SJTU-NICT at MRP 2019: Multi-Task Learning for End-to-End Uniform Semantic Graph Parsing

Zuchao Li[1,2,3], **Hai Zhao**[1,2,3,*], **Zhuosheng Zhang**[1,2,3],
Rui Wang[4,*], **Masao Utiyama**[4], and **Eiichiro Sumita**[4]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University (SJTU)
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
[3]MoE Key Lab of Artificial Intelligence, AI Institude, Shanghai Jiao Tong University, China
[4]National Institute of Information and Communications Technology (NICT), Kyoto, Japan

Abstract

This paper describes our SJTU-NICT's system for participating in the shared task on Cross-Framework Meaning Representation Parsing (MRP) at the 2019 Conference for Computational Language Learning (CoNLL). Our system uses a graph-based approach to model a variety of semantic graph parsing tasks. Our main contributions in the submitted system are summarized as follows: 1. Our model is fully end-to-end and is capable of being trained only on the given training set which does not rely on any other extra training source including the companion data provided by the organizer; 2. We extend our graph pruning algorithm to a variety of semantic graphs, solving the problem of excessive semantic graph search space; 3. We introduce multi-task learning for multiple objectives within the same framework. The evaluation results show that our system achieved second place in the overall F_1 score and achieved the best F_1 score on the DM framework.

1 Introduction

In recent years, the semantic graph parsing has received a lot of attention from researchers. However, due to the variety of semantic graph flavors, the framework-specific "balkanization" of semantic parsing is worth noting. The 2019 Conference on Computational Language Learning (CoNLL) hosts a shared task on Cross-Framework Meaning Representation Parsing (MRP 2019) (Oepen et al., 2019). From the perspective of the formal representation of semantic graphs, MRP 2019 uses the directed graphs to unify the five different semantic representation frameworks: DELPH-IN MRS Bi-Lexical Dependencies (DM), Prague Semantic Dependencies (PSD), Elementary Dependency Structures (EDS), Universal Conceptual Cognitive Annotation (UCCA), and Abstract Meaning Representation (AMR). Wherein, the directed graph is represented by a $\langle \mathcal{T}, \mathcal{N}, \mathcal{E} \rangle$ triplet, $\mathcal{N}$ represents a set of nodes that constitutes the semantic graph, $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ represents a set of edges that express a specific semantic relationship ($\mathcal{N}$, $\mathcal{E}$ contains a specific attribute corresponding to the semantic framework), and $\mathcal{T}$ represents nodes with a degree of zero in $\mathcal{N}$, usually corresponding to the most central semantic entity.

Though the semantic graph parsing task is uniformly modeled into a directed graph generation task, according to the relationship between nodes in the directed graph and the surface lexical units in the sentence, the five semantic graph frameworks can be divided into three different categories according to the alignment degree between graph nodes and lexical semantics: (1) graph nodes and surface lexical units anchor correspondence strictly (i.e., DM, PSD, EDS), (2) partial graph nodes and surface lexical units anchor correspondence strictly (i.e., UCCA), and (3) graph nodes and surface lexical units have no anchor correspondence (i.e., AMR). As there is a case of anchoring multiple nodes in the

* Corresponding authors. †This work was conductd when Zuchao Li and Zhuosheng Zhang visited NICT as internship students. Email: charlee@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn, zhangzs@sjtu.edu.cn, {wangrui, mutiyama, eiichiro.sumita}@nict.go.jp. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100) and Key Projects of National Natural Science Foundation of China (U1836222 and 61733011). This work was partially conducted under the program "Research and Development of Enhanced Multilingual and Multipurpose Speech Translation Systems" of the Ministry of Internal Affairs and Communications (MIC), Japan. Masao Utiyama is partly supported by JSPS KAKENHI Grant Number 19H05660. Rui Wang was partially supported by JSPS grant-in-aid for early-career scientists (19K20354): "Unsupervised Neural Machine Translation in Universal Scenarios" and NICT tenure-track researcher startup fund "Toward Intelligent Machine Translation".

Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 CoNLL, pages 45–54

https://doi.org/10.18653/v1/K19-2004

corresponding graph of the directed graph of EDS, our system further treats EDS as one type, and DM and PSD as another type.

Based on the experiences of Jiang et al. (2019) and Zhang et al. (2019a) and our previous works on the *Dependency Parsing* (Li et al., 2018a,b,d; Zhou and Zhao, 2019; Zhou et al., 2019), *Semantic Role Labeling* (He et al., 2018b; Cai et al., 2018; Li et al., 2018c, 2019b; He et al., 2019), *Universal Conceptual Cognitive Annotation* (Jiang et al., 2019), *Abstract Mean Representation* (Zhang et al., 2019a), *Machine Translation* (Xiao et al., 2019; Sun et al., 2019; Chen et al., 2019), *Language Modeling* (Li et al., 2019a; Zhang et al., 2019c,b) tasks, we create three graph parsing models based on the semantic graph flavors: (1) Strictly anchored (DM, PSD, EDS): scores the surface lexical units as nodes of the graph, and performs edge training based on the expression of the candidate graph nodes, (2) Non-strictly anchored (UCCA): treats it as a special constituent tree parsing task and uses an additional component to recover the remote edges, and (3) Completely unanchored (i.e., AMR): uses the Seq2seq model to generate the nodes and then performs edge scoring on the generated graph nodes. In order to maintain the end-to-end style of our system, we use the multi-task learning method to jointly train and predict the attributes of nodes and edges together with themselves. We use the pre-trained language model BERT as the encoder. In the training phase, in order to prevent the nodes from falling into local optimum and the edges unable to get enough training, we use the random sampling method on the golden graph nodes to push as many correct nodes as possible to join the edge training. According to the official results of the evaluation, our system ranked second place in the overall F_1 metric among the 16 participating systems. On the DM framework, our system achieved the best results. Our system on other 4 frameworks (PSD, EDS, UCCA, and AMR) are all ranked the third place.

2 Tasks and Modeling

In this section, we will introduce this shared task and our modeling approach. Our key idea is to use a graph-based approach rather than a transition-based one; therefore, all the modeling and optimization methods we have on these frameworks are graph-based. The CoNLL shared task combines the following five frameworks for graph-based meaning representation: DM, PSD, EDS, UCCA, and AMR.

2.1 DM and PSD

The DM (Ivanova et al., 2012) and PSD (Hajic et al., 2012; Miyao et al., 2014) are two independently developed syntactic-semantic annotations which project semantic forms onto bi-lexical dependencies in a lossy manner.

In the representation of the DM and PSD frameworks, the graph nodes and surface lexical units are strictly anchored. There is an explicit, one-to-many anchoring onto sub-strings of the underlying sentence. These graphs are neither fully connected nor rooted. The graphs of DM and PSD have the following features:

- There is only a one-to-one correspondence[1] between the graph node and the span in the sentence.
- Graph nodes can have multiple in-edges or out-edges.
- Graph nodes can be completely isolated, with no in-edges or out-edges.
- There is at most one edge between any two graph nodes.

According to the above properties, the task is modeled as follows: Given a sentence $S = \{w_1, w_2, ..., w_n\}$, enumerate all the span in the sentence $span_{i,j} = \{w_i, w_{i+1}, ..., w_j\}, (i <= j)$, which is used as a candidate graph node and is fed into the node classifier $classifier_n$[2] to filter the truly graph nodes: $node_k = classifier_n(span_{i,j})$, and then uses the edge classifier $classifier_e$ to obtain the semantic relationship between the two graph nodes $edge_{k_1,k_2} = classifier_e(node_{k_1}, node_{k_2})$.

2.2 EDS

EDS is a variable-free semantic dependency graph representation proposed by Oepen and Lønning

[1] Due to one span in the sentence includes several surface lexical units; therefore the one-to-many anchoring becomes one-to-one correspondence.

[2] Here, we summarize the general terminology of using only the $classifier$. In practice, it is possible that the $classifier$ contains several attribute classifiers, depending on how many attributes of the node or edge need to be predicted.

(2006) which encode the English Resource Semantics (ERS) (Flickinger et al., 2014). The EDS conversion from under-specified logical forms of the full ERS to variable-free graphs discards partial semantic information which makes the graph abstractly.

In the representation of the EDS framework, the graph nodes are independent of surface lexical units. For each graph node, there is an explicit, many-to-many anchoring onto sub-strings of the underlying sentence. The EDS graph has the following features:

- There is a many-to-one correspondence between the graph node and the span in the sentence.
- Graph nodes do not correspond to individual surface tokens in the sentence.
- Graph nodes can not be completely isolated and have at least one in- or out-edge.

According to the above features, since there is a many-to-one correspondence between the graph nodes and the spans in the sentences, it is impossible to use the modeling method of DM and PSD simply. Therefore, we adopt a `pseudo` node method to solve the problem. The transformation is carried out: the pseudo node has a one-to-one relationship with the span in the sentence. The edge between nodes in the graph is transformed into the edge of the pseudo node, and two attributes are added for the edge: the source node label and the target node label which are used to indicate the node label in the original EDS graph. In this way, the many-to-one relationship is converted into a one-to-one relationship. After conversion, we can model the problem using in the same way as DM and PSD as described in Subsection 2.1.

2.3 UCCA

UCCA is a multi-layer linguistic framework for semantic annotation proposed by Abend and Rappoport (2013). UCCA aims to recognize the level of semantic granularity which abstracts away from syntactic paraphrases in a typologically-motivated, cross-linguistic fashion and does not need to rely on language-specific resources.

In the representation of the UCCA framework, some nodes have a one-to-one correspondence with the span in the sentence, which is called terminal nodes[3]. Other nodes do not have any corresponding relationship with the span, which is introduced as a notion of a semantic constituency that transcends the pure dependency graphs to represent the semantic granularity. The UCCA graph has the following features:

- There is a one-to-one correspondence between the terminal nodes and the spans in the sentence.
- Graph nodes may have multiple parents, among which one is annotated as the primary parent and others as remote parents.
- The primary edges between nodes and their primary parents form a tree structure, whereas the remote edges between nodes and their remote parents enable the reentrancy, forming directed acyclic graphs (DAGs).
- The non-terminal nodes may exist discontinuous leaves; in which some terminal nodes are not its descendants.

Based on the above features and inspired by Nivre and Nilsson (2005), we transform the tree composed of primary edges (and nodes) into a constituent syntax tree structure, which is modeled using the constituent syntax tree parsing schema. Use an additional classifier for the remote edges prediction and recovery.

2.4 AMR

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) parsing is the task of transducing natural language text into AMR, which is a graph-based formalism used for capturing sentence-level semantics. The AMR framework backgrounds notions of compositionality and derivation, therefore, without explicit correspondence between graph nodes and lexical units.

In the representation of AMR framework, the graph nodes are obtained by composition, derivation, lexical decomposition, normalization towards verb senses and so on. The features of the AMR graphs built on these graph nodes is similar

[3]MRP-transformed UCCA graph differs from on the terminal nodes from the original UCCA graph. In the original UCCA graph representation, terminal nodes refer to words, and in the MRP-transformed UCCA graph, terminal nodes refer to the lowest layer of non-terminal nodes in the original UCCA graph due to a node can contain multiple words in the MRP-transformed UCCA graph.

to the dependency syntax tree except for the reentrancy. Therefore, if the node is determined, modeling can be performed using the method of dependency syntax parsing. However, we can't get the nodes of the graph directly from the sentence due to the nature of the AMR framework. Therefore, inspired by Zhang et al. (2019a), we model the nodes determination as sequence generation tasks using the Seq2seq model and then parse the tree structure on the generated nodes.

3 Data and Preprocessing

3.1 Data

The CoNLL shared task provides a training dataset of 5 subtasks, of which DM, PSD, and EDS are from Wall Street Journal (WSJ) text of Penn Treebank (Marcus et al., 1993) and contain 35,656 sentences. The UCCA training data comes from the English Web Treebank's reviews text (Bies et al., 2012) and the English Wikipedia celebrity articles, with a total data volume of 5,672 sentences. AMR annotation data are drawn from a variety of texts, including online discussion forums, newswires, folktales, novels, and Wikipedia articles, which contain a total of 56,240 sentences.

3.2 Tokenization, Lemmatization, and Anchor conversion

Since the sentence in the training dataset is the original text and no tokenization is performed, and the subsequent processing requires the word root form, we use the Stanford NLP toolkit[4] (Manning et al., 2014) to tokenize and lemmatize the original text. As the graph node anchor in the original data is defined at the character level, we need to convert the anchor to the word level. In this process, due to the difference in tokenization criteria and the existence of tokenizing errors, some graph nodes will be converted into the same one in the process of conversion to word-level anchor. Therefore, we performed some post-processing modifications on the tokenization results of the Stanford NLP toolkit to ensure that the graph nodes after the conversion to the word level anchor correspond to the previous character level, without increasing or decreasing the nodes.

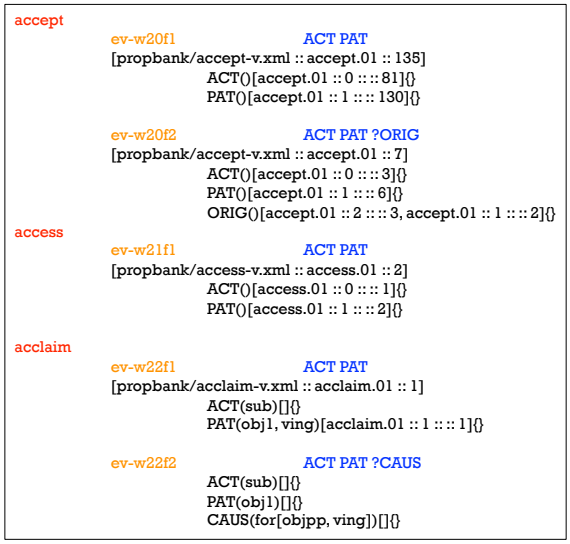

Figure 1: Examples of the most frequent frame-to-frameset mapping extracted from "rng_pb_links.txt".

3.3 Frame Label Projection in PSD Framework

The node label in the PSD framework is a special item id for Engvallex-to-PropBank mapping dictionary. The node label contains the item id of the item in the dictionary and the format id of the item. Such as: [access: ev-w22f1 ACT PAT] where 22 is the item id (word id) and 1 is the format id. Therefore, it is not convenient to use the classifier directly for prediction on the raw node label. Due to the word has a one-to-many relationship with the item id, we cannot obtain this item id by word directly. By observing "rng_pb_links.txt"[5] as shown in Figure 1, the item id has a one-to-one correspondence with its usage pattern string (like "ACT PAT") in the case of word determination, and the usage pattern has duplicates among different words, the number is much smaller than all item ids size; thus it is more suitable as a learning goal. In the subsequent recovery process, we can use lemma and the usage pattern to restore to the item id.

3.4 Graph to Constituent Tree Conversion in UCCA Framework

As described in subsection 2.3, the features, and modeling approach, we need to preprocess the UCCA graph in the training set, transforming the graph into a constituent tree by removing the remote edges and the edges that cause the

[4] https://stanfordnlp.github.io/CoreNLP/index.html.

[5] https://ufal.mff.cuni.cz/pcedt2.0/publications/eng_pb_links.txt

discontinuous leaves. We have adopted the same transformation method as Jiang et al. (2019). The simple steps are as follows:

1) Remote edges removal. In the UCCA MRP representation graph, we remove all edges with the $remote = True$ attribute. The label of the primary edge corresponding to the remote edge is added with a "remote" suffix to distinguish the node with the remote relationship from the ordinary node and subsequent recovery of the remote edge.

2) Constituency continuity formation. Since the current mainstream constituent parsing method requires continuity of constituency, we need to process the discontinuous nodes of the tree species obtained in the previous step. For detailed conversion steps, see Algorithm 1.

Algorithm 1 Constituency continuity formation

Input:
A tree with discontinuous leaves, T_d;
Output:
A constituent tree, T_c;

set $T(t) = T_d$
repeat
  set $n(t)$ is a non-descendant node with discontinuous leaves;
  find the discontinuous spans S_d in the range of $n(t)$;
  for each span $s \in S_d$ **do**
    for each word $w \in s$ **do**
      find a maximum range parent node n_p of word w whose range size is less than s;
      move node n_p to be the child of $n(t)$, and concatenate the original edge label with "ancestor-d" where d represents the original number of edges between the ancestor of n_p and $n(t)$;
      remove all the children words of n_p from s;
    end for
  end for
until $T(t)$ is a constituent tree
set $T_c = T(t)$

3) Edge labels move down. Constituent syntax parsing generally uses parenthetical notation to represent the constituent syntax tree structure, so in order to keep the model consistent, we also move the edge label down to the child node. Since the UCCA graphs need not be rooted trees, we add a "ROOT" dummy node to ensure that the transformed tree is a rooted tree.

3.5 Graph to Tree Conversion in AMR Framework

AMR graph is rooted, directed, and most acyclic. However, AMR is a graph instead of a tree due to it allows re-entrant semantic relations. In order to adopt the tree model for AMR parsing, we need to convert the AMR graph to a tree in the preprocessing step. Following the practice of Zhang et al. (2019a), we duplicate the nodes that have a re-entrant relation. In order to recover the original graph, we assign an index to each node, named `reentrancy index`. Duplicate nodes will be assigned the same index.

3.6 Anonymization in AMR Framework

Anonymization is an important AMR preprocessing method to reduce the data sparsity issue (Werling et al., 2015; Peng et al., 2017; Guo and Lu, 2018). Following the practice of Zhang et al. (2019a), we first remove senses, wiki links, and polarity attributes in the training dataset. Secondly, we anonymize sub-graphs of named entities which is labeled by one of AMR's fine-grained entity types that contain a `name` role, and other entities which end with `-entity`[6].

4 Models

To handle different flavors of representation, our system has three types of models: `Anchoring-based Pruning Parsing Model`, `Constituent Parsing Model`, `Seq2seq-based Parsing Model`.

4.1 Anchoring-based Pruning Parsing Model

The anchoring-based pruning parsing model is suitable for frameworks where the graph nodes are strictly one-to-one with the sentence span, such as DM, PSD, and the transformed EDS framework. The key idea of the anchoring-based pruning parsing model is to obtain the candidate graph nodes by enumerating the sentence span, and then use a scorer to pruning the candidate graph nodes and perform parsing on these graph nodes.

[6]For details of preprocessing, please refer to (Zhang et al., 2019a).

Formally, for major structural parsing goals, given a sentence $S = \{w_1, w_2, ..., w_n\}$, where n is the sequence length, we aim to predict a set of labeled graph node-pair (sentence span-pair) relations $Y \subseteq \mathcal{N} \times \mathcal{N} \times \mathcal{L}$, where $\mathcal{N} = \{(w_i, ..., w_j)|1 \leq i \leq j \leq n\}$ contains all the spans (graph nodes), and $\mathcal{L}$ is the space of the edge labels (semantic roles), including a null label ϵ indicating no edge between the node-pairs.

As our model deals with $\mathcal{O}(n^2)$ possible sentence spans (graph nodes), it needs to consider $\mathcal{O}(n^4|\mathcal{L}|)$ possible relations, which is computationally impractical. To overcome this issue, motivated by our previous work (Li et al., 2019b) and the work of (He et al., 2018a), we limit the maximum width of the candidate spans to fixed number W, which reduces the overall number of relational factors need to be considered by the model to $\mathcal{O}(n^2|\mathcal{L}|)$. In order to make the training goal denser, we also introduce a unary scorer $\phi_{node}(\cdot)$ and the candidate nodes are ranked and pruned by their unary score in descending order. The size of candidates reserved after the pruning operation is limited to λn. Candidates that are pruned do not participate in computing the edge relation prediction, which can also further reduce the computational complexity and memory requirements. These parameters W and λ are determined based on the statistics on the training dataset of each framework.

Neural Architecture Our model builds the candidate graph nodes representation based on the BERT (Devlin et al., 2019) encoder outputs, i.e., for each token w_i, the contextualized vector from BERT encoder is denoted as x_i. The candidate span (i, j) representation h consists of two endpoint contextualized vectors (x_i, x_j) where i and j are the start and end position of the span in the sentence:

$$h = [x_i; x_j]. \tag{1}$$

The node unary scorer $\phi_{node}(\cdot)$ is implemented with feed-forward networks based on the candidate graph nodes representation h:

$$\phi_{node}(\cdot) = sigmoid(\mathbf{MLP}_{node}(h)). \tag{2}$$

The edge relation classifier $\phi_{rel}(\cdot)$ is implemented with biaffine attention mechanism. Following Dozat and Manning (2017), we apply two seperate MLPs to the source and target nodes respectively, producing identity-specified representation: $r_{src} = \mathbf{MLP}_{src}(h)$ and $r_{tgt} = \mathbf{MLP}_{tgt}(h)$. We perform a biaffine operation to compute the relation labeling score.

$$\phi_{rel}(\cdot) = r_{src}^T \mathbf{W}_{rel} r_{tgt} + \mathbf{U}_{rel}^T r_{src} + \mathbf{V}_{rel}^T r_{tgt} + \mathbf{b}_{rel}, \tag{3}$$

where $\mathbf{W}_{rel}$, $\mathbf{U}_{rel}$, $\mathbf{V}_{rel}$, and $\mathbf{b}_{rel}$ are the weight matrix of the bi-linear term, the two weight vectors of the linear terms, and the bias vector, respectively.

Training Loss For the node scoring goal, we use the binary cross-entropy loss between the target and the output. For the edge classification, we implement it with the standard cross-entropy loss.

Multi-Task Learning Each framework still has some other goals to learn. The DM framework includes `top node`, `node pos tag`, and `node frame label`. The PSD includes `top node`, `node pos tag`, and `node frame label`. The EDS includes `edge source label` and `edge target label`. Overall, we use multi-tasking learning strategy, shared hidden representation, The `top node` uses the same mechanism as node scoring, using binary cross-entropy as loss implementation. The `node pos tag` and `node frame label` use independent feed-forward classifier, using cross-entropy as loss implementation. The `edge source label` and `edge target label` use a biaffine scorer consistent with the edge label, using cross-entropy loss as well. We accumulate the loss of all goals together.

4.2 Constituent Parsing Model

For the UCCA framework, we directly adopt the minimal span-based parser of Stern et al. (2017) on the converted constituent trees. A constituency tree can be regarded as a collection of labeled spans over a sentence. There are two components in the constituent parsing model: one is to assign the scores directly to span existence which determines the tree structure, and the other one assigns scores to span labels which provides the labeled outputs.

Neural Architecture In this model, we also build the candidate span representation h based on the BERT encoder outputs due to a span's correct label and its quality as a constituent depend heavily on the context in which it appears. Different from the previous span representation, in

this model, the representation h of span (i, j) is the concatenation of the two endpoint contextualized vectors differences:

$$h = x_j - x_i. \tag{4}$$

The span splitting unary scorer $\phi_{split}(\cdot)$ and the span label scorer $\phi_{label}(\cdot)$ are both implemented as feed-forward networks which take as input the span representation and output either a single span score or a vector of labeling scores. For the tree inference, we adopt the greedy top-down searching strategy introduced in Stern et al. (2017).

In order to recover the full UCCA graph, the model needs to learn the remote edge target. The remote edge target is similar to the previous edge target, which is to predict the relationship between the node pairs, so we also use the relational classifier $\phi_{rel}(\cdot)$. As the same in the previous model, there is also a null label ϵ indicating no edge between the node-pairs.

4.3 Seq2seq-based Parsing Model

The AMR framework backgrounds notions of compositionality and derivation and, accordingly, declines to make explicit how elements of the graph correspond to the surface utterance. Although most AMR parsing research presupposes a preprocessing step that aligns graph nodes with (possibly discontinuous) sets of tokens in the underlying input, these correspondences need extra annotation and training. This does not match our requirements for the model to be end-to-end. Therefore, we consider the AMR tree with indexed nodes as the prediction target (proposed by Zhang et al. (2019a)). The approach of AMR parsing is formulized as a two-stage process: node prediction (concept identification) and edge prediction (relation identification).

Formally, given a sentence $S = \{w_1, w_2, ..., w_n\}$, the model sequentially decodes a list of nodes $N = \{u_1, u_2, ..., u_m\}$ and their reentrancy indices $D = \{d_1, d_2, .., d_m\}$. Then, the model is required to search for the highest scoring parsing tree similar to dependency parsing.

Neural Architecture For node prediction, we adopt the widely-used Seq2seq model $Seq2seq(\cdot)$ with pointer-generator network (Vinyals et al., 2015). The pointer-network has the advantage of copying words from the source text while still retaining the ability to produce novel words through the generator. The node representation h is obtained as the decoder hidden state of the Seq2seq model:

$$h = Seq2seq(\cdot). \tag{5}$$

For the edge prediction, we also adopt the biaffine attention mechanism to score all possible head-dependent pairs like dependency parsing. The relation classifier $\phi_{rel}(\cdot)$ is the same as the previous:

$$\begin{aligned}\phi_{arc}(\cdot) = h_{head}^T \mathbf{W}_{arc} h_{dep} + \mathbf{U}_{arc}^T h_{head} \\ + \mathbf{V}_{arc}^T h_{dep} + \mathbf{b}_{arc},\end{aligned} \tag{6}$$

where $\mathbf{W}_{arc}$, $\mathbf{U}_{arc}$, $\mathbf{V}_{arc}$, and $\mathbf{b}_{arc}$ are the weight matrix of the bi-linear term, the two weight vectors of the linear terms, and the bias vector, respectively.

DM & PSD & EDS	
node_space_dim	128
max_seq_len	100
DM & PSD	
max_span_width (W)	5
pruning_reserve_ratio* (λ)	1.0
EDS	
max_span_width (W)	8
pruning_reserve_ratio* (λ)	1.2
UCCA	
split_space_dim (W)	128
AMR	
decoder_type	RNN
decoder_hidden_dim	512
decoder_num_layers	3
Deep biaffine classfier	
edge_space_dim	512
edge_label_space_dim	128
Optimizer	
optimizer_type	Adam
learning_rate	$5e^{-5}$
max_grad_norm	1.0

Table 1: Hyper-parameter settings for our final submission. **pruning_reserve_ratio* (λ) is for the sentence length n, not the number of candidate nodes.

5 Experiments

5.1 Setup

We first describe the final setup used for our final submission. We use the

	P	R	F_1	RANK
tops	0.92	0.91	0.915	**1**
labels	0.73	0.70	0.712	2
properties	0.70	0.67	0.687	2
anchors	0.78	0.77	0.776	1
edges	0.80	0.75	0.777	2
attributes	0.13	0.07	0.094	2
all	0.87	0.83	0.853	**2**

Table 2: The official evaluation scores of different "atomic" component pieces on all the test dataset.

	P	R	F_1	RANK
DM	0.96	0.95	0.9550	**1**
PSD	0.91	0.91	0.9119	3
EDS	0.95	0.86	0.8990	3
UCCA	0.80	0.76	0.7780	3
AMR	0.75	0.69	0.7197	3
all	0.87	0.83	0.853	**2**

Table 3: The official evaluation scores of different frameworks on all the test dataset.

`pytorch-transformers`[7] as our codebase to develop the downstream parsing models. The weights of pre-trained language model BERT with whole word masking[8] are used to initialize the encoder of our models. Due to the absence of development dataset, we split the training dataset to 10 sections and 0-8 for training and 9 for development. Our model is trained using Adam (Kingma and Ba, 2014) up to 30 epochs for DM, PSD, and EDS, and 20 epochs for UCCA and 120 epochs for AMR, with early stopping strategy based on the MRP F_1 score[9] on the development dataset with `mtool`[10] toolkit. Table 1 lists the hyperparameters used in our full model. We apply the hidden dropout ($dropout\_rate = 0.1$) to the outputs of each module in our model.

5.2 Main Results

We list our official evaluation scores[11] on the all test dataset in Tables 2 and 3. Table 2 summarizes the MRP F_1 scores of the 6 graph components. The results listed in Table 2 shows that we obtained the state-of-the-art MRP F_1 score on the `top nodes` component. In Table 3, we assess the quality on each frameworks. Our model also achieved the best results on the DM framework. We observed a notable phenomenon that as the anchoring relationship between the graph node and the surface lexical units is getting farther, the difficulty of parsing is getting higher.

From the results of parsing on different frameworks, our results on the EDS framework have the biggest gap with other priority teams, probably because of the existence of multiple edges between the same pair of `pseudo` nodes in the EDS framework after our modeling transformation. Therefore, our subsequent experiments modeled the edges of EDS as multi-classification problems, and our results on the development dataset have been improved.

6 Conclusion and Future Work

In this paper, we present our end-to-end graph-based system participated in the CoNLL 2019 shared task on Cross-Framework Meaning Representation Parsing (MRP 2019). We extend existing models and make our model be end-to-end and does not depend on any other information (including the companion data provided by the organizer). We introduce our previous graph pruning algorithm to a variety of semantic graphs, solving the problem of excessive semantic graph search space and adopt multi-task learning for multiple objectives within the same framework. Specifically, we model the semantic graph task as a multi-objective learning task of nodes, edges, node attributes, and edge attributes. The nodes candidates are scored and then pruned within the model, thus controlling the overall graph search space, and finally forming an end-to-end style parsing system. We achieve state-of-the-art results on the `top nodes` component and DM framework.

For future work, we are going to integrate all the different frameworks into one single model, not just the same modeling approach. Based on the MRP representation method, a single model is used to generate various semantic graphs. Furthermore, we would like to extend our model to other more semantic parsing tasks.

[7] https://github.com/huggingface/pytorch-transformers.

[8] In our experiments, we use the BERT-Large, uncased (Whole Word Masking) with 24-layer, 1024-hidden, 16-heads, and 340M parameters released by Google, https://github.com/google-research/bert.

[9] http://mrp.nlpl.eu/index.php.

[10] https://github.com/cfmrp/mtool.

[11] The official evaluation results are at http://bit.ly/cfmrp19.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238.

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.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English web treebank. *Linguistic Data Consortium, Philadelphia, PA*.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2753–2765.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2019. Neural machine translation with sentence-level topic context. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

Dan Flickinger, Emily M Bender, and Stephan Oepen. 2014. Towards an encyclopedia of compositional semantics: Documenting the interface of the english resource grammar. In *LREC*, pages 875–881.

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.

Jan Hajic, Eva Hajicová, Jarmila Panevová, Petr Sgall, Ondrej Bojar, Silvie Cinková, Eva Fucíková, Marie Mikulová, Petr Pajas, Jan Popelka, et al. 2012. Announcing prague czech-english dependency treebank 2.0. In *LREC*, pages 3153–3160.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018a. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Melbourne, Australia. Association for Computational Linguistics.

Shexia He, Zuchao Li, and Hai Zhao. 2019. Syntax-aware multilingual semantic role labeling. *arXiv preprint arXiv:1909.00310*.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2061–2071.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom?: A contrastive study of syntacto-semantic dependencies. In *Proceedings of the sixth linguistic annotation workshop*, pages 2–11. Association for Computational Linguistics.

Wei Jiang, Zhenghua Li, Yu Zhang, and Min Zhang. 2019. Hlt@ suda at semeval-2019 task 1: Ucca graph parsing as constituent tree parsing. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 11–15.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiangtong Li, Hai Zhao, Zuchao Li, Wei Bi, and Xiaojiang Liu. 2019a. Subword elmo. *ArXiv*, abs/1909.08357.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018a. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214.

Zuchao Li, Jiaxun Cai, and Hai Zhao. 2018b. Effective representation for easy-first dependency parsing. *arXiv preprint arXiv:1811.03511*.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018c. A unified syntax-aware framework for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2401–2411.

Zuchao Li, Shexia He, Zhuosheng Zhang, and Hai Zhao. 2018d. Joint learning of pos and dependencies for multilingual universal dependency parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 65–73.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019b. Dependency or span, end-to-end uniform semantic role labeling. *arXiv preprint arXiv:1901.05280*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

Yusuke Miyao, Stephan Oepen, and Daniel Zeman. 2014. In-house: An ensemble of pre-existing off-the-shelf parsers. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 335–340.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 99–106. Association for Computational Linguistics.

Stephan Oepen, Omri Abend, Jan Hajič, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, and Milan Straka. 2019. MRP 2019: Cross-framework Meaning Representation Parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 1 – 26, Hong Kong, China.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based mrs banking. In *LREC*, pages 1250–1255.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural amr parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 366–375.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Haipeng Sun, Rui Wang, Kehai Chen, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2019. Unsupervised bilingual word embedding agreement for unsupervised neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1235–1245.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Keenon Werling, Gabor Angeli, and Christopher D Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 982–991.

Fengshun Xiao, Jiangtong Li, Hai Zhao, Rui Wang, and Kehai Chen. 2019. Lattice-based transformer encoder for neural machine translation. *arXiv preprint arXiv:1906.01282*.

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.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2019b. Semantics-aware bert for language understanding. *arXiv preprint arXiv:1909.02209*.

Zhuosheng Zhang, Hai Zhao, Kangwei Ling, Jiangtong Li, Zuchao Li, Shexia He, and Guohong Fu. 2019c. Effective subword segmentation for text comprehension. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(11):1664–1674.

Junru Zhou, Zuchao Li, and Hai Zhao. 2019. Parsing all: Syntax and semantics, dependencies and spans. *arXiv preprint arXiv:1908.11522*.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on penn treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408.