Variational Graph Autoencoding as Cheap Supervision for AMR Coreference Resolution

Irene Li[1]*, **Linfeng Song**[2]†, **Kun Xu**[2] and **Dong Yu**[2]
[1]Yale University, CT, USA
irene.li@yale.edu
[2]Tencent AI Lab, Bellevue, WA, USA
{lfsong,kxkunxu,dyu}@tencent.com

Abstract

Coreference resolution over semantic graphs like AMRs aims to group the graph nodes that represent the same entity. This is a crucial step for making document-level formal semantic representations. With annotated data on AMR coreference resolution, deep learning approaches have recently shown great potential for this task, yet they are usually data hungry and annotating data is costly. We propose a general pretraining method using variational graph autoencoder (VGAE) for AMR coreference resolution, which can leverage any general AMR corpus and even automatically parsed AMR data. Experiments on benchmarks show that the pretraining approach achieves performance gains of up to 6% absolute F1 points. Moreover, our model significantly improves on the previous state-of-the-art model by up to 11% F1 points.

1 Introduction

Abstract Meaning Representation (AMR) is a way to preserve the semantic meaning of a sentence in a graph (Banarescu et al., 2013). As shown in Figure 1, AMRs are directed and acyclic graphs where the nodes and edges indicate concepts and their semantic relations. As a sentence-level semantic representation, AMRs have been shown to be effective in many NLP tasks, including text summarization (Liu et al., 2015; Dohare et al., 2018), information extraction (Rao et al., 2017; Li et al., 2020b; Zhang and Ji, 2021), and machine translation (Song et al., 2019; Pham et al., 2020).

More recently, the NLP tasks that are beyond the single-sentence level (Nallapati et al., 2016; Rajpurkar et al., 2016; Li et al., 2017; Chen et al., 2021) are attracting rising attention, and thus representing multiple sentences with AMR becomes important. To expand AMRs to represent multiple sentences, the task of AMR coreference resolution (O'Gorman et al., 2018) has been proposed, aiming at recognizing the concepts from multiple AMRs that represent the same entity. Figure 1 illustrates the AMR graphs of two consecutive sentences in a news article. Given them as the input, an AMR coreference resolver needs to group *police* and *they* (colored with blue), as well as *shop* and the implicit mention *shop* (dashed and colored with pink). Unlike text-based coreference resolution, where dense textual information is available, AMR coreference resolution deals with sparsely connected graphs and implicit graph nodes. More importantly, only a handful of annotated data (around 8K AMRs) exists for AMR coreference resolution. Furthermore, annotating such coreference information and sentence AMRs requires linguists, making the annotation *very costly*. Both situations add extra difficulties to this task.

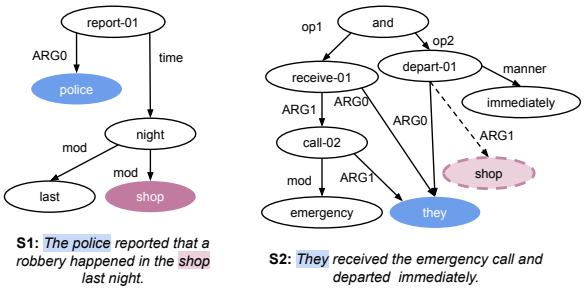

Figure 1: An example of multi-sentence AMR coreference resolution. It contains two coreference clusters, marked by blue and pink respectively: *police* in S1 and *They* in S2; *shop* in S1 and the implicit mention of *shop* (with dashed edge and node) in S2.

Early attempts on AMR coreference resolution adopt rule-based methods. For instance, Liu et al. (2015) only consider the nodes that represent entities (e.g., *police* in Figure 1), and they rely on string match to detect coreference. This method can cause errors, as concepts with the same surface string may not point to the same entity. It also fails

*Work done as an intern at Tencent AI Lab.
†Corresponding author.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 2790 - 2800

to recognize any situations that involve a pronoun (e.g., *police* and *they*). Anikina et al. (2020) build a pipeline system that uses a textual coreference resolution model (Lee et al., 2017) and a text-to-AMR aligner (Flanigan et al., 2014). Though this system can theoretically resolve many situations, in fact, it suffers from severe error propagation (Fu et al., 2021). With the availability of recent human-annotated data (O'Gorman et al., 2018) on AMR coreference resolution, later work starts exploring data-driven models. Fu et al. (2021) extend a standard text-based coreference model (Lee et al., 2017) on AMRs by replacing the LSTM encoder with a graph neural network (GNN). They show a significant performance boost over previous rule-based methods, and their generated document-level AMRs can help a downstream neural summarization system, demonstrating the potential of this task. However, the performance is still far from satisfactory, and they find that the main reason is the lack of annotated data. This calls for approaches that can leverage cheap and/or existing supervision signals to make further improvements.

In this paper, we propose a model and a corresponding pretraining method based on Variational Graph Autoencoder (VGAE) (Kipf and Welling, 2016b). Our model extends AMRCoref (Fu et al., 2021), the current state-of-the-art model, by replacing the core GNN encoder with an improved VGAE encoder. Our model can leverage the reconstruction loss and variational restriction from the VGAE module as additional supervision at no extra cost. Since the loss by our VGAE model can work on any AMR graphs, we also study pretraining our model on the full AMR bank[1] with gold or automatically parsed annotations. In this way, the training signal can be further enriched; thus, the data hunger issue can be alleviated. Though there exist some work applying VAEs and VGAEs on concept knowledge graphs (Li et al., 2020a), corpus-level graphs (Xie et al., 2021) and text (Su et al., 2018), we are the first to study VGAE on a graph-based formal semantic representation, to the best of our knowledge.

Experiments on the MS-AMR benchmark (O'Gorman et al., 2018) show that our model outperforms the previous state-of-the-art system by 11 absolute F1-score points. Besides, we find that pretraining with a larger AMR bank is helpful regardless of whether gold or silver AMR annotations are used. This indicates another potential boost on the performance if more automatically annotated data can be used. Code and pretrained models are made public[2].

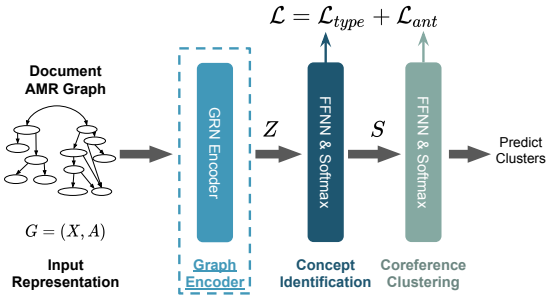

Figure 2: AMRCoref framework (Fu et al., 2021). The dashed rectangle indicates the core graph encoding component, which is also our main focus.

2 Baseline: AMRCoref

We take the end-to-end AMR coreference resolution model (**AMRCoref**, Fu et al. 2021) as our baseline system. Generally, it adapts a text-based end-to-end coreference model (Lee et al., 2017) on AMRs by clustering AMR nodes instead of text spans. Another major difference is that they also consider omitted AMR nodes (e.g., the dashed node *shop* in Figure 1), which are represented by their parent nodes and the corresponding relation (e.g., *depart-01* and *:ARG1*). As illustrated in Figure 2, AMRCoref consists of four essential modules: input representation, graph encoding, node type identification, and antecedent prediction.

2.1 Input Representation

As the first step of AMRCoref, it calculates the embedding $h_i^{(0)}$ for each AMR node x_i from its character-level embedding e_i^c, token-level embedding e_i^t and fixed embedding e_i^{bert} generated by a pretrained BERT model:

$$h_i^{(0)} = W^{concept}([e_i^c; e_i^t; e_i^{bert}]) + b^{concept}, \quad (1)$$

where $W^{concept}$ and $b^{concept}$ are model parameters. The character-level and token-level embeddings can be learned from scratch. One can choose to eliminate BERT embedding e_i^{bert} as a simple base model.

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

[2] https://github.com/IreneZihuiLi/VG-AMRCoref

2.2 Graph Encoder

Next, the representations $H^{(0)} = [h_1^{(0)}, \ldots, h_N^{(0)}]$ of all AMR nodes $X = [x_1, \ldots, x_N]$ are sent to a graph encoder together with the AMR edges. Since the input AMRs are disconnected (each AMR alone represents a sentence), Fu et al. (2021) heuristically connect the root nodes of these sentence AMRs to make a connected graph G. Specifically, $G = (X, A)$, where the edge set A consists of both the original AMR edges and the added ones between pairs of roots.

The graph encoder, f_{GRN}, is based on the Graph Recurrent Network (GRN, Song et al. 2018; Beck et al. 2018). It utilizes the gated operations of an LSTM (Hochreiter and Schmidhuber, 1997) step to simultaneously update each node representation h_i by exchanging information from its incoming N_{in}^i and outgoing neighbors N_{out}^i that can be easily obtained from the edge set A:

$$\begin{aligned} m_{in}^{(l-1)} &= \sum_{j \in N_{in}^i} [h_j^{(l-1)}; r_{ij}], \\ m_{out}^{(l-1)} &= \sum_{j \in N_{out}^i} [h_j^{(l-1)}; r_{ij}], \\ h_i^{(l)} &= \text{LSTM}(h_i^{(l-1)}, [m_{in}^{(l-1)}; m_{out}^{(l-1)}]), \end{aligned} \tag{2}$$

where each r_{ij} represents the embedding of the edge from x_i to x_j. After L steps of information exchange, $z_i = [h_i^{(0)}; h_i^{(L)}]$ is used as the representation of node x_i for the next step.

2.3 Concept Identification

The concept identification subtask is to determine the type for each AMR node from 6 predefined candidate types. Taking Figure 1 as an example, these types are: `func` (functional node like *and*), `ent` (entity node like *police*), `ver` (regular verbal node like *report-01*), `ver`x ($x \in [0, 1, 2]$) (verbal node with implicit argument like *depart-01*).

Given the node representation z_i from the graph encoder, a feed-forward network (FFNN^{type}) with softmax activation is adopted to calculate the probability distribution for its node type p_i^{type}:

$$p_i^{type} = \text{softmax}(\text{FFNN}^{type}(z_i)). \tag{3}$$

This subtask is introduced for detecting implicit mentions as shown in Figure 1, and it can also provide additional supervision defined by cross-entropy loss:

$$\mathcal{L}_{type} = -\frac{1}{N} \sum_{i=1}^{N} \log p_i^{type}[\hat{t}_i], \tag{4}$$

where $\hat{t}_i$ is the index of the correct node type for node x_i.

2.4 Coreference Clustering

In the last step, coreference clusters are predicted by finding the antecedent for each AMR node. Taking node x_i for example, the score of a precedent node x_j being its antecedent is defined as:

$$\begin{aligned} s(x_j, x_i) &= f^m(x_j) + f^m(x_i) + f^{ant}(x_j, x_i), \\ f^m(x_i) &= \text{FFNN}^m([z_i; p_i^{type}]), \\ f^{an}(x_j, x_i) &= \text{FFNN}^{ant}([z_j, z_i]), \end{aligned} \tag{5}$$

where FFNN^m classifies if the given node involves in a coreference link, and FFNN^{ant} determines if the given node pair form a coreference relation. Next, the scores are normalized into a probability distribution via a softmax layer, and the probability $p_{i,j}$ for x_j being the antecedent of x_i is:

$$p_{x_j, x_i} = \frac{e^{s(x_j, x_i)}}{\sum_{x' \in \mathcal{Y}(x_i)} e^{s(x', x_i)}}, \tag{6}$$

where $\mathcal{Y}(x_i)$ represents all precedents of x_i. The antecedent loss is a marginal log-likelihood on all correct antecedents of all the nodes, given the gold clustering for node i is $\text{GOLD}(x_i)$:

$$\mathcal{L}_{ant} = -\log \prod_{i=1}^{N} \sum_{\hat{x} \in \mathcal{Y}(x_i) \cap \text{GOLD}(x_i)} p_{\hat{x}, x_i}. \tag{7}$$

Finally, the training loss is a combination of antecedent loss and node type prediction loss:

$$\mathcal{L} = \mathcal{L}_{type} + \mathcal{L}_{ant}. \tag{8}$$

3 Proposed Method: VG-AMRCoref

This section describes our proposed model (**VG-AMRCoref**) that adopts Variational Graph Autoencoder (VGAE) to enable the cheap supervision of graph reconstruction. For fair comparison, we replace the original graph encoder of *AMRCoref* (Figure 2) with our optimized VGAE module. By doing so, we make it possible to pretrain our model on other standard AMR data for stronger robustness and generalizability. We illustrate the model framework in Figure 3.

3.1 VGAE-based Graph Encoding

After obtaining node embeddings $H^{(0)}$ and the edge set A from the Concept Representation step

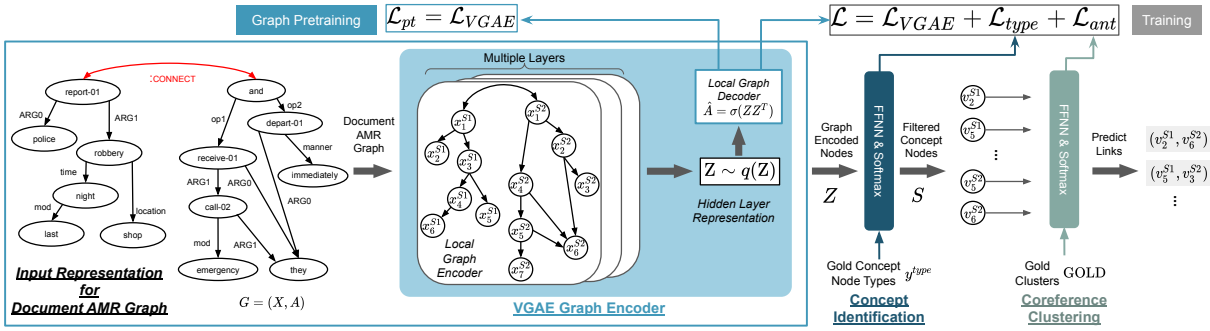

Figure 3: VG-AMRCoref model illustration: The model consists of three main components: Graph Encoder, Concept Identification and Coreference Cluster Prediction.

(Sec.2.1), a VGAE graph encoder is applied to further encode the input graph nodes into the representations with more contextual information. VGAE consists of a local graph encoder and a a local graph decoder.

Local Graph Encoder The local graph encoder functions as a typical graph neural network, where the node features in the lth layer are defines as:

$$H^{(l)} = f(H^{(l-1)}, A). \tag{9}$$

A typical VGAE model usually applies a Graph Convolutional Network (GCN) (Kipf and Welling, 2016a) as its local graph encoder f_{GCN}. Eq. 9 can be further defined as:

$$f_{GCN}\left(H^{(l)}, A\right) = \phi\left(\widetilde{D}^{\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right), \tag{10}$$

where $\phi(\cdot)$ is the Sigmoid activation function, $\widetilde{A} = A+I$, I is the identity matrix, and $\widetilde{D}$ is the diagonal node degree matrix of $\widetilde{A}$.

We study equipping the vanilla VGAE model with other major graph encoders, such as Graph Attention Network (GAT, Veličković et al. 2017) and Graph Recurrent Network (GRN, Beck et al. 2018; Song et al. 2018), to better capture the contextual information of each node. The GAT encoder f_{GAT} considers attention from the neighbors:

$$f_{GAT}\left(H^{(l)}, A\right) = \phi(\sum \boldsymbol{\alpha} W^{(l-1)} H^{(l-1)}),$$
$$\boldsymbol{\alpha} = \text{Attention}(H^{(l-1)}), \tag{11}$$

and the definition of the GRN encoder f_{GRN} is given in Eq. 2.

This local graph encoder also takes L layers. Same with the baseline (Sec. 2.2), we choose the hidden layer features after encoding to be $Z = [H^{(0)}; H^{(L)}]$ for the next step. Besides, Z indicates the stochastic latent variable, and it is modeled by a Gaussian prior distribution $\prod_i \mathcal{N}(z_i, 0, I)$. For $z_i \in Z$:

$$q(z_i|X, A) = \mathcal{N}(z_i|\boldsymbol{\mu}_i, diag(\sigma_i^2)), \tag{12}$$

we have $\boldsymbol{\mu} = f_{\boldsymbol{\mu}}(X, A)$ and $\log \boldsymbol{\sigma} = f_{\boldsymbol{\sigma}}(X, A)$.

Local Graph Decoder The hidden layer representation Z is also fed into a local graph decoder of VGAE. This decoder reconstructs the edge set A from Z. Typically, it is calculated by dot-product:

$$A' = \sigma(ZZ^T),$$
$$p(A' \mid Z) = \prod_{i=1}^{N}\prod_{j=1}^{N} p\left(A'_{ij} \mid z_i, z_j\right). \tag{13}$$

The loss from the VGAE module $\mathcal{L}_{VGAE}$ is defined by the reconstruction loss on the edge set $\mathcal{L}_{edge}$ and the variational restriction on the hidden parameters $\mathcal{L}_{var}$:

$$\begin{aligned}\mathcal{L}_{VGAE} &= \mathcal{L}_{edge} + \mathcal{L}_{var} \\ &= E_{q(Z|X,A)}[\log p(A'|Z)] \\ &- KL[q(Z|X, A)||p(Z)],\end{aligned} \tag{14}$$

where $KL[q(\cdot)||p(\cdot)]$ is the Kullback-Leibler divergence between q and p.

3.2 Task Training

Next, the encoded AMR graph node Z from Eq. 12 is sent to the Concept Identification and Coreference Clustering step, which are described in Sec. 2.3 and 2.4. As shown in Figure 3, the overall loss $\mathcal{L}$ comes from three parts: VGAE loss $\mathcal{L}_{VGAE}$, concept type loss $\mathcal{L}_{type}$ and the antecedent loss $\mathcal{L}_{ant}$, referring to Eq. 14, 4 and 7, respectively:

$$\mathcal{L} = \mathcal{L}_{VGAE} + \mathcal{L}_{type} + \mathcal{L}_{ant}. \tag{15}$$

Data	#Doc	#AMR	#Links	#Nodes
MS-AMR Train	273	7,705	12,003	86,704
MS-AMR Dev	9	121	216	1,599
MS-AMR Test	9	201	404	2,745
LP Test	6	282	463	2,333
Pretraining				
AMR-gold	6,254	49,405	591,918	631,128
AMR-silver	6,227	48,409	468,961	625,040

Table 1: Statistics on the datasets we use for AMR Coreference Resolution.

Encoder	MUC	B^3	$CEAF_{\phi4}$	Avg. F1
GRN	62.31	46.45	44.35	51.04
GCN	69.19	54.00	52.17	58.45
GAT	**70.39**	**55.18**	**52.69**	**59.42**

Table 2: Development results on MS-AMR regarding multiple local graph encoders (**Encoder**).

3.3 Graph Encoder Pretraining

Eq. 14 shows that VGAE can be trained in a self-supervised way, which only needs node features X and the edge set A. So we propose to pretrain the VGAE graph encoder using AMR graphs when only AMR graphs are available. In this pretraining stage, the loss function $\mathcal{L}_{pt}$ is defined as:

$$\mathcal{L}_{pt} = \mathcal{L}_{VGAE}. \tag{16}$$

After pretraining, the VGAE graph encoder will be fine-tuned on the coreference resolution downstream task.

4 Experiments

4.1 Experimental Settings

Datasets Following previous work, we choose the **MS-AMR** benchmark (O'Gorman et al., 2018), which has manually annotated coreference information over gold AMRs. It contains 273 documents for training, 9 for development and 9 for testing. In addition to the in-domain test set, we also evaluate on the Little Prince data (**LP**) that is annotated by (Fu et al., 2021) for out-of-domain evaluation. For pretraining, we choose the AMR bank 3.0 (LDC2020T02), the largest AMR corpus with only regular sentence-level AMRs. Please note that these AMRs are manually labeled and do not contain comprehensive document-level coreference annotations, thus they can not be utilized for task training. We consider this dataset as **AMR-gold**. To reduce the reliance on the annotated dataset, we conduct another setting, **AMR-silver**: we take the sentences of the AMR-gold dataset and apply a well-trained neural AMR parser (Van Noord and Bos, 2017) to generate silver AMR graphs. When doing this, a few documents failed because of post-processing issues[3], so one may notice that it has slight differences with AMR-gold, but we consider this to be acceptable. Smatch F1 score (Cai and Knight, 2013) on the silver results is 0.71, indicating an acceptable AMR parsing quality. We show the statistics in Table 1.

Evaluation Metrics To be consistent with previous work (Fu et al., 2021), we apply three evaluation metrics and an average F1 of all: MUC F1 (Vilain et al., 1995), B^3 F1 (Bagga and Baldwin, 1998) and $CEAF_{\phi4}$ F1 (Luo, 2005).

Hyperparameters For all of the experiment, we follow Fu et al. (2021) to set hyperparameters for fair comparison. For instance, the character embedding and concept type dimension are 32; the concept embedding dimension is 256. The pretrained `BERT-base-cased` model is used. We choose the number of local graph encoder layer of VGAE to be 3, an empirical value following Fu et al. (2021), and provide more details in the Ablation Study later. The optimizer is Adam (Kingma and Ba, 2017). We report average results on 5 runs with different random seeds.

Baselines We choose to compare with the following 4 models. **Rule-based** (Liu et al., 2015): it merges entity nodes with the same surface string to build document AMRs. **Pipeline** (Anikina et al., 2020): it combines a pretrained text-based coreference model and an AMR-to-text aligner into a pipeline, where the text-based coreference resolution results are projected onto AMRs via AMR-to-text alignments. **AMRCoref** and **AMRCoref+bert** are the baselines (Section 2) without and with BERT features, respectively.

4.2 Main Results

Since the local graph decoder has multiple choices including GRN, GCN and GAT, as described in Eq. 9, so we compare the performance on the development set to select the best setting in Table 2. Results show that our model can get the best performance when applying GAT, so we choose this setting in the main experiments.

Table 3 shows the main results on the test set. Here we study three variations of our proposed model: **VG-AMRCoref** learns node em-

[3]More details: https://github.com/RikVN/AMR

Model	In-domain Test Set				Out-domain Test Set			
	MUC	B^3	$CEAF_{\phi4}$	**Avg. F1**	MUC	B^3	$CEAF_{\phi4}$	**Avg. F1**
Rule-based (Liu et al., 2015)	50.80	41.10	22.40	38.10	53.30	41.70	25.90	40.30
Pipeline (Anikina et al., 2020)	58.00	43.00	25.00	42.00	55.20	42.30	26.70	41.40
AMRCoref (Fu et al., 2021)	66.10	49.70	38.10	51.30	64.40	45.80	31.40	47.20
AMRCoref + bert (Fu et al., 2021)	72.50	64.10	50.60	62.40	69.90	61.90	48.50	60.10
Ours								
VG-AMRCoref (GRN)	80.63	56.97	42.10	59.90±0.93	62.03	46.54	42.69	50.42±2.28
VG-AMRCoref	85.96	74.01	56.29	72.08±1.00	74.52	50.36	44.09	56.33±2.43
VG-AMRCoref + pretrain	88.62	75.54	**57.40**	**73.85±1.16**	78.27	55.43	**52.82**	62.18±1.79
VG-AMRCoref + pretrain + bert	**90.25**	**76.43**	53.80	73.49±1.28	**82.89**	**58.59**	48.97	**63.48±1.63**

Table 3: Main results: we compare variations of our proposed model with selected baselines, and report both in-domain and out-domain performances.

beddings from scratch; **VG-AMRCoref+pretrain** first pretrains the VGAE encoder using AMR-gold, and then fine-tune on the task; **VG-AMRCoref+pretrain+bert** is a model that adds pretrained BERT embeddings further. These three models are using GAT as the graph encoder. To compare with Fu et al. (2021) that applies a GRN as the graph encoder, we also conducte the **VG-AMRCoref (GRN)** that applies the same encoder. Both VG-AMRCoref (GRN) and VG-AMRCoref can be fairly compared with AMRCoref, given that they use the same training data and are under the same setting (without BERT). When applying GRN, our model improves about 8.6% and 3.2% Average F1 gains on in- and out-domain. When applying GAT, we could have a significant improvement, specifically, 20.7% and 9.1% Average F1 gains on in- and out-domain. With pretraining, VG-AMRCoref+pretrain performs better than VG-AMRCoref, improving 1.8% and 5.8% on the Average F1 score. This shows that our graph pretraining approach that learns from external data is effective, especially on the out-domain. Finally, we can notice that small gains can be found in the two domains when integrating with BERT knowledge. A possible reason is that only fixed BERT embeddings are applied. Since AMRCoref is undertrained, we see BERT improves the F1 scores by a large margin there. Overall, our best model outperforms the best baseline by around 11.1% and 3.4% on in- and out-domain. Besides, though there is a performance gap between the in- and out- domain test sets, our model shows improvements on both two domains.

One may notice a significant gap between the dev and test results when comparing Table 2 and 3, which is also reported by Fu et al. (2021). After a careful check on the data, we find that the average cluster sizes of the dev and test sets are 3.6 and 5.6, respectively. Since the model predicts as correct if the predicted ancestor is in the same cluster as the current mention, a larger cluster size gives better chances to make correct decisions. We also calculate the average distance between a mention to its closest ancestor, and the values for the dev and test sets are 7.1 and 5.8. This also indicates that the dev set is even more difficult.

Model	MUC	B^3	$CEAF_{\phi4}$	**Avg. F1**
GAT Encoder	84.26	71.39	49.70	68.45
+ VGAE $\mathcal{L}_{var}$	86.29	71.84	54.47	70.87
+ VGAE $\mathcal{L}_{edge}$	85.96	74.01	56.29	72.08

Table 4: Ablation study on VGAE loss components: results on MS-AMR Test set.

4.3 Ablation Study

We include ablation study on VGAE loss, number of graph layers, and the affect of pretraining data size.

VGAE Loss We first study how the VGAE loss from Eq. 14 can affect model performance. We start with a basic setting: applying GAT as the graph encoder (GAT Encoder). Then we add variational restriction (+VGAE $\mathcal{L}_{var}$), as well as the reconstruction loss of edge set (+VGAE $\mathcal{L}_{edge}$). We show the results on the MS-AMR test set in Table 4. With variational loss $\mathcal{L}_{var}$, we see an improvement of about 2.4% of Average F1. And with the edge set reconstruction loss $\mathcal{L}_{edge}$, we see the Average F1 increases again by 1.4%. In total, we see an overall improvement of 3.6% with the VGAE loss.

Number of Graph Layers Previous study shows that more graph layers may hurt the per-

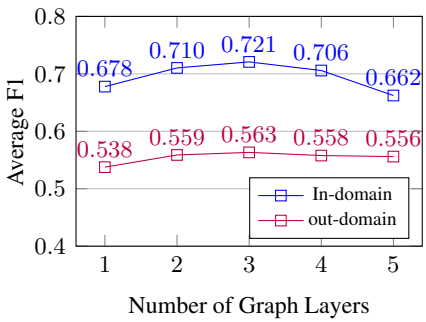

Figure 4: Ablation study on number of graph layers: results on in- and out- domain test sets using VG-AMRCoref model.

formance (Zhou et al., 2020; Fu et al., 2021), due to the over-smoothing issue led by message passing over multiple layers on the graph. We compare 1 to 5 graph layers in the VGAE encoder, and show the Average F1 score of two domains (test set) in Figure 4. When the number of layers is 3, the model achieves the best performance on both in- and out-domain. The performance increases from 1 to 3, and decreases from 3 to 5. This observation is consistent with the AMRCoref model.

Pretraining Data Size Our main results have shown that pretraining on the AMR-gold dataset makes a significant difference, especially for out-domain. We further investigate if our model can benefit from silver AMR data. We compare the Average F1 score with different pretraining sizes of AMR-gold and AMR-silver in Figure 5. In both domains, the x-axis shows the number of pretraining data size. Gold and silver datasets have the same trend: more pretraining data leads to better performance. Though pretraining using the silver dataset is slightly worse than the gold dataset, our model can still improve. Specifically, while the AMR parser (Van Noord and Bos, 2017) is not the current state-of-the-art, results show that applying silver dataset is positive. In the future, we plan to optimize with better AMR parsers and larger datasets to see if the silver data may achieve even better results than the gold dataset.

5 Case Study

To further understand the predicted results of our model, we compare our best performed model (VG-AMRCoref+pretrain+bert) and the best baseline model (AMRCoref+bert) with two case studies.

Figure 6 shows one example taken from the LP test set. Given that the whole document is too long,

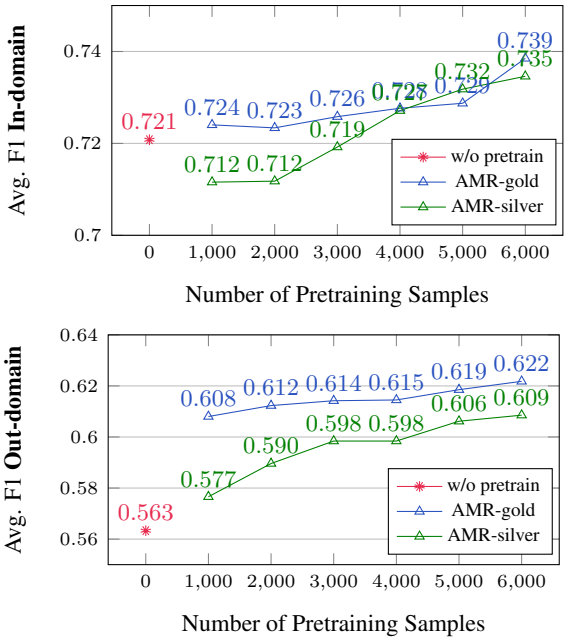

Figure 5: Ablation study on pretraining data size: results on MS-AMR test set.

we keep a part of the content and highlight the coreference cluster tokens with different colors to indicate ground truth, base model prediction and our model prediction. Note that we illustrate both AMRs and original sentences to show the context better, while the sentences were not directly participated in the training and testing. This content piece shows a dialogue between two characters: `me` and `little prince`. In the ground truth, the coreference cluster is indicating little prince, and this can be easily recognized from the token *prince* in S1 and the token *he* in S5 and S6. However, to find out if the token *I* in S3 belongs to this cluster, one needs to read from S1. Because dialogues are going in turns, it is important to figure out which character said S3. Here, the answer should be `little prince` (token *I* means himself) and should be included in the cluster. This could be challenging due to the deep understanding of the previous content and also the difficulty of long dependency. Our model successfully recognized the coreference tokens in this situation.

We illustrate another example from the MS-AMR test set in Figure 7. As can be observed form the ground truth, the highlighted tokens are indicating the coreference cluster of the main character in this article, `I`. The base model predicts a wrong answer in S1 (*who*), and misses the correct token *I* in that sentence. While both models ignore the token *I* in S2 and S3, compare with the base model,

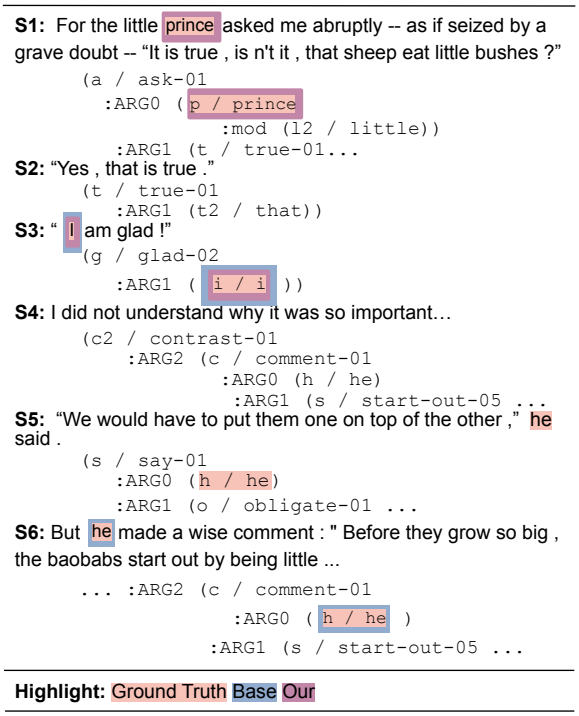

Figure 6: An example from LP Test set: for better understanding, we also put the original sentences here with the AMRs. (Best viewed in Color.)

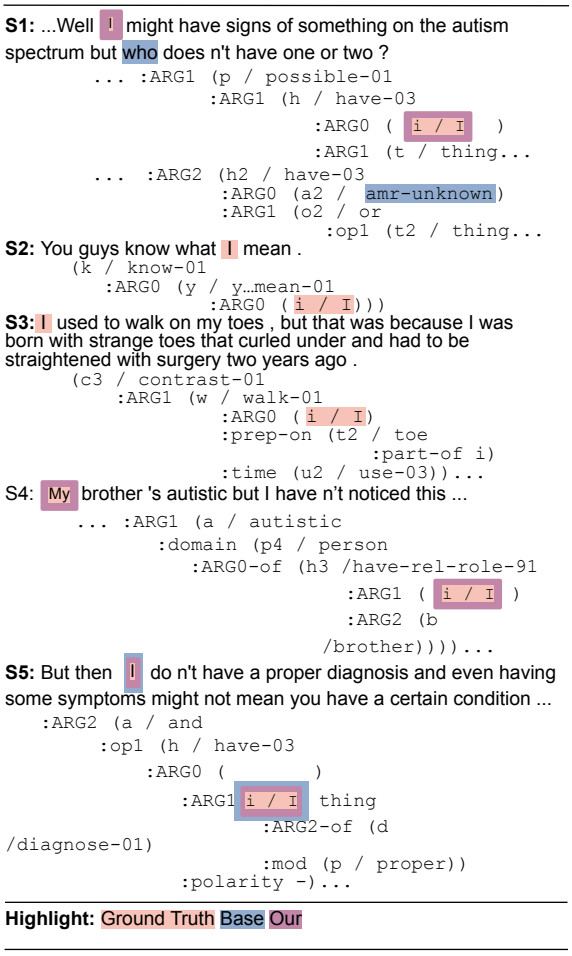

Figure 7: An example from MS-AMR Test set: for a better understanding, we also put the original sentences here with the AMRs. (Best viewed in Color.)

our model is able to identify more correct coreference tokens. Consistent with the previous example case, both models tend to predict only a part of the ground truth that they are more confident with, in order to keep a reasonable good performance.

While automatic evaluation only shows the overall performance, our case studies provide some interesting observations. The base and our model tend to predict fewer coreference nodes than the ground truth, but our model can capture larger and more accurate coreference clusters than the base model.

6 Related Work

Encoding AMRs using Graph Neural Networks Recently, graph neural networks (GNNs) have shown their simplicity and effectiveness in many NLP tasks, especially in encoding graph-structured input, such as knowledge graphs and other task-specific graphs (Li et al., 2020a; Xiong and Gao, 2019; Yin et al., 2019; Song et al., 2020b). Some methods are proposed to encode AMR graphs. For example, Graph Convolutional Networks (GCNs, Kipf and Welling 2016a) and some variations are well-studied for AMRs (Zhang et al., 2020; Cai and Lam, 2020). On the other hand, Song et al. (2019) applied Graph Recurrent Networks (GRNs, Song et al. 2018) on AMRs, achieving reasonable performance for neural machine translation. As a variant of GAT (Veličković et al., 2017), relation-aware self-attention (Shaw et al., 2018) is recently proposed and has been shown more effective (Zhu et al., 2019; Song et al., 2020a) than other GNN variants on presenting AMRs for text generation. We have similar observations where GAT gives better results over GCN and GRN on encoding AMRs for AMR coreference resolution.

Graph Pretraining Previous work shows that pretraining a model may bring better generalizability and performance gain, such as the pretrained language model, BERT (Devlin et al., 2018). There is limited research that focuses on pretraining graph neural networks. The work by Hu et al. (2019) proposes two methods to pretrain GNNs in both

individual node level and the entire graph level. Though there are a few attempts to pretrain GNNs in a similar way with BERT, i.e., Graph Transformer (Dwivedi and Bresson, 2020), and Knowledge Graph Pretraining (Yu et al., 2020), there is still limited study in other NLP tasks. Our work fills this gap by taking advantage of knowledge learned from external data.

Coreference Resolution Coreference resolution has long been an active research topic in NLP. Recently, Clark and Manning (2016) proposed a reinforcement learning approach to optimize a neural mention-ranking model for coreference. The first end-to-end neural coreference resolution method (Lee et al., 2017) targets span embeddings from context-dependent boundary representations using a head-finding attention mechanism. Then, Kantor and Globerson (2019) proposed the Entity Equalization mechanism to capture mentions in clusters using a neural network. Applying these textual coreference methods to AMR graphs requires extra AMR-to-text alignment, which can cause severe error propagation.

To promote multi-sentence AMR coreference resolution, O'Gorman et al. (2018) annotated MS-AMR dataset, which considered coreferences, implicit role coreferences and bridging relations. Very recent work by Fu et al. (2021) is the first end-to-end AMR coreference resolution model for multi-sentence. This model achieves better and robust performance compared with selected baselines.

7 Conclusion

This work proposed a new model (VG-AMRCoref) that is capable of self-supervised training for multi-sentence AMR coreference resolution. It applies VGAEs to encode document-level AMRs, significantly improving performance by up to 11% on the F1 score. We further proposed a simple but effective graph pretraining method using VGAEs, which can simultaneously boost in in-domain and out-domain performances. Analysis shows that potential boost performance may happen if more automatically parsed AMR data is available. One future work will focus on applying larger scale silver AMR datasets for pretraining to improve AMR coreference resolution. Another future direction is to investigate the generated document-level AMRs on more downstream tasks, like question answering and dialogue understanding.

References

Tatiana Anikina, Alexander Koller, and Michael Roth. 2020. Predicting coreference in Abstract Meaning Representations. In *Proceedings of the Third Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 33–38, Barcelona, Spain (online). Association for Computational Linguistics.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Citeseer.

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*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

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

Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7464–7471. AAAI Press.

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

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. Dialsumm: A real-life scenario dialogue summarization dataset. *arXiv preprint arXiv:2105.06762*.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models.

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*.

Shibhansh Dohare, Vivek Gupta, and Harish Karnick. 2018. Unsupervised semantic abstractive summarization. In *Proceedings of ACL 2018, Student Research Workshop*, pages 74–83.

Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*.

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.

Qiankun Fu, Linfeng Song, Wenyu Du, and Yue Zhang. 2021. End-to-end AMR corefencence resolution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4204–4214, Online. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2019. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*.

Ben Kantor and Amir Globerson. 2019. Coreference resolution with entity equalization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 673–677, Florence, Italy. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Thomas N Kipf and Max Welling. 2016a. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Thomas N Kipf and Max Welling. 2016b. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045*.

Irene Li, Alexander Fabbri, Swapnil Hingmire, and Dragomir Radev. 2020a. R-vgae: Relational-variational graph autoencoder for unsupervised prerequisite chain learning. *arXiv preprint arXiv:2004.10610*.

Xuelian Li, Bi Wang, Lixin Li, Zhiqiang Gao, Qian Liu, Hancheng Xu, and Lanting Fang. 2020b. Deep2s: improving aspect extraction in opinion mining with deep semantic representation. *IEEE Access*, 8:104026–104038.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086, Denver, Colorado. Association for Computational Linguistics.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Tim O'Gorman, Michael Regan, Kira Griffitt, Ulf Hermjakob, Kevin Knight, and Martha Palmer. 2018. AMR beyond the sentence: the multi-sentence AMR corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3693–3702, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Viet Pham, Long HB Nguyen, and Dien Dinh. 2020. Semantic convolutional neural machine translation using amr for english-vietnamese. In *Proceedings of the 2020 International Conference on Computer Communication and Information Systems*, pages 52–56.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Sudha Rao, Daniel Marcu, Kevin Knight, and Hal Daumé III. 2017. Biomedical event extraction using Abstract Meaning Representation. In *BioNLP 2017*, pages 126–135, Vancouver, Canada,. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using amr. *Transactions of the Association for Computational Linguistics*, 7:19–31.

Linfeng Song, Ante Wang, Jinsong Su, Yue Zhang, Kun Xu, Yubin Ge, and Dong Yu. 2020a. Structural information preserving for graph-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7987–7998.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2020b. Evidence integration for multi-hop reading comprehension with graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1616–1626. Association for Computational Linguistics.

Jinsong Su, Shan Wu, Biao Zhang, Changxing Wu, Yue Qin, and Deyi Xiong. 2018. A neural generative autoencoder for bilingual word embeddings. *Information Sciences*, 424:287–300.

Rik Van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *arXiv preprint arXiv:1705.09980*.

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

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*.

Qianqian Xie, Jimin Huang, Pan Du, Min Peng, and Jian-Yun Nie. 2021. Inductive topic variational graph auto-encoder for text classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 4218–4227. Association for Computational Linguistics.

Fan Xiong and Jianliang Gao. 2019. Entity alignment for cross-lingual knowledge graph with graph convolutional networks. In *IJCAI*, pages 6480–6481.

Yongjing Yin, Linfeng Song, Jinsong Su, Jiali Zeng, Chulun Zhou, and Jiebo Luo. 2019. Graph-based neural sentence ordering. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5387–5393. International Joint Conferences on Artificial Intelligence Organization.

Donghan Yu, Chenguang Zhu, Yiming Yang, and Michael Zeng. 2020. Jaket: Joint pre-training of knowledge graph and language understanding. *arXiv preprint arXiv:2010.00796*.

Yan Zhang, Zhijiang Guo, Zhiyang Teng, Wei Lu, Shay B. Cohen, Zuozhu Liu, and Lidong Bing. 2020. Lightweight, dynamic graph convolutional networks for AMR-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2162–2172, Online. Association for Computational Linguistics.

Zixuan Zhang and Heng Ji. 2021. Abstract meaning representation guided graph encoding and decoding for joint information extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–49.

Qiji Zhou, Yue Zhang, Donghong Ji, and Hao Tang. 2020. AMR parsing with latent structural information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4306–4319, Online. Association for Computational Linguistics.

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 *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5459–5468.