AMR Parsing with Latent Structural Information

Qiji Zhou[1†], **Yue Zhang**[2,3], **Donghong Ji**[1*], **Hao Tang**[1]
[1]Key Laboratory of Aerospace Information Security and Trusted Computing,
Ministry of Education, School of Cyber Science and Engineering, Wuhan University, China
{qiji.zhou,dhji,tanghaopro}@whu.edu.cn
[2]School of Engineering, Westlake University
[3]Institute of Advanced Technology, Westlake Institute for Advanced Study
yue.zhang@wias.org.cn

Abstract

Abstract Meaning Representations (AMRs) capture sentence-level semantics structural representations to broad-coverage natural sentences. We investigate parsing AMR with explicit dependency structures and interpretable latent structures. We generate the latent soft structure without additional annotations, and fuse both dependency and latent structure via an extended graph neural networks. The fused structural information helps our experiments results to achieve the best reported results on both AMR 2.0 (77.5% Smatch F1 on LDC2017T10) and AMR 1.0 (71.8% Smatch F1 on LDC2014T12).

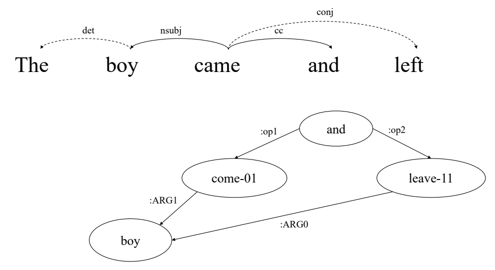

Figure 1: An example of AMR graph and its corresponding syntactic dependencies for the sentence "*The boy came and left*". The dashed lines denote the connected relations in the syntactic dependencies but not appear in the AMR graph.

1 Introduction

Abstract Meaning Representations (AMRs) (Banarescu et al., 2013) model sentence level semantics as rooted, directed, acyclic graphs. Nodes in the graph are concepts which represent the events, objects and features of the input sentence, and edges between nodes represent semantic relations. AMR introduces re-entrance relation to depict the node reuse in the graphs. It has been adopted in downstream NLP tasks, including text summarization (Liu et al., 2015; Dohare and Karnick, 2017), question answering (Mitra and Baral, 2016) and machine translation (Jones et al., 2012; Song et al., 2019).

AMR parsing aims to transform natural language sentences into AMR semantic graphs. Similar to constituent parsing and dependency parsing (Nivre, 2008; Dozat and Manning, 2017), AMR parsers mainly employ two parsing techniques: transition-based parsing (Wang et al., 2016; Damonte et al., 2017; Wang and Xue, 2017; Liu et al., 2018; Guo and Lu, 2018) use a sequence of transition actions to incrementally construct the graph, while graph-based parsing (Flanigan et al., 2014; Lyu and Titov, 2018; Zhang et al., 2019a; Cai and Lam, 2019) divides the task into concept identification and relation extraction stages and then generate a full AMR graph with decoding algorithms such as greedy and maximum spanning tree (MST). Additionally, reinforcement learning (Naseem et al., 2019) and sequence-to-sequence (Konstas et al., 2017) have been exploited in AMR parsing as well.

Previous works (Wang et al., 2016; Artzi et al., 2015) shows that structural information can bring benefit to AMR parsing. Illustrated by Figure 1, for example syntactic dependencies can convey the main predicate-argument structure. However, dependency structural information may be noisy due to the error propagation of external parsers. Moreover, AMR concentrates on semantic relations, which can be different from syntactic dependencies. For instance, in Figure 1, AMR prefers to select the coordination (i.e. "**and**") as the root, which is different from syntactic dependencies (i.e. "**came**").

Given the above observations, we investigate the effectiveness of latent syntactic dependencies for

[†]Part of work was done when the author was visiting Westlake University
[*]Corresponding author.

AMR parsing. Different from existing work (Wang et al., 2016), which uses a dependency parser to provide explicit syntactic structures, we make use of a two-parameter distribution (Bastings et al., 2019) to induce latent graphs, which is differentiable under reparameterization (Kingma and Welling, 2014). We thus build a end-to-end model for AMR parsing with induced latent dependency structures as a middle layer, which is tuned in AMR training and thus can be more aligned to the need of AMR structure.

For better investigating the correlation between induced and gold syntax, and better combine the strengths, we additionally consider fusing gold and induced structural dependencies into an align-free AMR parser (Zhang et al., 2019a). Specifically, we first obtain the input sentence's syntactic dependencies[1] and treat the input sentence as prior of the probabilistic graph generator for inferring the latent graph. Second, we propose an extended graph neural network (GNN) for encoding above structural information. Subsequently we feed the encoded structural information into a two stage align-free AMR parser (Zhang et al., 2019a) for promoting AMR parsing.

To our knowledge, we are the first to incorporate syntactic latent structure in AMR parsing. Experimented results show that our model achieves 77.5% and 71.8% SMATCH F1 on standard AMR benchmarks LDC2017T10 and LDC2014T12, respectively, outperforming all previous best reported results. Beyond that, to some extent, our model can interpret the probabilistic relations between the input words in AMR parsing by generating the latent graph[2].

2 Baseline: Align-Free AMR Parsing

We adopt the parser of Zhang et al. (2019a) as our baseline, which treats AMR parsing as sequence-to-graph transduction.

2.1 Task Formalization

Our baseline splits AMR parsing into a two-stage procedure: **concept identification** and **edge prediction**. The first task aims to identify the concepts (nodes) in AMR graph from input tokens, and the second task is designed to predict semantic relations between identified concepts.

[1]We employ Stanford CoreNLP (Manning et al., 2014) to get the dependencies.

[2]Our code will be available at: `https://github.com/zhouqiji/ACL2020_AMR_Parsing`.

Formally, for a given input sequence of words $\boldsymbol{w} = \langle w_1, ..., w_n \rangle$, the goal of concept identification in our baseline is sequentially predicting the concept nodes $\boldsymbol{u} = \langle u_1, ..., u_m \rangle$ in the output AMR graph, and deterministically assigning corresponding indices $\boldsymbol{d} = \langle d_1, ..., d_m \rangle$.

$$P(\boldsymbol{u}) = \prod_{i=1}^{m} P(u_i \mid u_{<i}, d_{<i}, \boldsymbol{w}),$$

After identifying the concept nodes c and their corresponding indices d, we predict the semantic relations in the searching space $\mathcal{R}(u)$.

$$\text{Predict}(\boldsymbol{u}) = \arg\max_{r \in \mathcal{R}(\boldsymbol{u})} \sum_{(u_i,u_j)\in r} \text{score}(u_i, u_j),$$

where $r = \{(u_i, u_j) \mid 1 \le i, j \le m\}$ is a set of directed relations between concept nodes.

2.2 Align-Free Concept Identification

Our baseline extends the pointer-generator network with *self-copy* mechanism for concept identification (See et al., 2017; Zhang et al., 2018a). The extended model can copy the nodes not only from the source text, but also from the previously generated list of nodes on the target side.

The concept identifier firstly encodes the input sentence into concatenated vector embeddings with GloVe (Pennington et al., 2014), BERT (Devlin et al., 2019), POS (part-of-speech) and character-level (Kim et al., 2016) embeddings. Subsequently, we encode the embedded sentence by a two-layer bidirectional LSTM (Schuster and Paliwal, 1997; Hochreiter and Schmidhuber, 1997):

$$h_i^l = [\overrightarrow{f}^l(h_i^{l-1}, h_{i-1}^l); \overleftarrow{f}^l(h_i^{l-1}, h_{i+1}^l)],$$

where h_i^l is the l-th layer encoded hidden state at the time step i and h_i^0 is the embedded token w_i.

Different from the encoding stage, the decoder does not use pre-trained BERT embeddings, but employs a two-layer LSTM to generate the decoding hidden state s_t^l at each time step:

$$s_t^l = f^l(s_t^{l-1}, s_{t-1}^l),$$

where s_t^{l-1} and s_{t-1}^l are hidden states from last layer and previous time step respectively, and s_0^l is the concatenation of the last bi-directional encoding hidden states. In addition, s_t^0 is generated from the concatenation of the previous node u_{t-1} embedding and the attention vector $\widetilde{s}_{t-1}$, which combine both source and target information:

$$\widetilde{s}_t = \tanh(W_c[c_t; s_t^l] + b_c),$$

where W_c and b_c are trainable parameters, c_t is the context vector calculated by the attention weighted encoding hidden states and the *source attention* distribution a_{src}^t following Bahdanau et al. (2015)

The produced attention vector $\widetilde{s}$ is used to generate the *vocabulary* distribution:

$$P_{\text{vocab}} = \text{softmax}(W_{\text{vocab}}\widetilde{s}_t + b_{\text{vocab}}),$$

as well as the *target attention* distribution:

$$e_{\text{tgt}}^t = v_{\text{tgt}}^\top \tanh(W_{\text{tgt}}\widetilde{s}_{1:t-1} + U_{\text{tgt}}\widetilde{s}_t + b_{\text{tgt}}),$$
$$a_{\text{tgt}}^t = \text{softmax}(e_{\text{tgt}}^t),$$

The *source-side* copy probability p_{src}, *target-side* copy probability p_{tgt} and *generation* probability p_{gen} are calculated by $\widetilde{s}$, which can be treated as generation switches:

$$[p_{\text{src}}, p_{\text{tgt}}, p_{\text{gen}}] = \text{softmax}(W_{\text{switch}}\widetilde{s}_t + b_{\text{switch}}),$$

The *final distribution* is defined below, if v_t is copied from existing nodes:

$$P^{(\text{node})}(u_t) = p_{\text{tgt}} \sum_{i:u_i=u_t}^{t-1} a_{\text{tgt}}^t[i],$$

otherwise:

$$P^{(\text{node})}(u_t) = p_{\text{gen}} P_{\text{vocab}}(u_t) + p_{\text{src}} \sum_{i:w_i=u_t}^{n} a_{\text{src}}^t[i],$$

where $a^t[i]$ is the i-th element of a^t, and then deterministically assigned the existing indices to the identified nodes based on whether the node is generated from the *target-side* distribution.

2.3 Edge Prediction

Our baseline employs a deep biaffine attention classifier for semantic edge prediction (Dozat and Manning, 2017), which have been widely used in graph-based structure parsing (Peng et al., 2017; Lyu and Titov, 2018; Zhang et al., 2019a).

For a node u_t, the probability of u_k being the head node of u_t and the probability of edge (u_k, u_t) are defined below:

$$P_t^{(\text{head})}(u_k) = \frac{\exp\left(\text{score}_{k,t}^{(\text{edge})}\right)}{\sum_{j=1}^{m} \exp\left(\text{score}_{j,t}^{(\text{edge})}\right)},$$

$$P_{k,t}^{(\text{label})}(l) = \frac{\exp\left(\text{score}_{k,t}^{(\text{label})}[l]\right)}{\sum_{l'} \exp\left(\text{score}_{k,t}^{(\text{label})}[l']\right)},$$

where score$^{(\text{score})}$ and label$^{(\text{edge})}$ are calculated via bi-affine attentions.

3 Model

The overall structure of our model is shown in Figure 2. First, we use an external dependency parser (Manning et al., 2014) to obtain the explicit structural information, and obtain the latent structural information via a probabilistic latent graph generator. We then combine both explicit and latent structural information by encoding the input sentence through an extended graph neural network. Finally, we incorporate our model with an align-free AMR parser for parsing AMR graphs with the benefit of structural information.

3.1 Latent Graph Generator

We generate the latent graph of input sentence via the *HardKuma* distribution (Bastings et al., 2019), which has both continuous and discrete behaviours. *HardKuma* can generate samples from the **closed** interval $[0, 1]$ probabilisitcally . This feature allows us to predict soft connections probabilities between input words, which can be seen as a latent graph. Specifically, we treat embedded input words as a prior of a two-parameters distribution, and then sample a *soft* adjacency matrix between input words for representing a dependency.

HardKuma Distribution The *HardKuma* distribution is derived from the *Kumaraswamy* distribution (*Kuma*) (Kumaraswamy, 1980), which is a two-parameters distribution over an **open** interval $(0, 1)$, i.e., $K \sim \text{Kuma}(\boldsymbol{a}, \boldsymbol{b})$, where $\boldsymbol{a} \in \mathbb{R}_{>0}$ and $\boldsymbol{b} \in \mathbb{R}_{>0}$. The *Kuma* distribution is similar to Beta distribution, but its CDF function has a simpler analytical solution and inverse of the CDF is:

$$C_K^{-1}(u; \boldsymbol{a}, \boldsymbol{b}) = \left(1 - (1-u)^{1/\boldsymbol{b}}\right)^{1/\boldsymbol{a}},$$

We can generate the samples by:

$$C_K^{-1}(U; \boldsymbol{\alpha}, \boldsymbol{\beta}) \sim \text{Kuma}(\boldsymbol{\alpha}, \boldsymbol{\beta}),$$

where $U \sim \mathcal{U}(0, 1)$ is the uniform distribution, and we can reconstruct this inverse CDF function by the reparameterizing fashion (Kingma and Welling, 2014; Nalisnick and Smyth, 2017).

In order to include the two discrete points 0 and 1, *HardKuma* employs a *stretch-and-rectify* method with support (Louizos et al., 2017), which leads

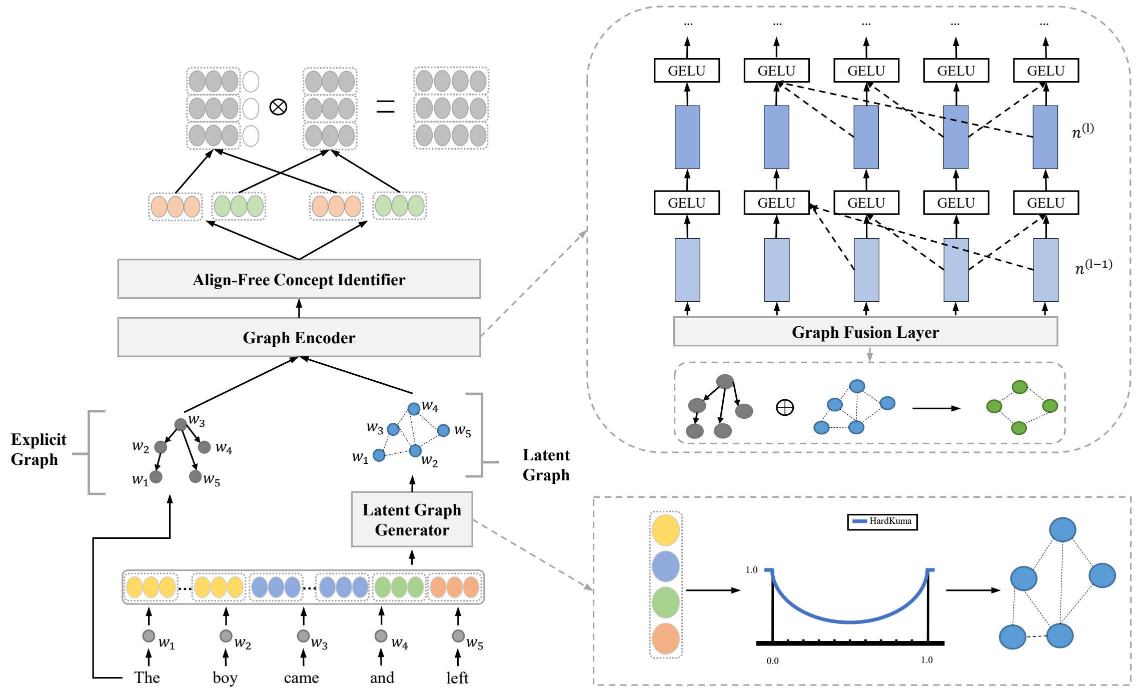

Figure 2: Stretch of the model which has four main components: (1) A latent graph generator for producing the soft-connected latent graph (§§3.1); (2) An extended syntactic graph convolutional network for encoding the structural information (§§3.2); (3) An align-free concept identification for concept node generation (§§2.2); (4) A deep biaffine classifier for relation edge prediction (§§2.3).

the variable $T \sim \text{Kuma}(\boldsymbol{a}, \boldsymbol{b}, l, r)$ to be sampled from *Kuma* distribution with an **open** interval (l, r) where $l < 0$ and $r > 0$. The new CDF is:

$$C_T(t; \boldsymbol{a}, \boldsymbol{b}, l, r) = C_K((t-l)/(r-l); \boldsymbol{a}, \boldsymbol{b}),$$

We pass the stretched variable $T \sim \text{Kuma}(\boldsymbol{a}, \boldsymbol{b}, l, r)$ through a hard-sigmoid function (i.e., $h = \min(1, \max(0, t))$) to obtain the rectified variable $H \sim \text{HardKuma}(\boldsymbol{a}, \boldsymbol{b}, l, r)$. Therefore, the rectified variable covers the **closed** interval $[0, 1]$. Note that all negative values of t are deterministically mapped to 0. In contrast, all samples $t > 1$ are mapped to 1[3]. Because the rectified variable is sampled based on *Kuma* distribution, *HardKuma* first sample a uniform variable over **open** interval $(0, 1)$ from uniform distribution $U \sim \mathcal{U}(0, 1)$, and then generate a *Kuma* variable through inverse CDF:

$$k = C_K^{-1}(u; \boldsymbol{a}, \boldsymbol{b}),$$

Second, we transform the *Kuma* variable for covering the stretched support:

$$t = l + (r - l)k,$$

[3]Details of derivations can be found at (Bastings et al., 2019).

Finally, we rectify the stretched variable including **closed** interval $[0, 1]$ via a hard-sigmoid function:

$$h = \min(1, \max(0, t)).$$

Latent Graph We generate the latent graph of input words w by sampling from *HardKuma* distribution with trained parameters $\boldsymbol{a}$ and $\boldsymbol{b}$. We first calculate the prior $\boldsymbol{c}$ of $(\boldsymbol{a}, \boldsymbol{b})$ by employing multi-head self-attention (Vaswani et al., 2017):

$$\boldsymbol{c}_a = \text{Transfomer}_a(\boldsymbol{v}),$$
$$\boldsymbol{c}_b = \text{Transfomer}_b(\boldsymbol{v}),$$

where $\boldsymbol{v} = \langle v_1, ..., v_n \rangle$ is the embedded input words. Subsequently, we compute $\boldsymbol{a}$ and $\boldsymbol{b}$ as:

$$\boldsymbol{a} = \text{Norm}(\boldsymbol{c}_a \boldsymbol{c}_a^T),$$
$$\boldsymbol{b} = \text{Norm}(\boldsymbol{c}_b \boldsymbol{c}_b^T),$$

where $\boldsymbol{a}_i = \langle a_{i1}, ..., a_{in} \rangle$ and $\boldsymbol{b}_i = \langle b_{i1}, ..., b_{in} \rangle$, $\boldsymbol{c}_a, \boldsymbol{c}_b \in \mathbb{R}^{n \times n}$ and $\text{Norm}(x)$ is the normalization function. Hence, the latent graph $\boldsymbol{L}$ is sampled via learned parameters $\boldsymbol{a}$ and $\boldsymbol{b}$:

$$l_{ij} \sim \text{HardKuma}(a_{ij}, b_{ij}, l, r).$$

3.2 Graph Encoder

For a syntactic graph with n nodes, the cell $A_{ij} = 1$ in the corresponding adjacent matrix represents that an edge connects word w_i to word w_j. An L-layer syntactic GCN of l-th layer can be used to represent A, where the hidden vector for each word w_i at the $l-th$ layer is:

$$h_i^{(l)} = \sigma(\sum_{j=1}^{n} \tilde{A}_{ij} W^{(l)} h_j^{(l-1)} / d_i + b^{(l)}),$$

where $\tilde{\boldsymbol{A}} = A + I$ with the $n \times n$ identity matrix I, $d_i = \sum_{j=1}^{n} \tilde{A}_{ij}$ is the degree of word w_i in the graph for normalizing the activation to avoid the word representation with significantly different magnitudes (Marcheggiani and Titov, 2017; Kipf and Welling, 2017), and σ is a nonlinear activation function.

In order to take benefits from both explicit and latent structural information in AMR parsing, we extend the Syntactic-GCN (Marcheggiani and Titov, 2017; Zhang et al., 2018b) with a graph fusion layer and omit labels in the graph (i.e. we only consider the connected relation in GCN). Specifically, we propose to merge the parsed syntactic dependencies and sampled latent graph through a graph fusion layer:

$$\boldsymbol{F} = \boldsymbol{\pi L} + (1 - \boldsymbol{\pi})\boldsymbol{D}$$

where $\boldsymbol{\pi}$ is trainable gate variables are calculated via the sigmoid function, $\boldsymbol{D}$ and $\boldsymbol{L}$ are the parsed syntactic dependencies and generated latent graph respectively, and F represent the fused soft graph. Furthermore, $\boldsymbol{F}$ is a $n \times n$ adjacent matrix for the input words $\boldsymbol{w}$, different from the sparse adjacent matrix $\boldsymbol{A}$, F_{ij} denote a soft connection degree from word w_i to word w_j. We adapt syntactic-GCN with a fused adjacent matrix F, and employ a gate mechanism:

$$\begin{aligned} h_i^{(l)} &= GELU(\\ &\quad L_{norm}(\sum_{j=1}^{n} G_j (F_{ij} W^{(l)} h_j^{(l-1)} + b^{(l)}))), \end{aligned}$$

We use *GELU* (Hendrycks and Gimpel, 2016) as the activation function, and apply layer normalization L_{norm} (Ba et al., 2016) before passing the results into GELU. The scalar gate G_j is calculated by each edge-node pair :

$$G_j = \mu(h_j^{(l-1)} \cdot \hat{v}^{(l-1)} + \hat{b}^{(l-1)}),$$

where μ is the logistic sigmoid function, $\hat{v}$ and $\hat{b}$ are trainable parameters.

3.3 Training

Similar to our baseline (Zhang et al., 2019a), we linearize the AMR concepts nodes by a pre-order traversal over the training dataset. We obtain gradient estimates of $\mathcal{E}(\phi, \theta)$ through Monte Carlo sampling from:

$$\begin{aligned} \mathcal{E}(\phi, \theta) = \mathbb{E}_{\mathcal{U}(0,I)} [&\log P(node|u_t, g_\phi(\boldsymbol{u}, \boldsymbol{w}), \theta) \\ &+ \log P_t(head|u_k, g_\phi(\boldsymbol{u}, \boldsymbol{w}), \theta) \\ &+ \log P_{k,t}(label|l, g_\phi(\boldsymbol{u}, \boldsymbol{w}), \theta)] \\ &+ \lambda \text{covloss}_t \end{aligned}$$

where u_t is the reference node at time step t with reference head u_k and l is the reference edge label between u_k and u_j. The form $g_\phi(\boldsymbol{u}, \boldsymbol{w})$ is short for the latent graph samples from uniform distribution to *HardKuma* distribution (§§3.1).

Different from Bastings et al. (2019), we do not limit the sparsity of sampled latent graphs, i.e. we do not control the proportion of zeros in the latent graph, because we prefer to retain the probabilistic connection information of each word in w. Finally, we introduce *coverage loss* into our estimation due to reduce duplication of node generation (See et al., 2017).

3.4 Parsing

We directly generate the latent graph by the PDF function of *HardKuma* distribution with trained parameters $\boldsymbol{a}$ and $\boldsymbol{b}$. In the concept identification stage, we decode the node from the final probability distribution $P^{(\text{node})}(u_t)$ at each time step, and apply beam search for sequentially generating the concept nodes $\boldsymbol{u}$ and deterministically assigning corresponding indices $\boldsymbol{d}$. For edge prediction, we use a bi-affine classifier to calculate the edge scores under the generated nodes $\boldsymbol{u}$ and indices $\boldsymbol{d}$:

$$\boldsymbol{S} = \{\text{score}_{i,j}^{(\text{edge})} \mid 0 \leq i, j \leq m\}.$$

Similar to Zhang et al. (2019a), we apply a *maximum spanning tree* (MST) algorithm (Chu, 1965; Edmonds, 1967) to generate complete AMR graph and restore the re-entrance relations by merging the receptive nodes via their indices.

4 Experiments

4.1 Setup

We use two standard AMR corpora: AMR1.0 (LDC2014T12) and AMR 2.0 (LDC2017T10). AMR 1.0 contains 13051 sentences in total. AMR

Data	Parser	F1(%)
AMR 2.0	Cai and Lam (2019)	73.2
	Lyu and Titov (2018)	74.4±0.2
	Lindemann et al. (2019)	75.3±0.1
	Naseem et al. (2019)	75.5
	Zhang et al. (2019a)	76.3±0.1
	- w/o BERT	74.6
	Zhang et al. (2019b)	77.0±0.1
	Ours	**77.5**±0.2
	- w/o BERT	75.5±0.2
AMR 1.0	Flanigan et al. (2016)	66.0
	Pust et al. (2015)	67.1
	Wang and Xue (2017)	68.1
	Guo and Lu (2018)	68.3±0.4
	Zhang et al. (2019a)	70.2±0.1
	- w/o BERT	68.8
	Zhang et al. (2019b)	71.3±0.1
	Ours	**71.8**±0.2
	- w/o BERT	70.0±0.2

Table 1: Main results of SMATCH F1 on AMR 2.0 (LDC2017T10) and 1.0 (LDC2014T12) test sets. Results are evaluated over 3 runs.

Metric	N'19	Z'19$_a$	Z'19$_b$	Ours
SMATCH	75.5	76.3	77	**77.5**
Unlabeled	80	79	80	**80.4**
No WSD	76	77	78	**78.2**
Reentrancies	56	60	61	**61.1**
Concepts	**86**	85	**86**	85.9
Named Ent.	**83**	78	79	78.8
Wikification	80	86	86	**86.5**
Negation	67	75	**77**	76.1
SRL	**72**	70	71	71.0

Table 2: Fine-grained F1 scores on the AMR 2.0 (LDC2017T10) test set. N'18 is Naseem et al. (2019); Z'19$_a$ is Zhang et al. (2019a); Z'19$_b$ is Zhang et al. (2019b)

2.0 is larger which is split into 36521, 1368 and 1371 sentences in training, development and testing sets respectively. We treat in AMR 2.0 as the main dataset in our experiments since it is larger.

We tune hyperparameters on the development set, and store the checkpoints under best development results for evaluation. We employ the pre-processing and post-processing methods from Zhang et al. (2019a), and get the syntactic dependencies via Stanford Corenlp (Manning et al., 2014). We train our model jointly with the Adam optimizer (Kingma and Ba, 2015). The learning rate is decayed based on the results of development set in training. Training takes approximately 22 hours on two Nivida GeForce GTX 2080 Ti.

4.2 Results

Main Results We compare the SMATCH F1 scores (Cai and Knight, 2013) against previous best reported models and other recent AMR parsers. Table 1 summarizes the results on both AMR 1.0 and AMR 2.0 data sets. For AMR 2.0, with the benefit from the fused structural information, we improve our baseline (Zhang et al., 2019a) by 1.2% F1 in the full model, and 0.9% F1 is gained without pre-trained BERT embeddings[4]. In addition, our model outperform the best reported model (Zhang et al., 2019b) by 0.5% F1. On AMR 1.0, there are only about 10k sentences for training. We outperform the best results by 0.5% Smatch F1. We observe that for the smaller data set, our model has a greater improvement of 1.6% F1 than for the larger data set (1.2% F1 comparing with our baseline.)

Fine-grained Results Table 2 shows fined-grained parsing results of each sub-tasks in AMR 2.0, which are evaluated by the enhance AMR evaluation tools (Damonte et al., 2017). We notice that our model brings more than 1% average improvement to our baseline (Zhang et al., 2019a) for most sub-tasks, in particular, the unlabeled is gained 1.4% F1 score increasing with the structural information, and the sub-task of no WSD, reentrancies, negation and SRL are all improved more than 1.0% score under our graph encoder. In addition, our model achieves comparable results to the best reported method (Zhang et al., 2019b) for each sub-task.

Ablation Study We investigate the impacts of different structural information in our model on AMR 2.0 with main sub-tasks[5]. Table 3 shows the fused structure perform better in most sub-task than explicit and latent structure. In particular, the model with explicit structures (i.e. both explicit and fused)

[4]We use pre-tained bert-base embedings without fine-tuning.

[5]We set all the hyper-parameters to the same.

Metric	Explicit	Latent	Fused
SMATCH	77.4	77.4	**77.5**
Unlabeled	80.2	80.1	**80.4**
Reentrancies	**61.1**	60.6	**61.1**
Concepts	85.6	**86.0**	85.7
Negation	75.6	75.1	**76.1**
SRL	70.8	70.9	**71.0**

Table 3: Ablation studies of the results for AMR2.0 (LDC2017T10) on different kind of structural information in our model.

Graph Type	UAS
Fused	84.9%
Latent	64.1%

Table 4: The UAS of fused and latent graph by calculating from the corresponding explicit dependencies in test set (we calculate the UAS by predicting the maximum probability heads in the latent graph).

outperform the model with only latent structure by 0.5% F1 in Reentrancies sub-task, which demonstrates that the explicit dependencies information can improve the this sub-task. Latent structure perform better in concepts sub-task, and fused structure brings more information to the negation sub-task which obtain 0.5% and 1.0% improvement than explicit and latent structure respectively.

Additionally, we can notice that both latent and explicit models outperform the previous best reported Smatch F1 score, and fused model reach the best results. It shows that different types of structural information can help the AMR parsing, we discuss the connection tendencies of each structure in (§§4.3).

4.3 Discussion

Experiment results show that both the explicit structure and latent structure can improve the performance of AMR parsing, and latent structural information reduces the errors in sub-tasks such as concept and SRL. Different from the discrete relation of explicit structures, the internal latent structure holds soft connection probabilities between words in the input sentence, so that, each fully-connected word receive information from all the other words.

Figure 3 depicts the latent and fused soft adjacent matrix of the input sentence *"The boy came and left"* respectively. It can be seen that the la-

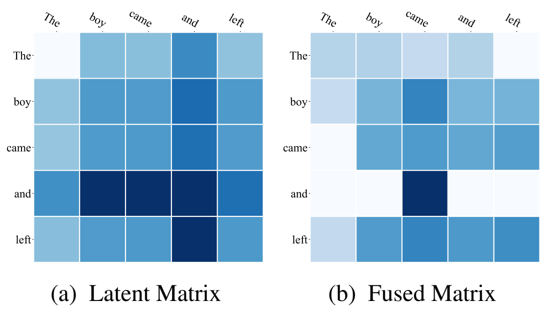

Figure 3: The latent soft adjacent matrix (a) and fused soft adjacent matrix (b) of the input sentence "*The boy came and left*".

tent matrix (Figure 3a) tries to retain information from most word pairs, and the AMR root *"and"* holds high connection probabilities to each word in the sentence. In addition, the main*predicates* and *arguments* in the sentence tend to be connected with high probabilities. The fused matrix (Figure 3b) holds similar connection probabilities to predicates and *arguments* in the sentence as well, and it reduces the connection degrees to the determiner *"The"* which does not appear in corresponding AMR graph. Moreover, the *syntactic root "came"* and semantic root "and" reserve most connection probabilistic to other words.

We compare the connections in different structures in Figure 4. The latent graph (Figure 4a) prefers to connect most words, and the main *predicates* and *arguments* in the graph have higher connection probabilities. The fused graph (Figure 4c) shows that our model provides core structural information between interpretable relations. Specifically, it holds similar potential relations to annotated AMR graph, and tries to alleviate the connection information to the words which are not aligned in AMR concept nodes.

Beyond that, we calculate the Unlabeled Attachment Score (UAS) for fused and latent graph in Table 4, the unsupervised latent graph captures less explicit edges than fused graph, and both fused and latent graph ignore some arcs on explicit graph. It shows that a lower UAS does not mean lower AMR parsing score and some arcs are more useful to AMR parsing but not in explicit gold trees. Consequently, we preserve the explicit and latent structure information simultaneously. The latent structure can not only improve AMR parsing, but also have ability to interpret the latent connections between input words.

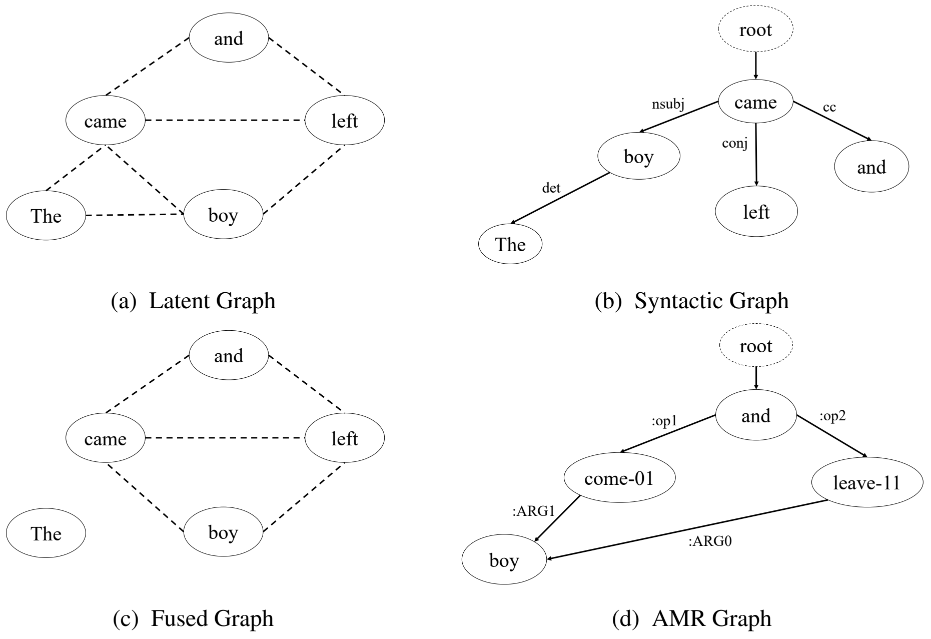

Figure 4: Different structures of the sentence "*The boy came and left* ". (a): The Latent Graph; (b): The Syntactic Graph; (c): The Fused Graph; (d): The AMR graph. (We construct the latent and fused graph by selecting the top 2 possible soft connections between each word, in addition, we ignore the edges whose connection probabilities are less than 0.5.).

5 Related Work

Transition-based AMR parsers (Wang et al., 2016; Damonte et al., 2017; Wang and Xue, 2017; Liu et al., 2018; Guo and Lu, 2018; Naseem et al., 2019) suffer from the lack of annotated alignments between words and concept notes is crucial in these models. Lyu and Titov (2018) treat the alignments as an latent variable for their probabilistic model, which jointly obtains the concepts, relations and alignments variables. Sequence-to-sequence AMR parsers transform AMR graphs into serialized sequences by external traversal rules, and then restore the generated the AMR sequence to avoid aligning issue (Konstas et al., 2017; van Noord and Bos, 2017). Moreover, Zhang et al. (2019a) extend a pointer generator (See et al., 2017), which can generate a node multiple times without alignment through the copy mechanism.

With regards to latent structure, Naradowsky et al. (2012) couples syntactically-oriented NLP tasks to combinatorially constrained hidden syntactic representations. Bowman et al. (2016); Yogatama et al. (2017) and Choi et al. (2018) generate unsupervised constituent tree for text classification. The latent constituent trees are shallower than human annotated, and it can boost the performance of downstream NLP tasks (e.g., text classification). Guo et al. (2019) and Ji et al. (2019) employ self-attention and bi-affine attention mechanism respectively to generate soft connected graphs, and then adopt GNNs to encode the soft structure to take advantage from the structural information to their works.

GCN and its variants are increasingly applied in embedding syntactic and semantic structures in NLP tasks (Kipf and Welling, 2017; Marcheggiani and Titov, 2017; Damonte and Cohen, 2019). Syntactic-GCN tries to alleviate the error propagation in external parsers with gates mechanism, it encodes both relations and labels with the gates, and filters the output of each GCN layer over the dependencies. (Marcheggiani and Titov, 2017; Bastings et al., 2017). Damonte and Cohen (2019) encodes AMR graphs via GCN to promote the AMR-to-text generation task.

6 Conclusion

We investigate latent structure for AMR parsing, and we denote that the inferred latent graph can interpret the connection probabilities between input words. Experiment results show that the latent structural information improve the best reported parsing performance on both AMR 2.0 (LDC2017T10) and AMR 1.0 (LDC2014T12). We also propose to incorporate the latent graph into other multi-task learning problems (Chen et al., 2019; Kurita and Søgaard, 2019).

Acknowledgments

We thank the anonymous reviewers for their detailed comments. We are grateful to Zhiyang Teng's discussions and suggestions. This work is supported by the National Natural Science Foundation of China (NSFC-61772378), the National Key Research and Development Program of China (No.2017YFC1200500) and the Major Projects of the National Social Science Foundation of China (No.11&ZD189). We also would like to acknowledge funding support from the Westlake University and Bright Dream Joint Institute for Intelligent Robotics.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal. Association for Computational Linguistics.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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.

Joost Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.

Joost 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, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1957–1967.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Deng Cai and Wai Lam. 2019. Core semantic first: A top-down approach for AMR parsing. 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 3797–3807, Hong Kong, China. Association for Computational Linguistics.

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.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. A multi-task approach for disentangling syntax and semantics in sentence representations. 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 2453–2464, Minneapolis, Minnesota. Association for Computational Linguistics.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5094–5101.

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

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.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Shibhansh Dohare and Harish Karnick. 2017. Text summarization using abstract meaning representation. *CoRR*, abs/1706.01678.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206, San Diego, California. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime 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, Baltimore, Maryland. Association for Computational Linguistics.

Zhijiang Guo and Wei Lu. 2018. Better transition-based AMR parsing with a refined search space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722, Brussels, Belgium. Association for Computational Linguistics.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 241–251, Florence, Italy. Association for Computational Linguistics.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

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

Tao Ji, Yuanbin Wu, and Man Lan. 2019. Graph-based dependency parsing with graph neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2475–2485, Florence, Italy. Association for Computational Linguistics.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of COLING 2012*, pages 1359–1376, Mumbai, India. The COLING 2012 Organizing Committee.

Yoon Kim, Yacine Jernite, David A. Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2741–2749.

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*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Ponnambalam Kumaraswamy. 1980. A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, 46(1-2):79–88.

Shuhei Kurita and Anders Søgaard. 2019. Multi-task semantic dependency parsing with policy gradient for learning easy-first strategies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2420–2430, Florence, Italy. Association for Computational Linguistics.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585, Florence, Italy. Association for Computational Linguistics.

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.

Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. 2018. An AMR aligner tuned by transition-based parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2422–2430.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2017. Learning sparse neural networks through l_0 regularization. *CoRR*, abs/1712.01312.

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.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Arindam Mitra and Chitta Baral. 2016. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2779–2785.

Eric T. Nalisnick and Padhraic Smyth. 2017. Stick-breaking variational autoencoders. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Jason Naradowsky, Sebastian Riedel, and David Smith. 2012. Improving NLP through marginalization of hidden syntactic structure. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 810–820, Jeju Island, Korea. 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.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *CoRR*, abs/1705.09980.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2037–2048, Vancouver, Canada. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1143–1154, Lisbon, Portugal. Association for Computational Linguistics.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45(11):2673–2681.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at semeval-2016 task 8: An extended transition-based AMR parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 1173–1178.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of AMR parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268, Copenhagen, Denmark. Association for Computational Linguistics.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019a. 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.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. Broad-coverage semantic parsing as transduction. 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 3784–3796, Hong Kong, China. Association for Computational Linguistics.

Sheng Zhang, Xutai Ma, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2018a. Cross-lingual decompositional semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1664–1675, Brussels, Belgium. Association for Computational Linguistics.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018b. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium. Association for Computational Linguistics.

A Appendix

A.1 Details of Model Structures and Parameters

GloVe embeddings	
dim	300
BERT embeddings	
source	BERT-base-cased
dim	768
POS tag embeddings	
dim	100
CharCNN	
num_filters	100
ngram_filter_sizes	[3]
Graph Encoder	
gcn_hidden_dim	512
gcn_layers	1
Latent Graph Generator	
HardKuma_support	[-0.1, 1.1]
k_dim	64
v_dim	64
n_heads	8
Encoder	
hidden_size	512
num_layers	2
Decoder	
hidden_size	1024
num_layers	2
Deep Biaffine Classifier	
edge_hidden_size	256
label_hidden_size	128
Optimizer	
type	Adam
learning_rate	0.001
max_grad_norm	5.0
Coverage loss weight λ	1.0
Beam size	5
Dropout	0.33
Batch Size	
train_batch_size	64
test_batch_size	32

Table 5: Hyper-parameter settings

We select the best hyper-parameters under the results of the development set, and we fix the hyper-parameters at the test stage. We use two-layer highway LSTM as the encoder and two-layer LSTM as the decoder for the align-free node generator. Table 5 shows the details.

A.2 More Examples

To discuss the generated latent graph in different situations, We provide two examples from the test set on the next page.

Figure 5 gives the analysis of an interrogative sentence: "*What advice could you give me?*". It shows that the latent graph of the sentence is going to hold the most information between predicates and arguments. Both the AMR root "advice" and the dependency root "give" are paid more attention from other words, and the fused graph retains more information of the predicates and arguments in the original sentence as well.

For a longer sentence with multiple predicate-argument structures, Figure 6 depicts the latent and fused graph of the sentence "*You could go to the library on saturdays and do a good 8 hours of studying there.*". In this case, the corresponding latent graph becomes shallower, and the AMR root "and" holds most information from other words. Besides, the fused graph indicates that predicates will receive more information from other words, and to some extent, phrases tend to be connected by the fused graph generator.

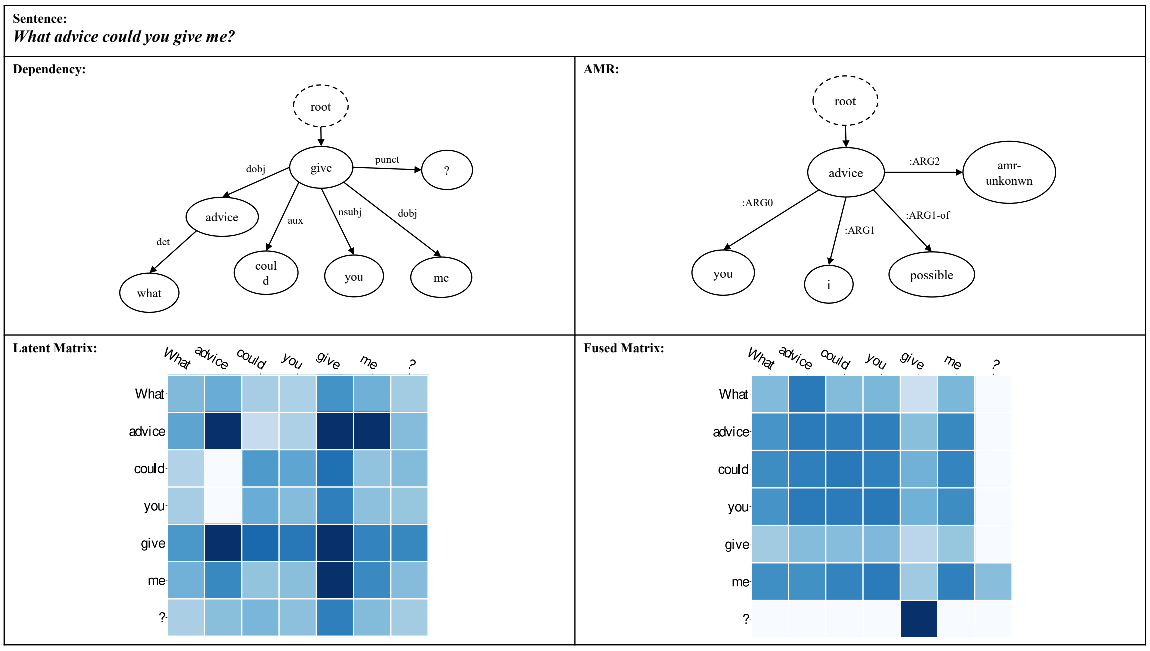

Figure 5: An analysis of the sentence “*What advice could you give me?*”.

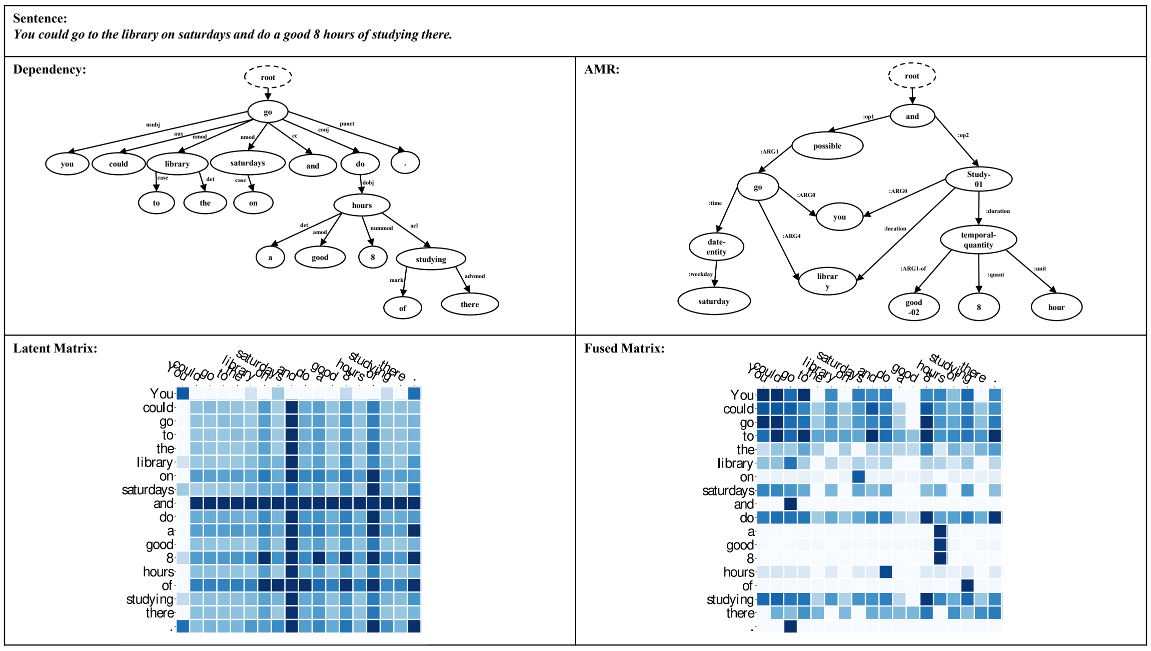
Sentence:
You could go to the library on saturdays and do a good 8 hours of studying there.
Dependency:
AMR:
Latent Matrix:
Fused Matrix:

Figure 6: An analysis of the sentence “*You could go to the library on Saturdays and do a good 8 hours of studying there.*”.