Modeling Graph Structure in Transformer for Better AMR-to-Text Generation

Jie Zhu[1] **Junhui Li**[1*] **Muhua Zhu**[2]
Longhua Qian[1] **Min Zhang**[1] **Guodong Zhou**[1]
[1]School of Computer Science and Technology, Soochow University, Suzhou, China
[2]Alibaba Group, Hangzhou, China
zhujie951121@gmail.com, {lijunhui, qianlonghua, minzhang, gdzhou}@suda.edu.cn
muhua.zmh@alibaba-inc.com

Abstract

Recent studies on AMR-to-text generation often formalize the task as a sequence-to-sequence (seq2seq) learning problem by converting an Abstract Meaning Representation (AMR) graph into a word sequence. Graph structures are further modeled into the seq2seq framework in order to utilize the structural information in the AMR graphs. However, previous approaches only consider the relations between directly connected concepts while ignoring the rich structure in AMR graphs. In this paper we eliminate such a strong limitation and propose a novel structure-aware self-attention approach to better modeling the relations between indirectly connected concepts in the state-of-the-art seq2seq model, i.e., the *Transformer*. In particular, a few different methods are explored to learn structural representations between two concepts. Experimental results on English AMR benchmark datasets show that our approach significantly outperforms the state of the art with 29.66 and 31.82 BLEU scores on LDC2015E86 and LDC2017T10, respectively. To the best of our knowledge, these are the best results achieved so far by supervised models on the benchmarks.

1 Introduction

AMR-to-text generation is a task of automatically generating a natural language sentence from an Abstract Meaning Representation (AMR) graph. Due to the importance of AMR as a widely adopted semantic formalism in representing the meaning of a sentence (Banarescu et al., 2013), AMR has become popular in semantic representation and AMR-to-text generation has been drawing more and more attention in the last decade. As the example in Figure 1(a) shows, nodes, such as *he* and *convict-01*, represent semantic concepts and edges, such as ":ARG1" and ":quant", refer to semantic relations between the concepts. Since two concepts close in an AMR graph may map into two segments that are distant in the corresponding sentence, AMR-to-text generation is challenging. For example in Figure 1, the neighboring concepts *he* and *convict-01* correspond to the words *he* and *convicted* which locate at the different ends of the sentence.

To address the above mentioned challenge, recent studies on AMR-to-text generation regard the task as a sequence-to-sequence (seq2seq) learning problem by properly linearizing an AMR graph into a sequence (Konstas et al., 2017). Such an input representation, however, is apt to lose useful structural information due to the removal of reentrant structures for linearization. To better model graph structures, previous studies propose various graph-based seq2seq models to incorporate graphs as an additional input representation (Song et al., 2018; Beck et al., 2018; Damonte and Cohen, 2019). Although such graph-to-sequence models can achieve the state-of-the-art results, they focus on modeling one-hop relations only. That is, they only model concept pairs connected directly by an edge (Song et al., 2018; Beck et al., 2018), and as a result, ignore explicit structural information of indirectly connected concepts in AMR graphs, e.g. the relation between concepts *he* and *possible* in Figure 1.

To make better use of structural information in an AMR graph, we attempt to model arbitrary concept pairs no matter whether directly connected or not. To this end, we extend the encoder in the state-of-the-art seq2seq model, i.e., the Transformer (Vaswani et al., 2017) and propose structure-aware self-attention encoding approach. In particular, several distinct methods are proposed to learn structure representations for the new self-attention mechanism.

*Corresponding Author: Junhui Li.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 5459–5468,
Hong Kong, China, November 3–7, 2019.

Empirical studies on two English benchmarks show that our approach significantly advances the state of the art for AMR-to-text generation, with the performance improvement of 4.16 BLEU score on LDC2015E86 and 4.39 BLEU score on LDC2017T10 respectively over the strong baseline. Overall, this paper makes the following contributions.

- To the best of our knowledge, this is the first work that applies the Transformer to the task of AMR-to-text generation. On the basis of the Transformer, we build a strong baseline that reaches the state of the art.
- We propose a new self-attention mechanism to incorporate richer structural information in AMR graphs. Experimental results on two benchmarks demonstrate the effectiveness of the proposed approach.
- Benefiting from the strong baseline and the structure-aware self-attention mechanism, we greatly advance the state of the art in the task.

2 AMR-to-Text Generation with Graph Structure Modeling

We start by describing the implementation of our baseline system, a state-of-the-art seq2seq model which is originally used for neural machine translation and syntactic parsing (Vaswani et al., 2017). Then we detail the proposed approach to incorporating structural information from AMR graphs.

2.1 Transformer-based Baseline

Transformer: Our baseline system builds on the Transformer which employs an encoder-decoder framework, consisting of stacked encoder and decoder layers. Each encoder layer has two sublayers: self-attention layer followed by a position-wise feed forward layer. Self-attention layer employs multiple attention heads and the results from each attention head are concatenated and transformed to form the output of the self-attention layer. Each attention head uses scaled dot-product attention which takes a sequence $x = (x_1, \cdots, x_n)$ of n elements as input and computes a new sequence $z = (z_1, \cdots, z_n)$ of the same length:

$$z = Attention\,(x) \quad (1)$$

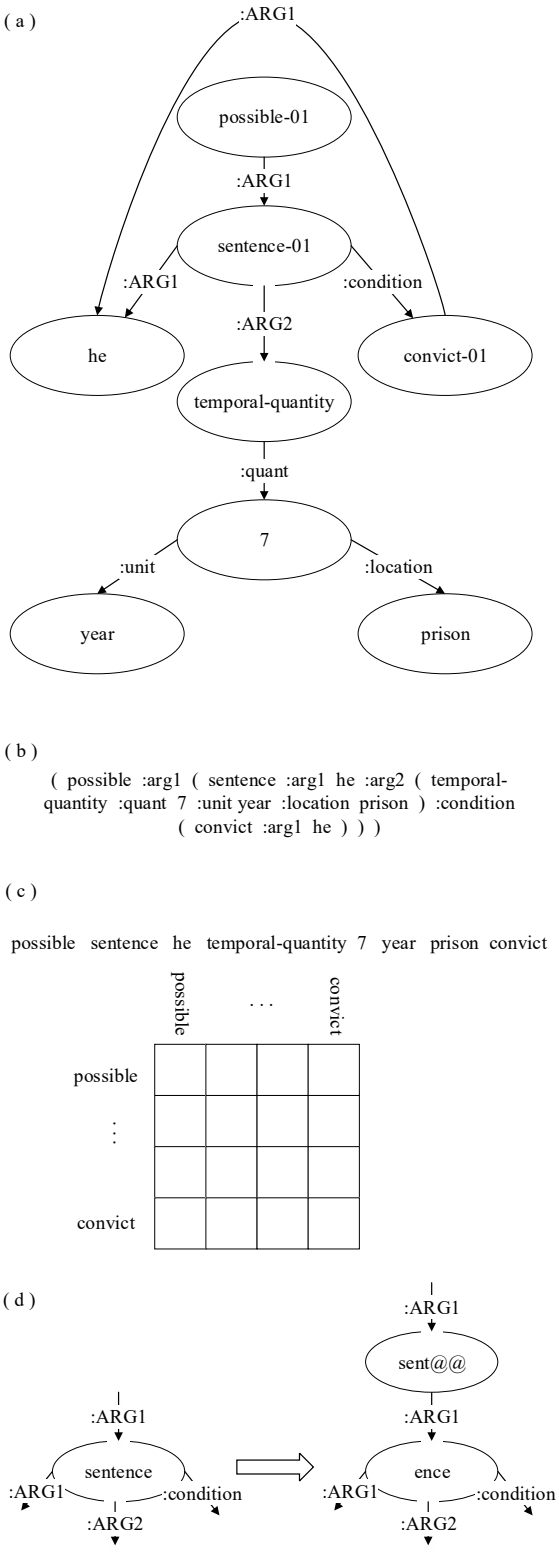

Figure 1: (a) An example of AMR graph for the sentence of *He could be sentenced to 7 years in prison if convicted.* (b) input to our baseline system, the seq2seq Transformer. (c) input to our proposed system based on structure-aware self-attention. (d) An example of graph structure extensions to sub-word units.

where $x_i \in \mathbb{R}^{d_x}$ and $z \in \mathbb{R}^{n \times d_z}$. Each output element z_i is a weighted sum of a linear transformation of input elements:

$$z_i = \sum_{j=1}^{n} \alpha_{ij} \left(x_j W^V \right) \quad (2)$$

where $W^V \in \mathbb{R}^{d_x \times d_z}$ is matrix of parameters. The vectors $\alpha_i = (\alpha_{i1}, \cdots, \alpha_{in})$ in Equation 2 are obtained by the self-attention model, which captures the correspondences between x_i and others. Specifically, the attention weight α_{ij} of each element x_j is computed using a softmax function:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{n} exp(e_{ik})} \quad (3)$$

where

$$e_{ij} = \frac{\left(x_i W^Q \right) \left(x_j W^K \right)^T}{\sqrt{d_z}} \quad (4)$$

is an alignment function which measures how well the input elements x_i and x_j match. $W^Q, W^K \in \mathbb{R}^{d_x \times d_z}$ are parameters to be learned.

Input Representation: We use the depth-first traversal strategy as in Konstas et al. (2017) to linearize AMR graphs and to obtain simplified AMRs. We remove variables, wiki links and sense tags before linearization. Figure 1(b) shows an example linearization result for the AMR graph in Figure 1(a). Note that the reentrant concept *he* in Figure 1 (a) maps to two different tokens in the linearized sequence.

Vocabulary: Training AMR-to-text generation systems solely on labeled data may suffer from data sparseness. To attack this problem, previous works adopt techniques like anonymization to remove named entities and rare words (Konstas et al., 2017), or apply a copy mechanism (Gulcehre et al., 2016) such that the models can learn to copy rare words from the input sequence. In this paper we instead use two simple yet effective techniques. One is to apply Byte Pair Encoding (BPE) (Sennrich et al., 2016) to split words into smaller, more frequent sub-word units. The other is to use a shared vocabulary for both source and target sides. Experiments in Section 3.2 demonstrate the necessity of the techniques in building a strong baseline.

2.2 Modeling Graph Structures in Transformer

Input Representation: We also use the depth-first traversal strategy to linearize AMR graphs and to obtain simplified AMRs which only consist of concepts. As shown in Figure 1 (c), the input sequence is much shorter than the input sequence in the baseline. Besides, we also obtain a matrix which records the graph structure between every concept pair, which implies their semantic relationship (Section2.3).

Vocabulary: To be compatible with sub-words, we extend the original AMR graph, if necessary, to include the structures of sub-words. As *sentence-01* in Figure 1(a) is segmented into *sent@@ ence-01*, we split the original node into two connected ones with an edge labeled as the incoming edge of the first unit. Figure 1(d) shows the graph structure for sub-words *sent@@ ence-01*.

Structure-Aware Self-Attention: Motivated by Shaw et al. (2018), we extend the conventional self-attention architecture to explicitly encode the relation between an element pair (x_i, x_j) in the alignment model by replacing Equation 4 with Equation 5. Note that the relation $r_{ij} \in \mathbb{R}^{d_z}$ is the vector representation for element pair (x_i, x_j), and will be learned in Section 2.3.

$$e_{ij} = \frac{\left(x_i W^Q \right) \left(x_j W^K + r_{ij} W^R \right)^T}{\sqrt{d_z}} \quad (5)$$

where $W^R \in \mathbb{R}^{d_z \times d_z}$ is a parameter matrix. Then, we update Equation 2 accordingly to propagate structure information to the sublayer output by:

$$z_i = \sum_{j=1}^{n} \alpha_{ij} \left(x_j W^V + r_{ij} W^F \right) \quad (6)$$

where $W^F \in \mathbb{R}^{d_z \times d_z}$ is a parameter matrix.

2.3 Learning Graph Structure Representation for Concept Pairs

The above structure-aware self-attention is capable of incorporating graph structure between concept pairs. In this section, we explore a few methods to learn the representation for concept pairs. We use a sequence of edge labels, along the path from x_i to x_j to indicate the AMR graph structure between concepts x_i and x_j.[1] In order to distinguish the edge direction, we add a direction symbol to each label with $\uparrow$ for climbing up along the path, and $\downarrow$ for going down. Specifically, for the special case of $i == j$, we use *None* as the path. Table 1 demonstrates structural label sequences between a few concept pairs in Figure 1.

[1]While there may exist two or more paths connecting x_i and x_j, we simply choose the shortest one.

x_i	x_j	Structural label sequence
he	convict-01	:ARG1↑
he	7	:ARG1↑ :ARG2↓ :quant↓
he	he	None

Table 1: Examples of structural path between a few concept pairs in Figure 1.

Now, given a structural path with a label sequence $s = s_1, \cdots, s_k$ and its d_x-sized corresponding label embedding sequence $l = l_1, \cdots, l_k$, we use the following methods to obtain its representation vector r, which maps to r_{ij} in Equation 5 and Equation 6.

Feature-based

A natural way to represent the structural path is to view it as a string feature. To this end, we combine the labels in the structural path into a string. Unsurprisingly, this will end up with a large number of features. We keep the most frequent ones (i.e., 20K in our experiments) in the feature vocabulary and map all others into a special feature *UNK*. Each feature in the vocabulary will be mapped into a randomly initialized vector.

Avg-based

To overcome the data sparsity in the above feature-based method, we view the structural path as a label sequence. Then we simply use the averaged label embedding as the representation vector of the sequence, i.e.,

$$r = \frac{\sum_{i=1}^{k} l_i}{k} \tag{7}$$

Sum-based

Sum-based method simply returns the sum of all label embeddings in the sequence, i.e.,

$$r = \sum_{i=1}^{k} l_i \tag{8}$$

Self-Attention-based (SA-based for short)

As shown in Figure 2, given the label sequence $s = s_1, \cdots, s_k$, we first obtain the sequence e, whose element is the addition of a word embedding and the corresponding position embedding. Then we use the self-attention, as presented in Eq. 1 to obtain its hidden states h, i.e, $h = Attention(e)$, where $h_i \in \mathbb{R}^{d_z}$. Our aim is to encode a variable length sentence into a d_z-sized vector. Motivated by (Lin et al., 2017), we achieve this by choosing a linear combination of

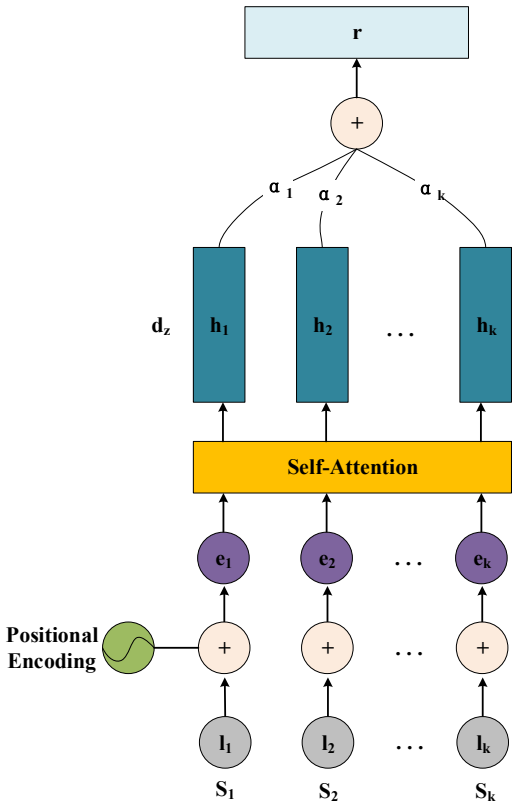

Figure 2: Self-Attention-based method.

the k vectors in h. Computing the linear combination requires an attention mechanism which takes the whole hidden states h as input, and outputs a vector of weights α:

$$\alpha = softmax(W^2 \tanh(W^1 h^T)) \tag{9}$$

where $W^1 \in \mathbb{R}^{d_w \times d_z}$ and $W^2 \in \mathbb{R}^{d_w}$. Then the label sequence representation vector is the weighted sum of its hidden states:

$$r = \sum_{i=1}^{k} \alpha_i h_i \tag{10}$$

CNN-based

Motivated by (Kalchbrenner et al., 2014), we use convolutional neural network (CNN) to convolute the label sequence l into a vector r, as follow:

$$\begin{aligned} conv = Conv1D(&kernel\_size = (m), \\ &strides = 1, \\ &filters = d_z, \\ &input\_shape = d_z \\ &activation =' relu') \end{aligned} \tag{11}$$

$$r = conv\,(l) \tag{12}$$

where kernel size m is set to 4 in our experiments.

3 Experimentation

3.1 Experimental Settings

For evaluation of our approach, we use the sentences annotated with AMRs from the LDC release LDC2015E86 and LDC2017T10. The two datasets contain 16,833 and 36,521 training AMRs, respectively, and share 1,368 development AMRs and 1,371 testing AMRs. We segment words into sub-word units by BPE (Sennrich et al., 2016) with 10K operations on LDC2015E86 and 20K operations on LDC2017T10.

For efficiently learning graph structure representation for concept pairs (except the feature-based method), we limit the maximum label sequence length to 4 and ignore the labels exceeding the maximum. In SA-based method, we set the filter size d_w as 128.

We use *OpenNMT* (Klein et al., 2017) as the implementation of the Transformer seq2seq model.[2] In parameter setting, we set the number of layers in both the encoder and decoder to 6. For optimization we use Adam with $\beta 1 = 0.1$ (Kingma and Ba, 2015). The number of heads is set to 8. In addition, we set the embedding and the hidden sizes to 512 and the batch token-size to 4096. Accordingly, the d_x and d_z in Section 2 are 64. In all experiments, we train the models for 300K steps on a single K40 GPU.

For performance evaluation, we use BLEU (Papineni et al., 2002), Meteor (Banerjee and Lavie, 2005; Denkowski and Lavie, 2014), and CHRF++ (Popovi, 2017) as metrics. We report results of single models that are tuned on the development set.

We make our code available at `https://github.com/Amazing-J/structural-transformer`.

3.2 Experimental Results

We first show the performance of our baseline system. As mentioned before, BPE and sharing vocabulary are two techniques we applied to relieving data sparsity. Table 2 presents the results of the ablation test on the development set of LDC2015E86 by either removing BPE, or vocabulary sharing, or both of them from the baseline system. From the results we can see that BPE and vocabulary sharing are critical to building our baseline system (an improvement from 18.77 to 24.93 in BLEU), revealing the fact that they are two effective ways to address the issue of data sparseness for AMR-to-text generation.

Model	BLEU	Meteor	CHRF++
Baseline	24.93	33.20	60.30
-BPE	23.02	31.60	58.09
-Share Vocab.	23.24	31.78	58.43
-Both	18.77	28.04	51.88

Table 2: Ablation results of our baseline system on the LDC2015E86 development set.

Table 3 presents the comparison of our approach and related works on the test sets of LDC2015E86 and LDC2017T10. From the results we can see that the Transformer-based baseline outperforms most of graph-to-sequence models and is comparable with the latest work by Guo et al. (2019). The strong performance of the baseline is attributed to the capability of the Transformer to encode global and implicit structural information in AMR graphs. By comparing the five methods of learning graph structure representations, we have the following observations.

- All of them achieve significant improvements over the baseline: the biggest improvements are 4.16 and 4.39 BLEU scores on LDC2015E86 and LDC2017T10, respectively.
- Methods using continuous representations (such as SA-based and CNN-based) outperform the methods using discrete representations (such as feature-based).
- Compared to the baseline, the methods have very limited affect on the sizes of model parameters (see the column of *#P (M)* in Table 3).

Finally, our best-performing models are the best among all the single and supervised models.

4 Analysis

In this section, we use LDC2017T10 as our benchmark dataset to demonstrate how our proposed approach achieves higher performance than the baseline. As representative, we use CNN-based method to obtain structural representation.

[2] `https://github.com/OpenNMT/OpenNMT-py`

System		LDC2015E86				LDC2017T10		
		BLEU	Meteor	CHRF++	#P (M)	BLEU	Meteor	CHRF++
Baseline		25.50	33.16	59.88	49.1	27.43	34.62	61.85
Our Approach	feature-based	27.23	34.53	61.55	49.4	30.18	35.83	63.20
	avg-based	28.37	35.10	62.29	49.1	29.56	35.24	62.86
	sum-based	28.69	34.97	62.05	49.1	29.92	35.68	63.04
	SA-based	**29.66**	**35.45**	**63.00**	49.3	31.54	36.02	63.84
	CNN-based	29.10	35.00	62.10	49.2	**31.82**	**36.38**	**64.05**
Previous works with single models								
Konstas et al. (2017)*		22.00	-	-	-	-	-	-
Cao and Clark (2019)*		23.5	-	-	-	26.8	-	-
Song et al. (2018)†		23.30	-	-	-	-	-	-
Beck et al. (2018)†		-	-	-	-	23.3	-	50.4
Damonte and Cohen (2019)†		24.40	23.60	-	-	24.54	24.07	-
Guo et al. (2019)†		25.7	-	-	-	27.6	-	57.3
Song et al. (2016)‡		22.44	-	-	-	-	-	-
Previous works with either ensemble models or unlabelled data, or both								
Konstas et al. (2017)*		33.8	-	-	-	-	-	-
Song et al. (2018)†		33.0	-	-	-	-	-	-
Beck et al. (2018)†		-	-	-	-	27.5	-	53.5
Guo et al. (2019)†		35.3	-	-	-	-	-	-

Table 3: Comparison results of our approaches and related studies on the test sets of LDC2015E86 and LDC2017T10. #P indicates the size of parameters in millions. * indicates seq2seq-based systems while † for graph-based models, and ‡ for other models. All our proposed systems are significant over the baseline at 0.01, tested by bootstrap resampling (Koehn, 2004).

System	BLEU
Baseline	27.43
Our approach	31.82
No indirectly connected concept pairs	29.92

Table 4: Performance on the test set of our approach with or without modeling structural information of indirectly connected concept pairs.

4.1 Effect of Modeling Structural Information of Indirectly Connected Concept Pairs

Our approach is capable of modeling arbitrary concept pairs no matter whether directly connected or not. To investigate the effect of modeling structural information of indirectly connected concept pairs, we ignore their structural information by mapping all structural label sequences between two indirectly connected concept pairs into *None*. In this way, the structural label sequence for *he* and *7* in Table 1, for example, will be *None*.

Table 4 compares the performance of our approach with or without modeling structural information of indirectly connected concept pairs. It shows that by modeling structural information of indirectly connected concept pairs, our approach improves the performance on the test set from 29.92 to 31.82 in BLEU scores. It also shows that even without modeling structural information of indirectly connected concept pairs, our approach achieves better performance than the baseline.

4.2 Effect on AMR Graphs with Different Sizes of Reentrancies

Linearizing an AMR graph into a sequence unavoidably loses information about reentrancies (nodes with multiple parents). This poses a challenge for the baseline since there exists on obvious sign that the first *he* and the second *he*, as shown in Figure 1 (b), refer to the same person. By contrast, our approach models reentrancies explicitly. Therefore, it is expected that the benefit of our approach is more evident for those AMR graphs containing more reentrancies. To test this hypothesis, we partition source AMR graphs to different groups by their numbers of reentrancies and evaluate their performance respectively. As shown in Figure 3, the performance gap be-

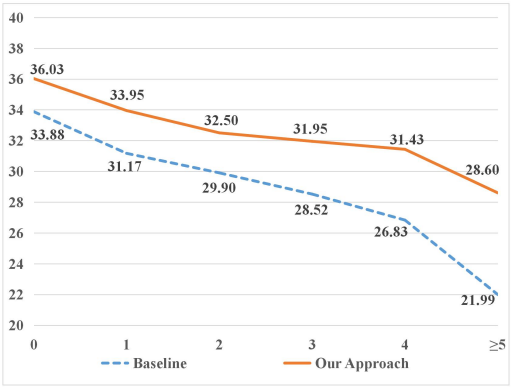

Figure 3: Performance (in BLEU) on the test set with respect to the reentrancy numbers of the input AMR graphs.

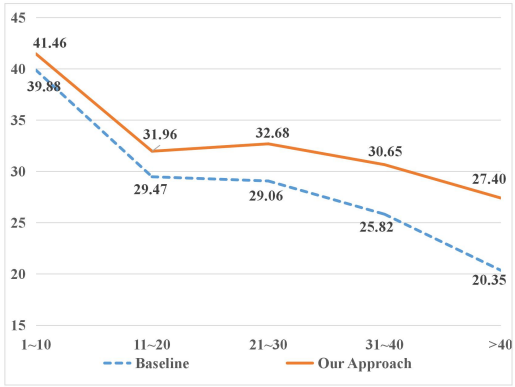

Figure 4: Performance (in BLEU) on the test set with respect to the size of the input AMR graphs.

tween our approach and the baseline goes widest for AMR graphs with more than 5 reentrancies, on which our approach outperforms the baseline by 6.61 BLEU scores.

4.3 Effect on AMR Graphs with Different Sizes

When we encode an AMR graph with plenty concepts, linearizing it into a sequence tends to lose great amount of structural information. In order to test the hypothesis that graphs with more concepts contribute more to the improvement, we partition source AMR graphs to different groups by their sizes (i.e., numbers of concepts) and evaluate their performance respectively. Figure 4 shows the results which indicate that modeling graph structures significantly outperforms the baseline over all AMR lengths. We also observe that the performance gap between the baseline and our approach increases when AMR graphs become big, revealing that the baseline seq2seq model is far from capturing deep structural details of big AMR graphs. Figure 4 also indicates that text generation becomes difficult for big AMR graphs. We think that the low performance on big AMR graphs is mainly attributed to two reasons:

- Big AMR graphs are usually mapped into long sentences while seq2seq model tends to stop early for long inputs. As a result, the length ratio[3] for AMRs with more than 40 concepts is 0.906, much lower than that for AMRs with less concepts.
- Big AMR graphs are more likely to have reentrancies, which makes the generation more challenging.

[3]Length ratio is the length of generation output, divided by the length of reference.

4.4 Case Study

In order to better understand the model performance, Figure 5 presents a few examples studied in Song et al. (2018) (Example (1)) and Damonte and Cohen (2019) (Examples (2) - (5)).

In Example (1), though our baseline recovers a propositional phrase for the noun *staff* and another one for the noun *funding*, it fails to recognize the anaphora and antecedent relation between the two propositional phrases. In contrast, our approach successfully recognizes *:prep-for c* as a reentrancy node and generates one propositional phrase shared by both nouns *staff* and *funding*. In Example (2), we note that although AMR graphs lack tense information, the baseline generates output with inconsistent tense (i.e., *do* and *found*) while our approach consistently prefers past tense for the two clauses. In Example (3), only our approach correctly uses *people* as the subject of the predicate *can*. In Example (4), the baseline fails to predict the direct object *you* for predicate *recommend*. Finally in Example (5), the baseline fails to recognize the subject-predicate relation between noun *communicate* and verb *need*. Overall, we note that compared to the baseline, our approach produces more accurate output and deal with reentrancies more properly.

Comparing the generation of our approach and graph-based models in Song et al. (2018) and Damonte and Cohen (2019), we observe that our generation is more close to the reference in sentence structure. Due to the absence of tense informa-

(1) (p/ provide-01
:ARG0 (a / agree-01)
:ARG1 (a2 / and
:op1 (s / staff
:prep-for (c / center
:mod (r / research-01)))
:op2 (f / fund-01
:prep-for c)))
REF: the agreement will provide staff and funding for the research center .
SEQ1: agreed to provide research and institutes in the center .
G2S: the agreement provides the staff of the research center and the funding .
Baseline: the agreement provides staff for research centres and funding for center .
Our approach: the agreement provided staff and funding for research centers .

(2) **REF:** i dont tell him but he finds out ,
SEQ2: i did n't tell him but he was out .
GRAPH: i do n't tell him but he found out .
Baseline: i do n't tell him but he found out .
Our approach: i did n't tell him but he found out .

(3) **REF:** if you tell people they can help you ,
SEQ2: if you tell him , you can help you !
GRAPH: if you tell them , you can help you .
Baseline: if you tell people , you might help you !
Our approach: if you tell people , people can help you !

(4) **REF:** i 'd recommend you go and see your doctor too .
SEQ2: i recommend you go to see your doctor who is going to see your doctor .
GRAPH: i recommend you going to see your doctor too .
Baseline: i would recommend going to see your doctor too .
Our approach: i would recommend you to go to see your doctor too .

(5) **REF:** tell your ex that all communication needs to go through the lawyer .
SEQ2: tell that all the communication go through lawyer .
GRAPH: tell your ex the need to go through a lawyer .
Baseline: tell your ex you need to go all the communication by lawyer .
Our approach: tell your ex that all communication needs to go through lawyers .

Figure 5: Examples of generation from AMR graphs. (1) is from Song et al. (2018), (2) - (5) are from Damonte and Cohen (2019). REF is the reference sentence. SEQ1 and G2S are the outputs of the seq2seq and the graph2seq models in Song et al. (2018), respectively. SEQ2 and GRAPH are the outputs of the seq2seq and the graph models in Damonte and Cohen (2019), respectively.

tion in AMR graphs, our model tends to use past tense, as *provided* and *did* in Example (1) and (2). Similarly, without information concerning singular form and plural form, our model is more likely to use plural nouns, as *centers* and *lawyers* in Example (1) and (5).

5 Related Work

Most studies in AMR-to-text generation regard it as a translation problem and are motivated by the recent advances in both statistical machine translation (SMT) and neural machine translation (NMT). Flanigan et al. (2016) first transform an AMR graph into a tree, then specify a number of tree-to-string transduction rules based on alignments that are used to drive a tree-based SMT model (Graehl and Knight, 2004). Pourdamghani et al. (2016) develop a method that learns to linearize AMR graphs into AMR strings, and then feed them into a phrase-based SMT model (Koehn et al., 2003). Song et al. (2017) use synchronous node replacement grammar (SNRG) to generate text. Different from synchronous context-free grammar in hierarchical phrase-based SMT (Chiang, 2007), SNRG is a grammar over graphs.

Moving to neural seq2seq approaches, Konstas et al. (2017) successfully apply seq2seq model together with large-scale unlabeled data for both text-to-AMR parsing and AMR-to-text generation. With special interest in the target side syn-

tax, Cao and Clark (2019) use seq2seq models to generate target syntactic structure, and then the surface form. To prevent the information loss in linearizing AMR graphs into sequences, (Song et al., 2018; Beck et al., 2018) propose graph-to-sequence models to encode graph structure directly. Focusing on reentrancies, Damonte and Cohen (2019) propose stacking encoders which consist of BiLSTM (Graves et al., 2013), TreeLSTMs (Tai et al., 2015), and Graph Convolutional Network (GCN) (Duvenaud et al., 2015; Kipf and Welling, 2016). Guo et al. (2019) propose densely connected GCN to better capture both local and non-local features. However, all the aforementioned graph-based models only consider the relations between nodes that are directly connected, thus lose the structural information between nodes that are indirectly connected via an edge path.

Recent studies also extend the Transformer to encode structural information for other NLP applications. Shaw et al. (2018) propose relation-aware self-attention to capture relative positions of word pairs for neural machine translation. Ge et al. (2019) extend the relation-aware self-attention to capture syntactic and semantic structures. Our model is inspired by theirs but aims to encode structural label sequences of concept pairs. Koncel-Kedziorski et al. (2019) propose graph Transformer to encode graph structure. Similar to the GCN, it focuses on the relations between directly connected nodes.

6 Conclusion and Future Work

In this paper we proposed a structure-aware self-attention for the task of AMR-to-text generation. The major idea of our approach is to encode long-distance relations between concepts in AMR graphs into the self-attention encoder in the Transformer. In the setting of supervised learning, our models achieved the best experimental results ever reported on two English benchmarks.

Previous studies have shown the effectiveness of using large-scale unlabelled data. In future work, we would like to do semi-supervised learning and use silver data to test how much improvements could be further achieved.

Acknowledgments

We thank the anonymous reviewers for their insightful comments and suggestions. We are grateful to Linfeng Song for fruitful discussions. This work is supported by the National Natural Science Foundation of China (Grant No. 61876120, 61673290, 61525205), and the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

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 7th Linguistic Annotation Workshop & Interoperability with Discourse*, pages 178–186.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of ACL*, pages 65–72.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of ACL*, pages 273–283.

Kris Cao and Stephen Clark. 2019. Factorising amr generation through syntax. In *Proceedings of NAACL*, pages 2157–2163.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33:201–228.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of NAACL*, pages 3649–3658.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of WMT*, pages 376–380.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timonthy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of NIPS*, pages 2224–2232.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proceedings of NAACL*, pages 731–739.

DongLai Ge, Junhui Li, Muhua Zhu, and Shoushan Li. 2019. Modeling source syntax and semantics for neural amr parsing. In *Proceedings of IJCAI*, pages 4975–4981.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proceedings of NAACL*, pages 105–112.

Alex Graves, Abdel rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP*, pages 6645–6649.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of ACL*, pages 140–149.

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

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*, pages 655–665.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL, System Demonstrations*, pages 67–72.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*, pages 127–133.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proceedings of NAACL*, pages 2284–2293.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of ACL*, pages 146–157.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of ICLR*.

Kishore Papineni, Salim Roukos, Ward Todd, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Maja Popovi. 2017. chrf++: words helping character n-grams. In *Proceedings of WMT*, pages 612–618.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating english from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL*, pages 1715–1725.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of NAACL*, pages 464–468.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. Amr-to-text generation with synchronous node replacement grammar. In *Proceedings of ACL*, pages 7–13.

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. Amr-to-text generation as a traveling salesman problem. In *Proceedings of EMNLP*, pages 2084–2089.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of ACL*, pages 1616–1626.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, pages 1556–1566.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N.Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008.