natural language description from structural or semi-structural data, such as tables (Wiseman et al., 2017), Abstract Meaning Representation (AMR) graphs (Banarescu et al., 2013), and Knowledge Graphs (KG) (Cheng et al., 2020). It helps people get the key points of the input data and makes the stored information accessible to a broader audience of end-users. There have been several practical application scenarios in this field, such as biography generation (Lebret et al., 2016), basketball news generation (Wiseman et al., 2017), and advertising text generation (Shao et al., 2019). This paper focuses on

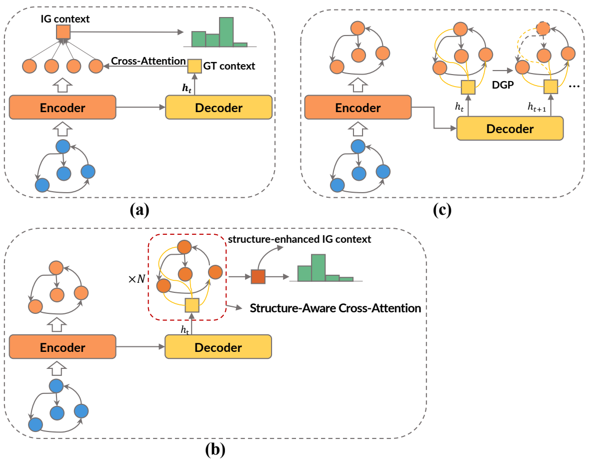


Figure 1: (a) denotes an encoder-decoder framework with the cross-attention mechanism where IG and GT contexts denote the input graph and generated text graph contexts, respectively. (b) is an example of Structure-Aware Cross-Attention. The dotted lines in (c) denote the pruned edges and nodes.

generation from graph structures in AMR and KG, referred to as graph-to-text.

In recent years, encoder-decoder with the cross-attention mechanism has been the de facto framework for graph-to-text tasks (shown in Figure 1(a)). Given an input graph, the encoder first computes vector representations for the graph nodes. On the decoding side, Input Graph (IG) context vector is obtained via cross-attention based on the partially Generated Text (GT) at each time step, then the next target token is finally predicted. Unlike conventional text-to-text tasks, the structural nature of the input graph makes it unsuitable to naively apply sequential encoder-decoder architecture to the graph-to-text task. To alleviate this issue, recent studies (Song et al., 2018; Damonte and Cohen, 2019; Cai and Lam, 2020) proposed to utilize the graph encoder to capture the input graph structure. These works have demonstrated that explicitly modeling the graph structure can bring benefits to the model performance.

*Corresponding authors: Can Ma, Yongbin Li

Although equipped with the structure-aware modeling, it is still hard for the encoder to capture all specialized information for graph-to-text generation. It is evidenced by recent studies (Liu et al., 2019; Li et al., 2021) that a vanilla structural encoder cannot capture the accurate semantic representation of the input structural data effectively. Auxiliary supervision has been shown to be helpful, but effective auxiliary tasks are not easy to design and may not generalize well to different datasets. We suspect that it is challenging for the encoder to encode all relevant information into node representations in a single forward pass for all the decoding steps, especially if the input graph structure is complex. Besides the encoder side, few works have focused on the decoder side for graph-to-text tasks. Considering the ordinary cross-attention mechanism, the representations of input data obtained from the encoder are still treated as an *unordered* node representation sequence. We conjuncture that this *plain* cross-attention does not take full advantage of the input graph structure and therefore may harm the model performance.

Current models with graph encoder and cross-attention may yield inaccurate input graph context representation due to the deficiency on both encoder and decoder as we discussed before. To tackle the above problems and avoid introducing auxiliary tasks, we propose a novel **Structure-Aware Cross-Attention** (SACA) mechanism. Apart from the *plain* cross-attention, our SACA re-encodes the input graph conditioning on the newly generated context in a *structure-aware* fashion. Other than a single forward pass, specialized representations from the source side are built adaptively at each decoding step, which makes the decoder easily exploit relevant-only information for prediction. More specifically, as shown in Figure 1(b), we construct a joint graph, in which we explicitly treat the generated text context vector as an additional node and connect it with nodes in the input graph at each decoding step. We implement SACA using the relational graph attention network (RGAT, Shaw et al. 2018). Furthermore, we stack multiple layers of SACAs to perform deep interactions between the generated text context vector and input node representations. Finally, we fetch the node representation corresponding to the newly added node as the structure-enhanced input graph context to predict the target token.

In practice, we notice that some nodes become irrelevant and uninformative as the decoding goes on. These nodes are distracting and can disturb the generation process. Intuitively, the decoder should dynamically discard the unrelated parts of the graph at different decoding steps. In other words, the joint graph structure should be dynamically adjusted. To this end, we adapt SACA and propose its variant **Dynamic Graph Pruning** (DGP) mechanism (shown in Figure 1(c)). DGP prunes the structure of the joint graph via the gate mechanism to achieve sparse connections between the nodes based on the generated text context.

We conduct experiments on two graph-to-text datasets, LDC2020T02[1] and ENT-DESC (Cheng et al., 2020), to verify the effectiveness of the proposed approach. Empirical results show that our proposed methods achieve new state-of-the-art results on the two datasets. Further experiments indicate that SACA and DGP do not reduce the diversity of the generated text and can better handle complex graphs. Meanwhile, additional investigation reveals that SACA and DGP only bring minor increase on the model size and inference time.

2 Related Works

Graph-to-text is a challenging task which aims at generating a descriptive text from the structured knowledge, such Knowledge Graph (KG), and Abstract Meaning Representation (AMR) graphs. It is helpful for interpretability of KGs in general (Schmitt et al., 2020) and knowledge-based question answering (Hui et al., 2022; Wang et al., 2022; Fu et al., 2020; Qin et al., 2022).

In recent years, most graph-to-text methods have been built based on the encoder-decoder architecture. This kind of method usually consists of a structural encoder and a decoder. The structural encoder aims to model the structure information into the representation of the input graph. Song et al. (2018) first propose the graph recurrent networks (GRNs) to encode the AMR node directly. And then, some works (Shi et al., 2020; Chen et al., 2020) introduce the Graph Neural Networks (GNNs) as the structural encoder, which updates the representations of nodes based on their immediate neighbors. To integrate both local and non-local features and learn a better structural representation of a graph, Guo et al. (2019) introduce the dense connection, allowing deeper GCNs. Unlike the local information aggregation scheme, Zhu et al.

[1]https://catalog.ldc.upenn.edu/LDC2020T02

(2019); Cai and Lam (2020) propose the Graph Transformer that uses explicit relation encoding and allows direct communication between two distant nodes.

A recently proposed neural abstractive Multi-Document Summarization (MDS) model, GraphSumm (Li et al., 2020), also considers the input graph structure during decoding. The biggest difference between Graphsum and our proposed SACA is that the former only introduces one graph attention layer in each decoder layer. SACA, on the other hand, injects graph structure into decoding by re-encoding the input graph. Specifically, it re-computes the input graph representation by conditioning it on the newly generated text at each decoding step.

Recent approaches try to apply the Pre-trained Language Models (PLMs) (Kenton and Toutanova, 2019; Raffel et al., 2019) into the graph-to-text generation. Particularly, Ribeiro et al. (2021) propose to utilize the adapter method (Pfeiffer et al., 2020) to encode graph structure into PLMs and only train graph structure-aware adapter parameters. In this way, they avoid catastrophic forgetting while maintaining the topological structure of the graph.

3 Approach

We expect that developing graph-to-text generation should benefit from the recent advance on pre-trained language models (PLMs) (Lewis et al., 2020; Raffel et al., 2019). To explicitly encode the input graph structure into PLMs while alleviating the catastrophic forgetting problem, we consider SA-RGCN (Ribeiro et al., 2021) as our baseline model. SA-RGCN is an adapter method to encode graph structure into PLMs. The overall illustration of our model architecture is shown in Figure 2(a). In this section, we first introduce how to represent the input graph and the architecture of our baseline SA-RGCN. Then, we depict our proposed Structure-Aware Cross-Attention (SACA) in details. Lastly, we adapt SACA and propose its variant Dynamic Graph Pruning (DGP).

3.1 Graph Representation

Let $\mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0, \mathcal{R}_0)$ denote a multi-relational and directed graph with nodes $v_i \in \mathcal{V}_0$ and labeled edges $(v_i, r, v_j) \in \mathcal{E}_0$, where $r \in \mathcal{R}_0$ is the relation type. Following previous work (Beck et al., 2018), we convert each input graph into a Levi graph $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$, which is an unlabeled and connected bipartite graph. Specifically, each labeled edge $(v_i, r, v_j) \in \mathcal{E}_0$ is transformed into two unlabeled edges $(v_i, r), (r, v_j) \in \mathcal{E}_l$. In addition, we add a reverse edge (v_j, v_i) for each default edge (v_i, v_j). Therefore, each Levi graph $\mathcal{G}_l$ contains two type relations $\mathcal{R}_l = \{d, r\}$, where d and r denote the default and reverse edge, respectively. To better take advantage of the PLMs, we convert each $\mathcal{G}_l$ into a new token graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where each token of a node in $\mathcal{V}_l$ becomes a node $v \in \mathcal{V}$.

3.2 Pretrained LMs with Structural Adapters

To inject graph structural bias into PLMs, we incorporate the structural adapter (Ribeiro et al., 2021) into the PLMs encoder. As shown in Figure 2 (a), we add a structural adapter after each transformer encoder block on the encoder. Figure 2 (b) illustrates the architecture of a structural adapter, in where a relational GCN (RGCN) (Schlichtkrull et al., 2018) layer computes the node representation based on the local neighborhood of node $v \in \mathcal{V}$. Formally, at each layer l, given the encoder layer representation h_v^l, a structural adapter computes the representation for v by the following:

$$g_v^l = \sum_{r \in \mathcal{R}} \sum_{u \in N_r(v)} \frac{1}{|\mathcal{N}_r(v)|} W_r^l \mathrm{LN}(h_v^l), \quad (1)$$

$$z_v^l = W_e^l(\sigma(g_v^l)) + h_v^l, \quad (2)$$

where LN$(\cdot)$ denotes layer normalization. $\mathcal{N}_r(v)$ is the sef of immediate neighbors under relation $r \in \mathcal{R}$. W_r^l encodes the edge type between the nodes u and v. σ is the activation function.

We add an FNN adapter after each transformer decoder block to adapt the language model to the graph-to-text task. Given the output $\hat{h}_v^l$ of the l th transformer decoder block, the adapter representation is computed as:

$$\hat{z}_v^l = W_o^l(\sigma(W_p^l \mathrm{LN}(\hat{h}_v^l))) + \hat{h}_v^l, \quad (3)$$

where W_o^l and W_d^l denote learnable parameters.

3.3 Structure-Aware Cross-Attention

We argue that the input graph context representation obtained by the *plain* cross-attention may be inaccurate. The reason is twofold. First, it is not easy for the graph encoder to capture all specialized information required for generation in a single forward pass. Therefore, a single encoder without any auxiliary assistant may not be effective in capturing the accurate semantic representation (Liu

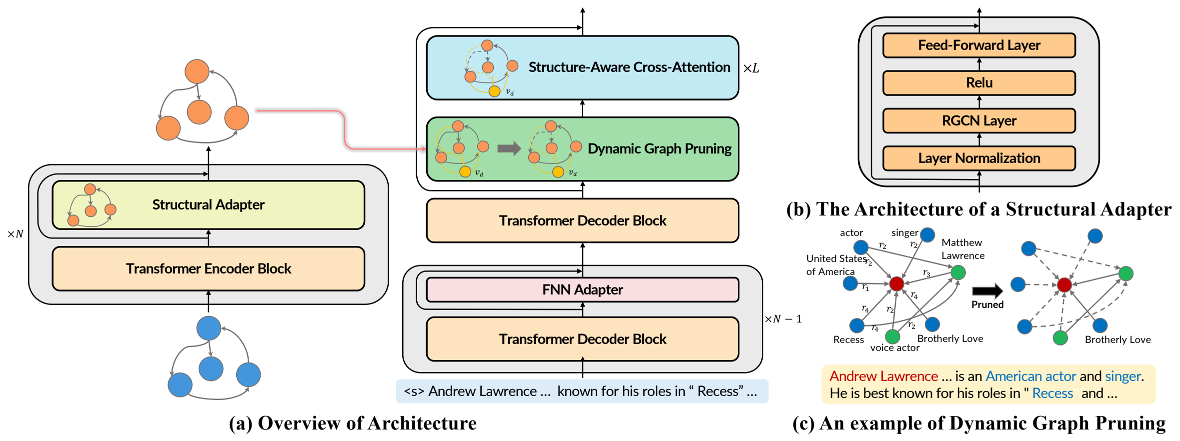

Structural Adapter
Transformer Encoder Block
×N
Structure-Aware Cross-Attention
×L
Dynamic Graph Pruning
Transformer Decoder Block
FNN Adapter
Transformer Decoder Block
×N − 1
<s> Andrew Lawrence ... known for his roles in " Recess" ...
(a) Overview of Architecture
Feed-Forward Layer
Relu
RGCN Layer
Layer Normalization
(b) The Architecture of a Structural Adapter
actor
singer
Matthew Lawrence
United States of America
Recess
voice actor
Brotherly Love
Pruned
Brotherly Love
Andrew Lawrence ... is an American actor and singer.
He is best known for his roles in " Recess and ...
(c) An example of Dynamic Graph Pruning

Figure 2: Illustration of the proposed model architecture. (a) is an overview of our model. (b) is the architecture of a structural adapter. (c) is an example of Dynamic Graph Pruning, where $r_1 \sim r_4$ denote the relations: "**country of citizenship**", "**occupation**", "**sibling**", and "**cast member**", respectively. The dummy lines in (c) denote the pruned edges.

et al., 2019; Li et al., 2021). In other words, the graph representation encoded by the graph encoder may be inaccurate. Second, during decoding, the decoder treats structural data as an unordered node sequence, which ignores the input graph structure. However, the graph structure has been proven to play an essential role in the graph representation and may offer clues about which nodes are more related to the generated text context.

To tackle the above challenge, we propose a Structure-Aware Cross-Attention (SACA) mechanism, which re-encodes the input graph representation by conditioning on the newly generated context. Specifically, we first build a joint graph, in which we view the generated text (GT) context as a new node v_d and explicitly connect it to each node in the input graph $\mathcal{G}$ at each decoding step. The corresponding reverse edges are also added. The joint graph can be formulated as $\mathcal{G}_{joint} = (\mathcal{V}_{joint}, \mathcal{E}_{joint}, \mathcal{R})$, where $\mathcal{V}_{joint} = \{v_d\} \cup \mathcal{V}$ and $\mathcal{E}_{joint} = \{(v_i, v_d), (v_d, v_i); v_i \in \mathcal{V}\} \cup \mathcal{E}$. We use the representations from the encoder for the node from $\mathcal{V}$ and the hidden state from the last transformer decoder block as the representation for the GT context node.

To induce the representations for the nodes in the joint graph $\mathcal{G}_{joint}$ and facilitate introducing Dynamic Graph Pruning (in Section 3.4), we consider graph neural network built on graph attention framework (GAT) (Shaw et al., 2018). Moreover, we employ the relational graph attention network (RGAT) implemented by Shaw et al. (2018) to model the relation between neighbor nodes. Specifically, at each RGAT layer l, we update the representation h_v^l of each node $v \in \mathcal{G}_{joint}$ by:

$$s_{v,u} = \frac{W^q {h_v^l}^T (W^k h_u^l + E^r_{\mathcal{R}(v,u)})}{\sqrt{m}}, \quad (4)$$

$$\alpha_{v,u} = \frac{e^{s_{v,u}}}{\sum_{u' \in \mathcal{N}_v} e^{s_{v,u'}}}, \quad (5)$$

$$h_v^{l+1} = \sigma(\sum_{u \in \mathcal{N}_v} \alpha_{v,u} W_v h_u^l), \quad (6)$$

where $E^r_{\mathcal{R}(v,u)}$ means the embedding of the relation between node v and u. m denotes the hidden dimension of RGAT. Finally, the representation vector $h_{v_d}^L$ corresponding to the GT context node v_d is fetched and used as the structure-enhanced input graph context vector for token prediction.

In conclusion, SACA provides two advantages. First, it re-encodes the input graph by conditioning its representation on the newly generated context. As a result, we build specialized representations which make it easier for the decoder to exploit relevant-only information for prediction at each decoding step. Second, the re-encoding explicitly injects structural bias into input graph context modeling, helping the decoder obtain a more accurate input graph context vector. The proposed SACA can be plugged after the last transformer decoder block as shown in Figure 2 (a).

3.4 Dynamic Graph Pruning

In practice, we notice that some nodes become irrelevant and uninformative as the decoding goes on. These unrelated nodes are distracting and can even disturb the subsequent generation. Intuitively, the decoder should dynamically prune the joint graph at different decoding steps. For this purpose, we adapt SACA and propose its variant Dynamic Graph Pruning (DGP) mechanism, which aims to dynamically drop the redundant nodes in the joint graph according to the generated text during decoding. The DGP employs the gate mechanism to sparse the connection between a node and its immediate neighbors in the joint graph to achieve graph pruning. Specifically, at each decoding step t, for each node v in the joint graph, we formulate its gate as bellow:

$$g_v = sigmoid(W_g^T tanh(W_e h_v + W_d h_t^d)), \quad (7)$$

where W_g, W_e, and W_d are learnable parameters. And h_v is the representation of node v and h_t^d is the decoder hidden state at decoding step t, which is usually considered as the representation of the generated text context. The value of gate $g_v \in R$ decides whether the node v_i should be dropped or not. Correspondingly, we apply the gate value to multiple SACA layers invariably by modifying the attention weights in SACA (Equation 5) as follows:

$$\alpha_{v,u} = \frac{g_u \odot e^{s_{v,u}}}{\sum_{u' \in \mathcal{N}_v} g_{u'} \odot e^{s_{v,u'}}}. \quad (8)$$

Intuitively, if the value of gate g_u is close to 0, the connections between node u with all its immediate neighbors will be largely weaken. That is, the node is removed from the joint graph. Specifically, the attention score $\alpha_{v,u}$ measures the relevance between any two nodes, v and u, in the joint graph, while the gate g_v models the relevance between the node v and the generated text context h_t.

As a shown example in Figure 2 (c), the red node represents the main entity. Initially, the main entity connects with all its neighbor nodes. As the decoding goes on, some nodes are redundant for the subsequent generation. For example, the nodes “actor“ has been described, and node “voice actor“ is also covered by the generated text. Therefore, DGP discards these nodes by giving them gates with small values.

We observed that the values of the gates calculated by Equation 7 are almost equal to 1, indicating that the model does not actively learn to prune a graph. Inspired by Xue et al. (2020), we further introduce a regularization item, encouraging the network to turn off more gates and generate more sparse connections between nodes in the input graph. We formulate it as follows:

	ENT-DESC	LDC2020T02
#train/dev/test	88,650/11,081/11,081	55,635/1,722/1,898
#relations	967	157
Avg #nodes	18.0	14.2
Avg #triples	27.4	14.8
Avg length	31.0	95.0

Table 1: Dataset statistics of ENT-DESC and LDC2020T02.

$$L_{DGP} = \frac{\sum_{t=1}^{|y|} \|Gate_t\|_1}{|y|}, \quad (9)$$

where $Gate_t = \{g_v | v \in \mathcal{V}\}$. $\|*\|_1$ means $L1$ norm regularizer.

3.5 Training

Given a reference output $y = \{y_1, y_2, ..., y_T\}$ and an input graph $\mathcal{G}$, we use the cross-entropy loss as the objective function of graph-to-text generation:

$$L_{lm} = \sum_{t=1}^{|y|} \log p(y_t | y_{1:t-1}, \mathcal{G}, \theta). \quad (10)$$

Finally, the overall objective function consists of two parts:

$$L = L_{lm} + \lambda L_{DGP}, \quad (11)$$

where λ is a tunable hyper-parameter and is used to make a trade-off between the cross-entropy loss and the regularization item. Intuitively, the L_{DGP} object encourages the model to learn how to prune the graph, and the L_{lm} trains the model to generate the text according to the graph and restrains DGP from pruning too much.

4 Experiments

4.1 Datasets

We demonstrate the effectiveness of our models on two graph-to-text datasets: LDC2020T02 and ENT-DESC (Cheng et al., 2020) LDC2020T02 is an AMR-to-Text dataset and has 55,635/1,722/1,898 instances for training, development, and testing. We follow Ribeiro et al. (2021) to preprocess the AMR graphs and tokenize the sentences. Each instance contains a sentence and an AMR graph. ENT-DESC is a large-scale and challenging dataset generating text from the Knowledge Graph (KG-to-Text). Each instance contains a KG consisting of a main entity and a few topic-related entities. The

Models	LDC2020T02				
	BLEU	METEOR	ChRF++	$\mathcal{M}$	BERTScore
LDGCN (Zhang et al., 2020b)	34.3	38.2	63.7	-	-
SPRING (Bevilacqua et al., 2021)	44.9	-	72.9	-	-
FINETUNE (Ribeiro et al., 2021)	41.6$\pm$0.6	-	70.4$\pm$0.5	78.5$\pm$0.2	96.0$\pm$0.1
ADAPT (Ribeiro et al., 2021)	43.0$\pm$0.2	-	71.3$\pm$0.1	79.3$\pm$0.1	96.2$\pm$0.1
SA-RGCN (Ribeiro et al., 2021)	48.0$\pm$0.2	-	73.2$\pm$0.1	80.1$\pm$0.3	96.3$\pm$0.1
FINETUNE‡	41.55$\pm$0.58	42.06$\pm$0.21	70.62$\pm$0.34	78.30$\pm$0.32	96.02$\pm$0.12
SA-RGCN‡	47.85$\pm$0.22	45.11$\pm$0.16	73.53$\pm$0.19	80.31$\pm$0.24	96.41$\pm$0.03
Ours	**48.78**$\pm$0.08	**46.12**$\pm$0.12	**74.35**$\pm$0.09	**80.69**$\pm$0.41	**96.62**$\pm$0.02
Models	ENT-DESC				
	BLEU	METEOR	ChRF++	ROUGE-L	PARENT
S2S (Bahdanau et al., 2015)	6.8	10.8	-	40.7	10.0
GraphTransformer (Koncel-Kedziorski et al., 2019)	19.1	16.1	-	54.3	21.4
GRN (Beck et al., 2018)	24.4	18.9	-	55.5	21.3
GCN (Marcheggiani and Perez-Beltrachini, 2018)	24.8	19.3	-	56.2	21.8
DeepGCN (Guo et al., 2019)	24.9	19.3	-	56.2	21.8
MGCN + CNN (Cheng et al., 2020)	26.4	20.4	-	57.4	24.2
FINETUNE‡	32.39$\pm$0.12	30.39$\pm$0.02	53.87$\pm$0.06	56.27$\pm$0.05	42.35$\pm$0.18
SA-RGCN‡	34.06$\pm$0.31	31.54$\pm$0.04	57.78$\pm$0.06	58.42$\pm$0.04	43.32$\pm$0.18
Ours	**34.87**$\pm$0.36	**32.37**$\pm$0.11	**58.41**$\pm$0.22	**58.97**$\pm$0.14	**43.70**$\pm$0.12

Table 2: Main results of models on LDC2020T02 and ENT-DESC test datasets. ‡ means our reimplementation. The other results are copied from the original paper. Mean ($\pm$s.d.) over 4 seeds.

target text consists of sentences that verbalize the main entity in KG. ENT-DESC lacks explicit alignment between the input and the output. Therefore, some knowledge in the input graph may be noise. We follow official training, development, and test splits of 88,650/11,081/11,081 instances. Table 1 summarizes the detailed statistics of LDC2020T02 and ENT-DESC.

4.2 Settings

Our implementation is based on Hugging Face (Wolf et al., 2019). The RGCN and RGAT are implemented based on PyTorch Geometric (Fey and Lenssen, 2019). We initialize our models by T5 (Raffel et al., 2019). To make a fair comparision, we following the same experimental setting with SA-RGCN (Ribeiro et al., 2021). We set the hidden dimensions of both *Structural Adapter* and SACA to 256. And we use T5$_{base}$ for all experiments on ENT-DESC and T5$_{large}$ on LDC2020T02 for a fair comparison with baselines. We use the AdamW optimizer (Loshchilov and Hutter, 2018) and employ a linearly decreasing learning rate schedule without warm-up. The learning rate is fixed as 10^{-4}. We set the training batch size as 4 for all experiments. We freeze the T5 parameters and only update the newly added parameters during training. We tune the hyper-parameter λ in Equation 11 from the set $[1^{-2}, 5^{-3}, 1^{-3}, 5^{-4}]$, and select the best one on the development set. We stack $L = 2$ RGAT layers in Structure-Aware Cross-Attention. During decoding, we use beam search with a beam size 5. We use BLEU (Papineni et al., 2002) for the early stopping criterion. All experiments are trained on Nvidia Tesla V100 32GB GPUs.

Following previous works, on both datasets, we evaluate the results with BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011), and ChRF++ (Popović, 2015) on both datasets. On LDC2020T02, following Ribeiro et al. (2021), we utilize the meaning ($\mathcal{M}$) component of the $\mathcal{MF}$-score (Opitz and Frank, 2021) to measure how well the source AMR graph can be reconstructed from the generated sentence (refer to A.1 for more details). We use BERTScore (Zhang et al., 2020a) allowing a semantic evaluation that depends less on the surface forms. On ENT-DESC, We add ROUGE-L (Lin, 2004) and employ PARENT (Dhingra et al., 2019) for evaluating the faithfulness. We conduct experiments over 4 different seeds and report the average scores on them.

4.3 Main Results

We compare our method with recent state-of-the-art methods (please refer to A.4 for more details). Table 2 summarizes the results on LDC2020T02 and ENT-DESC test sets. FINETUNE is a method that transforms the input graph into a sequence and finetunes T5 directly. It does not consider the input graph structure. For LDC2020T02, our method outperforms the previous state-of-the-art model by 0.78 BLEU and 1.15 ChRF++. Com-

Models	BLEU	METEOR	$\mathcal{M}$	Dis-1	Dis-2
GOLD	-	-	81.00	23.82	71.76
ADAPT	45.22	43.28	79.56	23.20	71.40
Ours	**47.85**	**45.80**	**80.37**	23.46	71.75
w/o DGP	47.68	45.51	80.21	23.51	72.08
w/o SACA & DGP	47.20	45.05	80.01	23.38	71.69
w/o StrucAdapt	45.43	43.54	79.75	23.32	71.65

Table 3: Ablation study of models on LDC2020T02 development dataset. **GOLD** indicates the ground-truth sentences. Dis-1 and Dis-2 denote Distinct1 and Distinct2, respectively.

Models	# Additional Params (million)	Latency (s)
ADAPT	28.72 (3.3%)	1.41
SA-RGCN	37.80 (4.9%)	1.49
+ SACA	39.21 (5.0%)	1.54
+ SACA & DGP	41.31 (5.0%)	1.55

Table 4: Impact on parameter and speed.

pared with our implemented SA-RGCN, we improve 1.01 METEOR. Moreover, our method raises 0.38 $\mathcal{M}$, which indicates that it can generate more faithful sentences to the input graphs. The improvement on *BERTScore* shows that the sentence generated by our method is more similar to the ground truth on the semantic level. For ENT-DESC, we notice FINETUNE performs better than all previous methods. SA-RGCN, which encodes graph structure into T5, furtherly improves the performance. And our model exceeds all previous works and achieves new state-of-the-art results on all metrics. The above results indicate that our proposed methods can improve the model on fluency and faithfulness.

4.4 Analysis and Discussion

Ablation Study The overall performance on the two datasets shows the superiority of our proposed Structure-Aware Cross-Attention (SACA) and Dynamic Graph Pruning (DGP). To demonstrate the effectiveness of each component, we conduct ablation studies on LDC2020T02 development sets and minus one particular component at a time to understand its impact on the performance. Especially, **w/o DGP** denotes we remove the dynamic graph pruning module and the training objective L_{DGP}. **ADAPT** and **w/o StrucAdapt** denote replacing each structural adapter in SA-RGCN's and our encoders with an FNN adapter, respectively. **W/o StrucAdapt** means that the model only considers the structural information during decoding. The results are summarized in Table 3. Particularly, we observe the performance drops after removing SACA or DGP. This indicates that injecting the structural information into input graph context modeling (SACA) and dynamically removing the redundant nodes (DPG) are beneficial. Regarding the $\mathcal{M}$ score, our model and **ADAPT** are close to **GOLD**. The AMR parser utilized by $\mathcal{M}$, **ADAPT** as well as our method are all initialized by T5. And the AMR paring and AMR-to-Text are dual tasks actually. Therefore, the $\mathcal{M}$ score is biased and the results of our model and **ADAPT** are somehow inflated. Additionally, we utilize Distinct-1 and Distinct-2 (Li et al., 2016) to evaluate the diversity of the output text. We observe that SACA and DGP have little effect on Distinct-1 and Distinct-2. This implies that they will not reduce the diversity of the output text.

We notice that, compared with **ADAPT**, **w/o StrucAdapt** shows a slight improvement. This indicates it is necessary to explicitly model the graph structure in the encoder, even though structural bias has been injected into the input graph context modeling during decoding. We believe this may be attributed to SACA relying on the input graph representation encoded by the encoder. Because our SACA is designed to exploit the relevant-only information for prediction, it re-encodes the input graph by conditioning its representation on the newly generated context. Therefore, the initial representation for the input graph is important.

Impact on the parameter and speed Furthermore, we investigate the impact of SACA and DPG on the model parameters and inference speed on LDC2020T02 development. Specifically, we calculate the additional parameters of each model with respect to T5$_{large}$. And we set the batch size to 1 to calculate the average decoding time for generating all examples. The results summarized in Table 4 indicate that SACA and DGP only bring minor increase on the model size and inference time.

Impact on the Graph Properties To examine the robustness of our proposed methods, we investigate the model's performance concerning different graph properties (graph size, graph diameter, and reentrancies) on LDC2020T02 and ENT-DESC. Following previous works (Cheng et al., 2020; Ribeiro et al., 2021), we use BLEU as the metric. The results are summarized in Table 5 and Table 6, respectively. For LDC2020T02, we firstly note that the BLEU scores decrease as the graph size increases since the larger graph is often

Graph Size	1-30	31-60	>60
# Examples	840	678	380
SA-RGCN	54.10	44.89	46.12
Ours	$54.55_{+0.45}$	$45.88_{+0.99}$	$46.72_{+0.60}$
Graph Diameter	1-8	9-12	>12
# Examples	824	603	471
SA-RGCN	56.98	43.12	46.07
Ours	$57.01_{+0.03}$	$43.59_{+0.47}$	$46.99_{+0.92}$
Reentrancies	<= 1	2	>2
# Examples	913	549	436
SA-RGCN	53.60	44.03	43.30
Ours	$54.16_{+0.56}$	$44.55_{+0.52}$	$44.53_{+1.23}$

Table 5: BLEU scores with respect to graph size, graph diameter and number of reentrancies on LDC2020T02 test set.

Graph Size	1-20	21-40	>40
# Examples	3,559	5,069	2,453
SA-RGCN	33.01	38.86	28.54
Ours	$33.67_{+0.66}$	$39.44_{+0.58}$	$29.02_{+0.48}$
Graph Diameter	1-3	4-5	>5
# Examples	2,227	5,017	3,787
SA-RGCN	30.52	34.41	35.83
Ours	$31.14_{+0.62}$	$34.83_{+0.45}$	$36.55_{+0.72}$
Reentrancies	< 6	6-10	>10
# Examples	2,277	5,017	3,787
SA-RGCN	27.57	36.58	37.17
Ours	$28.03_{+0.46}$	$37.17_{+0.59}$	$37.81_{+0.64}$

Table 6: BLEU scores with respect to graph size, graph diameter and number of reentrancies on ENT-DEST test set.

Models	**BLEU**	**METEOR**	**ROUGE-L**
GraphWriter	14.30	18.80	-
GraphWriter‡	14.13 ± 0.10	18.92 ± 0.28	27.61 ± 0.16
Ours	**15.59** ±0.35	**19.70** ±0.21	**28.47** ±0.14

Table 7: Generalization Study on AGENDA test dataset. ‡ means our reimplementation.

complex. Our method achieves a clear improvement when handling graphs with > 30 nodes. And then we observe that the BLEU gap between our method and SA-RGCN becomes larger for a relatively larger graph diameter. Reentrancies are the nodes with multiple parents. A graph with more reentrancies is typically more complex (Wang et al., 2020). As shown in the last section in Table 5, our method has an improvement of $+1.23$ BLEU points compared to SA-RGCN when graphs contain > 2 reentrancies. To sum up, the results on the LDC2020T02 dataset show the advantage of our model in dealing with the AMR graph with more complex structures.

As shown in Table 6, both models perform differently on ENT-DESC than on LDC2020T02. First, we notice that both models perform the best when the graph size is between 31 and 50, and they perform poorly when the graph size is too small or too large. Cheng et al. (2020) also observed the finding, and they believe this is due to the insufficient or very noisy input information for generation. Additionally, both models perform better when graph diameter or number of the reentrancies increase. The reason is that, in the ENT-DESC, the knowledge graph with a small diameter or number of the reentrancies contains more noisy information for the generation. Please refer to A.2 for more details. The BLEU gap between our method and SA-RGCN is the largest when the graph diameter > 5 or the number of reentrancies > 10. The above results demonstrate that our approach makes SA-RGCN better at handling complex knowledge graphs.

We investigate how the model behaves on different types of graphs (AMR and KG). And the results demonstrate that our model deals better with complex structures. We believe the improvement comes from two aspects. First, on the one hand, it is challenging for an encoder to encode all relevant information into node representations in a single forward pass, especially if the graph structure is complex. On the other hand, the re-encoding in SACA makes the decoder easily exploit the relevant-only information for prediction and explicitly injects the structural information at each decoding step. Second, DGP dynamically removes the nodes which are redundant for the subsequent generation, which makes the decoder pay more attention to the relevant nodes.

4.5 Generalization Study

Institutionally, our proposed methods can not only be applied to PLMs but also RNN based models. In other words, we can easily combine the SACA and DGP with previous RNN based works. To examine the generalization of SACA and DGP, we choose GraphWriter (Koncel-Kedziorski et al., 2019) as the baseline, which consists of a multi-layer graph transformer encoder and an attention-based decoder with a copy mechanism. Further, to make a fair comparison, we conduct the generalization experiment on AGENDA dataset (Koncel-Kedziorski et al., 2019). We simply replace the *plain* cross-attention in GraphWriter with our proposed SACA. Additionally, we add the DGP layer before the SACA. The experiments are under the same settings as described in GraphWriter. As

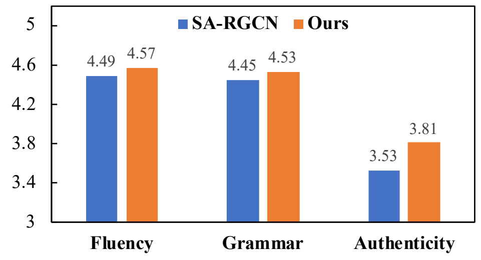


Figure 3: Human evaluation results on ENT-DESC test set.

shown in Table 7, we observe that our proposed model significantly improves the performance of GraphWriter. The results indicate that SACA and DGP are not only effective well on PLMs-based models but also potent for RNN-based models.

4.6 Human Evaluation

Considering that the knowledge graph is more readable than AMR, we do human evaluations on the ENT-DESC test set to examine whether human judgments corroborate improvements in automatic evaluation metrics. Following Cheng et al. (2020), from outputs generated by the baseline model SA-RGCN and our final model (Ours). We distribute the outputs of different systems to three annotators with linguistic backgrounds. The annotators have no knowledge in advance about which model the generated text comes from. Specifically, we give each participant all main entities' neighbors, 1-hop and 2-hop connections between main entities, and topic-related entities as references. They are required to score the generated text from 1 to 5 in terms of three criteria: Fluency (is the sentence fluent?), Grammar (is the sentence grammatical?), and Authenticity (is the sentence more related to the input graph?). For each criterion, we calculate the final score by averaging the scores from all annotators. As shown in Figure 3, our model outperforms the baseline SA-RGCN on Fluency and Grammar metrics. For Authenticity, the improvement is more significant. The performance validates the benefit of our proposed SACA and DGP modules in capturing more accurate input graph context representations. We supply a case study in A.3.

5 Conclusions

In this work, we make two main contributions. First, we propose Structure-Aware Cross-Attention (SACA) to make decoder easily exploit relevant-only information for prediction. Apart from the plain cross-attention, SACA re-encodes the input graph conditioning on the newly generated context while explicitly considering the input graph structure. The second one is that we adapt SACA and propose its variant Dynamic Graph Pruning (DGP) mechanism. In detail, the DGP dynamically prunes the structure of the joint graph at different decoding steps according to the generated text. Experimental results conducted on two graph-to-text datasets, LDC2020T02 and ENT-DESC, show the effectiveness of our method. The empirical and analysis results on both datasets show that the proposed methods can improve the model's performance on complex graphs while only bringing minor increase on the model size and inference time.

References

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *ACL*.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Proc. of AAAI*.

Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. In *Proc. of AAAI*.

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020. Kgpt: Knowledge-grounded pre-training for data-to-text generation.

Liying Cheng, Dekun Wu, Lidong Bing, Yan Zhang, Zhanming Jie, Wei Lu, and Luo Si. 2020. ENT-DESC: Entity description generation by exploring knowledge graph. In *EMNLP*.

Marco Damonte and Shay B Cohen. 2019. Structural neural encoders for amr-to-text generation. In *Proc. of AACL*.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the sixth workshop on statistical machine translation*.

Bhuwan Dhingra, Manaal Faruqui, Ankur P. Parikh, Ming-Wei Chang, Dipanjan Das, and William W. Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proc. of ACL*.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Bin Fu, Yunqi Qiu, Chengguang Tang, Yang Li, Haiyang Yu, and Jian Sun. 2020. A survey on complex question answering over knowledge base: Recent advances and challenges. *arXiv preprint arXiv:2007.13069*.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*.

Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, and Yongbin Li. 2022. S^{2}SQL: Injecting syntax to question-schema interaction graph encoder for text-to-SQL parsers. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1254–1262, Dublin, Ireland. Association for Computational Linguistics.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of AACL*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proc. of AACL*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proc. of NAACL*.

Liang Li, Can Ma, Yinliang Yue, and Dayong Hu. 2021. Improving encoder by auxiliary supervision tasks for table-to-text generation. In *ACL*.

Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. 2020. Leveraging graph to improve abstractive multi-document summarization. In *Proc. of ACL*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*.

Tianyu Liu, Fuli Luo, Qiaolin Xia, Shuming Ma, Baobao Chang, and Zhifang Sui. 2019. Hierarchical encoder with auxiliary supervision for neural table-to-text generation: Learning better representation for tables. In *Proc. of AAAI*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *ICLR*.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*.

Juri Opitz and Anette Frank. 2021. Towards a decomposable metric for explainable evaluation of text generation from AMR. In *Proc. of EACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *EMNLP*.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*.

Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, et al. 2022. A survey on text-to-sql parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. Structural adapters in pretrained language models for AMR-to-Text generation. In *Proc. of EMNLP*.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*.

Martin Schmitt, Sahand Sharifzadeh, Volker Tresp, and Hinrich Schütze. 2020. An unsupervised joint system for text generation from knowledge graphs and semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7117–7130, Online. Association for Computational Linguistics.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019. Long and diverse text generation with planning-based hierarchical variational model. In *EMNLP-IJCNLP*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proc. of NAACL*.

Yunzhou Shi, Zhiling Luo, Pengcheng Zhu, Feng Ji, Wei Zhou, Haiqing Chen, and Yujiu Yang. 2020. G2t: Generating fluent descriptions for knowledge graph. In *Proc. of SIGIR*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. In *ACL*.

Lihan Wang, Bowen Qin, Binyuan Hui, Bowen Li, Min Yang, Bailin Wang, Binhua Li, Jian Sun, Fei Huang, Luo Si, et al. 2022. Proton: Probing schema linking information from pre-trained language models for text-to-sql parsing. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1889–1898.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. Amr-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *Proc. of EMNLP*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Lanqing Xue, Xiaopeng Li, and Nevin L Zhang. 2020. Not all attention is needed: Gated attention network for sequence data. In *Proc. of AAAI*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020a. Bertscore: Evaluating text generation with BERT. In *Proc. of ICLR*.

Yan Zhang, Zhijiang Guo, Zhiyang Teng, Wei Lu, Shay B Cohen, Zuozhu Liu, and Lidong Bing. 2020b. Lightweight, dynamic graph convolutional networks for amr-to-text generation. In *EMNLP*.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better amr-to-text generation. In *EMNLP-IJCNLP*.

A Appendix

A.1 $\mathcal{MF}$-score

The $\mathcal{M}$ (Meaning Preservation) component of the $\mathcal{MF}$-score (Opitz and Frank, 2021) is utilized to measure how well the source AMR graph can be reconstructed from the generated sentence. It reconstructs the AMR with a SOTA parser and computes the relative graph overlap of the reconstruction and the source AMR using graph matching. $\mathcal{M}$ employs the python library amrlib[2] (version 0.5.0) to make AMR parse, where the parser is a T5-based model.

A.2 Distribution on Graph Size

On the ENT-DESC test set, previous study (Cheng et al., 2020) and our experimental results (in Table 6) suggest that the model performs the best when the graph size lies in the range of $21 - 40$ and has a poorer performance when the number of triples is too small or too large. It should be due to the fact that the input information is insufficient or very noisy. However, we find that the model performance increases as the graph diameter and reentrancies increase. For further investigation, we calculate the distribution of graph diameter and reentrancies broken down by graph size, respectively. The results are summarized in Figure 4. As shown in Figure 4(a), the proportion of graphs with size $21 - 40$ increases as the graph diameter increases. As shown in Figure 4(b), the results on graph reentrancy follow a pattern similar to graph diameter. In a word, in ENT-DESC, the noise decreases as the graph diameter and reentrancies increase, so the model performs better.

A.3 Case Study

As shown in Figure 5, we further take a typical example from our human study to better understand how our method improves the mode's performance. Given the Knowledge Graph containing the main entity “Andrew Lawrence" and all its related entities, we aim to generate a description about the main entity. We notice that both the baseline and our model can identify the main entity. However, the baseline outputs a sentence describing the relation between “Andrew Lawrence" and “Matthew Lawrence". The relation is not existing in the input graph. Moreover, it repeatedly generates the entity “Brotherly Love" and misses the related entity “Recess". Compared with it, our model generates the sentences faithful to the input graph and correctly covers the main entity and most topic-related entities. We consider this is because the SACA helps the decoder obtain a more accurate input graph context, and the DGP removes the redundant nodes as the decoding stage progresses.

[2]https://github.com/bjascob/amrlib/tree/0.5.0

A.4 Baseline Models

On the AMR-to-Text task LDC2020T02, we compare our method with several baselines including:

- **LDGCN** (Zhang et al., 2020b) is a a dynamic fusion mechanism, which captures richer non-local interactions by synthesizing higher order information from the input graphs. A weight tied convolutions to reduce memory usage is applied.

- **SPRING** (Bevilacqua et al., 2021) casts Text-to-AMR and AMR-to-Text as a symmetric transduction task and proposes a graph linearization and extending a pretrained encoder-decoder model.

On the KG-to-Text task ENT-DESC, we compare our method with several baselines including:

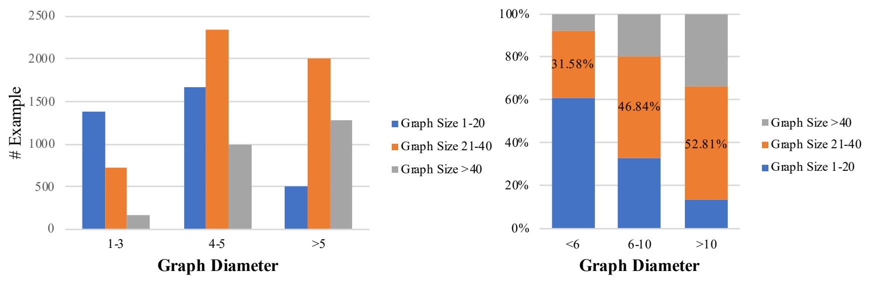
- **s2s** (Bahdanau et al., 2015) is a encoder-decoder based model, which allows a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly.

- **GraphTransformer** (Koncel-Kedziorski et al., 2019) introduces a novel graph transforming encoder which can leverage the relational structure of such knowledge graphs without imposing linearization or hierarchical constraints.

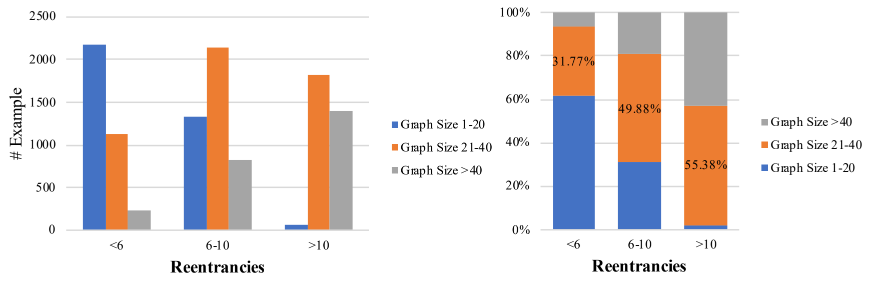
- **GRN** (Beck et al., 2018) couples the recently proposed Gated Graph Neural Networks with an input transformation that allows nodes and edges to have their own hidden representations.

- **GCN** (Marcheggiani and Perez-Beltrachini, 2018) proposes an alternative encoder based on graph convolutional networks that directly exploits the input structure.

- **DeepGCN** (Guo et al., 2019) introduces a dense connection strategy, which is able to integrate both local and non-local features to



(a) The distribution of graph diameter by graph size.



(b) The distribution of graph reentrancies by graph size.

Figure 4: The clustered column charts of graph diameter and reentrancies by graph size.

Gold	Andrew James Lawrence (born January 12, 1988) is an American actor and singer. He is known for his roles as Andy Roman in "Brotherly Love" and T.J. Detweiler in "Recess".
SA-RGAT	Andrew Lawrence (born January 12, 1988) is an American actor, voice actor, and singer. He is best known for his roles in the films "Brotherly Love" and "Brotherly Love". He is the younger brother of Matthew Lawrence.
Ours	Andrew Lawrence (born January 12, 1988) is an American actor and singer. He is best known for his roles in "Recess" and "Brotherly Love".

Figure 5: An example of generated sentences. The main entity is highlighted in red, topic-related entities are highlighted in blue, and the sentence that is not faithful to the input graph is in green.

learn a better structural representation of a graph.

- **MGCN + CNN** (Cheng et al., 2020) is a multi-graph structure that is able to represent the original graph information more comprehensively. We do not report the results of MGCN + CNN + delex. Because it applies the delexicalization technique on the ENT-DESC dataset, which delexicalizes the main entity and topic-related entities by replacing these entities with tokens indicating the entity types and indices. The delexicalization technique greatly boosts their performance on ROUGE-L. They do not release the code about delexicalization, and we can not reproduce it.

What's more, FINETUNE, ADAPT and SA-RGCN are T5-based models proposed in (Ribeiro et al., 2021).