SJTU at MRP 2019: A Transition-Based Multi-Task Parser for Cross-Framework Meaning Representation Parsing

Hongxiao Bai[1,2,3], **Hai Zhao**[1,2,3,*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[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 Institute, Shanghai Jiao Tong University
baippa@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

Abstract

This paper describes the system of our team *SJTU* for our participation in the CoNLL 2019 Shared Task: Cross-Framework Meaning Representation Parsing. The goal of the task is to advance data-driven parsing into graph-structured representations of sentence meaning. This task includes five meaning representation frameworks: DM, PSD, EDS, UCCA, and AMR. These frameworks have different properties and structures. To tackle all the frameworks in one model, it is needed to find out the commonality of them. In our work, we define a set of the transition actions to once-for-all tackle all the frameworks and train a transition-based model to parse the meaning representation. The adopted multi-task model also can allow learning for one framework to benefit the others. In the final official evaluation of the shared task, our system achieves 42% F_1 unified MRP metric score.

1 Introduction

Semantic understanding of texts is very important in Natural Language Processing (NLP), in which, Meaning Representation Parsing (MRP) attracts attentions of many researchers. This task is to encode a sentence into a semantic graph, which usually is directed. Compared with dependency parsing (Ma and Zhao, 2012; Li et al., 2018a; Zhou and Zhao, 2019) or semantic role labeling (Zhao et al., 2009a,b; Li et al., 2018b; Guan et al., 2019), this task is much harder since its representation is a graph which may incorporate both syntactical and semantic information. These general graphs are more expressive and arguably more adequate target structures for sentence-level analysis beyond shallow syntax and in particular for representations of the semantic structure. Many works have shown that these meaning representations are beneficial to other tasks such as machine translation and abstractive summarization. However, there are several types of meaning representations with different definitions, structures, and abstractions, which hinder the applications.

The CoNLL 2019 Shared Task (Oepen et al., 2019) combines formally and linguistically different meaning representation in graph form on a uniform training and evaluation setup for the first time. This task includes five MRP frameworks: DM, PSD, EDS, UCCA, and AMR. These frameworks have different anchoring types, i.e., the tightness of correspondence between graph nodes and sentence tokens with different abstractions. The nodes in DM and PSD are all the surface tokens in the sentences. In EDS and UCCA, the anchoring is flexible so that arbitrary parts of the sentence (e.g. sub-token or multi-token sequences) may be node anchors, as well as multiple nodes anchored to overlapping sub-strings. Further, AMR has even no anchoring but with the strongest expressive ability.

For each of these frameworks, the common methods for their parsing are transition-based method and graph-based method. The former parses sentences by making a sequence of transition actions according to the present state which usually consists of a stack, a buffer, and a processed edge set, while the latter gets nodes first and predicts the edges between these nodes.

In our system, we use the transition-based model to do the cross-framework meaning representation parsing, since we can define a set of transition actions and incorporate all the frameworks into our system, and the shared part of the model can learn from all the data from different frameworks. Our model is modified from

[*]Corresponding author. 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 (No. U1836222 and No. 61733011).

TUPA (Transition-based UCCA Parser) (Hershcovich et al., 2017, 2018) in terms of neural networks, which is powerful in a lot of NLP tasks (Cai and Zhao, 2016; Zhang et al., 2016; Qin et al., 2016; Vaswani et al., 2017; Cai et al., 2017; Wang et al., 2017; Qin et al., 2017; Bai and Zhao, 2018; He et al., 2018; Cai et al., 2018; Zhang and Zhao, 2018; Zhang et al., 2018a,b; Zhu et al., 2018; Huang and Zhao, 2018; Li et al., 2018c; Wu et al., 2018; Zhang et al., 2019; Xiao et al., 2019). Neural networks can encode the texts into a dense representation. We put the parsing job of all the frameworks to one model and use a multi-task setting to jointly train the system. In the final official evaluation of the shared task, our system achieves $42\% F_1$ unified MRP metric score.

The rest of this paper is organized as follows. Section 2 introduces these frameworks. Section 3 shows our model. Section 4 gives the settings of our model and test results.

2 Framework Schemes

This shared task considers five meaning representation frameworks. In this section, we briefly introduce these frameworks and figure out the traits of these frameworks.

2.1 DM and PSD

DELPH-IN MRS Bi-Lexical Dependencies (DM) (Ivanova et al., 2012) and Prague Semantic Dependencies (PSD) (Hajič et al., 2012; Miyao et al., 2014) use bi-lexical semantic dependencies to represent the meaning with different annotations. Graph nodes in DM and PSD correspond to surface tokens, and graphs are neither fully connected nor rooted trees, that is, some tokens from the underlying sentence remain structurally isolated, and for some nodes, there are multiple incoming edges.

2.2 EDS

Elementary Dependency Structures (EDS) (Oepen and Lønning, 2006) is a variable-free semantic dependency graph, where graph nodes correspond to logical predictions and edges to labeled argument positions. The variable-free feature makes these graphs quite similar to Abstract Meaning Representation (AMR). Nodes in EDS are in principle independent of surface lexical units, but for each node, there is an explicit and many-to-many anchoring onto sub-strings of the underlying sentence.

2.3 UCCA

Universal Conceptual Cognitive Annotation (UCCA) (Abend and Rappoport, 2013) targets to a more semantic way rather than only syntactically and can be extended to cross-linguistic settings. UCCA representations are directed acyclic graphs (DAGs), where terminal nodes correspond to the text tokens and non-terminal nodes to semantic units with more abstract meanings. Edges are labeled, indicating the role of a child in the relation. UCCA enable reentrancy to allow a node to participate in several semantic relations.

2.4 AMR

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) tries to abstract out all the semantic information from the sentences. The AMR graphs are rooted directed graphs, in which both nodes and edges are labeled, and reentrancy is also allowed. AMR declines to make explicit how elements of the graph correspond to the surface utterance and the nodes are abstract. So similar to EDS, it is also needed to generate nodes from semantic information, but AMR is harder since even no anchor is available. AMR graphs quite generally appear to be more abstractive compared to the other frameworks.

2.5 Framework Summary

	DM	PSD	EDS	UCCA	AMR
Node Labels	●	●	●	-	●
Node Properties	●	●	●	-	●
Node Anchoring	●	●	●	●	-
Generated Node	-	-	●	○	●
Edge Attributes	-	-	-	●	-

Table 1: Framework properties. Generated node means the nodes in the graph are not the superficial tokens and ○ means in UCCA they are empty non-terminal nodes.

These frameworks have different structures and different complexity. The graphs of these frameworks all have a top node or root node, and edges are all directed and labeled. Other properties are summarized in Table 1. By analyzing these properties, we can design a transition set to accommodate all these frameworks.

	Action	Current State	Resulting State	Description
1	Drop	$[\sigma\|s_0, b_0\|\beta, A]$	$[\sigma\|s_0, \beta, A]$	drop the word that does not convey any semantics (the first element of the buffer)
2	New(c)	$[\sigma\|s_0, b_0\|\beta, A]$	$[\sigma\|s_0, c\|b_0\|\beta, A]$	generate a new node c and push it into the buffer
3	Add	$[\sigma\|s_0, b_0\|\beta, A]$	$[\sigma\|s_0, o\|b_0\|\beta, A]$	generate a non-terminal node for UCCA and push it into the buffer
4	Left(r)	$[\sigma\|s_0, b_0\|\beta, A]$	$[\sigma\|s_0, b_0\|\beta, A \cup s_0 \xleftarrow{r} b_0]$	make left-arc with label r (edge label)
5	Right(r)	$[\sigma\|s_0, b_0\|\beta, A]$	$[\sigma\|s_0, b_0\|\beta, A \cup s_0 \xrightarrow{r} b_0]$	make right-arc with label r (edge label)
6	Swap	$[\sigma\|s_1\|s_0, \beta, A]$	$[\sigma\|s_0, s_1\|\beta, A]$	swap the top two nodes in stack and then put the top one in the buffer
7	Shift	$[\sigma\|s_0, b_0\|\beta, A]$	$[\sigma\|s_0\|b_0, \beta, A]$	shift the first node of the buffer to the stack
8	Reduce	$[\sigma\|s_0, b_0\|\beta, A]$	$[\sigma, b_0\|\beta, A]$	if the top node of the stack is processed, pop it from stack

Table 2: The transition system. σ is the stack and s_0 is the top element of the stack. β is the buffer and b_0 is the first element of the buffer. A is the set containing all processed edges. The $[\sigma|s_0, b_0|\beta, A]$ denotes one state of the transition procedure. For initialization, the σ and A are empty and β contains all tokens in the sentence (for AMR, only words that can be aligned to the graph are kept).

3 Model Description

For the joint learning task, we select a multi-task transition-based model. Following we will describe the transition set, the model, and the training/inference.

3.1 Transition Set

For a transition-based system, a transition action set is needed, and an oracle is also needed to generate gold-standard actions during training. We define the transition set to cover all meaning representation frameworks then these tasks can be learned consistently. Our transition system has a stack, a buffer, and a set of processed edges. Given a sentence consisting of a sequence of tokens $t_0, t_1, \cdots, t_n$, we put all these tokens to the buffer as initialization. During training, an oracle will generate a gold-standard action sequence, and during inference, the model will predict the action sequence and recover it to a graph. Table 2 summarizes all the actions. In these actions, actions 4, 5, 6, 7, 8 are used by all the frameworks, actions 1, 2 are used by EDS and AMR, action 3 is used by UCCA. If one action is not used by the framework, then the oracle will not generate this action for it, and during inference, the action is only selected from the legal actions for task-specified classifiers.

3.2 Model

Figure 1 depicts our model. $x_1, x_2, \cdots, x_i$ denotes the input tokens. Our model architecture

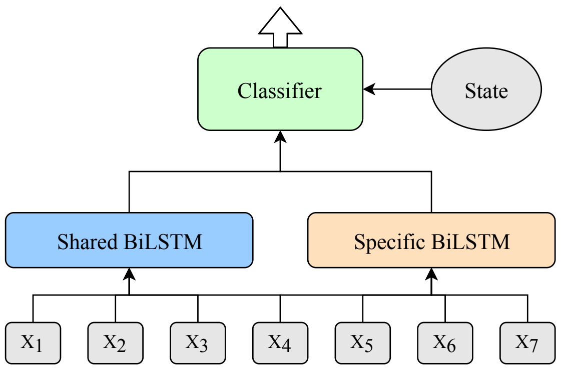

Figure 1: Model overview.

follows TUPA. The model uses a bi-directional LSTM (Hochreiter and Schmidhuber, 1997) to encode the sentence and a multi-layer perceptron (MLP) with a softmax layer for classification. Following Hershcovich et al. (2018), in the model, we have shared embedding components and a shared LSTM module, and for each framework, we have a task-specified LSTM module and a corresponding classifier. For each framework, the outputs of shared LSTM and task-specified LSTM are concatenated and fed into the task-specified classifier for action prediction. For the word embeddings, we use the pre-trained GloVe embeddings (Pennington et al., 2014) and the pre-trained BERT (Devlin et al., 2019). For each token, there are also embeddings for lemma, POS tag, and syntactic dependency label. These embeddings together with token embeddings and BERT outputs are concatenated and sent to the BiLSTMs as input. These embeddings and pre-trained models are tuned during training.

Besides the neural model, we also add handmade features to the classifier. We use features representing the existing node labels related to the top four stack elements and the first three buffer elements. We also use the last three actions taken by the parser, and if there are less than three actions before, use zero embeddings instead. For all these features, we use vector embeddings to represent them, that is, node labels and transition actions are embedded to vectors. All these embeddings are initialized randomly. These features embeddings are concatenated as a feature vector for the state.

The final hidden state vectors of shared and specific BiLSTMs and the feature vector of the state are concatenated and fed as input to the action classifiers. The training is done with an oracle that yields the set of all optimal transitions at a given state. The actual transition performed in training is the one with the highest score given by the classifier, which is trained to maximize the sum of log-likelihoods of all optimal transitions at each step.

In addition to the main model, we also apply two classifiers for property prediction of DM and PSD. The classifier is an MLP and the input is the concatenated output vectors of each token from shared and specific BiLSTM since the nodes are one-to-one corresponding to the tokens in the sentence.

3.3 Training and Inference Procedures

The training and test data have companion data processed by UDPipe (Straka, 2018). For all the input sentences, we use the tokenization, lemma, POS tagging, dependency parsing, and anchor information results from UDPipe data. Then the anchors of the output graphs are directly obtained from the UDPipe data. For EDS and AMR, the anchors are derived from the first token in the buffer.

Since AMR has no anchoring between nodes and texts, so we use the alignments generated from JAMR (Flanigan et al., 2014) and the tokens and nodes which have no alignments are discarded. Then the oracle can generate an action sequence for AMR during training. We also do pre-process on AMR and EDS graphs by expanding the node properties of graphs in the two frameworks, that is, the property key is seen as edge label and property value is seen as node label. We collect these edge labels and convert these nodes and edges to properties. For DM and PSD, the *pos* node property is from *XPOS* in UDPipe data, and the *frame* property are predicted by additional classifiers. UCCA has edge attribute *remote* to reflect the reentrancy and we neglect the edge attribute in our transition system for convenience. So we add the attribute *remote* to the later predicted edges that link the used nodes. For node labels, we use the lemma corresponding to the token in the sentences as node label for DM and PSD, and we generate node label for EDS and AMR in the *New* action.

During training, an oracle is used to generate action sequences. We use a dynamic oracle which outputs a set of optimal transitions from a given state, and from the resulting state, the gold standard graph is still reachable. For example, for EDS and AMR, if the first element of the buffer is a token and it has aligned unprocessed nodes, then a node with small id is generated by the *New* and put to the buffer. For UCCA, if the top node in the stack connects to a non-terminal node which is not generated yet, then the *Add* this non-terminal node. If this token has no aligned nodes remaining, then the *Drop* is applied. If the top element of the stack and the first element of the buffer are nodes and the node in the buffer is the child of an unprocessed edge, then the *Right* action is applied. Similarly, we have the *Left* action. If the top element of the stack has no unprocessed edges, then the *Reduce* is applied. If the stack is empty and the buffer has elements, then the *Shift* is applied. If no other actions can be found, then we do the *Swap* action.

For inference, after the action sequence is predicted, we can generate a graph from this sequence. However, this graph may not conform to the graph rules of the respective framework. So we prune the generated graph. The pruning method includes: deleting the repeated nodes and edges, deleting the nodes containing empty labels of EDS and AMR, deleting the edges attached to the deleted nodes.

4 Experiments and Results

4.1 Data settings

Our system is trained and evaluated on the data provided by the shared task. The data size is shown in Table 5. We randomly sample out 3% of the training data in each framework as the development set. After the hyperparameters are de-

	tops	labels	properties	anchors	edges	attributes	all	rank	base all
all MRP	52.7	42.8	32.1	54.7	29.5	0.2	43.0	10	**45.3**
lpps MRP	60.2	44.3	24.9	56.0	30.8	0.1	45.1	10	**50.6**
all MRP DM	47.8	56.5	34.7	70.2	31.4	0.0	**43.2**	11	42.7
lpps MRP DM	61.2	55.8	33.1	70.4	27.9	0.0	**41.9**	11	39.5
all MRP PSD	48.4	60.7	39.0	72.8	23.4	0.0	47.6	11	**52.7**
lpps MRP PSD	53.3	60.6	40.5	72.7	24.8	0.0	48.8	11	**54.5**
all MRP EDS	32.6	53.1	52.8	64.8	40.5	0.0	53.2	8	**74.0**
lpps MRP EDS	43.2	53.6	40.0	67.8	43.2	0.0	55.3	8	**74.8**
all MRP UCCA	81.8	0.0	0.0	66.0	19.4	0.8	**32.7**	9	23.7
lpps MRP UCCA	75.6	0.0	0.0	69.3	23.4	0.7	35.3	9	**41.0**
all MRP AMR	53.2	43.8	34.1	0.0	33.0	0.0	**38.5**	9	33.8
lpps MRP AMR	67.7	51.4	10.8	0.0	34.8	0.0	**44.1**	9	43.4

Table 3: Test results.$F_1(\%)$ scores for tops, labels, properties, anchors, edges, attributes, and all unified score. MRP is the results of all frameworks and others are for specific frameworks. For the test data, "all" denotes all test data and "lpps" denotes the 100 sentences in *The Little Prince*. *base* denotes the results of TUPA baseline.

	labeled F	labeled M	labeled rank	unlabeled F	unlabeled M	unlabeled rank
all DM	37.9 (56.2)	1.7 (7.2)	11	41.6 (64.3)	1.8 (8.5)	12
lpps DM	33.5 (55.7)	0.0 (14.0)	11	37.8 (64.7)	0.0 (17.0)	12
all PSD	34.0 (50.1)	2.2 (8.6)	12	45.9 (66.0)	4.1 (22.0)	12
lpps PSD	35.9 (55.3)	0.0 (15.0)	12	45.7 (68.8)	0.0 (27.0)	12

Table 4: SDP results for DM and PSD. F denotes $F_1(\%)$ score and M denotes exact match score(%). The scores in the brackets are from TUPA baseline.

termined, we train our system on all the training data. The shared task also evaluates the system on the 100 annotated sentences from *The Little Prince* which denote as "lpps" in the Results section.

	Training	Test
DM	35,656	3359
PSD	35,656	3359
EDS	35,656	3359
UCCA	6,572	1131
AMR	56,240	1998

Table 5: Number of sentences of each framework in training set and test set.

4.2 Model Settings

We implement our model with PyTorch[1] and tuned on the development set. During inference, we use greedy decoding to get the action sequence. Table 6 shows the hyperparameter settings. The optimizer is Adam (Kingma and Ba, 2015). The dropout is applied to the embeddings, the outputs of BiLSTMs, and the outputs of the first MLP layers. If the length of one sentence is larger than the max length, then the exceeding tokens are discarded. Other features denote the node labels in the stack and buffer, and the previous actions introduced in Section 3.2.

[1]https://pytorch.org/

4.3 Results

The evaluation is blindly conducted. The MRP score results are shown in Table 3. For framework specified metric, the SDP results for DM and PSD are reported in Table 4, the EDM results for EDS are reported in Table 7, and the SMATCH results for AMR are reported in Table 9. Table 3 also contains the comparison results with the TUPA baseline (Hershcovich and Arviv, 2019). For some of the frameworks, our model is better than the TUPA baseline.

4.4 Analysis

Though following the same model architecture and dynamic oracle of TUPA, we adopt a different transition set with a different feature set and setting. For example, UCCA only generates a node when an unprocessed edge is met and the node is on it, and UCCA has separate actions to predict

Hyperparameter	Value
Max sentence length	100
GloVe embedding dim	300
BERT output dim	1024
Lemma embedding dim	200
POS-tag embedding dim	20
Dependency embedding dim	20
Other feature dim	10
BiLSTM layers	2
BiLSTM dim	300
MLP layers	2
MLP dim	50
Dropout	0.2
Optimizer	Adam
Learning rate	0.001
Adam β_1	0.9
Adam β_2	0.999

Table 6: Model hyperparameters.

	all	lpps
tops	28.8	40.5
names	50.6	52.8
arguments	34.7	35.5
properties	53.5	40.0
all	43.5 (65.6)	44.9 (66.0)
rank	8	8

Table 7: EDM $F_1(\%)$ results for EDS. The scores in the brackets are from TUPA baseline.

node label, edge label, node property, and edge attribute. Whereas we have actions to generate nodes (*New* and *Add*) and the node or edge label is predicted when the node or the edge is generated. However, our set does not have actions for node properties and edge attributes, which has been introduced in Section 3.3. The motivation for designing our transition set is to use fewer actions to parse a sentence.

For the results, we find the MRP metric may be imperfect for every framework. For example, the MRP results for UCCA of ours and the baseline are comparative, whereas, for the UCCA task-specific metric, ours (Table 8) are much lower than TUPA. That is, a better MRP result may not reflect a better task-specific result. This is due to some items calculated by MRP that are not in UCCA graphs such as labels and properties, and the edge overlapping search methods are different. The gap comes from our transition set, which is not well suitable for UCCA, and this is mainly due to that we generate non-terminal node separately whereas TUPA directly generates edges attached to the non-terminal nodes, and our method may even illegally connect two terminal nodes. Other frameworks have the same issue such as DM. These task-specific metrics pay more attention to edges and have a different overlapping search method compared with MRP metric, which is more similar to the AMR specific metric SMATCH.

	all	lpps
labeled primary	4.7	5.6
labeled remote	0.6	1.6
labeled all	4.5 (22.4)	5.5 (28.4)
labeled rank	9	9
unlabeled primary	6.5	7.7
unlabeled remote	1.1	3.3
unlabeled all	6.3 (27.1)	7.5 (33.1)
unlabeled rank	9	9

Table 8: UCCA $F_1(\%)$ results. The scores in the brackets are from TUPA baseline.

	all	lpps
F_1	37.3 (32.8)	41.1 (41.1)
rank	9	9

Table 9: SMATCH $F_1(\%)$ results for AMR. SMATCH is the specific evaluation metric for AMR. The scores in the brackets are from TUPA baseline.

	Ours			TUPA		
	P	R	F	P	R	F
MRP	**46.0**	43.0	43.0	39.0	**57.0**	**45.3**
DM	**36.0**	53.0	**43.2**	31.0	**69.0**	42.7
PSD	**48.0**	48.0	47.6	45.0	**63.0**	**52.7**
EDS	**75.0**	41.0	53.2	74.0	**74.0**	**74.0**
UCCA	**31.0**	35.0	**32.7**	17.0	**38.0**	23.7
AMR	**40.0**	37.0	**38.5**	29.0	**41.0**	33.8

Table 10: Precision, Recall, and F_1 score comparisons on all MRP results.

Only for AMR, our MRP results and SMATCH results are both better, which may be due to the separate *New* action and expanding the properties as nodes.

In Table 10, we compare the precision, recall, and F_1 results for MRP metric, and we can find that though the F_1 scores are comparative, our precision scores are much higher than TUPA, whereas recall scores are much lower. That is, we can predict the elements in the graph more accurately, but our model misses too much nodes and edges. This is due to that the new node action and the separate property classifiers can bring better element prediction. Fewer actions also make the prediction more accurate. However, the design of the oracle and the training may have flaws, so some tokens are dropped and some edges are not predicted out, which makes the low recall. Our parser tends to predict a smaller graph, so for some frameworks which tend to have bigger graphs, such as PSD and EDS, the MRP results of our parser are worse.

5 Conclusion

In this paper, we describe our transition-based multi-task parsing system for the CoNLL 2019 Shared Task: Cross-Framework Meaning Representation Parsing. In our system, we integrate all the frameworks into one transition-based neural model using shared features, and we focus more on unified overall MRP metric results. The results of the blind test show that our system achieves 42% F_1 unified MRP metric score. Compared with baseline TUPA, our parser has higher precision but lower recall, for future work, we will optimize our transition set and oracle for better performance.

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 (ACL) (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria.

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 571–583, Santa Fe, New Mexico, USA.

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.

Deng Cai and Hai Zhao. 2016. Neural Word Segmentation Learning for Chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 409–420, Berlin, Germany.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for Chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 2: Short Papers)*, pages 608–615, Vancouver, Canada.

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 (COLING)*, pages 2753–2765, Santa Fe, New Mexico, USA.

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

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 (ACL) (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland.

Chaoyu Guan, Yuhao Cheng, and Hai Zhao. 2019. Semantic role labeling with associated memory network. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL:HLT), Volume 1 (Long and Short Papers)*, pages 3361–3371, Minneapolis, Minnesota.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English dependency treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3153–3160, Istanbul, Turkey.

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

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 1127–1138, Vancouver, Canada.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 373–385, Melbourne, Australia.

Daniel Hershcovich and Ofir Arviv. 2019. TUPA at MRP 2019. A multi-task baseline system. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 27–38, Hong Kong, China.

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

Yafang Huang and Hai Zhao. 2018. Chinese pinyin aided IME, input what you have not keystroked yet. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2923–2929, Brussels, Belgium.

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, Jeju, Republic of Korea.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA.

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

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

Zuchao Li, Shexia He, Zhuosheng Zhang, and Hai Zhao. 2018c. 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, Brussels, Belgium.

Xuezhe Ma and Hai Zhao. 2012. Fourth-order dependency parsing. In *Proceedings of COLING 2012: Posters*, pages 785–796, Mumbai, India.

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, Dublin, Ireland.

Stephan Oepen, Omri Abend, Jan Hajič, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, and Nianwen Xue. 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 *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy.

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.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016. A stacking gated neural architecture for implicit discourse relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2263–2270, Austin, Texas.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 1006–1017, Vancouver, Canada.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS) 30*, pages 5998–6008.

Hao Wang, Hai Zhao, and Zhisong Zhang. 2017. A transition-based system for universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 191–197, Vancouver, Canada.

Yingting Wu, Hai Zhao, and Jia-Jun Tong. 2018. Multilingual universal dependency parsing from raw text with low-resource language enhancement. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 74–80, Brussels, Belgium.

Fengshun Xiao, Jiangtong Li, Hai Zhao, Rui Wang, and Kehai Chen. 2019. Lattice-based transformer encoder for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association* *for Computational Linguistics (ACL)*, pages 3090–3097, Florence, Italy.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1382–1392, Berlin, Germany.

Zhuosheng Zhang, Yafang Huang, and Hai Zhao. 2018a. Subword-augmented embedding for cloze reading comprehension. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 1802–1814, Santa Fe, New Mexico, USA.

Zhuosheng Zhang, Yafang Huang, and Hai Zhao. 2019. Open vocabulary learning for neural Chinese pinyin IME. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1584–1594, Florence, Italy.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao, and Gongshen Liu. 2018b. Modeling multi-turn conversation with deep utterance aggregation. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 3740–3752, Santa Fe, New Mexico, USA.

Zhuosheng Zhang and Hai Zhao. 2018. One-shot learning for question-answering in Gaokao history challenge. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 449–461, Santa Fe, New Mexico, USA.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009a. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 61–66, Boulder, Colorado.

Hai Zhao, Wenliang Chen, and Chunyu Kit. 2009b. Semantic dependency parsing of NomBank and PropBank: An efficient integrated approach via a large-scale feature selection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 30–39, Singapore.

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 (ACL)*, pages 2396–2408, Florence, Italy.

Pengfei Zhu, Zhuosheng Zhang, Jiangtong Li, Yafang Huang, and Hai Zhao. 2018. Lingke: a fine-grained multi-turn chatbot for customer service. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING): System Demonstrations*, pages 108–112, Santa Fe, New Mexico.