# An Empirical Study of Span Representations in Argumentation Structure Parsing

**Tatsuki Kuribayashi**[1], **Hiroki Ouchi**[2,1], **Naoya Inoue**[1,2],
**Paul Reisert**[2,1], **Toshinori Miyoshi**[3], **Jun Suzuki**[1,2], and **Kentaro Inui**[1,2]
[1]Tohoku University [2]RIKEN Center for Advanced Intelligence Project (AIP)
[3]Research & Development Group, Hitachi, Ltd.
{kuribayashi, naoya-i, jun.suzuki, inui} @ecei.tohoku.ac.jp
{hiroki.ouchi, paul.reisert} @riken.jp
toshinori.miyoshi.pd@hitachi.com

## Abstract

For several natural language processing (NLP) tasks, span representation is attracting considerable attention as a promising new technique; a common basis for an effective design has been established. With such basis, exploring task-dependent extensions for argumentation structure parsing (ASP) becomes an interesting research direction. This study investigates (i) span representation originally developed for other NLP tasks and (ii) a simple task-dependent extension for ASP. Our extensive experiments and analysis show that these representations yield high performance for ASP and provide some challenging types of instances to be parsed.

## 1 Introduction

Argumentation structure parsing (ASP) is the task of identifying argumentation structures in argumentative text. Figure 1 shows an example of an argumentative text and its structure. The structure forms a tree, the nodes of which are referred to as argumentative discourse units (ADUs) and the edges represent argumentative relations between the ADUs. ASP systems must identify such edges, edge labels (e.g., SUPPORT and ATTACK), and node labels (e.g., PREMISE, CLAIM, and MAJOR-CLAIM). A key to achieving high performance is feature representation design for segmental discourse units (spans), such as ADUs. The aim of this study is to update the foundation of span representation design for ASP.

Potash et al. (2017) introduced a model exploiting neural network-based span representation for ASP and reported state-of-the-art performance. Similarly, for other natural language processing (NLP) tasks, such as syntactic and semantic parsing, neural network-based span representation design is attracting considerable attention as a promising new technique; some effective designs have been reported (Wang and Chang, 2016; Stern et al., 2017; He et al., 2018). Starting from such basis, task-dependent extensions for ASP are an interesting direction to explore.

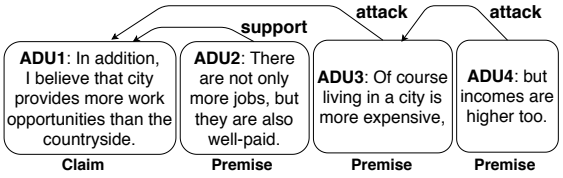


Figure 1: Example of an argumentative text and its structure. ADUs denote argumentative discourse units, which are fundamental units of arguments.

Although the neural network-based approach (Potash et al., 2017; Eger et al., 2017) achieves high performances for ASP, it does not explicitly take into account useful linguistic clues. However, prior works demonstrate that linguistic features, particularly discourse connectives, are strong clues to predict the structure for ASP (Lawrence and Reed, 2015; Stab and Gurevych, 2017). We integrate such linguistic properties into span representation design as task-dependent extensions for ASP.

In summary, our contributions are as follows.

- We investigate (i) span representation originally developed for other NLP tasks and (ii) a simple task-dependent extension for ASP.
- Empirical results show that such representations improve the performance of ASP and yield state-of-the-art scores.
- Extensive analysis reveals that such representations especially improve the performance when parsing argumentative texts with a complex structure (deeper ADU trees).

To facilitate ASP research, our model code and scripts made publicly available[1].

[1] https://github.com/kuribayashi4/span_based_argumentation_parser



*Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4691–4698
Florence, Italy, July 28 - August 2, 2019. 

## 2 Related work

**Argumentation structure parsing** In recent years, the persuasive essay corpus (PEC), a large-scale annotated dataset for argument structures, has been published (Stab and Gurevych, 2017); moreover, a variety of models have been proposed for ASP. There are two main approaches for ASP: (i) a discrete feature-based approach (Stab and Gurevych, 2017; Nguyen and Litman, 2016; Peldszus and Stede, 2015) and (ii) a neural network-based approach (Potash et al., 2017; Eger et al., 2017). Because neural network-based models have achieved high performance for this task (Potash et al., 2017), this study also explores the neural network-based approach.

**Discourse connectives** In discourse parsing (Mann and Thompson, 1987; Prasad et al., 2008), which is closely related to ASP, discourse connectives are strong clues for identifying discourse relations (Marcu, 2000; Braud and Denis, 2016). Exploring effective ways to use the discourse connective information has received wide attention in various NLP fields (Sileo et al., 2019; Pan et al., 2018).

**Span representation** Span representation design is gaining considerable attention for several NLP tasks, such as syntactic parsing (Wang and Chang, 2016; Stern et al., 2017; Kitaev and Klein, 2018), semantic role labeling (He et al., 2018; Ouchi et al., 2018), and coreference resolution (Lee et al., 2017, 2018). One common practice for effective design is to use bidirectional long short-term memory networks (BiLSTMs). Wang and Chang (2016) proposed span representation called "LSTM-minus," which represents each span $(i, j)$ as the difference between the LSTM's hidden states over time steps, i.e. $\mathbf{h}_j - \mathbf{h}_i$. The work most similar to ours is Li et al. (2016), where LSTM-minus is used for discourse structure prediction. This study extends it by integrating discourse properties into span representation.

## 3 Model

This section describes (i) an LSTM-minus-based span representation and (ii) a task-specific span representation for ASP.

### 3.1 Overview

The given input is a paragraph consisting of $T$ tokens $w_{1:T} = (w_1, w_2, \cdots, w_T)$ and preidentified segmental discourse units. In terms of segmental discourse units to be captured, we decompose ADUs into argumentative components (ACs) and argumentative markers (AMs). ACs correspond to claims and premises relevant to the writer's argumentation. AMs, which play a crucial role in controlling argumentative flows, are conjunctive expressions frequently used in argumentative texts (see Table 1). We explore span representation of ADUs by leveraging AC and AM information as task-dependent extensions for ASP. Each ADU span is denoted as $(i, j)$, where $i$ and $j$ are word indices ($1 \leq i \leq j \leq T$). We contextualize each representation on all of the ADUs in a paragraph by using BiLSTMs. These vectors are then fed to each task-specific classifier.

| |
|---|
| *the other reason is that* |
| *consequently,* |
| *in conclusion, from the above views, although* |
| *first, as you can see that* |
| *in this essay , the reasons for why i agree that* |
| *it is a debatable subject that* |
| *unfortunately* |
| *although some argue that* |
| *furthermore , it 's undeniable that the* |
| *in short,* |
| *another thing that put big cities in front of small towns is* |
| *in conclusion, despite the contribution of it to the society,* |
| *however, some say that* |

Table 1: Examples of the AMs.

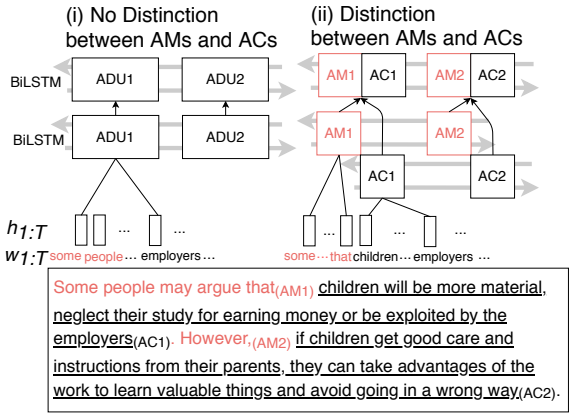


Figure 2: Example of a part of an argumentative essay (AMs are highlighted in red and ACs are underlined), and the models with or without the AC/AM distinction.

### 3.2 LSTM-minus-based span representation

We first build a model based on the LSTM-minus-based span representation (Wang and Chang, 2016), as shown in Figure 2 (left). This representation makes no distinction between AMs and ACs

and deals with them as one combined span $(i, j)$.

$$\begin{aligned}\mathbf{w}_{1:T} &= f^{\text{emb}}(w_{1:T}) \ , \\ \mathbf{h}_{1:T} &= \text{BiLSTM}(\mathbf{w}_{1:T}) \ , \\ \mathbf{h}_{(i,j)} &= [\overrightarrow{\mathbf{h}}_j - \overrightarrow{\mathbf{h}}_{i-1}; \overleftarrow{\mathbf{h}}_i - \overleftarrow{\mathbf{h}}_{j+1}; \\ &\quad \overrightarrow{\mathbf{h}}_{i-1}; \overleftarrow{\mathbf{h}}_{j+1}; \phi(w_{i:j})] \ . \end{aligned} \quad (1)$$

Here, the word embedding layer $f^{\text{emb}}$ maps the input tokens $w_{1:T}$ to a sequence of embeddings $\mathbf{w}_{1:T}$. Taking $\mathbf{w}_{1:T}$ as input, the BiLSTM layer then outputs a sequence of hidden states $\mathbf{h}_{1:T}$. Finally, from $\mathbf{h}_{1:T}$, the LSTM-minus-based representation $\mathbf{h}_{(i,j)}$ is assigned to each combined span $(i, j)$. Following Potash et al. (2017), we also concatenate the additional discrete feature vector $\phi(w_{i:j})$[2] of bag-of-words (BoW) and position information of each span.

### 3.3 Distinction between AMs and ACs

As a task-specific extension of the LSTM-minus-based span model (henceforth, the LSTM model), we build a model that distinguishes between AMs $(i, k)$ and ACs $(k + 1, j)$, as shown in Figure 2 (right). Similar to the LSTM model, each AM $(i, k)$ is assigned its span representation $\mathbf{h}_{(i,k)}$, and each AC $(k + 1, j)$ is assigned $\mathbf{h}_{(k+1,j)}$. Then, using two independent BiLSTMs, the model contextualizes AM spans and AC spans, respectively. The distinction and contextualization enable the model to better capture the argumentative flow; for example, the model can learn patterns of AM sequences (e.g., *in my opinion* → *for example* → *however*).

Finally, each contextualize AC span representation $\mathbf{h}^{\text{ctx}}_{(k+1,j)}$ is concatenated with its preceding contextualized AM span representation $\mathbf{h}^{\text{ctx}}_{(i,k)}$ and discrete feature vectors $\phi(w_{i:j})$, i.e. $\mathbf{h}_{(i,j)} = [\mathbf{h}^{\text{ctx}}_{(i,k)}; \mathbf{h}^{\text{ctx}}_{(k+1,j)}, \phi(w_{i:j})]$.

### 3.4 Output layers

Based on the contextualized ADU representations, we compute the class probability for each subtask. As the output function, we use the softmax function. In AC/link type classification, the softmax function computes the probability of each type. In link identification, it computes the probability that the $m$-th ADU has a directed link to the $h$-th ADU.

[2]Details of discrete features are shown in the Appendix A.2

### 3.5 Training

We assume that we have access to a training set $\mathcal{D}$,

$$\begin{aligned}\mathcal{D} &= \{(X, Y^{\text{link}}, Y^{\text{ac-type}}, Y^{\text{link-type}})_d\}_{d=1}^{|\mathcal{D}|} \ , \\ X &= \{w_{1:T}, s^{\text{ac}}_{1:M}, s^{\text{am}}_{1:M}\} \ , \\ Y^{\text{link}} &= \{h_1, \cdots, h_M\} \ , \\ Y^{\text{ac-type}} &= \{r_1, \cdots, r_M\} \ , \\ Y^{\text{link-type}} &= \{t_1, \cdots, t_M\} \ , \end{aligned}$$

where $s^{\text{ac}}$ and $s^{\text{am}}$ denote AC span and AM span, respectively. $Y^{\text{link}}$ is the set of candidate spans (including a root object) to which the given span has a directed link. $Y^{\text{ac-type}}$ and $Y^{\text{link-type}}$ are the set of all possible categories defined in each task, e.g., $Y^{\text{link-type}} = \{\text{SUPPORT}, \text{ATTACK}\}$.

To train model parameters, we minimize the joint cross-entropy loss function,

$$\begin{aligned}\mathcal{L} = -\sum_{(X, Y^{\text{link}}, Y^{\text{type}}) \in \mathcal{D}} \big[ & \alpha \, \ell^{\text{link}}(X, Y^{\text{link}}) \\ & + \beta \, \ell^{\text{ac-type}}(X, Y^{\text{ac-type}}) \\ & + (1 - \alpha - \beta) \, \ell^{\text{link-type}}(X, Y^{\text{link-type}}) \big] \ , \end{aligned}$$

where the three loss functions $\ell^{\text{link}}$, $\ell^{\text{ac-type}}$, and $\ell^{\text{link-type}}$ are interpolated with $\alpha$ and $\beta$. Each loss function is defined as follows:

$$\ell^{\text{task}}(X, Y^{\text{task}}) = \sum_{y \in Y^{\text{task}}} \log \text{P}(y|m) \ ,$$

where $m$ denotes the $m$-th ADU. We show the details of model training and hyperparameters in the Appendix A.3.

## 4 Experiments

### 4.1 Experimental setup

**Task setting** The ASP task consists of the following subtasks: (i) AC segmentation, (ii) AC type classification (ATC), (iii) link indentification (LI), and (iv) link type classification (LTC).

Some previous studies (Potash et al., 2017; Niculae et al., 2017; Peldszus and Stede, 2015) skipped the AC segmentation task and used gold AC segmentation for the other three subtasks. Following these studies, we adopted the same task setting.

**Dataset** We used the PEC (Stab and Gurevych, 2017) and arg-microtext corpus (MTC) (Peldszus and Stede, 2016) for our experiments. The PEC consists of 402 essays (1,833 paragraphs) posted on an online portal[3]. We used the same train/test

[3]https://essayforum.com/

split used in Stab and Gurevych (2017) and randomly selected 10% of the training set as the validation set in the PEC. The scores obtained for the PEC were averaged across three distinct trials using different random seeds. The MTC contains 112 short texts. Because this dataset is small, we conducted 10 sets of five-fold cross-validation and obtained the average scores across all the splits. We list the statistics of the datasets in the Appendix C.

**Argumentative marker extraction** In the PEC, while ACs are annotated, AMs are not. We thus extracted AMs using simple rules. The tokens preceding an AC were extracted as its AMs. Note that AMs are not beyond the sentence and do not overlap with other ACs. We considered pairs of ACs and the preceding AMs as ADUs.

In the MTC, there are annotations of ADUs, which include both ACs and AMs. First, we created an AM list from the PEC and Penn Discourse TreeBank (Prasad et al., 2008). If an ADU began with the phrases (choosing the longest one) specified in the AM list, we extracted the phrase as the AM and the following tokens as the AC.

We collected 1,131 AMs, which had 5.38 tokens on average. We show the details of extraction procedures in the Appendix D.

**Implementation details** We used GloVe (Pennington et al., 2014) and ELMo embeddings (Peters et al., 2018), which have different properties. To explore the performance of the three subtasks, the models jointly predicted the three subtasks.

**Baseline** As a baseline, we used the current state-of-the-art model using span representation based on BoW (Potash et al., 2017) (BoW span model[4]).

## 4.2 Results

We calculated macro-F1 and individual class F1 scores for the three tasks. For the overall performance, we calculated averaged macro-F1 scores across all the three tasks. As a statistical significance test, we used bootstrap resampling (Koehn, 2004).

Table 2 shows the results of our experiments. We obtained the following three tendencies: (i) compared with the BoW span model, the LSTM models show significantly better performance, (ii) the span representation capturing the distinction between ACs and AMs (LSTM+dist) improves the performance especially in LI and LTC, and (iii) the ELMo embeddings boost performance. We also compared the LSTM+dist model with the existing models. Table 3 shows that our span-based models achieved new state-of-the-art scores on the PEC and yielded competitive results on the MTC compared to the current state-of-the-art model.

[4]Details of BoW span representation are shown in the Appendix A.1

| Data. | Embed. | Span rep. | Overall | LI | LTC | ATC |
|---|---|---|---|---|---|---|
| PEC | ELMo | LSTM+dist | **81.8** | **80.7** | **79.0** | 85.7 |
| | | LSTM | **81.8** | 80.4$^{\dagger}$ | 78.2$^{\dagger}$ | **86.9**$^{\dagger}$ |
| | | BoW | 77.1 | 76.2 | 72.3 | 82.9 |
| | GloVe | LSTM+dist | **79.7** | **78.8** | **76.5** | **83.9** |
| | | LSTM | 78.8 | 77.7$^{\dagger}$ | 75.0$^{\dagger}$ | 83.7 |
| | | BoW | 76.1 | 74.2 | 71.3 | 82.8 |
| MTC | ELMo | LSTM+dist | **78.2** | **73.9** | **77.2**$^{\ddagger}$ | **83.4** |
| | | LSTM | 75.0 | 73.0$^{\dagger}$ | 71.5 | 80.5 |
| | | BoW | 73.3 | 71.2 | 67.5 | 81.2 |
| | GloVe | LSTM+dist | **76.5** | **72.6** | **75.4**$^{\ddagger}$ | **81.5** |
| | | LSTM | 70.4 | 70.1 | 64.1 | 76.9 |
| | | BoW | 71.1 | 69.2 | 64.8$^{\dagger}$ | 79.3 |

Table 2: Comparison among the LSTM-dist, LSTM, and BoW based representation. The results of LSTM model marked with † are statistically significant compared to the BoW representation ($p < 0.05$). The results of LSTM+dist model marked with ‡ are statistically significant compared to the LSTM representation ($p < 0.05$)

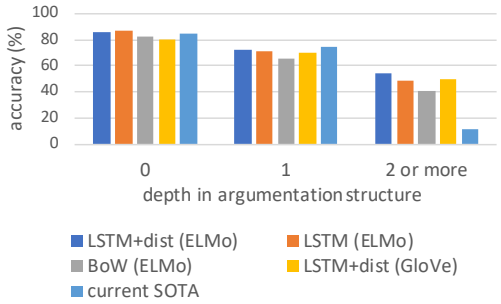


Figure 3: Accuracy in LI according to different depths.

## 4.3 Analysis

We conducted a detailed analysis on the PEC for investigating the types of instances that were easy or difficult for the span models and the current state-of-the-art model to predict, focusing on the most challenging subtask, i.e., LI.

Stab and Gurevych (2017) reported that their model tends to output shallow trees even if the corresponding gold trees are deeper. We therefore investigated the performance according to different depths of ADU trees. Figure 3 indi-

| Data. | Model | Overall Avg. | LI Macro | LI Link | LI No-Link | LTC Macro | LTC Support | LTC Attack | ATC Macro | ATC MC | ATC Claim | ATC Premise |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PEC | LSTM+dist (ELMo) | 81.8 | 80.7 | 67.8 | 93.7 | 79.0 | 96.8 | 61.1 | 85.7 | 91.6 | 73.3 | 92.1 |
| | Joint PointerNet (Potash et al., 2017) | - | 76.7 | 60.8 | 92.5 | - | - | - | 84.9 | 89.4 | 73.2 | 92.1 |
| | St. SVM-full (Niculae et al., 2017) | - | - | 60.1 | - | - | - | - | 77.6 | 78.2 | 64.5 | 90.2 |
| | ILP Joint (Stab and Gurevych 2017) | 75.2 | 75.1 | 58.5 | 91.8 | 68.0 | 94.7 | 41.3 | 82.6 | 89.1 | 68.2 | 90.3 |
| MTC | LSTM+dist (ELMo) | 78.2 | 73.9 | 57.5 | 90.3 | 77.2 | 84.2 | 70.3 | 83.5 | - | 72.9 | 94.0 |
| | Joint PointerNet (Potash et al., 2017) | - | 74.0 | 57.7 | 90.3 | - | - | - | 81.3 | - | 69.2 | 93.4 |
| | New Best EG (Afantenos et al., 2018) | 78.5 | 72.2 | - | - | 75.7 | - | - | 87.6 | - | - | - |
| | ILP Joint (Stab and Gurevych 2017) | 76.2 | 68.3 | 48.6 | 88.1 | 74.5 | 85.5 | 62.8 | 85.7 | - | 77.0 | 94.3 |

Table 3: Comparison with existing models on the PEC and the MTC. MC denotes MAJORCLAIM.

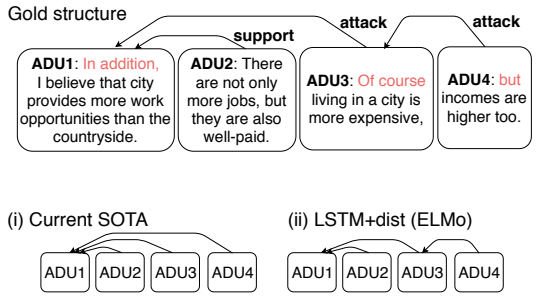


Figure 4: Example of the outputs predicted by the current state-of-the-art model and the LSTM-dist model.

| Model | acc. |
|---|---|
| LSTM+dist (ELMo) | **71.0** |
| LSTM (ELMo) | 68.9 |
| BoW (ELMo) | 60.7 |
| Joint Pointer Net. (Current SOTA) | 50.3 |

Table 4: Accuracy for predicting the relations which form an ATTACK relation chain.

cates the accuracy of predicting outgoing links (i.e. parents) from each AC at each depth. Overall, it is difficult to predict the links for deeper ADUs. While the performance of the state-of-the-art model (Joint Pointer Net) sharply dropped for deeper ACs (two or more), the LSTM models (LSTM and LSTM+dist) could robustly predict the correct links for deeper ADUs.

Figure 4 shows an example of an argumentation structure with a chain of relations (ADU4→ADU3→ADU1) and the outputs predicted by the state-of-the-art model and the LSTM+dist model. The writer of this text first states an expected opposite opinion (ADU3) and then re-attacks the opposite opinion (ADU4). In such a structure, while the state-of-the-art model failed to predict the higher-order relations, the LSTM+dist model succeeded. Furthermore, Table 4 shows that the distinction between ACs and AMs had a positive effect on predicting the chains of ATTACK relations. Such ATTACK relation chains have been regarded as an important characteristic of argumentation structure (Peldszus and Stede, 2013; Freeman, 2011) and frequently appear especially in the MTC. Thus, the performance difference between the LSTM model and the LSTM+dist model (see Table 2) on the MTC is relatively greater than that of the PEC. One possible reason for this positive effect is that typical AM flows (such as *of course*→*but*), which the LSTM+dist model explicitly captures, can be a clue for predicting higher-order chains.

## 5 Conclusion and Future work

This work has studied span representations for ASP. Specifically, we have investigated (i) an LSTM-minus-based span representation originally developed for other NLP tasks and (ii) a task-specific extended representation capturing the AM/AC distinction for ASP. The experimental results have demonstrated the effects of these representations in ASP and that the span representation capturing the AM/AC distinction achieves state-of-the-art results for three subtasks. The empirical analysis has showed that there is room for improvement in the LI for deeper-level ADUs. One interesting line of our future work is to investigate the performance of our model in an end-to-end setting (including AC segmentation). Another direction is to explore span representations in several related tasks such as RST-style discourse parsing or new span-related argumentation mining tasks (Trautmann et al., 2019).

## 6 Acknowledgements

This work was supported by JST CREST Grant Number JPMJCR1513, Japan. We would like to thank the laboratory members who gave us advice and all reviewers of this work for their useful comments and feedback.

## References


Chloé Braud and Pascal Denis. 2016. Learning connective-based word representations for implicit discourse relation identification. In *Proceedings of EMNLP*, pages 203–213.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *Proceedings of ACL*, pages 11–22.

James B Freeman. 2011. *Argument Structure:: Representation and Theory*, volume 18. Springer Science & Business Media.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of ACL*, pages 364–369.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of ACL*, pages 2676–2686.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395.

John Lawrence and Chris Reed. 2015. Combining argument mining techniques. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 127–136.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of EMNLP*, pages 188–197.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of NAACL-HLT*, pages 687–692.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *Proceedings of EMNLP*, pages 362–371.

William C Mann and Sandra A Thompson. 1987. Rhetorical structure theory: Description and construction of text structures. In *Natural Language Generation*, pages 85–95.

Daniel Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT press.

Huy V Nguyen and Diane J Litman. 2016. Context-aware Argumentative Relation Mining. *Proceedings of ACL*, pages 1127–1137.

Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. Argument Mining with Structured SVMs and RNNs. In *Proceedings of ACL*, pages 985–995.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of EMNLP*, pages 1630–1642.

Boyuan Pan, Yazheng Yang, Zhou Zhao, Yueting Zhuang, Deng Cai, and Xiaofei He. 2018. Discourse marker augmented network with reinforcement learning for natural language inference. In *Proceedings of ACL*, pages 989–999.

Andreas Peldszus and Manfred Stede. 2013. From Argument Diagrams to Argumentation Mining in Texts: a survey. *International Journal of Cognitive Informatics and Natural Intelligence*, 7(1):1–31.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in MST-style discourse parsing for argumentation mining. In *Proceedings of EMNLP*, pages 938–948.

Andreas Peldszus and Manfred Stede. 2016. An Annotated Corpus of Argumentative Microtexts. *Studies in Logic and Argumentation*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. Here's My Point: Joint Pointer Architecture for Argument Mining. In *Proceedings of EMNLP*, pages 1375–1384.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *Proceedings of the Sixth Conference on Language Resources and Evaluation*, pages 2961–2968.

Damien Sileo, Tim Van-De-Cruys, Camille Pradel, and Philippe Muller. 2019. Mining discourse markers for unsupervised sentence representation learning. In *Proceedings of NAACL-HLT*.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. In *Proceedings of Computational Linguistics*, volume 43, pages 619–659. MIT Press.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of ACL*, pages 818–827.

Dietrich Trautmann, Johannes Daxenberger, Christian Stab, Hinrich Schütze, and Iryna Gurevych. 2019. Robust argument unit recognition and classification. In *arXiv:1904.09688*.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of ACL*, pages 2306–2315.

# A Model

## A.1 Features of BoW models

Following Potash et al. (2017), we used the following features as ADU representations in BoW models: (i) one-hot vector of BoW; (ii) embedding representation calculated by average, max, and min pooling across the token embeddings; (iii) position of the ADU in an essay and paragraph and whether ADU is in opening, body, or closing paragraph (one-hot vector).

In BoW models with ELMo, using all of the versions of embedding representations (avg, min, and max pooling) hinders the performance. We only used embedding representations calculated by max pooling in BoW models with ELMo.

## A.2 Discrete features of LSTM-minus-based models

As mentioned in Section 3, we additionally used discrete features in designing ADU representation of LSTM and LSTM-dist models ($\phi(w_{i:j})$ in equation 1). We used (i) Bag-of-words and (iii) position information as additional discrete features. In LSTM-dist models, while ADU span representation $\mathbf{h}_{(i,j)}$ is concatenated with its discrete features $\phi(w_{i:j})$, AC and CE span representation ($\mathbf{h}_{(i,k)}$ and $\mathbf{h}_{(k+1j)}$) has no discrete features.

## A.3 Output layers

**AC/Link type classification layer** Taking each span vector $\mathbf{h}_{(i,j)}$, the AC (or link) type classification layer computes the probability that the span $m = (i, j)$ is classified into each AC/Link type $r \in \mathcal{R}$.

$$\text{score}^{\text{type}}_{m,r} = \mathbf{w}^{\text{type}}_r \cdot \mathbf{h}_n + b^{\text{type}}_r \ ,$$

$$\text{P}(r|m) = \frac{\exp(\text{score}^{\text{type}}_{m,r})}{\sum_{r' \in \mathcal{R}} \exp(\text{score}^{\text{type}}_{m,r'})} \ ,$$

where $\mathbf{w}^{\text{type}}_r$ is a parameter vector and $b^{\text{type}}_r$ is a bias parameter associated with a type $r$. In ATC, $\mathcal{R} = \{\text{PREMISE}, \text{CLAIM}, \text{MAJORCLAIM}^5\}$. In LTC, $\mathcal{R} = \{\text{SUPPORT}, \text{ATTACK}\}$.

**Link identification layer** Taking each span vector $\mathbf{h}_{(i,j)}$, the link identification layer computes the probability that span $m = (i, j)$ has a directed link to span $h = (i', j')$.

$$\text{score}^{\text{link}}_{m,h} = \mathbf{w}^{\text{link}} \cdot [\mathbf{h}^{\text{link}}_m; \mathbf{h}^{\text{link}}_h; \mathbf{h}^{\text{link}}_m \odot \mathbf{h}^{\text{link}}_h; \phi(h, m)] \ ,$$

$$\text{P}(h|m) = \frac{\exp(\text{score}^{\text{link}}_{m,h})}{\sum_{h'=1}^{M} \exp(\text{score}^{\text{link}}_{m,h'})} \ ,$$

where $\mathbf{w}^{\text{link}}$ is a parameter vector. $\phi(h, m)$ is one-hot vector of relative position between $h$-th span and $m$-th span. $R$ is the set of ADU spans in the same paragraph (including a root object). To decode the tree structure, we use the maximum spanning tree algorithm based on the probabilities calculated by the softmax function.

[5] Major claims appear only in the PEC.

| Name | Value |
|---|---|
| **Word embeddings** | |
| - Glove | 300 dim. |
| - ELMo | 1024 dim. |
| BiLSTMs | 256 (300 in the LSTM models) dim. |
| Mini-batch size | 16 |
| Optimizer | Adam |
| Learning rate | 0.001 |
| Epoch | 500 (1000 in the MTC) |
| **Loss interpolation** | |
| - $\alpha$ | 0.5 |
| - $\beta$ | 0.25 |
| **Dropout ratio** | |
| - Output layer | 0.5 (0.9 in the MTC) |
| - BiLSTMs | 0.1 (0.9 in the MTC) |
| - ELMo | 0.1 |

Table 5: Hyperparameters for our span-based model.

# B Hyperparameters

Table 5 shows the hyperparameters used in our experiments.

**Network setup** We use 300-dimensional GloVe and 1024-dimensional ELMo embeddings.

There was concern that LSTM-dist models outperformed the other models just because they had larger parameters. For fair comparison among BoW, LSTM, and LSTM-dist models, we preliminary conducted experiments with increasing the number of parameters in BoW and LSTM models (changed the dimensions of the hidden units of all the BiLSTMs from 256 to 300 and added layers in the LSTM used in ADU-level contextualization). While parameter increased versions of BoW models did not perform better, LSTM models slightly improve by increasing their parameters.

The dimensions of the hidden units of all the BiLSTMs used in BoW and LSTM-dist models

| | | PEC | MTC |
|---|---|---|---|
| Texts | All essays | 402 | - |
| | Paragraphs | 1,833 | 112 |
| | Sentences | 7,116 | - |
| | Tokens | 147,271 | - |
| ACs | All ACs | 6,089 | 576 |
| | Major claim | 751 | - |
| | Claim | 1,506 | 112 |
| | Premise | 3,832 | 464 |
| Links | All links | 3,832 | 464 |
| | Support | 3,613 | 290 |
| | Attack | 219 | 174 |

Table 6: Statistics of the PEC and the MTC.

are set to 256 and the BiLSTMs have one hidden layer. In the LSTM models, the dimensions of the hidden units are set to 300 and LSTM used in ADU-level contextualization has two layers.

**Regularization** We apply dropout to the input vectors of the output layer with a dropout ratio of 0.5 in the PEC. We also apply dropout to all the BiLSTMs and word embeddings with 0.1 in the PEC. In the MTC, we increase the ratio of dropout (Table 5)

**Model selection** We select the models that achieve the highest averaged macro-F1 across the three tasks on the validation set.

## C Data

Table 6 shows the statistics of the PEC and the MTC. The PEC is about ten times larger than the MTC.

## D Argumentative Marker Extraction

We distinguish *because* in the following examples:

- *A.* **Because** *B, C.*
- *A* **because** *B. C.*

As a simple solution, we include a period in the AMs. Then, AMs of the examples above are . *because* and *because*, respectively. If there is no lexical AM in AC, then only end of sentence symbols (i.e., ., *!*, and *?*) in the preceding sentence become its AMs. In the PEC, 63% of ADUs have AMs. Sometimes AMs are inserted in the middle of sentences (e.g. *Others,* **however***, think that these children may disrupt their school work and should be allowed to leave school early to find a job.*). In this case, we do not extract AMs. In fact, this is a rare case (e.g. *, however,* exists in 1% of the paragraphs in the PEC.).

| Joint Tasks | | | Performance | | |
|---|---|---|---|---|---|
| LI | LTC | ATC | LI | LTC | ATC |
| ✓ | ✓ | ✓ | 80.7 | 79.0 | 85.7 |
| ✓ | ✓ | | 80.2 | 78.6 | - |
| ✓ | | ✓ | **81.1** | - | **87.3** |
| | ✓ | ✓ | - | 78.0 | 86.0 |
| ✓ | | | 78.3 | - | - |
| | ✓ | | | **79.6** | |
| | | ✓ | - | - | 85.6 |

Table 7: Joint learning results on the PEC.

## E Joint learning analysis

We additionally analyzed the effectiveness of joint learning with the LSTM+dist model. In jointly learning three tasks, we set $\alpha = 0.5$, $\beta = 0.25$ in the joint cross-entropy loss function. In jointly learning two tasks, we set 0.5 as each task's weight in the joint cross-entropy loss function. Table 7 shows the result. The check mark in Table 7 means that the model jointly learns the task. The numbers denote macro-F1 scores. We found that link type prediction task does not benefit from joint learning.