End-to-End AMR Coreference Resolution

Qiankun Fu[1,2,3], **Linfeng Song**[4], **Wenyu Du**[2,3], **Yue Zhang**[2,3]
1. Zhejiang University
2. School of Engineering, Westlake University
3. Institute of Advanced Technology, Westlake Institute for Advanced Study
4. Tencent AI Lab, Bellevue, WA, USA
fqiankun@gmail.com and lfsong@tencent.com
{duwenyu, zhangyue}@westlake.edu.cn

Abstract

Although parsing to Abstract Meaning Representation (AMR) has become very popular and AMR has been shown effective on many sentence-level tasks, little work has studied how to generate AMRs that can represent multi-sentence information. We introduce the first end-to-end AMR coreference resolution model in order to build multi-sentence AMRs. Compared with the previous pipeline and rule-based approaches, our model alleviates error propagation and it is more robust for both in-domain and out-domain situations. Besides, the document-level AMRs obtained by our model can significantly improve over the AMRs generated by a rule-based method (Liu et al., 2015) on text summarization.

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a semantic formalism for natural language understanding. It represents a sentence as a rooted, directed and acyclic graph, where nodes (e.g., "*Bill*" in Figure 1) represents concepts and edges (e.g., "*:arg0*") are the semantic relations. Encompassing knowledge of named entities, semantic roles and coreference structures, AMR has been proven effective for downstream tasks, including information extraction (Rao et al., 2017), text summarization (Liu et al., 2015; Hardy and Vlachos, 2018; Liao et al., 2018), paraphrase detection (Issa Alaa Aldine et al., 2018), event detection (Li et al., 2015), machine translation (Song et al., 2019b) and dialogue understanding (Bonial et al., 2020).

Existing work on AMR mainly focuses on individual sentences (Lyu and Titov, 2018; Naseem et al., 2019; Ge et al., 2019; Zhang et al., 2019; Cai and Lam, 2020a; Zhou et al., 2020). On the other hand, with the advance of neural networks in NLP, tasks involving multiple sentences with cross-sentence reasoning (e.g., text summarization, reading comprehension and dialogue response generation) have received increasing research attention. Given the effectiveness of AMR on sentence-level tasks (Pan et al., 2015; Rao et al., 2017; Issa Alaa Aldine et al., 2018; Song et al., 2019b), it is important to extend sentence-level AMRs into the multi-sentence level. To this end, a prerequisite step is AMR coreference resolution, which aims to find the AMR components referring to the same entity. Figure 1 shows the AMR graphs of two consecutive sentences in a document. An AMR coreference resolution model need to identify two coreference cases: "*he*" refers to "*Bill*" in the first graph, and "*arrive-01*" omits an argument "*:arg3*" that refers to "*Paris*".

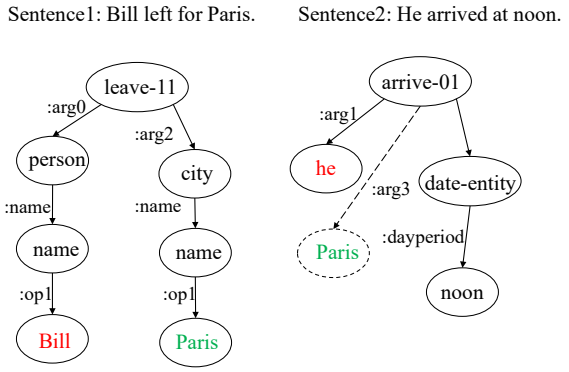

Figure 1: Multi-sentence AMR example, where nodes with the same non-black color are coreferential and the dotted ellipse represents an implicit role coreference.

Relatively little research has been done on AMR coreference resolution. Initial attempts (Liu et al., 2015) merge the nodes that have the same surface string. To minimize noise, only named entities and date entities are considered, and they do not consider merging non-identical nodes (e.g., "*Bill*" and "*he*" in Figure 1) that are also frequent in real-life situation. Subsequent work considers more

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 4204–4214
August 1–6, 2021.

co-reference cases by either manually annotating AMR coreference information (O'Gorman et al., 2018) or taking a pipeline system (Anikina et al., 2020) consisting of a textual coreference resolution model (Lee et al., 2018) and an AMR-to-text aligner (Flanigan et al., 2014). Yet there is little research on automatically resolving coreference ambiguities directly on AMR, making use of AMR graph-structural features.

In this work, we formulate AMR coreference resolution as a missing-link prediction problem over AMR graphs, where the input consists of multiple sentence-level AMRs, and the goal is to recover the missing coreference links connecting the AMR nodes that represent to the same entity. There are two types of links. The first type corresponds to the **standard** situation, where the edge connects two entity nodes (e.g., "*Bill*" and "*he*" in Figure 1) that refer to the same entity. The second type is the **implicit role coreference**, where one node (e.g., "*Paris*" in Figure 1) is a dropped argument ("*:arg3*") of other predicate node ("*arrive-01*").

We propose an AMR coreference resolution model by extending an end-to-end text-based coreference resolution model (Lee et al., 2017). In particular, we use a graph neural network to represent input AMRs for inducing expressive features. To enable cross-sentence information exchange, we make connections between sentence-level AMRs by linking their root nodes. Besides, we introduce a concept identification module to distinguish functional graph nodes (non-concept nodes, e.g., "*person*" in Figure 1), entity nodes (e.g., "*Bill*"), verbal nodes with implicit role (e.g., "*arrive-01*") and other regular nodes (e.g., "*leave-11*") to help improve the performance. The final antecedent prediction is conducted between the selected nodes and all their possible antecedent candidates, following previous work on textual coreference resolution (Lee et al., 2017).

Experiments on the MS-AMR benchmark[1] (O'Gorman et al., 2018) show that our model outperforms competitive baselines by a large margin. To verify the effectiveness and generalization of our proposed model, we annotate an out-of-domain test set over the gold AMR Little Prince 3.0 data following the guidelines of O'Gorman et al. (2018), and the corresponding results show that our model is consistently more robust than the baselines in domain-transfer scenarios. Finally, results on document abstractive summarization show that our document AMRs lead to much better summary quality compared to the document AMRs by Liu et al. (2015). This further verifies the practical value of our approach. Our code and data is available at https://github.com/Sean-Blank/AMRcoref

[1]It consists gold coreference links on gold AMRs.

2 Model

Formally, an input instance of AMR coreference resolution consists of multiple sentence-level AMRs $G_1, G_2, ..., G_n$, where each G_i can be written as $G_i = \langle V_i, E_i \rangle$ with V_i and E_i representing the corresponding nodes and edges for G_i. We consider a document-level AMR graph $\hat{G} = [G_1, G_2, ..., G_n; \hat{e}_1, \hat{e}_2, ..., \hat{e}_m]$, where each $\hat{e}_i$ is a coreference link connecting two nodes from different sentence-level AMRs. The task of AMR coreference resolution aims to recover $\hat{e}_1, ..., \hat{e}_m$, which are missing from the inputs. Figure 2 shows the architecture of our model, which consists of a graph encoder (§ 2.1), a concept identifier (§ 2.2), and an antecedent prediction module (§ 2.3).

2.1 Representing Input AMRs using GRN

Given sentence-level AMRs $G_1, ..., G_n$ as the input, randomly initialized word embeddings are adopted to represent each node v_k as a dense vector $\boldsymbol{e}_k$. To alleviate data sparsity and to obtain better node representation, character embeddings $\boldsymbol{e}_k^{char}$ are computed by using a character-level CNN. We concatenate both $\boldsymbol{e}_k$ and $\boldsymbol{e}_k^{char}$ embeddings for each concept before using a linear projection to form the initial representation:

$$\boldsymbol{x}_k = \boldsymbol{W}^{node}([\boldsymbol{e}_k; \boldsymbol{e}_k^{char}]) + \boldsymbol{b}^{node}, \quad (1)$$

where $\boldsymbol{W}^{node}$ and $\boldsymbol{b}^{node}$ are model parameters.

To enable global information exchange across different sentence-level AMRs, we construct a *draft* document-level graph by connecting the root nodes of each AMR subgraph as shown in Figure 2. This is important because AMR coreference resolution involves cross-sentence reasoning. We then adopt Graph Recurrent Network (GRN, Song et al., 2018; Zhang et al., 2018; Beck et al., 2018) to obtain rich document-level node representations. GRN is one type of graph neural network that iteratively updates its node representations with the message passing framework (Scarselli et al., 2009). Compared with alternatives such as Graph Convolutional Network (GCN, Kipf and Welling 2017;

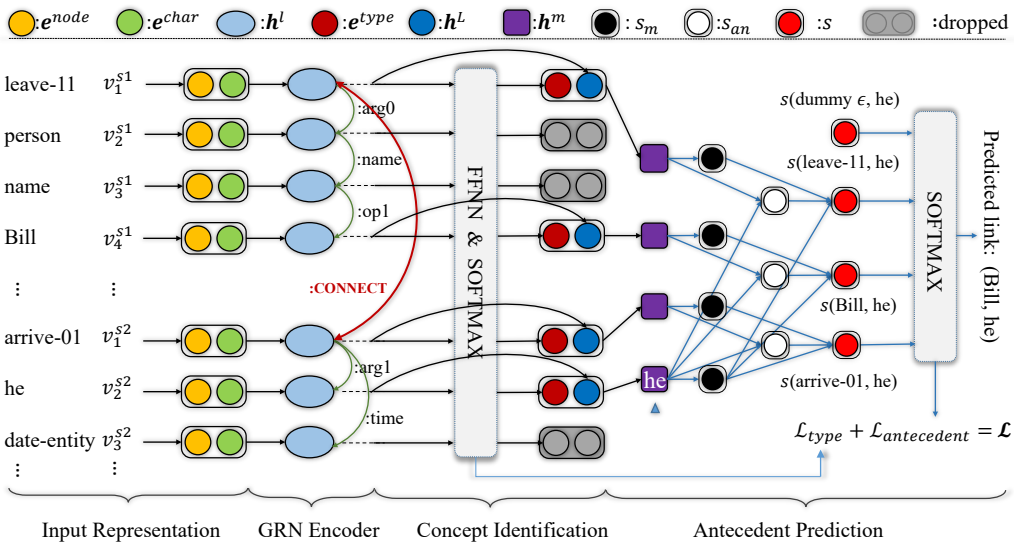

Figure 2: Model framework for end-to-end AMR coreference resolution.

Bastings et al. 2017) and Graph Attention Network (GAT, Veličković et al. 2018), GRN has been shown to give competitive results.

Message passing In the message passing framework, a node v_k receives information from its directly connected neighbor nodes at each layer l. We use a hidden state vector $\boldsymbol{h}_k^l$ to represent each node, and the initial state $\boldsymbol{h}_k^0$ is defined as a vector of zeros.

In the first step at each message passing layer, the concept representation of each neighbor of v_k is combined with the corresponding edge representation to make a message $\boldsymbol{x}_{k,j}$. This is because edges contain semantic information that are important for learning global representation and subsequent reasoning. Formally, a neighbor v_j of node v_k can be represented as

$$\boldsymbol{x}_{k,j} = \boldsymbol{W}^{node}([\boldsymbol{e}_j; \boldsymbol{e}_j^{char}; \boldsymbol{e}_{k,j}^{label}]) + \boldsymbol{b}^{node}, \quad (2)$$

where $\boldsymbol{e}_{k,j}^{label}$ denotes the label embedding of the edge from node v_k and to v_j.

Next, representations of neighboring nodes from the incoming and outgoing directions are aggregated:

$$\begin{aligned} \boldsymbol{x}_k^{in} &= \sum_{i \in \mathcal{N}_{in}(k)} \boldsymbol{x}_{i,k}^l \\ \boldsymbol{x}_k^{out} &= \sum_{j \in \mathcal{N}_{out}(k)} \boldsymbol{x}_{k,j}^l \\ \boldsymbol{x}_k^l &= [\boldsymbol{x}_k^{in}, \boldsymbol{x}_k^{out}], \end{aligned} \quad (3)$$

where $\mathcal{N}_{in}(k)$ and $\mathcal{N}_{out}(k)$ denote the set of incoming and outgoing neighbors of v_k, respectively. Similarly, the hidden states from incoming and outgoing neighbors are also summed up:

$$\begin{aligned} \boldsymbol{m}_k^{in} &= \sum_{i \in \mathcal{N}_{in}(k)} \boldsymbol{h}_i^{l-1} \\ \boldsymbol{m}_k^{out} &= \sum_{j \in \mathcal{N}_{out}(k)} \boldsymbol{h}_j^{l-1} \\ \boldsymbol{m}_k^l &= [\boldsymbol{m}_k^{in}, \boldsymbol{m}_k^{out}], \end{aligned} \quad (4)$$

where $\boldsymbol{h}_j^{l-1}$ denotes the hidden state vector for node v_j at the previous $(l-1)$ layer. Finally, the message passing from layer $l-1$ to l is conducted following the gated operations of LSTM (Hochreiter and Schmidhuber, 1997):

$$\begin{aligned} \boldsymbol{i}_k^l &= \sigma(\boldsymbol{W}_i^m \boldsymbol{m}_k^l + \boldsymbol{W}_i^x \boldsymbol{x}_k^l + \boldsymbol{b}_i) \\ \boldsymbol{o}_k^l &= \sigma(\boldsymbol{W}_o^m \boldsymbol{m}_k^l + \boldsymbol{W}_o^x \boldsymbol{x}_k^l + \boldsymbol{b}_o) \\ \boldsymbol{f}_k^l &= \sigma(\boldsymbol{W}_f^m \boldsymbol{m}_k^l + \boldsymbol{W}_f^x \boldsymbol{x}_k^l + \boldsymbol{b}_f) \\ \boldsymbol{u}_k^l &= \sigma(\boldsymbol{W}_u^m \boldsymbol{m}_k^l + \boldsymbol{W}_u^x \boldsymbol{x}_k^l + \boldsymbol{b}_u) \\ \boldsymbol{c}_k^l &= \boldsymbol{f}_k^l \odot \boldsymbol{c}_k^{l-1} + \boldsymbol{i}_k^l \odot \boldsymbol{u}_k^l \\ \boldsymbol{h}_k^l &= \boldsymbol{o}_k^l \odot \tanh(\boldsymbol{c}_k^l), \end{aligned} \quad (5)$$

where $\boldsymbol{i}_k^l$, $\boldsymbol{o}_k^l$ and $\boldsymbol{f}_k^l$ are a set of input, output and forget gates to control information flow from different sources, $\boldsymbol{u}_k^l$ represents the input messages, $\boldsymbol{c}_k^l$ is the cell vector to record memory, and $\boldsymbol{c}_k^0$ is also initialized as a vector of zeros. $\boldsymbol{W}_z^m$, $\boldsymbol{W}_z^x$ and $\boldsymbol{b}_z$ ($z \in \{i, o, f, u\}$) are model parameters. We adopt L GRN layers in total, where L is determined by a development experiment. The output $\boldsymbol{h}_k^L$ at layer L is adopted as the representation of each node v_k for subsequent procedures.

2.2 Concept Identification

Concept identification aims to distinguish the AMR nodes in regard to its concept type. We consider 6 concept types $\mathcal{T}$ = {func, ent, ver_0, ver_1, ver_2, reg}, which denotes the functional nodes, entity concepts, verbal concepts ver_x with implicit arguments (i.e., "*:arg*x" $x \in \{0, 1, 2\}$[2]) and other regular nodes (e.g., "*leave-11*"), respectively. This module is comparable to the mention detection procedure in textual coreference resolution (Lee et al., 2017).

Formally, a concept representation $\boldsymbol{h}_k^L$ from the top GRN layer is concatenated with a learnable type embedding $\boldsymbol{e}_k^{type}(t)$ of type t for each concept v_k, and the corresponding type score $s_{type}^k(t)$ is computed using a feed-forward network:

$$s_{type}^k(t) = \text{FFNN}_{type}(\boldsymbol{W}^{type}[\boldsymbol{h}_k^L; \boldsymbol{e}_k^{type}(t)]), \quad (6)$$

where $\boldsymbol{W}^{type}$ is a mapping matrix. $\boldsymbol{e}_k^{type}(t)$ represents a concept-type embedding and is randomly initialized. A probability distribution $P(t|v_k)$ over all concept types $\mathcal{T}$ for each concept v_k is calculated as follows using a softmax layer:

$$P(t|v_k) = \frac{e^{s_{type}^k(t)}}{\sum_{t' \in \mathcal{T}} e^{s_{type}^k(t')}}. \quad (7)$$

Finally, we predicate the type t_k^* for each concept

$$t_k^* = \text{argmax}_{t \in \mathcal{T}} \; s_{type}^k(t), \quad (8)$$

and use it to filter the input nodes. In particular, functional concepts are dropped directly and the other concepts (i.e., ent, ver_0, ver_1, ver_2, reg) are selected as candidate nodes for antecedent prediction.

2.3 Antecedent Prediction

Given a selected node v_k by the concept identifier, the goal is to predict its antecedent y_k from all possible candidate nodes $\mathcal{Y}_k = \{\epsilon, y_\pi, ..., y_{k-1}\}$, where a dummy antecedent ϵ is adopted for the nodes that are not coreferent with any previous concepts. $\pi = \min(1, k - \psi)$, where ψ represents the maximum antecedents considered as candidates. As mentioned by previous work on textual coreference resolution (Lee et al., 2017), considering too many candidates can hurt the final performance. We conduct development experiments to decide the best ψ. The finally predicted coreference links implicitly determine the coreference clusters.

[2]We do not model other *:arg*x to avoid long tail issue.

Type information in § 2.2 can help to guide the antecedent prediction and ensure global type consistency. We combine the node hidden vector and its type representation as the final concept state:

$$\boldsymbol{h}_k^m = [\boldsymbol{h}_k^L; \boldsymbol{e}_k^{type}(t^*)], \quad (9)$$

where $\boldsymbol{e}_k^{type}(t^*)$ denotes the learned embedding of the concept type of node v_k.

Similar with Lee et al. (2017), the goal of the antecedent prediction module is to learn a distribution $Q(y_k)$ over the antecedents for each node v_k:

$$Q(y_k) = \frac{e^{s(k,y_k)}}{\sum_{y' \in \mathcal{Y}(k)} e^{s(k,y')}} \quad (10)$$

where $s(k, a)$ computes a coreference link score for each concept pair (v_k, v_a):

$$s(k, a) = s_m(k) + s_m(a) + s_{an}(k, a). \quad (11)$$

Here $a < k$, and $s_m(k)$ means whether concept v_k is a mention involved in a coreference link. It is calculated by using a feed-forward network:

$$s_m(k) = \text{FFNN}_m(\boldsymbol{h}_k^m). \quad (12)$$

$s_{an}(k, a)$ indicates whether mention v_a is an antecedent of v_k and measures the semantic similarity between v_k and v_a, computed with rich features using a feed-forward network:

$$s_{an}(k, a) = \text{FFNN}_{an}([\boldsymbol{h}_k^m, \boldsymbol{h}_a^m, \boldsymbol{h}_k^m \circ \boldsymbol{h}_a^m, \phi(k, a)]) \quad (13)$$

where $\circ$ denotes element-wise multiplication of each mention pair (v_k, v_a), and a feature vector $\phi(k, a)$ represents the normalized distance between two mentions and the speaker information if available. Following Lee et al. (2017), we also normalize the distance values by grouping them into the following buckets [1, 2, 3, 4, 5-7, 8-15, 16-31, 32-63, 64+]. All features (speaker, distance, concept type) are randomly initialized 32-dimensional embeddings jointly learned with the model.

2.4 Training

Our objective function takes two parts: $\mathcal{L}_{type}(\theta)$ (i.e., the concept-type identification loss), and $\mathcal{L}_{antecedent}$ (i.e., the antecedent prediction loss)

$$\mathcal{L}(\theta) = \mathcal{L}_{type}(\theta) + \lambda \mathcal{L}_{antecedent}(\theta), \quad (14)$$

where λ is the weight coefficient (we empirically set $\lambda = 0.1$ in this paper).

Data (portion)	#Doc	#AMR	#Links	#Nodes
MS-AMR (Train)	273	7705	12003	86704
MS-AMR (Dev)	9	121	216	1599
MS-AMR (Test)	9	201	404	2745
LP (Test)	6	282	463	2333

Table 1: Statistics of MS-AMR (first group) and our annotated out-of-domain test data based on LP corpus.

Concept Identification Loss. $\mathcal{L}_{type}$ measures whether our model can accurately identify meaningful concepts and learn the correct type representations. Specifically, given the concept set $V = \{v_1, ...v_N\}$, the concept identifier is trained to minimize an average cross-entropy loss:

$$\mathcal{L}_{type}(\theta) = -\frac{1}{N}\sum_{k=1}^{N} \log P(t_k^*|v_k), \quad (15)$$

where θ are the set of model parameters, $P(t_k^*|v_k)$ denotes the output probability of predicted type t_k^* for each node v_k as in Eq. 7.

Antecedent Prediction Loss. Given a training AMR document with gold coreference clusters $\text{GOLD}(k)|_{k=1}^{N}$ and antecedent candidates $\mathcal{Y}_k = \{\epsilon, y_\pi, ..., y_{k-1}\}$ for mention v_k, $\mathcal{L}_{antecedent}$ measures whether mentions are linked to their correct antecedent. Since the antecedents are latent, the antecedent loss is a marginal log-likelihood of all correct antecedents implied by gold clustering:

$$\mathcal{L}_{antecedent}(\theta) = \prod_{k=1}^{N} \sum_{y \in \mathcal{Y}_k \cap \text{GOLD}(k)} \log Q(y) \quad (16)$$

where $\text{GOLD}(k) = \epsilon$ if mention v_k does not belong to any gold cluster. $Q(y)$ is calculated using Eq. 10.

3 Experiments

We conduct experiments on the MS-AMR dataset[3] (O'Gorman et al., 2018), which is annotated over a previous gold AMR corpus (LDC2017T10). It has 293 annotated documents in total with an average of 27.4 AMRs per document, covering roughly 10% of the total AMR corpus. We split a dev data with the same size as the test set from the training set.

Following the annotation guidelines of MS-AMR, we manually annotate the AMR coreference resolution information over the development and test data of the Little Prince (**LP**) AMR corpus[4] and use it as an out-of-domain test set. For this dataset, we consider each chapter as a document. The data statistics are shown in Table 1.

3.1 Setup

Evaluation Metrics We use the standard evaluation metrics for coreference resolution evaluation, computed using the official CoNLL-2012 evaluation toolkit. Three measures include: MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998) and $CEAF_{\phi_4}$ (Luo, 2005). Following previous studies (Lee et al., 2018), the primary metric AVG-F is the unweighted average of the above three F-scores.

Baselines To study the effectiveness of end-to-end AMR coreference resolution, we compare our model with the following baselines:

- **Rule-based** (Liu et al., 2015): a heuristic method that builds a large document-level AMR graph by linking identical entities.
- **Pipeline** (Anikina et al., 2020): it uses an off-the-shelf coreference system (Lee et al., 2018) with SpanBERT (Joshi et al., 2020) embeddings, and an AMR-to-text aligner (Flanigan et al., 2014). The former generates coreference from text, and the later projects this information from text to AMRs.

Models We study two versions of our model *with* or *without* BERT features.

- **AMRcoref-base**: it corresponds to our model described in § 2 only with word embeddings.
- **AMRcoref-bert**: it denotes our model in § 2 except that the word embeddings ($\boldsymbol{e}_k$ in Eq. 1) are concatenated with BERT outputs. Specifically, we use a cased BERT-base model with fixed parameters to encode a sentence, taking an AMR-to-text aligner (Flanigan et al., 2014) to project BERT outputs to the corresponding AMR nodes.

Hyperparameters We set the dimension of concept embeddings to 256. Characters in the character CNN (§ 2.1) are represented as learned embeddings with 32 units and the convolution window sizes include 2, 3, and 4 characters, each consisting of 100 filters. We use Adam (Kingma and Ba, 2015) with a learning rate of 0.005 for optimization.

[3]The MS-AMR dataset considers 3 types of coreference links: regular, implicit and part-whole. We ignore the last type, which has been challenging and ignored since textual coreference resolution.

[4]https://amr.isi.edu/download/amr-bank-struct-v3.0.txt.

Model	In-domain test set				Out-domain test set			
	MUC	B^3	$CEAF_{\phi_4}$	Average F1	MUC	B^3	$CEAF_{\phi_4}$	Average F1
Rule-based	50.8	41.1	22.4	38.1	53.3	41.7	25.9	40.3
Pipeline	58.0	43.0	25.0	42.0	55.2	42.3	26.7	41.4
AMRcoref-base	66.1	49.7	38.1	51.3	64.4	45.8	31.4	47.2
AMRcoref-bert	**72.5**	**64.1**	**50.6**	**62.4**	**69.9**	**61.9**	**48.5**	**60.1**

Table 2: Main results on the MS-AMR data and LP test sets.

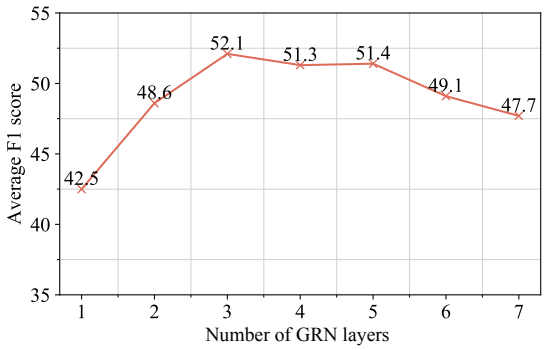

Figure 3: Development results of *AMRcoref-base* on the number of GRN layers.

Distances.	#Links	Cover(%)	F1
$\leqslant 50$	184	85.2	42.9
$\leqslant 100$	206	95.4	45.2
$\leqslant 150$	212	98.1	45.4
$\leqslant 200$	214	99.1	47.2
$\leqslant 250$	215	99.5	52.1
$\leqslant 300$	216	100.0	49.7
> 300	216	100.0	48.3

Table 3: Devset statistics on mention-gold-antecedent distance and the performances of *AMRcoref-base* using the distance as the search space.

3.2 Development Experiments

We first conduct development experiment to choose the values for the crucial hyperparameters.

GRN Encoder Layers The number of recurrent layers L in GRN defines the amount of message interactions. Large message passing layers may lead to over-smoothing problems, while small layers may result in weak graph representation (Qin et al., 2020; Zhang et al., 2018). Figure 3 shows development experiments of the *AMRcoref-base* model in this aspect. We observe large improvements when increasing the layers from 1 to 3, but further increase from 3 to 7 does not lead to further improvements. Therefore, we choose 3 layers for our final model.

Antecedent Candidates How many antecedents are considered as candidates (denoted as ψ in Section 2.3) for making each coreference decision is another important hyperparameter in a coreference resolution model (Lee et al., 2017). Intuitively, allowing more antecedents gives a higher upper bound, but that also introduces a larger search space. Table 3 shows the statistics of the distance between each mention and its gold antecedent and the devset performance of *AMRcoref-base model* that uses this distance as the search space. The performance of *AMRcoref-base* improves when increasing the search space, and the best performance was observed when 250 antecedents are considered as the search space. We choose ψ =250 in subsequent experiments.

3.3 Main Results

Table 2 shows the final in-domain results on the MS-AMR test set and out-domain results on the annotated Little Prince (**LP**) data, where we compare our model (*AMRcoref-base* and *AMRcoref-bert*) with the *Rule-based* and *Pipeline* baselines.

In-domain Results The *Rule-based* method performs the worst, because it only links the identical entity and suffers from low recall. The *Pipeline* model performs better than the *Rule-based* model due to better coverage, but it can suffer from error propagation in both textual coreference and inaccurate AMR aligner. In addition, it does not make use of AMR structure features, which is less sparse compared to text cues. Our proposed *AMRcoref-base* model outperforms the two baselines by a huge margin, gaining at least 9.3% and 13.2% average F1 scores, respectively. This verifies the effectiveness of the end-to-end framework.

Out-domain Results On the cross-domain **LP** data, our model largely outperforms both *Rule-based* method and the *Pipeline* model. Compared with the in-domain setting, there is minor drop on the out-of-domain dataset (4.1% and 2.3% F1 score for *AMRcoref-base* and *AMRcoref-bert* respectively). Neither the performances of *Rule-based* nor *Pipeline* change much on this dataset, which is because these systems are not trained on a certain domain. This also reflects the quality of our LP annotations, because of the consistent performance changes of both *AMRcoref-base* and *AMRcoref-bert* when switching from MS-AMR to LP.

Model	Average F1	Δ
AMRcoref-base	51.3	-
- concept identification	31.4	-19.9
+ gold mention	70.4	+19.1
+ bert concatenate	62.4	+11.1
+ bert graph	62.0	+10.7
- distance feature	49.2	-2.1
- speaker feature	49.4	-1.9
- character CNN	50.1	-1.2
- graph connections	49.0	-2.3

Table 4: Ablation study on the test set of MS-AMR.

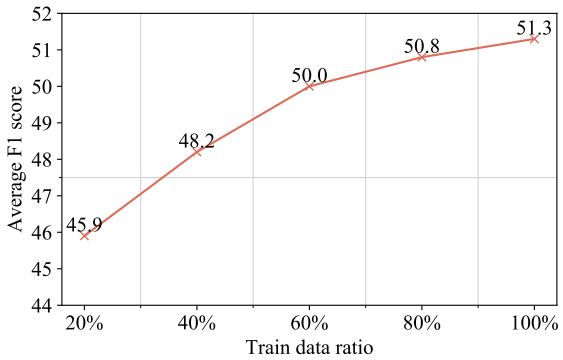

Figure 4: Testing results of *AMRcoref-base* regarding different ratios of training data used. The F1 score of *Pipeline* is 42.0% (Table 2).

3.4 Analysis

We analyze the effects of mention type, textual embedding and various extra features in this section.

Concept Identification As shown in the first group of Table 4, we conduct an ablation study on the concept identification module, which has been shown crucial on the textual coreference resolution (Lee et al., 2017). Removing the concept identifier from the *AMRcoref-base* model results in a large performance degradation of up to 19.9%, indicating that concept type information of the AMR node can positively guide the prediction of coreference links. On the other hand, when the concept identifier outputs are replaced with gold mentions, the results can be further improved by 19.1%. This indicates that better performances can be expected if concept identification can be further improved.

Injecting BERT knowledge As shown in the second group of Table 4, we study the influence of rich features from BERT in our model, which has been proven effective on text-based coreference resolution. Two alternatives of using BERT are studied, *concatenate* (i.e. *AMRcoref-bert*) denotes concatenating the AMR node embeddings with the corresponding textual BERT embedding, and *graph* means that we construct an AMR-token graph that connects AMR nodes and the corresponding tokens. We find that the *AMRcoref-base* model can be improved by a similar margin using both approaches. This is consistent with existing observations from other structured prediction tasks, such as constituent parsing (Kitaev et al., 2019) and dependency parsing (Li et al., 2019). Due to the limited scale of our training data, we expect the gain to be less with more training data.

Features Ablation As shown by the last group in Table 4, we investigate the impacts of each component in our proposed model on the development set of MS-AMR. We have the following observations. **First**, consistent with findings of Lee et al. (2017), the distance between a pair of AMR concepts is an important feature. The final model performance drops by 2.1% when removing the distance feature (Eq. 13). **Second**, the speaker indicator features (Eq. 13) contribute to our model by a 1.9% improvement. Intuitively, speaker information is helpful for pronoun coreference resolution in dialogues. For example, "*my package*" in one sentence may represent identical entity with "*your package*" in the next utterance. **Third**, the character CNN provides morphological information and a way to back off for out-of-vocabulary tokens. For AMR node representations, we see a modest contribution of 1.2% F1 score. **Finally**, we exploit the necessity of cross-sentence AMR connections. Compared to encoding each AMR graph individually, global information exchange across sentences can help to achieve a significant performance improvement.

Data Hunger Similar to other results, it is important to study how much data is necessary to obtain a strong performance (at least be better than the baseline). Figure 4 shows the performances when training the *AMRcoref-base* model on different portions of data. As the number of training samples increases, the performance of our model continuously improves. This shows that our model has room for further improvement with more training data. Moreover, our model even outperforms the *Pipeline* baseline when trained on only 20% data. This confirms the robustness of our end-to-end framework.

Effect of Document Length Figure 5 shows the performance on different MS-AMR document lengths (i.e., the number of AMR graphs in the document). We can see that both our model and the *Pipeline* model show performance decrease

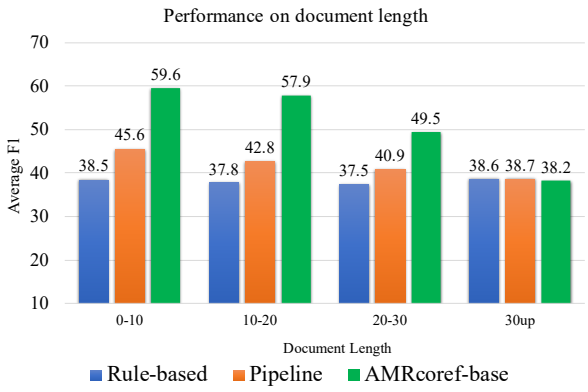

Figure 5: Testing results regarding document length.

when increasing input document length. This is likely because a longer document usually involves more complex coreference situations and brings more challenge for the encoder. Insufficient information interaction for distant nodes further leads to weaker inference performance. As expected, the *Rule-based* approach (Liu et al., 2015) is not significantly affected, but its result is still pretty low. When the document contains more than 30 sentences, the *AMRcoref-base* model slightly under-performs both the *Rule-based* method and the *Pipeline* baseline. One reason is that only a few training instances have a long document length, so we expect that the performance of our model can be further improved given more long documents.

3.5 Application on Summarization

Table 5 compares the summarization performances using the document-level AMRs generated by various methods on the LDC2015E86 benchmark (Knight et al., 2014). Following Liu et al. (2015), Rouge scores (R-1/2/L Lin 2004) are used as the metrics. To consume each document AMR and the corresponding text, we take a popular dual-to-sequence model (D2S, Song et al. 2019b), which extends the standard sequence-to-sequence framework with an additional graph encoder and a dual attention mechanism for extracting both text and graph contexts during decoding.

For previous work, summarization using AMR was first explored by Liu et al. (2015). They first use a rule-based method to build document AMRs and then take a statistic model to generate summaries. Dohare et al. (2017) improves this approach by selecting important sentences before building a document AMR. The *D2S-Rule-based* can be considered as a fair comparison with Liu et al. (2015) on the same summerization platform.

Model	R-1	R-2	R-L
Liu et al. (2015)	44.3	–	–
Dohare et al. (2017)	44.8	17.3	30.6
D2S-Rule-based	47.6	20.1	32.5
D2S-Pipeline	47.9	19.5	32.6
D2S-AMRcoref-base	48.4	20.4	33.2
D2S-AMRcoref-bert	**49.1**	**20.5**	**33.6**

Table 5: Test summarization results on LDC2015E86. R-1/2/L is short for Rouge-1/2/L.

The overall performance of the D2S models outperform the previous approaches, indicating that our experiments are conducted on a stronger baseline. Though *Pipeline* is better than *Rule-based* on AMR coreference resolution, *D2S-Pipeline* is comparable with *D2S-Rule-based* on the downstream summerization task. This shows that the error propagation issue of *Pipeline* can introduce further negative effects to a downstream application. On the other hand, both *D2S-AMRcoref-base* and *D2S-AMRcoref-bert* show much better results than the baselines across all Rouge metrics. This demonstrates that the improvements made by our end-to-end model is solid and can transfer to a downstream application. *D2S-AMRcoref-bert* achieves the best performance, which is consistent with the above experiments.

4 Related Work

Multi-sentence AMR Although some previous work (Szubert et al., 2020; Van Noord and Bos, 2017) explore the coreference phenomena of AMR, they mainly focus on the situation within a sentence. On the other hand, previous work on multi-sentence AMR primarily focuses on data annotation. Song et al. (2019a) annotate dropped pronouns over Chinese AMR but only deals with implicit roles in specific constructions. Gerber and Chai (2012) provide implicit role annotations, but the resources were limited to a small inventory of 5-10 predicate types rather than all implicit arguments. O'Gorman et al. (2018) annotated the MS-AMR dataset by simultaneously considering coreference, implicit role coreference and bridging relations. We consider coreference resolution as the prerequisite for creating multi-sentence AMRs, proposing the first end-to-end model for this task.

Coreference Resolution Coreference resolution is a fundamental problem in natural language processing. Neural network models have shown promising results over the years. Recent work (Lee et al., 2017, 2018; Kantor and Globerson, 2019)

tackled the problem end-to-end by jointly detecting mentions and predicting coreference. Lee et al. (2018) build a complete end-to-end system with the span-ranking architecture and higher-order inference technique. While previous work considers only text-level coreference, we investigate AMR co-reference resolution.

AMR Representation using GNN To encode AMR graphs, many variants of GNNs such as GRNs (Song et al., 2018; Beck et al., 2018), GCNs (Zhou et al., 2020; Zhang et al., 2020) and GATs (Damonte and Cohen, 2019; Cai and Lam, 2020b; Wang et al., 2020) have been introduced. We choose a classic GRN model following Song et al. (2018) to represent our document-level AMR graph and leave the exploiting on a more efficient GNN structure for future work.

5 Conclusion

We investigated a novel end-to-end multi-sentence AMR coreference resolution model using a graph neural network. Compared with previous rule-based and pipeline methods, our model better captures multi-sentence semantic information. Results on MS-AMR (in-domain) and LP (out-of-domain) datasets show the superiority and robustness of our model. In addition, experiments on the downstream text summarization task further demonstrate the effectiveness of the document-level AMRs produced by our model.

In future work, we plan to resolve both the cross-AMR coreference links and the sentence-level ones together with our model.

Acknowledgments

Linfeng Song is the corresponding author. We would like to thank the anonymous reviewers for their insightful comments. We gratefully acknowledge funding from the National Natural Science Foundation of China (NSFC No.61976180). It also receives supported by Tencent AI Lab Rhino-Bird Focused Research Program.

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.

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.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. 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, Melbourne, Australia. Association for Computational Linguistics.

Claire Bonial, Lucia Donatelli, Mitchell Abrams, Stephanie M. Lukin, Stephen Tratz, Matthew Marge, Ron Artstein, David Traum, and Clare Voss. 2020. Dialogue-AMR: Abstract Meaning Representation for dialogue. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 684–695, Marseille, France. European Language Resources Association.

Deng Cai and Wai Lam. 2020a. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020b. Graph transformer for graph-to-sequence learning. In *AAAI*, pages 7464–7471.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3649–3658, Minneapolis, Minnesota. Association for Computational Linguistics.

Shibhansh Dohare, Harish Karnick, and Vivek Gupta. 2017. Text summarization using abstract meaning representation. *arXiv preprint arXiv:1706.01678*.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discrim-

inative 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, Baltimore, Maryland. Association for Computational Linguistics.

DongLai Ge, Junhui Li, Muhua Zhu, and Shoushan Li. 2019. Modeling source syntax and semantics for neural amr parsing. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4975–4981. International Joint Conferences on Artificial Intelligence Organization.

Matthew Gerber and Joyce Y. Chai. 2012. Semantic role labeling of implicit arguments for nominal predicates. *Computational Linguistics*, 38(4):755–798.

Hardy Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using Abstract Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773, Brussels, Belgium. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ahmad Issa Alaa Aldine, Mounira Harzallah, Giuseppe Berio, Nicolas Béchet, and Ahmad Faour. 2018. EXPR at SemEval-2018 task 9: A combined approach for hypernym discovery. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 919–923, New Orleans, Louisiana. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

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. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Kevin Knight, Laura Baranescu, Claire Bonial, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, and Nathan Schneider. 2014. Deft phase 2 amr annotation r1 ldc2015e86. philadelphia: Linguistic data consortium. *Abstract meaning representation (AMR) annotation release*, 1.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. 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 687–692, New Orleans, Louisiana. Association for Computational Linguistics.

Xiang Li, Thien Huu Nguyen, Kai Cao, and Ralph Grishman. 2015. Improving event detection with Abstract Meaning Representation. In *Proceedings of the First Workshop on Computing News Storylines*, pages 11–15, Beijing, China. Association for Computational Linguistics.

Ying Li, Zhenghua Li, Min Zhang, Rui Wang, Sheng Li, and Luo Si. 2019. Self-attentive biaffine dependency parsing. In *IJCAI*, pages 5067–5073.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract Meaning Representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

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.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407, Melbourne, Australia. Association for Computational Linguistics.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4586–4592, Florence, Italy. Association for Computational Linguistics.

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.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of the 2015 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 1130–1139.

Libo Qin, Xiao Xu, Wanxiang Che, and Ting Liu. 2020. AGIF: An adaptive graph-interactive framework for joint multiple intent detection and slot filling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1807–1816, Online. Association for Computational Linguistics.

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.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Li Song, Yuan Wen, Sijia Ge, Bin Li, and Weiguang Qu. 2019a. An easier and efficient framework to annotate semantic roles: Evidence from the chinese amr corpus. In *Workshop on Chinese Lexical Semantics*, pages 474–485. Springer.

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

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 (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Ida Szubert, Marco Damonte, Shay B. Cohen, and Mark Steedman. 2020. The role of reentrancies in Abstract Meaning Representation parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2198–2207, Online. Association for Computational Linguistics.

Rik Van Noord and Johan Bos. 2017. Dealing with co-reference in neural semantic parsing. In *Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*, pages 41–49.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

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*.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. AMR-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

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

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.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state LSTM for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327, Melbourne, Australia. Association for Computational Linguistics.

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.