Towards Better Non-Tree Argument Mining: Proposition-Level Biaffine Parsing with Task-Specific Parameterization

Gaku Morio, Hiroaki Ozaki, Terufumi Morishita, Yuta Koreeda and **Kohsuke Yanai**
Hitachi, Ltd.
Research and Development Group
Kokubunji, Tokyo, Japan
{gaku.morio.vn, hiroaki.ozaki.yu, terufumi.morishita.wp,
yuta.koreeda.pb, kohsuke.yanai.cs}@hitachi.com

Abstract

State-of-the-art argument mining studies have advanced the techniques for predicting argument structures. However, the technology for capturing non-tree-structured arguments is still in its infancy. In this paper, we focus on non-tree argument mining with a neural model. We jointly predict proposition types and edges between propositions. Our proposed model incorporates (i) **task-specific parameterization (TSP)** that effectively encodes a sequence of propositions and (ii) a **proposition-level biaffine attention (PLBA)** that can predict a non-tree argument consisting of edges. Experimental results show that both TSP and PLBA boost edge prediction performance compared to baselines.

1 Introduction

Argument mining, a research area that focuses on predicting argumentation structures in a text, has been receiving much attention. To date, efforts in argument mining were devoted to predicting tree arguments in which a claim proposition is represented as a root and premise propositions are represented as leaves. For example, Stab and Gurevych (2017) introduced Argument Annotated Essays (hereafter, Essay), and researchers attempted to predict tree arguments in the corpus (Eger et al., 2017; Potash et al., 2017; Kuribayashi et al., 2019).

However, these techniques lack the capability of dealing with more flexible arguments such as reason edges where a proposition can have several parents. To this end, Park and Cardie (2018) provided a less restrictive argument mining dataset known as Cornell eRulemaking Corpus (CDCP), which contains flexible edges (see VALUES (a), (b), and TESTIMONY (e) in Figure 1). Figure 2 shows a distribution of outgoing edges for Essay and CDCP. Propositions in CDCP have sparse connections, making the majority of propositions isolated from the others. Besides, a proposition in Essay has at most one outgoing edge, while that in CDCP has a variable number of edges (i.e., there are about 200 propositions which have two or more outgoing edges). Therefore, it is important to work on the less restrictive arguments. Yet, it has not been deeply studied except a few studies (Niculae et al., 2017; Galassi et al., 2018).

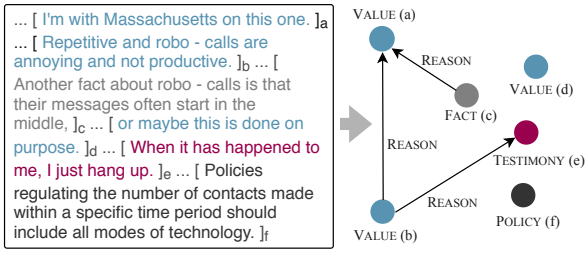


Figure 1: Example graph in the CDCP corpus

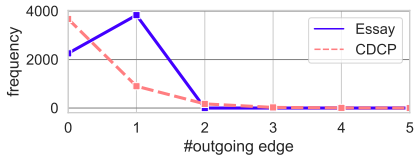


Figure 2: Distribution of the outgoing edges (i.e., Support/Attack or REASON/EVIDENCE relations) from a node (proposition) in Essay and CDCP corpora

In this paper, we present a novel model for non-tree argument mining. Different from the previous studies of Niculae et al. (2017); Galassi et al. (2018), we focus on an effective encoding for the propositions and a graph-based non-tree argument parsing technique. Given sentence or clause spans in an argument, our model jointly predicts proposition types for the spans, edges between the propositions and edge labels by employing following two architectures:

– **Task-Specific Parameterization (TSP)** is an effective encoding step for the proposition sequence. On top of a shared encoder, we prepare two dis-

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3259–3266

tinct attention-to-encoder layers to maintain task-specific representations. One is for the proposition type, and the other for the edges (and their labels). TSP employs our expectation that edge- and proposition type-specific representations should be separately obtained. This is because representations of proposition types and edges are relatively less bonded when compared to the tree-structured Essay where each premise proposition always has one outgoing edge.

– **Proposition-Level Biaffine Attention (PLBA)** is used to predict non-tree edges after the encoding step. Biaffine attention has recently been used for syntactic or semantic token-to-token dependency parsing (Dozat and Manning, 2017, 2018; Wang et al., 2019; Zhang et al., 2019; Li et al., 2019b,a). We extend the biaffine attention to predict proposition-to-proposition dependencies.

Experimental results on CDCP show that our proposed model improves performance. Analyses also show that task-specific information can be captured by TSP.

2 Dataset

We use CDCP (Park and Cardie, 2018; Niculae et al., 2017) with 731 arguments. The corpus provides five types of propositions (32 REFERENCE, 746 FACT, 1026 TESTIMONY, 2160 VALUE and 815 POLICY), and two types of argumentative edges (1307 REASON and 46 EVIDENCE). For example, FACT poses a truth value that can be verified with objective evidence: *That process usually takes as much as two years or more.* CDCP also provides directed edges between propositions and edge label. A proposition i is REASON for a proposition j if i provides rationale for j, or is EVIDENCE if it proves whether j is true or not.

3 Task Formalization

Input: We assume a text consisting of N tokens and M proposition spans is given. We denote the i-th proposition span as $(\mathrm{START}(i), \mathrm{END}(i))$ where $\mathrm{START}(i)$ and $\mathrm{END}(i)$ are the starting and ending token indices, respectively. Thus, $1 \leq \mathrm{START}(i) \leq \mathrm{END}(i) \leq N$.

Output: For each given span i, we predict its proposition *type*, outgoing *edges*, and edge *labels* (i.e., REASON and EVIDENCE), where the graph does not necessarily form a tree.

4 Approach

An overview of our proposed model is shown in Figure 3 (right). We encode propositions by TSP, and use PLBA to obtain non-tree arguments.

We use $\mathbf{w}_t$ to denote the concatenation of t-th set of word features, each set consisting of a surface, a part-of-speech tag, a GloVe vector (Pennington et al., 2014) and an optional ELMo vector (Peters et al., 2018). The input words for span i are fed into a bidirectional LSTM:

$$\mathbf{h}_{\mathrm{START}(i):\mathrm{END}(i)} = \textsc{BiLSTM}\left(\mathbf{w}_{\mathrm{START}(i):\mathrm{END}(i)}\right).$$

4.1 TSP: Task-Specific Parameterization

We provide task-specific encoding layers, one for proposition types and the other for edges (and their labels), on the top of the BILSTM. We expect the lower layers to extract task-universal representations and the upper layers to extract more task-specific representations (Liu et al., 2019; Ethayarajh, 2019). First, to be aware of informative tokens such as discourse markers, we obtain task-aware span representations for each task $\tau \in \{\mathrm{type}, \mathrm{edge}\}$:

$$a_{\tau,t} = \mathbf{v}_\tau^\top (W_\tau \mathbf{h}_t + \mathbf{b}_\tau),$$

$$s_{\tau,i,t} = \frac{\exp(a_{\tau,t})}{\sum_{k=\mathrm{START}(i)}^{\mathrm{END}(i)} \exp(a_{\tau,k})},$$

$$\mathbf{h}_{\tau,i}^{\mathrm{span\_att}} = \sum_{t=\mathrm{START}(i)}^{\mathrm{END}(i)} s_{\tau,i,t} \mathbf{h}_t,$$

where $\mathbf{v}_\tau$, W_τ and $\mathbf{b}_\tau$ are parameters. We note that $\mathbf{h}_{\tau,i}^{\mathrm{span\_att}} \in \{\mathbf{h}_{\mathrm{type},i}^{\mathrm{span\_att}}, \mathbf{h}_{\mathrm{edge},i}^{\mathrm{span\_att}}\}$. Then, each type- and edge-specific proposition span is represented as:

$$\mathbf{h}_{\mathrm{type},i}^{\mathrm{span}} = \mathbf{h}_{\mathrm{END}(i)} \oplus \mathbf{h}_{\mathrm{type},i}^{\mathrm{span\_att}} \oplus \phi(i),$$

$$\mathbf{h}_{\mathrm{edge},i}^{\mathrm{span}} = \mathbf{h}_{\mathrm{END}(i)} \oplus \mathbf{h}_{\mathrm{edge},i}^{\mathrm{span\_att}} \oplus \phi(i),$$

where $\oplus$ is a concatenation operation and $\phi(i)$ is a span length feature. The span representations are then fed into new BiLSTMs to encode task-specific proposition sequences:

$$\mathbf{s}_{\mathrm{type},i} = \textsc{BiLSTM}_{\mathrm{type}}(\mathbf{h}_{\mathrm{type},i}^{\mathrm{span}}),$$

$$\mathbf{s}_{\mathrm{edge},i} = \textsc{BiLSTM}_{\mathrm{edge}}(\mathbf{h}_{\mathrm{edge},i}^{\mathrm{span}}).$$

4.2 PLBA: Proposition-Level Biaffine Attention

To predict non-tree edges between propositions, we use biaffine attention (Dozat and Manning, 2018)

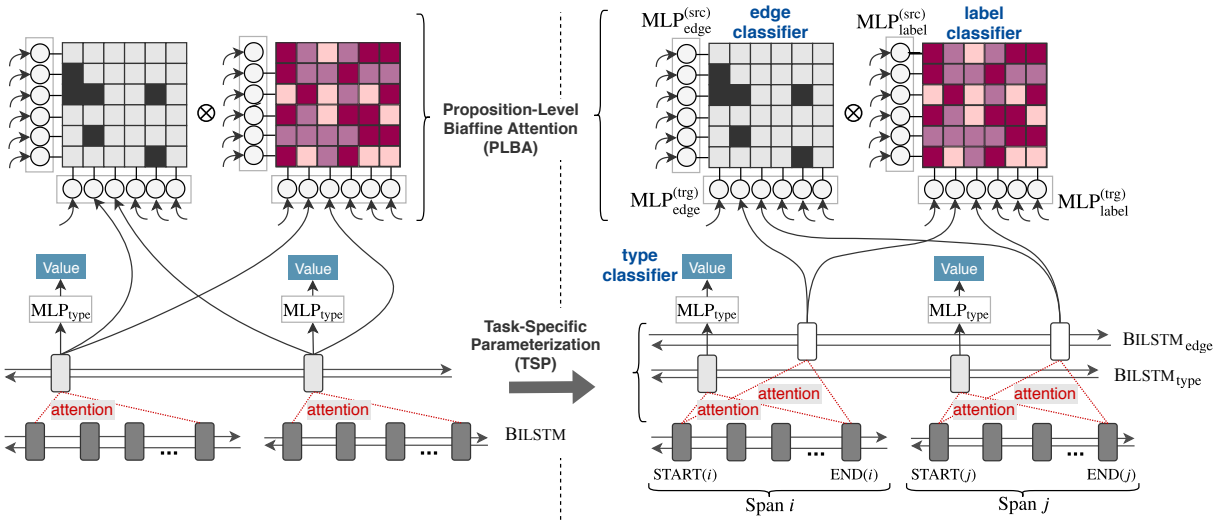


Figure 3: Simplified overview of (**left**) *non*-TSP model using a naive single attention-to-encoder system and (**right**) our proposed model. Note that, for each figure, only two propositions in six propositions are shown for the visibility.

that computes scores of all proposition pairs by the following operation:

$$\text{BIAFFINE}_k(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \mathbf{U}_k \mathbf{y},$$

where $\mathbf{U}_k$ is a parameter. We apply multi-layer perceptrons (MLPs) and a biaffine operation to a pair of edge-specific representations $(\mathbf{s}_{\text{edge},i}, \mathbf{s}_{\text{edge},j})$ to obtain a probability of a directed edge from i-th span to j-th span:

$$\mathbf{e}_i^{(\text{src})} = \text{MLP}_{\text{edge}}^{(\text{src})}\left(\mathbf{s}_{\text{edge},i}\right),$$
$$\mathbf{e}_j^{(\text{trg})} = \text{MLP}_{\text{edge}}^{(\text{trg})}\left(\mathbf{s}_{\text{edge},j}\right),$$
$$\hat{\text{edge}}_{i,j} = \text{sigmoid}\left(\text{BIAFFINE}_{\text{edge}}\left(\mathbf{e}_i^{(\text{src})}, \mathbf{e}_j^{(\text{trg})}\right)\right),$$

and the label for the edge (i, j) is calculated as

$$\boldsymbol{\ell}_i^{(\text{src})} = \text{MLP}_{\text{label}}^{(\text{src})}(\mathbf{s}_{\text{edge},i}),$$
$$\boldsymbol{\ell}_j^{(\text{trg})} = \text{MLP}_{\text{label}}^{(\text{trg})}(\mathbf{s}_{\text{edge},j}),$$
$$\hat{\text{label}}_{i,j} = \text{softmax}\left(\text{BIAFFINE}_{\text{label}}\left(\boldsymbol{\ell}_i^{(\text{src})}, \boldsymbol{\ell}_j^{(\text{trg})}\right)\right).$$

We train edges and labels by summing the losses, backpropagating gradients for the labels only through gold edges. At inference, the predicted labels are masked by the edges: $\hat{\text{edge}}_{i,j} \otimes \hat{\text{label}}_{i,j}$.

4.3 Joint Learning with Proposition Type

We classify the proposition type for span i with the type-specific representation: $\hat{\text{type}}_i = \text{softmax}\left(\text{MLP}_{\text{type}}\left(\mathbf{s}_{\text{type},i}\right)\right)$. Finally, we minimize the joint objective of edge loss $\mathcal{L}_i^{\text{edge}}$, label loss $\mathcal{L}_i^{\text{label}}$ and type loss $\mathcal{L}_i^{\text{type}}$:

$$\mathcal{L} = \sum_{i=1}^{M}\left(\lambda^{\text{edge}}\mathcal{L}_i^{\text{edge}} + \lambda^{\text{label}}\mathcal{L}_i^{\text{label}} + \lambda^{\text{type}}\mathcal{L}_i^{\text{type}}\right),$$

where λ are hyperparameters to adjust training.

5 Experiments

Following Niculae et al. (2017), we evaluate the test set of CDCP that contains 973 propositions and 272 edges. F1 scores for the proposition type prediction and the edge prediction along with their average are used for the evaluations. For the edge labels, we only consider the classification of EVIDENCE rather than macro-averaged scores because labels are highly imbalanced. We calculate label scores on gold edges.

5.1 Baselines

To the best of our knowledge, two existing studies are comparable in our task settings. The first set of baselines are factor-based models (SVM basic/full/strict ; RNN basic/full/strict; Niculae et al., 2017). Another set of baselines are neural residual models (deep basic PG/LG ; deep residual PG/LG; Galassi et al., 2018), which are the state-of-the-art models in terms of edge classification.

We also provided a non-TSP model for comparison where we use a joint aggregation to make $\mathbf{s}_{\text{type},i} = \mathbf{s}_{\text{edge},i}$. To this end, we provide a shared

model	edge	type avg.	avg.	label EVIDENCE
deep basic: LG	22.56	43.79	33.18	-
RNN: full	14.6	52.4	33.5	-
RNN: strict	10.5	65.9	38.2	-
deep basic: PG	22.45	63.31	42.88	-
RNN: basic	14.4	72.7	43.5	-
deep residual: PG	20.76	71.99	46.37	-
deep residual: LG	29.29	65.28	47.28	-
SVM: basic	24.7	71.6	48.1	-
SVM: full	25.1	73.5	49.3	-
SVM: strict	26.7	73.2	50.0	-
ours	**34.04**	78.91	56.48	18.73
+ checkpoint ensemble	33.84	**79.48**	**56.66**	**21.28**

Table 1: F1 comparison against the existing models on CDCP

representation for both type and edge:

$$\mathbf{h}^{\text{span}}_{\text{type\&edge},i} = \mathbf{h}^{\text{span}}_{\text{type},i} = \mathbf{h}^{\text{span}}_{\text{edge},i}, \\ = \mathbf{h}_{\text{END}(i)} \oplus \mathbf{h}^{\text{span\_att}}_{\text{type\&edge},i} \oplus \phi(i).$$

and we use a joint encoder:

$$\mathbf{s}_{\text{type\&edge},i} = \mathbf{s}_{\text{type},i} = \mathbf{s}_{\text{edge},i} \\ = \text{BILSTM}_{\text{type\&edge}}(\mathbf{h}^{\text{span}}_{\text{type\&edge},i}).$$

According to the change above, the non-TSP model also requires us to modify the pre-biaffine MLPs and the proposition type classifier (see Appendix for more details).

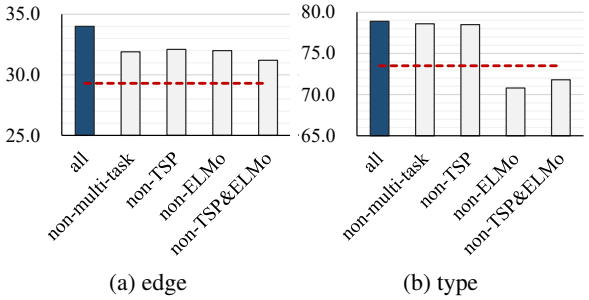
5.2 Implementation

GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018) were used as input embeddings. The hyperparameters were tuned with Optuna (Akiba et al., 2019) without using ELMo and TSP for fair comparison (see Appendix for more details). Each model was trained for 100 epochs with Adam (Kingma and Ba, 2015), and we selected a model that exhibited the highest average development F1 scores amongst all the classifiers.

6 Results

We ran the experiment 30 times with different random seeds. Table 1 shows their average scores, showing our models outperform all the baselines. F1 performance for each proposition type are: FACT=51.58, POLICY=83.32, REFERENCE=100.0, TESTIMONY=78.99, and VALUE=80.67. We also report the results of our model with checkpoint ensemble (Chen et al., 2017)[1], showing a stable performance for both the proposition type and EVIDENCE label classification.

[1]Different from the study, we simply employed the best three checkpoints.



(a) edge (b) type

Figure 4: Task-specific ablation study (F1 scores). The dashed red line indicates a state-of-the-art baseline.

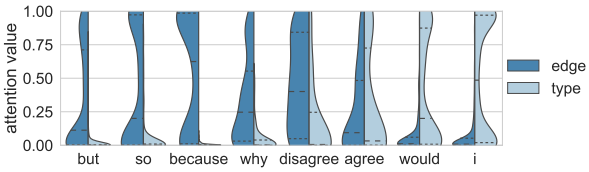


Figure 5: Attention weight analysis with a violin plot by a kernel density estimation

6.1 Ablation Study

Figure 4 shows ablation studies. The non-ELMo model already outperforms the state-of-the-art baseline in the edge prediction task, showing that PLBA is effective. Besides, ELMo boosted the type classification.

Figure 4a shows that the edge scores for the non-multi-task model are significantly lower, while Figure 4b shows that its type scores are barely affected. The result implies the edge task utilizes type information in the lower layer, but the type task is less dependent on edges. Besides, the edge scores for the non-TSP model are worse, indicating that TSP is effective in obtaining a stable performance. The result implies that TSP acquires edge-specific representations independently from types.

6.2 What Does TSP Learn?

To further analyze TSP, we investigated the task-specific token attention $s_{\tau,i,t}$. Figure 5 shows the attention distributions by a kernel density estimation for a number of selected tokens. The figure shows that not only discourse markers (i.e., *because*, *but* and *so*) but rhetorical or subjective claims (i.e., *why* and *disagree*) were focused in edge predictions. We found in the corpus that propositions with *disagree* and *why* are likely to be a top (claim) node. This suggests that these subjective statements can be

used for predicting the top nodes.

For proposition types, a number of first-person pronouns such as *I* were useful. We attribute this result to the TESTIMONY propositions which express personal experiences, e.g., *but I never received any notice from my original mortgage lender that my mortgage was sold.*

7 Related Work

Researchers in argument mining have been utilizing Essay (Stab and Gurevych, 2014), a tree argument corpus. For example, Persing and Ng (2016) employed integer linear programming. Eger et al. (2017) investigated argument mining as a dependency parsing problem with neural models. Potash et al. (2017) developed a pointer network architecture to predict edges. However, we cannot simply utilize them for non-tree arguments because these models were built upon the assumption that an argument forms a tree structure.

Non-tree arguments are relatively less emphasized. Niculae et al. (2017) attempted to resolve the problem with a factor-based model. Our study is primarily inspired by the semantic dependency parsing of Dozat and Manning (2018) and we predict the whole graph jointly. Galassi et al. (2018) proposed a deep learning-based model that utilizes residual connections to predict proposition pair relations.

8 Conclusion

This paper focused on non-tree argument mining. We provided an approach to effectively encode a proposition sequence and to predict non-tree edges. Experimental results showed that our proposed model outperforms baselines. This paper demonstrated that we could successfully analyze more flexible structures in arguments. For future work, we aim to develop a universal model to handle both tree and non-tree arguments.

Acknowledgments

We appreciate Prof. Dr. Naoaki Okazaki at Tokyo Institute of Technology for his helpful comments.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 2623–2631, New York, NY, USA. ACM.

Hugh Chen, Scott Lundberg, and Su-In Lee. 2017. Checkpoint ensembles: Ensemble methods from a single training process. *CoRR*, abs/1710.03282.

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*.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22, Vancouver, Canada. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. 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 55–65, Hong Kong, China. Association for Computational Linguistics.

Andrea Galassi, Marco Lippi, and Paolo Torroni. 2018. Argumentative link prediction using residual networks and multi-objective learning. In *Proceedings of the 5th Workshop on Argument Mining*, pages 1–10, Brussels, Belgium. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*.

Tatsuki Kuribayashi, Hiroki Ouchi, Naoya Inoue, Paul Reisert, Toshinori Miyoshi, Jun Suzuki, and Kentaro Inui. 2019. An empirical study of span representations in argumentation structure parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4691–4698, Florence, Italy. Association for Computational Linguistics.

Ying Li, Zhenghua Li, Min Zhang, Rui Wang, Sheng Li, and Luo Si. 2019a. Self-attentive biaffine dependency parsing. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial*

Intelligence, IJCAI-19, pages 5067–5073. International Joint Conferences on Artificial Intelligence Organization.

Zuchao Li, Hai Zhao, Zhuosheng Zhang, Rui Wang, Masao Utiyama, and Eiichiro Sumita. 2019b. SJTU-NICT at MRP 2019: Multi-task learning for end-to-end uniform semantic graph parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 45–54, Hong Kong. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. Argument mining with structured SVMs and RNNs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 985–995, Vancouver, Canada. Association for Computational Linguistics.

Joonsuk Park and Claire Cardie. 2018. A corpus of eRulemaking user comments for measuring evaluability of arguments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

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.

Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1384–1394, San Diego, California. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. Here's my point: Joint pointer architecture for argument mining. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1364–1373, Copenhagen, Denmark. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 1501–1510.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019. Second-order semantic dependency parsing with end-to-end neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4618, Florence, Italy. Association for Computational Linguistics.

Yue Zhang, Wei Jiang, Qingrong Xia, Junjie Cao, Rui Wang, Zhenghua Li, and Min Zhang. 2019. SUDA-Alibaba at MRP 2019: Graph-based models with BERT. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 149–157, Hong Kong. Association for Computational Linguistics.

A Appendices

A.1 Input Representation

Following the work of Kuribayashi et al. (2019) and Potash et al. (2017), we propose incorporating multiple types of token representation to provide rich input features. Specifically, the proposed system combines surface, part-of-speech (POS) tags, GloVe (Pennington et al., 2014) embedding, and ELMo (Peters et al., 2018) as input features for each token. The following descriptions explain how we acquire each input representation:

Surface Tokens are parsed by SpaCy (`https://spacy.io/`). Surfaces that appear less than four times are replaced by special `UNK` tokens.

POS tags We employ POS tags obtained by SpaCy.

GloVe We employ 300-dimensional GloVe vectors (obtained from `http://nlp.stanford.edu/data/glove.840B.300d.zip`).

ELMo We employ the pretrained ELMo (obtained from `https://s3-us-west-2.amazonaws.com/allennlp/models/elmo/2x4096_512_2048cnn_2xhighway/elmo_2x4096_512_2048cnn_2xhighway_weights.hdf5` and `elmo_2x4096_512_2048cnn_2xhighway_`

hyperparameter	value or search space
GloVe dimention	300
GloVe embedding linear	100
POS embedding linear	100
ELMo type	2x4096, 512_2048cnn_2xhighway
input dropout	0.25, 0.33, 0.45
BILSTM dimension	200, 300, 400
BILSTM stack	1
BILSTM$_\tau$ dimension	200, 300, 400
BILSTM$_\tau$ stack	2, 3
recurrent dropout of all BiLSTMs	0.25, 0.33, 0.45
output dropout of all BiLSTMs	0.25, 0.33, 0.45
dimention of all MLPs	600, 700
dropout of all MLPs	0.25, 0.33, 0.45
activation of all MLPs	ReLU
$(\lambda^{\text{edge}}, \lambda^{\text{label}}, \lambda^{\text{type}})$	(0.6, 0.2, 0.2), (0.4, 0.3, 0.3), (0.333, 0.333, 0.333)
learning rate	0.0012, 0.0011, 0.001, 0.0009, 0.0008
Adam β_1	0.9
Adam β_2	0.999
epoch	100
mini-batch size	16

Table 2: List of hyperparameters. Multiple values indicates that the hyperparameter was tuned within those values. Underlines show the selected hyperparameter by the Optuna framework.

`options.json`). Following Peters et al. (2018), we *mix* different layers of ELMo for each token:

$$\tilde{s_k} = \frac{\exp(s_k)}{\sum_{k'} \exp(s_{k'})},$$
$$\mathbf{w}^{\text{ELMo}}_{\text{START}(i):\text{END}(i)} = \sum_k \tilde{s_k} \text{ELMo}^k_{\text{START}(i):\text{END}(i)},$$

where $\text{ELMo}^k_{\text{START}(i):\text{END}(i)} (0 < k \leq N^{\text{ELMo}})$ is the hidden state of the k-th layer of the ELMo obtained by START(i) to END(i) tokens, $\text{ELMo}^0_{\text{START}(i):\text{END}(i)}$ are the features from character-level CNN in ELMo, and s_k are trainable parameters. The ELMo paramters are fixed by truncating backpropagation.

The surface and POS tag of a token are each embedded into a vector. A multi-layered perceptron (MLP) is applied to each surface and POS. All features are then concatenated to form input token representation:

$$\mathbf{w}_t = \mathbf{w}^{\text{surface}}_t \oplus \mathbf{w}^{\text{POS}}_t \oplus \mathbf{w}^{\text{GloVe}}_t,$$

Optionally, we can concatenate ELMo:

$$\mathbf{w}_t = \mathbf{w}^{\text{surface}}_t \oplus \mathbf{w}^{\text{POS}}_t \oplus \mathbf{w}^{\text{GloVe}}_t \oplus \mathbf{w}^{\text{ELMo}}_t.$$

A.2 Non-TSP Model

For non-TSP model in experiments, we provide a shared representation for both type and edge:

$$\begin{aligned}\mathbf{h}^{\text{span}}_{\text{type\&edge},i} &= \mathbf{h}^{\text{span}}_{\text{type},i} = \mathbf{h}^{\text{span}}_{\text{edge},i}, \\ &= \mathbf{h}_{\text{END}(i)} \oplus \mathbf{h}^{\text{span\_att}}_{\text{type\&edge},i} \oplus \phi(i).\end{aligned}$$

and we use a joint encoder:

$$\mathbf{s}_{\text{type\&edge},i} = \text{BILSTM}_{\text{type\&edge}}(\mathbf{h}^{\text{span}}_{\text{type\&edge},i}).$$

According to the change above, the non-TSP also requires us to modify the pre-biaffine operations:

$$\begin{aligned}\mathbf{e}^{(\text{src})}_i &= \text{MLP}^{(\text{src})}_{\text{edge}}\left(\mathbf{s}_{\text{type\&edge},i}\right), \\ \mathbf{e}^{(\text{trg})}_j &= \text{MLP}^{(\text{trg})}_{\text{edge}}\left(\mathbf{s}_{\text{type\&edge},j}\right), \\ \boldsymbol{\ell}^{(\text{src})}_i &= \text{MLP}^{(\text{src})}_{\text{label}}(\mathbf{s}_{\text{type\&edge},i}), \\ \boldsymbol{\ell}^{(\text{trg})}_j &= \text{MLP}^{(\text{trg})}_{\text{label}}(\mathbf{s}_{\text{type\&edge},j}),\end{aligned}$$

and the proposition type classifier:

$$\hat{\text{type}}_i = \text{softmax}\left(\text{MLP}_{\text{type}}\left(\mathbf{s}_{\text{type\&edge},i}\right)\right).$$

A.3 Hyperparameter Tuning

We tuned the hyperparameters using a subset considering our preliminary experiments. See Table 2 for hyperparameter search space and list of hyperparameters chosen by the Optuna framework (Akiba et al., 2019). We tried 20 hyperparameter sets. As can be seen from the table, the high dropout rate is effective. We estimate this is because the system can prevent an overfitting. We also found stacking BiLSTMs in TSP higher can improve performance, implying the semantics can be captured in upper layers.

A.4 Single-task Setup

For the single-task setup (*non-multi-task*), we provide each task-specific learning: type, edge, and edge label. Each model was optimized using its objective using the same hyperparameters.