AMR-to-Text Generation with Graph Structure Reconstruction and Coverage

Anonymous ACL submission

Abstract

Generating text from semantic representations such as AMR is a challenging task. Previous research formalizes this task as a graph-to-sequence learning problem and uses various graph neural networks to model the graph structure. Recently, methods based on pre-trained models improve the performance significantly due to pre-trained on a large text corpus. However, these pre-trained model-based methods take linearized AMR graphs as input and may lose the information of graph structure. In addition, these methods don't consider the coverage of the AMR graph. Therefore, some nodes in the graph may be lost or repeated in the generated text. To address these problems, we propose a graph structure and coverage enhanced model for this task. To enhance the information of graph structure, we design two auxiliary objectives, relationship prediction and distance prediction of nodes in AMR graphs. To consider the coverage of AMR graphs, we design a coverage mechanism to solve the problem of information under-translation or over-translation in AMR-to-text generation. Experimental results on three standard datasets show that our proposed method outperforms the existing methods significantly.

1 Introduction

In recent years, abstract meaning represents (AMR) has drawn increasing attention (Banarescu et al., 2013; Guo et al., 2019; Wang et al., 2020a; Bevilacqua et al., 2021) for its potential value in many applications. AMR abstracts away from the surface form of a sentence and encodes its meaning as a rooted, directed, and acyclic graph. The nodes in the graph represent concepts, and the edges represent the relationships between concepts. Figure 1 illustrates an AMR graph of a sentence.

The task of AMR-to-text is to recover text that represents the same meaning as the given AMR graph. The semantic information of the generated text has to be as consistent as possible with the original AMR graph. This task is extremely challenging due to the abstraction of functional words and syntactic realizations in the AMR graph and the lack of description of many details such as tense, number, and determinism.

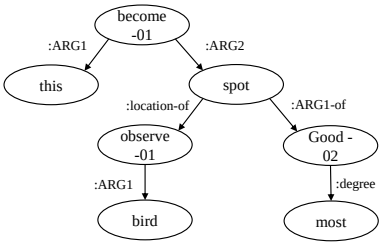

Figure 1: An AMR graph of the sentence: These have become the best spots to observe birds.

Existing neural machine translation methods have been explored for AMR-to-text generation (Konstas et al., 2017). First, the graph is transformed into a sequence. Then a sequence-to-sequence model (Seq2seq) is used to solve the problem (Song et al., 2017, 2018; Damonte and Cohen, 2019). This approach is simple and effective but risks losing information about the graph structure.

Recent studies have viewed this task as a graph-to-sequence learning problem, and these studies have proposed a variety of graph neural networks to encode graphs. Initially, several studies have used gated graph neural networks (GGNN) or graph convolutional networks (GCN) to directly encode AMR graphs (Beck et al., 2018; Guo et al., 2019). However, these graph encoders still could not significantly outperform the sequence encoders. Subsequently, graph transformers were introduced to this task with good results (Cai and Lam, 2020; Wang et al., 2020a).

However, these graph-to-sequence models are prone to misrepresentation of semantic relations between concepts, which suggests that some graph structure information is not captured by the node

representation. What's more, most previous methods doesn't consider the coverage of AMR graph. As a result, these methods may lose some nodes, or they may duplicate some nodes when generating text. For example, given AMR graph represented in Figure 1 as input, the above models may produce the following errors: 1) **semantic confusion**: compared to the gold sentence, it is translated to *these are the best locations for birds to observe them.* The model confuses the semantic relationship between *observe* and *bird*. This is a typical problem of semantic confusion; 2) **under-translation**: In another case, it is translated to *these are the best locations to observe.* The model ignores the object *bird*. This is a typical under-translation problem; 3) **over-translation**: it is translated to *these are the best locations to observe the best birds.* The model repeats the translation of *best*. This is a typical problem of over-translation.

More recently, as pre-trained language models, such as BERT (Devlin et al., 2018), BART (Lewis et al., 2020), T5 (Raffel et al., 2020), etc., have achieved good performance on many tasks (Zhang et al., 2020c; Bao et al., 2020; Zhang et al., 2020a), some researchers have used them for AMR-to-text generation. Specifically, Bevilacqua et al. (2021) proposed a method based on the pre-trained model BART and achieved the best score in this task. And the performance of their model were substantially ahead of the previous methods. Therefore, we focus on the pre-trained model based methods for this task. However, most of the pre-trained models are based on Seq2seq framework. As mentioned above, using Seq2seq based models for AMR-to-text task usually leads to loss of graph structure and under- and over-translation problems.

To tackle the above problems, we enhance a pre-trained model with graph structure reconstruction and coverage to improve the quality of AMR-to-text generation. To enhance the information of graph structure, we design two auxiliary objectives, relationship prediction and distance prediction of nodes in AMR graphs. With these two auxiliary objectives, we force the model to learn and reconstruct the graph structure. Inspired by the coverage mechanism in machine translation (Tu et al., 2016) and text summarization (See et al., 2017), we propose a novel coverage mechanism to address the under- and over-translation problems in AMR-to-Text generation. This mechanism encourages the model to cover all nodes and edges in the graph when decoding. With this mechanism, node loss or duplication is avoided. We conduct experiments on three benchmarks and the results show the effectiveness of our model.

In summary, our contributions are as follows:

- To solve the problem that linearizing AMR graphs lead to loss of graph structure information, we apply graph structure reconstruction to enhance a pre-trained model.
- The coverage mechanism is designed to solve the under- and over-translation problems that tend to occur in AMR-to-Text generation.
- Experimental results on three datasets demonstrate that our model achieves the best results on all metrics compared to existing methods.

2 Related Work

Without discussing the statistical methods, most of the current methods for AMR-to-Text generation can be roughly divided into three categories. Next, we detail the progress of the three categories of methods.

Seq2seq based models In previous work, the mainstream approach was to linearly serialize the AMR graph and then send these sequences to the seq2seq model. Konstas et al. (2017) defined this task as a translation task, translating sequences of AMR graph serialization into sentences. His model is based on an off-the-shelf RNN. Zhu et al. (2019) was the first to apply transformer to this task. However, these seq2seq based models have some drawbacks. When linearizing the AMR graph, structural information between graph nodes are lost. But the structural information plays an important role in AMR-to-text generation.

Graph2seq based models In the past years, this task has gradually become a graph2seq task due to the rise of graph neural networks. Many graph neural network-based models were used for this task. Beck et al. (2018) proposed to transform AMR graphs into Levi graphs to solve the problem of sparse data, and proposed gated graph neural networks. Guo et al. (2019) used a new GCN-based model to solve this task. Then, Wang et al. (2020a) modified a graph transformer-based model to solve the AMR-to-text task with good results. Currently, Wang et al. (2020b) has achieved an advanced score in this task. They use a GAT-based model with an additional structure reconstruction

objective to solve the semantic confusion problem in the AMR-to-text generation.

Pre-trained based models Recently, the pre-trained based models have achieved good performance in many NLP downstream tasks, including the AMR-to-text generation task. Mager et al. (2020) used GPT-2 and successfully achieved the best performance at that time. Then Bevilacqua et al. (2021) applied BART to this task and improved the performance further. Researchers also evaluated the performance of different pre-trained models for this task (Ribeiro et al., 2020). Compared to the non-pre-trained models, the pre-trained models improved performance with a large margin in AMR-to-text generation.

3 Methodology

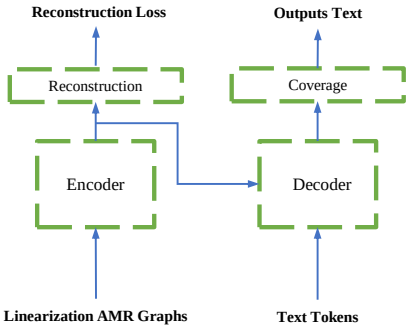

Figure 2: The architecture of our whole model

The overall structure of our model is shown in Figure 2. We use BART as our backbone model. A graph structure reconstruction is proposed to solve the semantic confusion, and a coverage mechanism is designed to solve the under- and over-translation. In this section, we will elaborate our model in three parts. First, we show how to transform the AMR graph into linear sequence. Second, we present how to reconstruct the graph structure in the linearized AMR graph. Finally, we elaborate how our coverage mechanism works and how it allows the model to avoid the under- and over-translation problems in AMR-to-text generation.

3.1 BART Inputs

Following the method proposed by Bevilacqua et al. (2021), we chose to linearize the AMR graph using a depth search-first algorithm. Special markers are used to replace the variable names that appear in the PENMAN (Goodman, 2020) linearization. The variable names generated by the PENMAN linearization tend to introduce certain confusion problem. The special token is a good solution to solve this confusion problem. It also facilitates the extraction of position of nodes and edges. Figure 3 shows specific differences between PENMAN and ours.

PM: (b / become-01 :ARG1 (t / this) :ARG2 (s / spot :location-of (o / observe-01 :ARG1 (b3 / bird)) :ARG1-of (g / good-02 :degree (m / most))))

Ours: (<P0> become-01 :ARG1 (<P1> this) :ARG2 (<P2> spot :location-of (<P3> observe-01 :ARG1 (<P4> bird)) :ARG1-of (<P5> good-02 :degree (<P6> most))))

Figure 3: Linearized sequences of PENMAN and ours for the AMR graph shown in Figure 1

As we know, BART uses a subword vocabulary whose tokenization is optimized to handle English, but it is not well suited to AMR notation. To avoid AMR notation being divided too finely and thus affecting performance, we expand BART's vocabulary with these notations. Specifically, we refer to the approach of Bevilacqua et al. (2021) and expand it with three types of tokens: 1) all relations and frames that occur at least five times in the training corpus; 2) all the constituent components of AMR tokens, such as *:op*; 3) all other special tokens added in the linearization.

To facilitate graph structure reconstruction, we must record and extract the position of nodes and edges in the sequence. Since node may be split into multiple subwords by BART during the tokenization process, it is not easy to extract. Therefore, we extract the position of special markers instead of node position. Since each node corresponds to a special marker, this substitution solves the problem caused by tokenization effectively.

3.2 Graph Structure Reconstruction

To force the model to reconstruct the graph structure in the linearized AMR graph, we propose to optimize two simple but effective auxiliary prediction objectives. The architecture of the graph reconstruction mechanism is shown in Figure 4. First, the positions of the specified nodes are extracted from the linearized sequence. Then, the encoder hidden states of the specified nodes are extracted in the encoder hidden states based on the positions of the nodes. Finally, the encoder hidden states of these nodes are used for the prediction objectives.

Relationship Prediction The node positions and the edge positions of the linearization AMR graphs are usually not closely related. The source, edge

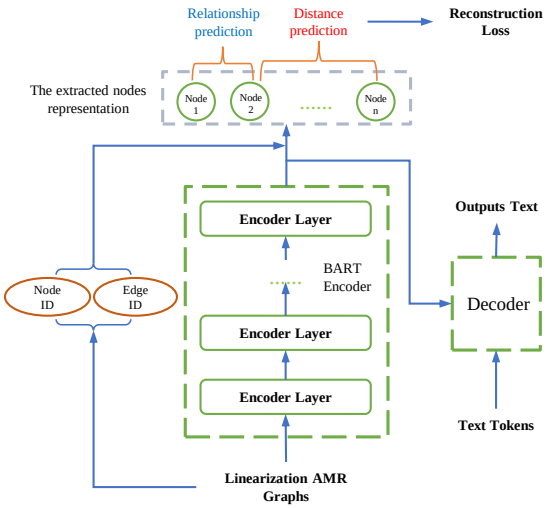

Figure 4: The architecture of Graph Structure Reconstruction

and target nodes are no longer adjacent to each other in the linear sequence. This leads to the loss of relevant structural information. To reconstruct the structure of the graph, we choose to predict the types of edges between different nodes in the linearized sequence. To make the prediction complete, we predict not only the types of edges between adjacent nodes, but also whether non-adjacent nodes are adjacent to each other.

The relationship prediction requires the model to predict the semantic relationship of a given node pair. For a given pair of nodes, we concatenate the representations of the two nodes and use a multi-layer perceptron to predict the corresponding semantic relationship as follows:

$$h_{ij}^r = \text{MLP}([h_i; h_j]) \quad (1)$$

$$\hat{r}_{i,j} = softmax(W_r[h_{ij}^r] + b_r) \quad (2)$$

where $W_r \in R^{(L+1)\times d_{model}}$, $b_r \in R^{L+1}$ and L is the number of semantic label types in the AMR graph, h_i is the encoder hidden states of the node i after linearization. For the pair of nodes that are adjacent in the graph, the gold relation label is exactly the given semantic relation $r_{i,j}$. For the pair of nodes that are not adjacent in the AMR graph, the gold label is *non-adjacent*.

Distance Prediction After we linearize the AMR graph, many adjacent nodes in the AMR graph become non-adjacent nodes in the sequence. Then this phenomenon means that our model cannot perceive the structural information in the original AMR graph, which leads to the problem of semantic confusion. So it is not enough to predict the semantic relationships of node pairs. In order to reconstruct the structure of the graph completely, we also predict the distances of node pairs to reduce the problem of semantic confusion.

The Distance prediction is to predict the distance between two nodes. The distance between a pair of nodes is defined as the length of the shortest path from node i to node j regardless of the direction of the edges. We use a multi-layer perceptron to predict the distance between two nodes via:

$$h_{ij}^d = \text{MLP}([h_i; h_j]) \quad (3)$$

$$\hat{d}_{i,j} = softmax(W_d[h_{ij}^d] + b_d) \quad (4)$$

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

Optimization Objective In order to learn better node representation and generate higher-quality text, we optimize the two proposed graph reconstruction objectives in the loss function. The relationship prediction objective is defined as follows:

$$\begin{aligned} L_r = &- \sum_{(i,j,r_{i,j})\in E} \log P(r_{i,j}|i,j,\theta) \\ &- \frac{1}{N}\lambda_n \sum_{(i,j,r_{i,j})\notin E} \log P(r_{i,j}|i,j,\theta) \end{aligned} \quad (5)$$

where $r_{i,j}$ is the golden label to the relationship between nodes v_i and v_j (note that when two nodes are not adjacent, the golden label is set to *non-adjacent*), θ is for the model parameters. N is the number of nodes, and E is the set of edges. We sample N pairs of non-adjacent nodes as negative samples. $\frac{1}{N}$ and λ_n is used to balance the weight of negative samples. $P(r_{i,j}|i,j,\theta)$ is computed from the predicted probability $\hat{r}_{i,j}$ in Equation 2. The distance prediction objective is defined as follows:

$$L_d = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} \log p(d_{i,j}|\theta) \quad (6)$$

where $d_{i,j}$ is the golden label to the distance between nodes v_i and v_j, $\frac{1}{N}$ is used to balance the weight of negative samples. $p(d_{i,j}|\theta)$ can computed from from the predicted probability $\hat{d}_{i,j}$ in Equation 4.

Finally, we add these two additional optimization objectives to original optimization objective and the final loss is:

$$L_{total} = L_{text} + \lambda_r * L_r + \lambda_d * L_d \quad (7)$$

where L_{text} is the loss of text generation. λ_r and λ_d

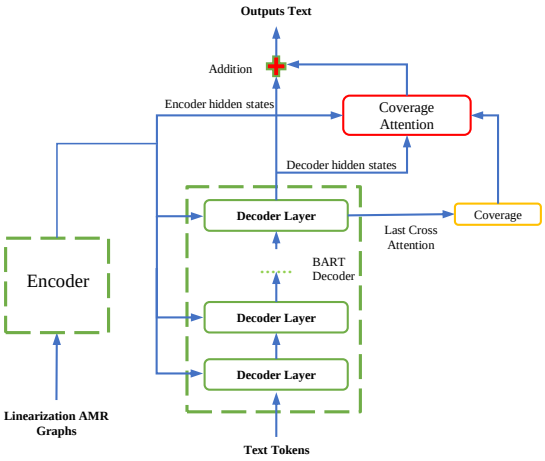

Figure 5: The architecture of Coverage Mechanism

are the hyperparameters to make a trade-off among different losses.

3.3 Coverage Mechanism

Repetition and omission are common problems in machine translation. In traditional statistical machine translation, a coverage mechanism is usually used to solve this problem (Koehn, 2009). Inspired by this idea, Tu et al. (2016) applied a coverage mechanism in neural machine translation to solve the under- and over-translation problems. And See et al. (2017) further simplified this mechanism and achieved good results in text summarization. They use a LSTM based decoder and update the coverage vector in chronological order (Greff et al., 2016). In short, these methods use coverage vector to track attention history on source tokens. When predicting next target token, the coverage vector is fed back to the attention model to help adjust future attention, which allows the model to take more account of untranslated source tokens.

However, most previous works on AMR-to-text generation ignored these two key problems (Wang et al., 2020b; Ribeiro et al., 2020; Zhang et al., 2020b). Therefore, we design a novel coverage mechanism to solve under- and over-translation problems in AMR-to-text generation. In contrast, we do not use a LSTM based model, which means that the coverage vector cannot be updated based on the time steps as Tu et al. (2016) and See et al. (2017) do. Next, we elaborate on our coverage mechanism in detail.

The architecture of coverage mechanism is shown in Figure 5. Since in transformer based model the outputs of all time steps are computed in parallel when training, adding dependencies between time steps can significantly slow down the training speed. Therefore, as an alternative, we treat a layer of the transformer as a time step. Therefore, the cross attention of the last decoder layer A_{last} in the decoder is chosen as our coverage vector C.

To guide attention with the coverage vector, we define a new attention layer after the decoder, which takes into account the historical attention distribution C. The coverage vector, the encoder hidden states and the decoder hidden states are used as inputs to this attention. The new attention distribution is computed as follows:

$$A_{cov} = softmax(\frac{H_{last} * E_{enc}^T}{\sqrt{d_{model}}} + W_c * C) \quad (8)$$

where H_{last} represents the hidden states of the last decoder layer, E_{enc} is the final hidden states of the encoder.

Then we multiply the obtained coverage attention with the hidden states of the encoder to obtain new weighted context vectors as follows:

$$H_{cov} = A_{cov} * E_{enc}^T \quad (9)$$

Subsequently, we add a residual layer, in which we add new weighted context vectors to the decoder hidden states to get the final hidden states as follows:

$$H_{final} = H_{last} + H_{cov} \quad (10)$$

Then, we add layer normalization as follows:

$$H_{norm} = \text{LayerNorm}(H_{final}) \quad (11)$$

Finally, the normalized result H_{norm} are used to predict words.

4 Experimental Settings

We now describe the experimental settings for AMR-to-text generation.

Datasets

We use three standard English AMR corpora as evaluation datasets, AMR 1.0 (LDC2014T12), AMR 2.0 (LDC2017T10), and AMR 3.0 (LDC2020T02). They contains 13,051, 39,260, and 59,255 manually-created sentence-AMR pairs respectively. Notice that AMR 2.0 is a superset of AMR 1.0 and AMR 3.0 is a superset of AMR 2.0. Each dataset is randomly split into training set, development set, and test set.

Setup

We mainly refer to the experimental parameter settings of Bevilacqua et al. (2021). The same settings are used on the BART-Large as specified in Huggingface's transformers. The model is trained for

Model	AMR 1.0			AMR 2.0		
	BLEU	METEOR	CHRF++	BLEU	METEOR	CHRF++
GraphLSTM (Song et al., 2018)	23.3	-	-	-	-	-
GGNN (Beck et al., 2018)	-	-	-	27.5	-	53.5
DenselyGCN (Guo et al., 2019)	28.2	-	-	30.4	-	59.6
GraphTransformer (Cai and Lam, 2020)	27.4	32.9	56.4	29.8	35.1	59.4
StructuralTransformer-SA (Zhu et al., 2019)	29.7	35.5	63.0	31.5	36.0	63.8
HetGT (Yao et al., 2020)	31.8	36.9	63.8	34.0	38.1	65.6
BetterG-Transformer (Wang et al., 2020b)	32.1	36.1	64.0	33.9	37.1	65.8
GPT-2 (Mager et al., 2020)	-	-	-	33.0	37.7	63.9
BART	40.4	40.1	70.7	42.7	40.7	72.2
SPRING	41.8	41.0	71.4	45.3	41.0	73.5
Ours(Recon + Coverage)	**42.8**	**41.5**	**72.2**	**45.4**	**42.4**	**73.6**

Table 1: Results on the test set of AMR 1.0 and AMR 2.0

30 epochs using cross-entropy with a batch size of 500. We use the RAdam optimizer (Liu et al., 2019) with a learning rate of $10^{-}5$. The gradient accumulation is 10 batches. Dropout is set to 0.25. A depth-first search method is used for linearization. Special tokens are added in the linearization process. The vocabulary of the BART model is expanded with these spcial tokens. For prediction, we follow the usual practice of neural machine translation (Yang et al., 2018) and set the beam size to 5.

Baseline Models

For AMR 1.0 and AMR 2.0 datasets, the baselines are divided into three main categories.

The first category of models uses graph neural network models to solve this task: **1) GraphLSTM** (Song et al., 2018) uses an LSTM structure to directly encode graph-level semantic graphs; **2) GGNN** (Beck et al., 2018) couples the Gated Graph Neural Networks with an input transformation; **3) DenselyGCN** (Guo et al., 2019) introduces a dense connection strategy, proposing a novel Densely Connected Graph Convolutional Networks.

The second category is a series of models of transformer variants (Vaswani et al., 2017): **1) GraphTransformer** (Cai and Lam, 2020), a graph-based parser iteratively refining an incrementally constructed graph; **2) StructuralTransformer-SA** (Zhu et al., 2019), a transformer-based method that enhances structural awareness of self-attention; **3) HetGT** (Yao et al., 2020), a graph transformer-based model for encoding the representation of heterogeneous subgraphs; **4) BetterG-Transformer** (Wang et al., 2020b), a transformer-based model that generates sentences with additional structural reconstruction goals.

The third category is systems that use pre-trained models: **1) GPT-2** (Mager et al., 2020) is a fine-tuned GPT-2 model (Radford et al., 2019) to predict Penman linearization of AMR graphs; **2) SPRING** (Bevilacqua et al., 2021) uses BART to make AMR-to-text and Text-to-AMR in one system; **3) BART** (Lewis et al., 2020) is also reported in Bevilacqua et al. (2021), which uses the pre-trained model BART without any changes. It linearizes the AMR graph using Penman and generates the text directly.

For AMR 3.0, since this dataset is the latest release, there are fewer baselines to compare, and we compare two baselines: **1) LDGCN** (Zhang et al., 2020b) is a Lightweight Dynamic Graph Convolutional Networks that capture richer non-local interactions by synthesizing higher-order information from the input graphs; **2) SPRING** (Bevilacqua et al., 2021) uses BART to make AMR-to-text and Text-to-AMR in one system.

Evaluation

We follow the previous methods and use three common natural language generation (NLG) measures for evaluation: BLEU (Papineni, 2002), chrF++ (Popović, 2017), and Meteor (Banerjee and Lavie, 2005). Tokenization is performed with the script provided by JAMR (Flanigan et al., 2014).

5 Results

5.1 Main Results

First, we test the baseline and our overall model on the AMR 1.0 and AMR 2.0 datasets. The results are shown in table 1. From table 1, we can see that our model achieves the best performance on all metrics. Specifically, on AMR 1.0, the BLEU score, METEOR score and CHRF++ score are improved by 1.0, 0.5 and 0.8 points, respectively. On AMR 2.0, the BLEU score, METEOR score and

Model	AMR 3.0		
	BLEU	METEOR	CHRF++
LDGCN (Zhang et al., 2020b)	34.3	38.2	63.7
SPRING	44.9	40.6	72.9
ours(Recon +Coverage)	**45.7**	**42.8**	**73.7**

Table 2: Results on the test set of AMR 3.0

Model	AMR 2.0			AMR 3.0		
	B	M	C	B	M	C
Base model	45.3	41.0	73.5	44.9	40.6	72.9
+Recon	44.9	42.2	73.2	45.6	42.7	73.6
+Coverage	45.2	42.4	73.3	45.1	42.3	73.2
+Recon +Coverage	**45.4**	**42.4**	**73.6**	**45.7**	**42.8**	**73.7**

Table 3: Ablation results on the test set of AMR 2.0 and AMR 3.0. B for the BLEU scores. M for the METEOR scores. C for the CHRF++ scores.

CHRF++ score are improved by 0.1, 1.4 and 0.1 points, respectively.

Compared with the first category of GNN based methods, the second category of transformer based methods performs better. Because transformer handles long distance dependencies better through introducing self-attention structure. Compared with transformer based models, the pre-trained based methods improve the performance significantly. The pre-trained models can leverage knowledge obtained by training on large-scale text corpus, thus they achieve better results than the non-pre-trained models. This is why we chose BART as our backbone model. Even so, our model outperforms the pre-trained based methods significantly, including BART baseline. This indicates that our proposed graph structure reconstruction and coverage mechanism contributes to improving the quality of AMR-to-text generation.

Second, we test baselines and our overall model on the AMR 3.0 dataset and Table 2 shows the results. The results also demonstrate the effectiveness of our model. Compared with the previous best method, our model improves BLEU score, METEOR score, and CHRF++ score by 0.8, 2.2, and 0.8, respectively.

5.2 Ablation Results

To verify the validity of two components in our model, we conduct ablation experiments on AMR 2.0 and AMR 3.0. Specifically, we compare our overall model with three variants: 1) **Base model**: our backbone model BART which only contains an encoder and a decoder; 2) **+Recon**: our backbone model with graph structure reconstruction component; 3) **+Coverage**: our backbone model with coverage mechanism. The results are shown in Table 3.

Compared to our backbone model, both of graph structure reconstruction and coverage mechanism improve the scores of the three main metrics. Specifically, on AMR 2.0, both individual models improve METEOR scores, by 1.2 and 1.4, respectively. On AMR 3.0, both individual models improve all three metrics. The graph structure reconstruction model improve BLEU, METEOR, and CHRF++, by 0.7, 2.1, and 0.7, respectively on AMR 3.0. The coverage mechanism model improve on AMR 3.0 for BLEU, METEOR, and CHRF++, by 0.2, 1.7, and 0.3, respectively. This shows that both of our proposed components achieve a score improvement.

Similarly, compared to our overall model, removing either of the two modules decreases its score. Specifically, removing the graphical structure reconstruction leads to a decrease in all metrics for both datasets. This demonstrates the effectiveness of the graph structure reconstruction module on solving the problem of semantic confusion. Without the coverage mechanism, the metrics scores are also inferior to our overall model. This shows that the coverage mechanism is effective in reducing the under- and over-translation problems. This demonstrates that both of components are useful, and removing either one have a negative effect on the results.

5.3 Case Study

We first perform error analysis on the results generated by baselines. Specifically, we calculate three metrics for each sample in the test set of AMR 2.0, and samples with the lower score are considered as bad cases. We analyze bad cases and summarized three common errors in AMR-to-text generation. Then we analyze results generated by our model. Compared with baselines, our model addresses two of these common errors: semantic confusion and information loss. Next, we introduce three common errors in detail and show how our model solves the first two errors. Due to space limitation, there is only one case for each common error, and more cases are shown in the Appendix A. And we only show the results of a strong baseline SPRING in case study.

Semantic confusion The first common error is semantic confusion. That is, the semantic relations of

the generated sentences are confused or opposite to the original meaning. An example is shown in Figure 6. The original sentence is "China considers Germany the most important trading partner of Europe", while our baseline translates as "China is considered Germany's the most important trading partner in Europe". This is a common example of confusing semantic relations. This problem may be due to the loss of graphical structure information during linearization, which leads to the confusion of subject-verb-object relationships in the sentences. And it can be seen that our model translates it correctly, which shows the effectiveness of our model. This indicates that the graph structure reconstruction module in our model distinguishes the semantic relationships between nodes well and improves the quality of AMR-to-text generation.

AMR Graph:
(c / consider-01
:ARG0 (c2 / country :wiki "China"
:name (n / name :op1 "China"))
:ARG1 (p / partner-01
:ARG1 (c4 / country :wiki "Germany"
:name (n3 / name :op1 "Germany"))
:mod (i / important
:degree (m / most))
:mod (t / trade-01)
:location (c3 / continent :wiki "Europe"
:name (n2 / name :op1 "Europe"))))
Gold: China considers Germany the most important trade partner of Europe.
Baseline: China is considered Germany's most important trading partner in Europe.
Ours: China considers Germany to be its most important trading partner in Europe.

Figure 6: A comparison of our model and the baseline model to generate the result cases

Information loss The problem of information loss is very common. We show an example of information loss in Figure 7. In the original sentence, "wen is expressing concern and sympathy for the situation in Iraq and the Iraqi people". However, our baseline model loses the object *the situation in Iraq* and generates only one object, which is different from the original sentence's meaning. Since baseline does not consider the coverage of AMR graphs, some nodes information may be missing when generating them. Our model, however, does not have this problem and generates the original meaning of the sentence intactly. This proves the effectiveness of our coverage mechanism, which ensures the integrity of the information by considering the coverage of the graph.

Correct but low scores The third type is that the generated sentences have the correct meaning, but

AMR Graph:
(s / state-01
:ARG0 (p / person :wiki "Wen _ Jiabao" :name (n / name :op1 "Wen"))
:ARG1 (a / and
:op1 (c / concern-01
:ARG0 (s2 / situation
:time (c3 / current)
:location (c4 / country :wiki "Iraq" :name (n3 / name :op1 "Iraq")))
:ARG1 (c2 / country :wiki "China" :name (n2 / name :op1 "China"))
:ARG1-of (d / deep-02))
:op2 (s3 / sympathize-01
:ARG0 c2
:ARG1 (p2 / people
:mod c4))))
Gold: Wen stated that China is deeply concerned with the current situation in Iraq and is sympathetic to the Iraqi people.
Baseline: Wen stated that China is deeply concerned and sympathetic to the Iraqi people.
Ours: Wen stated that China is deeply concerned about the current situation in Iraq and sympathizes with the Iraqi people.

Figure 7: A comparison of our model and the baseline model to generate the result cases

AMR Graph:
(p / possible-01 :polarity -
:ARG1 (v / verify-01
:ARG1 (t / thing
:ARG1-of (c / claim-01
:ARG0 (c2 / country :wiki "Iran"
:name (n / name :op1 "Iran"))))))
Gold: There is no way to verify Iran's claims.
Baseline: Iran's claims cannot be verified.
Ours: Iran's claims are impossible to verify.

Figure 8: A comparison of our model and the baseline model to generate the result cases

the scores are low on all three measures. We show an example related to this problem in Figure 8. The meaning of the generated sentence is the same as the original sentence, regardless of baseline model or our model. However, since the generated sentence differs in expression, the used metrics can not handle this difference. In other words, the third error is made by metrics, not by our model or baseline. Therefore, in the future, we can add human evaluation or design new automatic metrics in evaluation of AMR-to-text generation.

6 Conclusion

We propose a BART-based model with graph structure reconstruction and coverage mechanism to improve the quality of AMR-to-text. We design two auxiliary objectives, relationship prediction and distance prediction of nodes in AMR graphs, to enhance the information of the graph structure. To consider the coverage of AMR graphs, we design a coverage mechanism to address the under- and over-translation problems in AMR-to-text generation. Experimental results on three benchmarks show that our model achieves the best performance compared to existing methods.

References

L. Banarescu, C. Bonial, C. Shu, M. Georgescu, and N. Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Satanjeev Banerjee and Alon Lavie. 2005. 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.

Siqi Bao, Huang He, Fan Wang, Hua Wu, and Haifeng Wang. 2020. Plato: Pre-trained dialogue generation model with discrete latent variable. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 85–96.

D Beck, G. Haffari, and T. Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. *Meeting of the Association for Computational Linguistics*.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*.

Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7464–7471.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436.

M. W. Goodman. 2020. Penman: An open-source library and tool for amr graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2016. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232.

Z. Guo, Y. Zhang, Z. Teng, and W. Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*, 7(2):297–312.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

I. Konstas, S. Iyer, M. Yatskar, Y. Choi, and L. Zettlemoyer. 2017. 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)*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2019. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*.

M. Mager, R. F. Astudillo, T. Naseem, M. A. Sultan, and S. Roukos. 2020. Gpt-too: A language-model-first approach for amr-to-text generation. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

S. Papineni. 2002. Blue ; a method for automatic evaluation of machine translation. In *Meeting of the Association for Computational Linguistics*.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Leonardo FR Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation. *arXiv preprint arXiv:2007.08426*.

A. See, P. J. Liu, and CD Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

L. Song, Y. Zhang, Z. Wang, and D. Gildea. 2018. A graph-to-sequence model for amr-to-text generation. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. Amr-to-text generation with synchronous node replacement grammar. *arXiv preprint arXiv:1702.00500*.

Z. Tu, Z. Lu, L. Yang, X. Liu, and L. Hang. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

T. Wang, X. Wan, and H. Jin. 2020a. Amr-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8(1):19–33.

T. Wang, X. Wan, and S. Yao. 2020b. Better amr-to-text generation with graph structure reconstruction. In *Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence IJCAI-PRICAI-20*.

Yilin Yang, Liang Huang, and Mingbo Ma. 2018. Breaking the beam search curse: A study of (re-) scoring methods and stopping criteria for neural machine translation. *arXiv preprint arXiv:1808.09582*.

Shaowei Yao, Tianming Wang, and Xiaojun Wan. 2020. Heterogeneous graph transformer for graph-to-sequence learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7145–7154.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Y. Zhang, Z. Guo, Z. Teng, W. Lu, and L. Bing. 2020b. Lightweight, dynamic graph convolutional networks for amr-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B Dolan. 2020c. Dialogpt: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278.

J. Zhu, J. Li, M. Zhu, L. Qian, M. Zhang, and G. Zhou. 2019. Modeling graph structure in transformer for better amr-to-text generation. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

A Appendix

As stated in the main text, we report more case studies below.

A.1 Semantic confusion

AMR Graph:
(c / check-07
:ARG1 (t / terrorism
:ARG0-of (c2 / cross-02
:ARG1 (b / border))))
Gold: Check cross-border terrorism.
Baseline: Cross-border terrorism checks
Ours: Check cross-border terrorism.

AMR Graph:
(r / road
:destination (n / nowhere))
Gold: A road to no where.
Baseline: There is no road to nowhere.
Ours: A road to nowhere.

AMR Graph:
(p / possible-01
:ARG1 (h / have-03
:ARG0 (c / country :wiki "China" :name (n / name :op1 "China"))
:ARG1 (g / girl
:ARG1-of (p2 / pure-02)
:ARG1-of (i / innocent-01)))
:time (a / amr-unknown))
Gold: When can China have a pure, innocent girl?
Baseline: How can China have pure innocent girls?
Ours: When can China have pure innocent girls?

Figure 9: A comparison of our model and the baseline model to generate the result cases

The first common error is semantic confusion. That is, the semantic relations of the generated sentences are confused or opposite to the original meaning. Three examples are shown in the Figure 9. The original sentence of Example 1 is "Check cross-border terrorism ", while our baseline translates to "Cross-border terrorism checks". Example 2 is "A road to no where", while the baseline translation is "There is no road to nowhere". Both of these are common examples of confusing semantic relationships because they reverse the subject and object. Example 3 is "When can China have a pure, innocent girl?" but the baseline translation is "How can China have pure innocent girls? "This example confuses a time question with a manner question. This problem may be due to the loss of graphical structure information during linearization, which leads to the confusion of subject-verb-object relationships in the sentences. And it can be seen that our model can be translated correctly, which shows the validity of our model. This indicates that the graph structure reconstruction module in our model can distinguish the semantic relationships between nodes well and improve the quality of AMR-to-text generation.

A.2 Information loss

AMR Graph:
(s / station
:mod (t / television
:ARG1-of (l / local-02))
:domain (c / channel :wiki "ABS-CBN _Corporation" :name (n2 / name :op1 "ABS-CBN")
:mod (n / news))
:location (c2 / city :wiki "Manila" :name (n3 / name :op1 "Manila")))
Gold: ABS-CBN news channel is a local television station in Manila.
Baseline: ABS-CBN is a local television station in Manila.
Ours: ABS-CBN news channel is a local television station in Manila.

AMR Graph:
(m / multi-sentence
:snt1 (a / agree-01
:ARG1 (p / person
:ARG0-of (e / entertain-01)
:mod (j / just)
:ARG1-of (c / cheap-02)))
:snt2 (s3 / sense-02
:ARG1 (t2 / thing
:ARG2-of (r / repute-01
:ARG1 (t / they)
:location (c2 / country :wiki "China" :name (n / name :op1 "China")
:mod (f / feudal))))
:degree (s2 / somewhat)))
Gold: Agree, just cheap entertainers. Their reputation in feudal China somewhat makes sense.
Baseline: Agreed, they are just entertainers, their rep in feudal China makes somewhat sense.
Ours: Agree, just cheap entertainers. Their rep in feudal China makes somewhat sense.

Figure 10: A comparison of our model and the baseline model to generate the result cases

The problem of missing information is very common. We show several examples of missing information in Figure 10 and Figure 11 respectively. The original sentence of Example 1 is "ABS-CBN news channel is a local television station in Manila." However, our baseline model loses *news*, which is different from the meaning of the original sentence. This leads to a change in the subject of the sentence. Similarly, in Example 2, the original sentence reads "Agree, just cheap entertainers. Their reputation in feudal China somewhat makes sense." and the baseline loses the adjective *cheap*, which is the same as in Example 1. In Example 3, which is in Figure 11, the original sentence reads "Reportedly one of the most important drugs- and gun-smuggling routes in supplying Europe with cocaine runs from Colombia across the northern tier of the Amazon to Suriname ." The baseline translation is missing *Reportedly*. Since the baseline does not take into account the coverage of the AMR graph, some node information may be lost during generation. However, our model does not have this problem and generates the original meaning of the sentences intact. This proves the effectiveness of our coverage mechanism, which ensures the integrity of the information by considering the coverage of the graph.

AMR Graph:
(r / run-04
:ARG1 (r2 / route
:ARG1-of (i / include-91
:ARG2 (r3 / route
:mod (i2 / important
:degree (m / most)
:purpose (s2 / supply-01
:ARG1 (c / cocaine)
:ARG2 (c2 / continent :wiki "Europe"
:name (n / name :op1 "Europe"))))
:path-of (s / smuggle-01
:ARG1 (a / and
:op1 (d / drug)
:op2 (g / gun))))))
:ARG3 (c3 / country :wiki "Colombia"
:name (n2 / name :op1 "Colombia"))
:ARG4 (c4 / country :wiki "Suriname"
:name (n3 / name :op1 "Suriname"))
:ARG1-of (r5 / report-01)
:path (a2 / across
:location (t / tier
:mod (n4 / north)
:part-of (w / world-region :wiki "Amazon_basin"
:name (n5 / name :op1 "Amazon")))))
Gold: Reportedly one of the most important drug- and gun-smuggling routes in supplying Europe with cocaine runs from Colombia across the northern tier of the Amazon to Suriname .
Baseline: One of the most important routes for smuggling drugs and guns from Colombia to Europe runs across the northern Amazon basin to Suriname.
Ours: One of the most important routes for supplying cocaine to Europe reportedly runs across the northern tier of the Amazon from Colombia to Suriname smuggling drugs and guns.

Figure 11: A comparison of our model and the baseline model to generate the result cases

AMR Graph:
(t2 / train-01
:ARG2 (t / they)
:location (i / indoor)
:degree (m / most)
:location (h / home))
Gold: At home , they carry out indoor training mostly.
Baseline: They are mostly trained indoor at home .
Ours: They are mostly being trained indoor at home.

AMR Graph:
(d / do-02
:ARG0 (i / i)
:topic (d2 / disease :wiki "Obsessive–compulsive_disorder"
:name (n / name :op1 "OCD"))
:mod (e / emoticon :value ":P")
:subevent (c / class
:topic (p / psychology)
:time (t / today)))
Gold: I did about OCD in psycology today :P.
Baseline: :P I did a psychology class today on OCD.
Ours: In my psychology class today, I did about OCD (:P).

AMR Graph:
(d2 / disguise-01
:ARG1 (s / suffer-01
:quant (m / more))
:ARG2 (l / look-02
:ARG1 (g / glorious))
:ARG2-of (d / dispute-01
:ARG1-of (p / possible-01 :polarity -)))
Gold: More suffering is under the disguise of glorious looks, 'tis the undisputable fact.
Baseline: More suffering disguised as a glorious look is indisputable.
Ours: The more suffering disguised as a glorious look is indisputable.

Figure 12: A comparison of our model and the baseline model to generate the result cases

A.3 Correct but low scores

The third type is that the generated sentences have the correct meaning, but the scores are low on all three measures. We show three examples related to this problem in Figure 12. The meanings of the generated sentences are the same as the original sentences, regardless of the baseline model or our model. However, since the generated sentences differ in expression, the used metrics are not sufficient to reflect this difference. In other words, the third error is made by metrics, not by our model or baseline. Therefore, in the future, we can add human evaluation or design new automatic metrics in the evaluation of AMR-to-text generation.