A Generative Model for End-to-End Argument Mining with Reconstructed Positional Encoding and Constrained Pointer Mechanism

Jianzhu Bao[1,2*], **Yuhang He**[1,2*], **Yang Sun**[1,2], **Bin Liang**[1,2†], **Jiachen Du**[1,2], **Bing Qin**[1], **Min Yang**[3], **Ruifeng Xu**[1,2,4†]

[1]Harbin Institute of Technology, Shenzhen, China
[2]Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies
[3]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
[4]Peng Cheng Laboratory, Shenzhen, China
jianzhubao@gmail.com, yuhang.he.hitsz@outlook.com,sy95@mail.ustc.edu.cn
bin.liang@stu.hit.edu.cn, jacobvan199165@gmail.com
qinb@ir.hit.edu.cn, min.yang@siat.ac.cn, xuruifeng@hit.edu.cn

Abstract

Argument mining (AM) is a challenging task as it requires recognizing the complex argumentation structures involving multiple subtasks. To handle all subtasks of AM in an end-to-end fashion, previous works generally transform AM into a dependency parsing task. However, such methods largely require complex pre- and post-processing to realize the task transformation. In this paper, we investigate the end-to-end AM task from a novel perspective by proposing a generative framework, in which the expected outputs of AM are framed as a simple target sequence. Then, we employ a pre-trained sequence-to-sequence language model with a constrained pointer mechanism (CPM) to model the clues for all the subtasks of AM in the light of the target sequence. Furthermore, we devise a reconstructed positional encoding (RPE) to alleviate the order biases induced by the autoregressive generation paradigm. Experimental results show that our proposed framework achieves new state-of-the-art performance on two AM benchmarks.[1]

1 Introduction

As a fundamental task of computational argumentation, argument mining (AM) has drawn much research attention recently (Schaefer and Stede, 2021; Vecchi et al., 2021; Lawrence and Reed, 2019). The ultimate goal of AM is to analyze and understand argumentative text, so as to obtain structured argumentation knowledge that can support a diverse range of downstream tasks, such as argument persuasiveness prediction (Li et al., 2020; Huang et al., 2021), automated essay scoring (Ghosh et al., 2016; Nguyen and Litman, 2018; Song et al., 2020), argument generation (Hua et al., 2019; Slonim et al., 2021; Khatib et al., 2021), text summarization (Fabbri et al., 2021; Bar-Haim et al., 2020), etc.

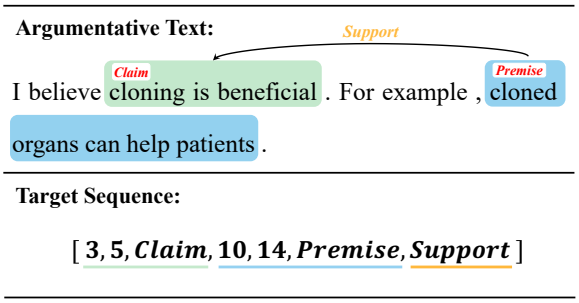

Figure 1: A simplified example of AM task. Two argument components are marked in green and blue, respectively, where the former is a *Claim* and the latter is a *Premise*. In addition, there is a *Support* relation from the *Premise* to the *Claim*. Our proposed target sequence corresponding to this example is shown at the bottom.

Given a piece of argumentative text as input, an end-to-end AM system needs to identify both the argument components (ACs) and the argumentative relations (ARs) between them. An example is shown in Figure 1. Specifically, AM generally comprises four fine-grained subtasks (Eger et al., 2017): 1) *component segmentation* detects the boundaries of fine-grained argumentative segments, which are known as ACs; 2) *component classification* classifies the ACs into the categories defined by argumentation schemes; 3) *relation detection* determines whether there is an AR between two ACs; 4) *relation classification* further classifies the types of the ARs. Following Persing and Ng (2016) and Ye and Teufel (2021), we refer the first two subtasks as *argument component identification* (ACI), and the last two subtasks as *argumentative relation identification* (ARI). The end-to-end AM task is highly challenging as it is difficult to solve all the AM subtasks synchronously in a unified framework.

*Equal Contribution
†Corresponding Authors
[1]Code is available at https://github.com/HITSZ-HLT/GMAM.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 10437 - 10449
December 7-11, 2022

Most previous work focuses on only a subset of the four fine-grained subtasks (Niculae et al., 2017; Reimers et al., 2019; Jo et al., 2019; Morio et al., 2020; Lenz et al., 2020; Ruiz-Dolz et al., 2021; Bao et al., 2021). However, only a limited number of studies are devoted to the end-to-end AM scenario (Persing and Ng, 2016; Eger et al., 2017; Ye and Teufel, 2021). Recent research efforts formulate the end-to-end AM as a dependency parsing task and apply existing dependency parsers to solve it (Ye and Teufel, 2021; Dozat and Manning, 2018). Such methods, however, require not only a tedious preprocessing process to transform the argumentation structure into an elaborately-designed dependency graph, but also a complex post-processing process to ensure that each dependency of the predicted output is completely consistent with that in the desired dependency graph of the AM task. Thus, developing an elegant, simple, and effective framework for the end-to-end AM task is still an important challenge of great significance.

Inspired by the recent success of the generative methods for information extraction (Yan et al., 2021b; Zhang et al., 2022), we propose to address the end-to-end AM task via a unified generative framework. We first devise a target sequence to express the outputs of all the subtasks of AM. An example is shown in Figure 1. Subsequently, the pre-trained BART (Lewis et al., 2020) model is adopted to model the dependencies between the target sequence and the input argumentative text through the pointer mechanism. Further, we introduce a reconstructed positional encoding (RPE) scheme in the BART decoder to alleviate the order biases induced by the autoregressive generation paradigm. In addition, considering the long length of the ACs and the pattern of the target sequence, we present a constrained pointer mechanism (CPM), which is manifested as an auxiliary task at the training time and as a constrained decoding method at the inference time. This constrained pointer mechanism can help the model to generate more accurate AC boundaries and fewer invalid target sequences. Compared to the previous dependency parsing-based method, it is more straightforward and easier to formalize the end-to-end AM task into a generation task. Also, in our proposed method, the predicted target sequence can be easily converted to the expected outputs of AM without complex post-processing.

We conduct extensive experiments on two AM benchmarks of different structures to show the superiority of our method. Experimental results show that our proposed method achieves substantial improvements over several strong baselines, yielding the state-of-the-art performance on both benchmark datasets. In addition, we carry out further analysis to show that the proposed RPE and CPM can significantly reduce the errors in the generated target sequence, thus leading to performance improvements.

2 Related Work

2.1 Argument Mining

AM traditionally involves four fine-grained subtasks. Early work usually exclusively studies a particular subtask, such as *component segmentation* (Moens et al., 2007; Florou et al., 2013; Goudas et al., 2014), *component classification* (Palau and Moens, 2009; Stab and Gurevych, 2014; Lippi and Torroni, 2015; Nguyen and Litman, 2015), *relation detection* (Palau and Moens, 2009; Stab and Gurevych, 2014), and *relation classification* (Ghosh et al., 2014; Boltuzic and Snajder, 2014; Peldszus, 2014; Cocarascu and Toni, 2017).

Recently, there has been a trend to study the joint modeling of multiple subtasks of AM. However, most work only addresses a subset of the four fine-grained subtasks of AM, instead of performing an end-to-end approach. For joint modeling *component segmentation* and *component classification*, Chernodub et al. (2019) built a neural sequence labeling model, while Wang et al. (2020) proposed a multi-scale model to recognize different types of ACs at corresponding levels. Since *component segmentation* is a token-level task while other three subtasks are at segment-level (Ye and Teufel, 2021), it is difficult to model them jointly. Thus, some previous studies ignore the *component segmentation* task and only jointly model the other three subtasks. Kuribayashi et al. (2019) explored the application of span representation in AM. Morio et al. (2020) incorporated task-specific parameterization and biaffine attention for improving non-tree AM. Many other works further ignore the *relation classification* task. Potash et al. (2017) employed a pointer network with the attention mechanism for structural prediction. Niculae et al. (2017) presented a factor graph model to impose structure constraints. Bao et al. (2021) proposed a transition-based neural network to construct argumentation graphs.

Compared to the studies above, there have been

relatively fewer researches on end-to-end AM. Persing and Ng (2016) performed joint inference in an Integer Linear Programming (ILP) framework. Eger et al. (2017) formalized the end-to-end AM task into multiple other tasks, including sequence labeling, end-to-end relation extraction, and dependency parsing. However, sequence labeling-based methods try to predict the distance between ACs by token-level classification, which is hard to learn and the results are suboptimal. The relation extraction-based models solve ACI and ARI sequentially but may result in poor performance due to error propagation. Although Ye and Teufel (2021) further extended the dependency parsing approach of Eger et al. (2017) and achieved promising performance, it requires tedious pre- and post-processing (e.g. label refinement, removing invalid or multiple edges).

2.2 Generative Methods for IE

With superior development of pre-training techniques, there has been a rising trend of adopting generative models to solve information extraction (IE) tasks, such as aspect-based sentiment analysis (Zhang et al., 2021b,a), named entity recognition (Ren et al., 2021; Cui et al., 2021; Zhang et al., 2022), event argument extraction (Li et al., 2021), etc.

Closely related to our work, some recent studies incorporate the pre-trained generative models with the pointer mechanism to better address IE tasks. Yan et al. (2021a) formalized all the subtasks of aspect-based sentiment analysis into generation tasks, and employed the pre-trained BART (Lewis et al., 2020) with the pointer mechanism to address them in a unified framework. Similarly, Yan et al. (2021b) explored solving multiple NER subtasks with pre-trained BART.

2.3 Pointer Mechanism

Pointer mechanism (Vinyals et al., 2015) aims to solve the problem of generating an output sequence that contains elements from the input sequence, which is usually based on a sequence-to-sequence model with the attention mechanism (Bahdanau et al., 2015). It has been applied to various tasks including dependency parsing (Ma et al., 2018; Liu et al., 2019; Fernández-González and Gómez-Rodríguez, 2020), named entity recognition (Yan et al., 2021b; Fei et al., 2021; Yang and Tu, 2022), text summarization (Miao and Blunsom, 2016; See et al., 2017; Paulus et al., 2018), etc.

In this paper, we modify the traditional pointer mechanism by imposing task-specific constraints to make it more suitable for the generative model for the end-to-end AM.

3 Task Formulation

Formally, for the end-to-end AM, the input is a piece of argumentative text $X = [w_1, w_2, \ldots, w_{n_x}]$ with n_x tokens. The first goal is to extract a set of ACs $A = \{a_i | a_i = (s_i, e_i, c_i)\}_{i=1}^{|A|}$, where a_i is the i-th AC, s_i and e_i respectively denote its start and end indexes, c_i represents its category label, such as "Claim", "Premise", etc. The second goal is to output a set of ARs $R = \{(a_i^{sc}, a_i^{tc}, r_i)\}_{i=1}^{|R|}$, where $a_i^{sc} \in A$ and $a_i^{tc} \in A$ denote the source and target ACs, r_i is the AR category label, such as "Support", "Attack", etc. Here, we denote the AC and AR category label lists as $L^c = [l_1^c, l_2^c, \ldots, l_{n_c}^c]$ and $L^r = [l_1^r, l_2^r, \ldots, l_{n_r}^r]$, where l_i^c/l_i^r is the i-th AC/AR category label, n_c/n_r is the number of all the possible AC/AR category labels.

To solve the end-to-end AM through a generative framework, we need to formulate it as a sequence-to-sequence generation task with X as the input source sequence. Also, the expected AM outputs A and R are transformed as the target sequence $Y = [T_1, T_2, \ldots, T_{|R|}]$, where the tuple $T_i = [s_i^{tc}, e_i^{tc}, c_i^{tc}, s_i^{sc}, e_i^{sc}, c_i^{sc}, r_i]$ represents the i-th AR in R. For the i-th AR, s_i^{sc}/e_i^{sc} and s_i^{tc}/e_i^{tc} respectively denote the start/end indexes of the source and the target ACs, c_i^{sc} and c_i^{tc} are their AC category labels.

Example. In Figure 1, there is only one AR, so the target sequence is $Y = [T_1] = [3, 5, Claim, 10, 14, Premise, Support]$. Also, the AC and AR category label lists for this example are $L^c = [MajorClaim, Claim, Premise]$ and $L^r = [Support, Attack]$, respectively.

4 Method

Inspired by Yan et al. (2021a,b), we utilize a BART-based generative framework as our basic model, which takes X as input and generates the target sequence Y with the vanilla pointer mechanism. Since ACs are much longer and have more ambiguous boundaries than named entities (Yan et al., 2021b) or aspect term (Yan et al., 2021a), it is more challenging to solve the end-to-end AM task by a generative framework. Hence, we introduce a

constrained pointer mechanism (CPM) to help the model generate more accurate AC boundaries and fewer invalid predictions. Further, we propose a reconstructed positional encoding (RPE) to alleviate the order biases introduced by the autoregressive paradigm in the basic model.

4.1 Basic Model

First, we feed X into the BART encoder to derive the hidden representations of the source sequence:

$$\mathbf{H}^e = \text{BART\_Encoder}(X) \tag{1}$$

where $\mathbf{H}^e \in \mathbb{R}^{n_x \times d}$, and d is the hidden dimension of BART.

Then, the BART decoder incorporates $\mathbf{H}^e$ and the previous decoder outputs $Y_{<t}$ to predict the current output. The hidden state of the last decoder layer at the time step t is:

$$\mathbf{h}_t^d = \text{BART\_Decoder}(\mathbf{H}^e, Y_{<t}) \tag{2}$$

where $\mathbf{h}_t^d \in \mathbb{R}^d$. Note that, during this procedure, each start/end index (i.e. s_i^{sc}, e_i^{sc}, s_i^{tc} or e_i^{tc}) in $Y_{<t}$ needs to be mapped to its corresponding token in X first.

Vanilla Pointer Mechanism At time step t, the vanilla pointer mechanism selects tokens from the input X through a pointer distribution $\bar{\mathbf{P}}_t \in \mathbb{R}^{n_x}$ over all the positions of X. In this way, the start/end indexes in the target sequence Y can be generated. However, we also expect the model to generate AC and AR category labels. Hence, we expand the pointer distribution $\bar{\mathbf{P}}_t$ as $\mathbf{P}_t \in \mathbb{R}^{n_x+n_c+n_r}$ by combining $\bar{\mathbf{P}}_t$ with the distributions over all the possible AC and AR category labels.

More precisely, by feeding X, L^c and L^r into the embedding layer of BART, we could obtain the token embedding matrix $\mathbf{E} \in \mathbb{R}^{n_x \times d}$, the AC category embedding matrix $\mathbf{L}^c \in \mathbb{R}^{n_c \times d}$ and the AR category embedding matrix $\mathbf{L}^r \in \mathbb{R}^{n_r \times d}$.

Following Yan et al. (2021a), the encoder output matrix $\mathbf{H}^e$ is combined with $\mathbf{E}$ to produce the representation matrix for the pointer mechanism:

$$\bar{\mathbf{H}}^e = \alpha(\text{MLP}_m(\mathbf{H}^e)) + (1-\alpha)\mathbf{E} \tag{3}$$

where α is a hyper-parameter, MLP_m is a multi-layer perceptron. Subsequently, the expanded pointer distribution $\mathbf{P}_t$ at the time step t can be derived by:

$$\mathbf{H}^p = [\bar{\mathbf{H}}^e; \mathbf{L}^c; \mathbf{L}^r] \tag{4}$$

$$\mathbf{P}_t = \text{Softmax}(\mathbf{H}^p \mathbf{h}_t^d) \tag{5}$$

where $\mathbf{P}_t \in \mathbb{R}^{n_x+n_c+n_r}$, and ; denotes the matrix concatenation operation in the first dimension. With $\mathbf{P}_t$, the probability $P(Y_t|Y_{<t}, X)$ of generating the i-th element of the target sequence Y can be obtained.

Finally, this model is optimized with the negative log-likelihood loss:

$$\mathcal{L}_b = -\sum_{t=1}^{|Y|} \log P(Y_t|Y_{<t}, X) \tag{6}$$

4.2 Constrained Pointer Mechanism

We refer to each position in $\mathbf{P}_t \in \mathbb{R}^{n_x+n_c+n_r}$ as a *pointer index*. The *pointer indexes* in range $I^x = [1, n_x]$ are *token indexes*, while the *pointer indexes* in range $I^c = [n_x+1, n_x+n_c]$ and $I^r = [n_x+n_c+1, n_x+n_c+n_r]$ are the *AC category indexes* and *AR category indexes*, respectively.

In each decoding time step, the vanilla pointer mechanism selects an index directly based on $\mathbf{P}_t$, which is not reasonable because the valid *pointer indexes* for each time step are not identical. To be specific, regarding the i-th tuple $T_i = [s_i^{tc}, e_i^{tc}, c_i^{tc}, s_i^{sc}, e_i^{sc}, c_i^{sc}, r_i]$ in the target sequence Y, when predicting the target AC's end index e_i^{tc} from its pointer distribution, all the *pointer indexes* less than the decoded start index s_i^{tc} are invalid, since the end index must be greater than the start index. Also, the *pointer indexes* in range I^c and I^r are also invalid, since e_i^{sc} must be a *token index* within range I^x.

To address this issue, we define the following three constraints:

- (1) When decoding an end index, it should be greater than its corresponding start index.
- (2) When decoding the start and end index of a source AC, they can not overlap with the target AC.
- (3) The valid *pointer indexes* must be consistent with the type of the expected output for the current time step. For example, when decoding the AR category label r_i, only the *pointer indexes* in range I^r (i.e. *AR category indexes*) are valid.

To introduce these constraints to the basic model, we further define a proxy distribution $\mathbf{Q}_t \in \mathbb{R}^{n_x+n_c+n_r}$ for the decoding time step t to simulate the real pointer distribution with constraints, where the values of the valid and invalid *pointer*

indexes are set to 1 and 0, respectively. To illustrate, for the example shown in Figure 1, the proxy distribution of each element in the target sequence is shown in Table 1.

With the proxy distribution $\mathbf{Q}_t$, we can easily incorporate the aforementioned three constraints into both the training and inference stages. Note that, during training, $\mathbf{Q}_t$ is constructed from the ground truth target sequence. During inference, it is constructed from the generated sequence in previous time steps.

Incorporating Constraints in Training. During training, we regard $\mathbf{Q}_t$ as the labels of an auxiliary binary classification task to steer the model in a multi-task learning manner. This auxiliary task can provide the model with supervision signals of whether each *pointer index* is valid, making it less likely to predict invalid results.

Concretely, we first calculate the probabilities of the binary classification task:

$$\bar{\mathbf{H}}^p = \mathrm{MLP}_a([\mathbf{H}^e; \mathbf{L}^c; \mathbf{L}^r]) \tag{7}$$

$$\mathbf{P}^a_t = \mathrm{Sigmoid}(\bar{\mathbf{H}}^p \mathbf{h}^d_t) \tag{8}$$

where $\mathbf{P}^a_t \in \mathbb{R}^{n_x+n_c+n_r}$. The training objective function for this auxiliary task is:

$$\begin{aligned}\mathcal{L}_a = -\sum_{t=1}^{|Y|}\sum_{i=1}^{n_x+n_c+n_r} & [\mathbf{Q}_{t,i}\log(\mathbf{P}^a_{t,i}) + \\ & (1-\mathbf{Q}_{t,i})\log(1-\mathbf{P}^a_{t,i})]\end{aligned} \tag{9}$$

where $\mathbf{P}^a_{t,i}$ and $\mathbf{Q}_{t,i}$ are the i-th elements in $\mathbf{P}^a_t$ and $\mathbf{Q}_t$, respectively.

During training, we combine $\mathcal{L}_a$ with the loss function of the basic model $\mathcal{L}_b$ as the joint training objective.

Incorporating Constraints in Inference. However, although incorporating constraints in training with multi-task learning can reduce the invalid predictions, it can not avoid this issue completely. Thus, during inference, we impose hard constraints to ensure that the prediction at each time step is valid.

More precisely, we directly regard $\mathbf{Q}_t$ as binary masks to set the probabilities of all the invalid *pointer indexes* to zero by Hadamard product:

$$\hat{\mathbf{P}}_t = \mathbf{Q}_t \odot \mathbf{P}_t \tag{10}$$

where $\hat{\mathbf{P}}_t \in \mathbb{R}^{n_x+n_c+n_r}$ is the constrained pointer distribution. Finally, we use this constrained pointer distribution instead of $\mathbf{P}_t$ to predict the target sequence.

PD	TS	I^x	I^c	I^r
$\mathbf{Q}_1$	3	[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]	[0,0,0]	[0,0]
$\mathbf{Q}_2$	5	[0,0,0,1,1,1,1,1,1,1,1,1,1,1,1]	[0,0,0]	[0,0]
$\mathbf{Q}_3$	C.	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]	[1,1,1]	[0,0]
$\mathbf{Q}_4$	10	[1,1,0,0,0,1,1,1,1,1,1,1,1,1,1]	[0,0,0]	[0,0]
$\mathbf{Q}_5$	14	[0,0,0,0,0,0,0,0,0,0,1,1,1,1,1]	[0,0,0]	[0,0]
$\mathbf{Q}_6$	P.	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]	[1,1,1]	[0,0]
$\mathbf{Q}_7$	S.	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]	[0,0,0]	[1,1]

Table 1: Proxy distributions (PD) for the example given in Figure 1, whose target sequence (TS) is $[3, 5, Claim, 10, 14, Premise, Support]$. I^x, I^c and I^r respectively denote the range of the *token indexes*, *AC category indexes* and the *AR category indexes*. C., P. and S. are the abbreviations for *Claim*, *Premise* and *Support*.

4.3 Reconstructed Positional Encoding

Similar to the findings in Zhang et al. (2022), we argue that there are order biases in the basic model described in Section 4.1 due to the autoregressive generation paradigm. In particular, the order of tuples in the target sequence Y is fixed, but there are actually no order relations among these tuples. Therefore, when the basic model generates the target sequence, the tuples that have been generated can have undesired effects on the tuples that are currently being generated. Intuitively, the positional encoding (PE) of BART's decoder is closely related to the order biases, since it represents the order information of the target sequence. Thus, to alleviate this issue, we propose to replace the original PE scheme in the BART decoder with a reconstructed positional encoding (RPE) scheme.

Original PE of BART's decoder. We denote the original position index for the target sequence Y as $Y^p = [1, 2, \ldots, |Y|]$, where each position index will be transformed into a positional embedding vector by the BART's embedding layer.

Reconstruction of Original PE. We substitute the original position indexes Y^p with $\hat{Y}^p = [T^p_1, T^p_2, \ldots, T^p_{|S|}]$, where $T^p_i = [1, 1, 2, 1, 1, 2, 2]$[2] represent the position index sequence of the i-th tuple $T_i = [s^{tc}_i, e^{tc}_i, c^{tc}_i, s^{sc}_i, e^{sc}_i, c^{sc}_i, r_i]$ in the target sequence. The rationale behind this design is two-fold: 1) From the intra-tuple perspective, for each tuple, we set an identical position index for all span-related elements (i.e. s^{sc}_i, e^{sc}_i, s^{tc}_i and e^{tc}_i) and another identical position index for all category-related elements (i.e. c^{sc}_i, c^{tc}_i and r_i). This enables

[2]We explore multiple reconstruction methods in our experiments and find that this strategy yields the best results. See Section 6.4 for details.

the model to better learn the difference between the two kinds of elements. 2) From the inter-tuple perspective, unlike the original positional encoding scheme where each tuple has a unique position index sequence, we assign an identical position index sequence (i.e. $[1, 1, 2, 1, 1, 2, 2]$) to all tuples. In this way, the order information among different tuples existing in the original positional encoding scheme can be eliminated, thus reducing the effect caused by the order bias.

5 Experimental Setups

5.1 Datasets

We evaluate our proposed model on two public AM benchmarks, that is, Argument Annotated Essays (AAE) (Stab and Gurevych, 2017) and Consumer Debt Collection Practices (CDCP) (Niculae et al., 2017).

The AAE benchmark consists of 402 persuasive essays annotated with three types ACs (*MajorClaim*, *Claim*, *Premise*) and four types of ARs (*Support*, *Attack*, *For*, *Against*). Note that, in our experiments, we respectively convert *For* and *Against* to *Support* and *Attack* according to the stance polarity.[3] The AC and AR category label lists of AAE are $L^c = [MajorClaim, Claim, Premise]$ and $L^r = [Support, Attack]$. Each essay in AAE contains several paragraphs, and there are 1,833 paragraphs in total (369 paragraphs are reserved for testing). Moreover, AAE is a tree structured benchmark, where ACs and ARs are constrained to form one or more directed trees within each paragraph.

The CDCP benchmark consists of 731 argumentative user comments about rule proposals, and 150 of them are held out for testing. In this benchmark, there are five types of ACs and two types of ARs with the category label lists $L^c = [Fact, Testimony, Value, Policy, Reference]$ and $L^r = [Reason, Evidence]$. Unlike the AAE benchmark, CDCP is a non-tree structured benchmark, where ACs and ARs in a comment can form a directed graph.

5.2 Baselines

We compare our proposed model with following baselines:

- **ILP:** A feature-based approach which jointly optimizes the subtasks of AM by Integer Linear Programming (ILP) (Persing and Ng, 2016; Eger et al., 2017).
- **LSTM-Parser:** A neural dependency parser-based on stack LSTM, which is proposed by Dyer et al. (2015) and is applied to the end-to-end AM task in Eger et al. (2017).
- **LSTM-ER:** An end-to-end relation extraction model combining both tree-structured and sequential LSTM (Miwa and Bansal, 2016), which is adapted for extracting argument structure by Eger et al. (2017).
- **BiPAM:** Another dependency parsing-based model for end-to-end AM, which is based on a biaffine neural network (Ye and Teufel, 2021). Note that, this model use BERT-Base (Devlin et al., 2019) as base model, which have a similar number of parameters with the BART-Base model we adopted.
- **BiPAM-syn:** The BiPAM model enhanced by explicit syntactic information produced by the Stanford syntactic dependency parser (Manning et al., 2014), which is the current state-of-the-art method.
- **BART-B:** The basic model described in Section 4.1, which is similar to the model in (Yan et al., 2021a).

5.3 Evaluation Metrics

Following previous works (Persing and Ng, 2016; Eger et al., 2017; Ye and Teufel, 2021), we employ micro F1 score to evaluate both the ACI (**C-F1**) and ARI (**R-F1**) task.

More precisely, for ACI, the true positive for calculating the C-F1 score is defined as the number of the predicted ACs that exactly match a gold standard AC, i.e., their boundaries and AC category labels are exactly the same. Similarly, for ARI, the true positive for calculating the R-F1 score is defined as the number of the predicted ARs that exactly match a gold standard AR, i.e., their source ACs, target ACs, and AR category labels are all identical.

5.4 Implementation Details

Following Ye and Teufel (2021), for the AAE benchmark, we train our model on the paragraph level since most ARs are within a single paragraph.

[3] Since *For*/*Against* only exists between *Claim* and *MajorClaim*, the predicted *Support*/*Attack* between *Claim* and *MajorClaim* can be easily reconverted to *For*/*Against* by post-processing.

Dataset	Methods	C-F1	R-F1
AAE	ILP	62.61	34.74
	LSTM-Parser	58.86	35.63
	LSTM-ER	70.83	45.52
	BiPAM	72.90	45.90
	BiPAM-syn	73.50	46.40
	BART-B	73.61	47.93
	Ours	**75.94**	**50.08**
CDCP	BiPAM*	41.15	10.34
	BART-B	56.15	13.76
	Ours	**57.72**	**16.57**

Table 2: Experimental results with baselines. The best scores are in bold. * denotes our implementations.

Dataset	Methods	C-F1	R-F1
AAE	Ours	**75.94**	**50.08**
	w/o RPE	74.27	48.22
	w/o CPMT	75.39	49.27
	w/o CPMI	75.33	49.36
	w/o CPM	74.07	48.28
CDCP	Ours	57.72	**16.57**
	w/o RPE	**58.13**	15.11
	w/o CPMT	57.11	15.14
	w/o CPMI	56.06	15.70
	w/o CPM	55.95	14.67

Table 3: The results of ablation experiments. RPE denotes the reconstructed positional encoding strategy. CPMT and CPMI are the abbreviation for the constrained pointer mechanism in training and inference stage, respectively.

For both AAE and CDCP benchmarks, we randomly choose 15% of the training set for validation. All experiments are conducted ten times with different random initializations, and the average scores are reported. Notably, there are few isolated ACs that are not involved in any AR. For such ACs, we introduce a special "none" token to construct a pseudo tuple. For example, if the *Claim* in Figure 1 is an isolated AC, then its corresponding pseudo tuple is $[3, 5, Claim, none, none, none, none]$.

We use the pre-trained BART-Base as the base model. The learning rate is set to 5e-5 and 8e-5 for AAE and CDCP benchmarks, respectively. AdamW (Loshchilov and Hutter, 2019) is used for optimization with ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-8$). We use a warm-up strategy with the warm-up ratio set to 0.01. Each MLP contains 2 layers with a hidden size of 768. Moreover, the dropout rate is set to 0.3 and the batch size is set to 32. Following Yan et al. (2021a), the hyperparameter α is set to 0.5, and beam search is used for decoding during inference with a beam size of 4. We train our model 75 epochs and select the best checkpoint base on the average of C-F1 and R-F1 on the validation set.

6 Results and Analysis

6.1 Main Results

Table 2 shows the overall performance of the baselines and our proposed model. Our model significantly ($p < 0.01$) outperforms the BiPAM-syn model by at least 2.44% and 3.68% on the C-F1 and R-F1 scores, respectively, achieving state-of-the-art performance on the AAE benchmark. On the CDCP benchmark, our model also outperforms the BiPAM by a large margin ($p < 0.01$) . Also, the basic BART model with the vanilla pointer mechanism (BART-B) can already surpass the current state-of-the-art model, BiPAM-syn, indicating that it might be more appropriate to formalize the end-to-end AM task as a generation task instead of a dependency parsing task. The performance of BART-B can be further improved by introducing our proposed RPE and CPM. In addition, it is worth noting that both LSTM-ER and BiPAM-syn are enhanced with explicit syntactic information, while our model does not require any other information except the input text and is still able to achieve significantly better results.

6.2 Ablation Study

We perform ablation experiments to reveal the effect of each module in our model on both the AAE and CDCP benchmarks. The results are shown in Table 3. Overall, all of our proposed strategies can bring performance improvements. In particular, applying the RPE to our generative model contributes about 1.86% and 1.46% R-F1 scores on AAE and CDCP, respectively, showing the effectiveness of our proposed RPE for alleviating the order biases. Surprisingly, RPM can slightly decrease the C-F1 score on CDCP. One big factor for this is that some samples in CDCP contain too many ARs, resulting in very long target sequences, and RPM may be unfavorable to capture such long-term dependency. Also, we can observe that, in both the training and inference stages, CPM contributes significantly on the model performance. Concretely, either removing CPMT or CPMI causes performance degrada-

Methods	AAE			CDCP		
	Length	Order	Overlap	Length	Order	Overlap
Ours	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
w/o CPMI	0.5%	1.9%	3.3%	1.3%	5.2%	11.2%
w/o CPM	0.9%	3.1%	4.5%	2.3%	5.4%	12.3%
w/o CPM RPE	0.2%	5.9%	7.2%	1.5%	6.8%	13.6%

Table 4: Invalid predictions analysis on the test set.

ID	RPE	AAE		CDCP	
		C-F1	R-F1	C-F1	R-F1
1	[1,1,2,1,1,2,2]	**75.94**	**50.08**	57.71	**16.46**
2	[1,1,2,1,1,2,3]	75.38	49.66	58.18	16.23
3	[1,2,3,4,5,6,7]	75.84	49.16	57.29	15.03
4	[1,2,3,1,2,3,3]	75.32	49.01	57.77	16.36
5	[1,2,3,1,2,3,4]	75.33	48.91	**58.42**	16.04
6	iden.	74.83	48.78	56.43	15.31
7	iden. w/ dist.	74.93	48.82	56.61	16.03
8	original	74.07	48.22	58.13	15.11

Table 5: Effect of different RPE. Here, "original" denotes using original positional encoding of BART decoder. "iden." denotes the positional embedding of each token in the target sequence is identical. "iden. w/ dist." denotes that the positional embeddings are identical within each tuple, but distinct among tuples. Each list in row 1-5 indicates the position index sequence for each tuple, which is identical among tuples.

tion. Further, removing both of them (*w/o* CPM) results in further decreases, showing that CPMT and CMPI can collaborate properly with each other to gain more performance improvement.

6.3 Invalid Predictions Analysis

According to Yan et al. (2021a,b), adapting generative models to IE tasks suffers from the issue of invalid prediction, since the generation of the target sequences is not fully controllable. Thus, to better demonstrate why our proposed RPE and CPM work, we carry out a detailed analysis of the invalid predictions. For our end-to-end AM task, we define three types of invalid predictions: 1) **Invalid Length:** The length of a valid tuple should be 7. 2) **Invalid Order:** In each tuple, the start index of an AC must be smaller than its end index. 3) **Invalid Overlap:** The predicted source and target AC spans in each tuple should not overlap with each other.

For each predicted target sequence in the test set, we consider it as an invalid prediction if it contains one of the aforementioned invalid types. The percentage of each type of invalid prediction is shown in Table 4. Our proposed model can avoid all the invalid predictions because of the hard constraints imposed by CPMI. Compared to removing both CPMI and CPMT (*w/o* CPM), only applying constraints by multi-task learning during training (*w/o* CPMI) results in fewer invalid predictions. By comparing (*w/o* CPM) and (*w/o* CPM & RPE), we discover that our proposed RPE can reduce the invalid order and invalid overlap predictions. We argue that the order biases can disrupt the decoding process, causing the model to produce more invalid predictions, whereas RPE can alleviate this issue. However, RPE can increase the invalid length predictions, probably because the original PE with strong order biases is more favorable for controlling the prediction length.

6.4 Effect of Different RPE

To find an appropriate positional encoding scheme for the decoder of our model, we explored various methods. As shown in Table 5, we can conclude that it is better to assign one same position index to all span-related elements (i.e. s_i^{sc}, e_i^{sc}, s_i^{tc} and e_i^{tc}), and set another position index to all category-related elements (i.e. c_i^{sc}, c_i^{tc} and r_i) (row 1). This is due to the fact that the span-related elements and the category-related elements have intrinsically different meanings, where the former are used to recognize the locations and boundaries of ACs, and the latter are used to identify the categories of ACs and ARs. This fact can also be confirmed in row 3 and row 6, where setting distinct or identical position index for all elements both decreases the performance. In addition, using different position index patterns among tuples (row 7 and row 8) does not work well, which further confirms the negative effect caused by the order biases between tuples. Not surprisingly, keeping the original PE of BART (row 8) achieves the worst results due to the order biases. We use $[1, 1, 2, 1, 1, 2, 2]$ as our final RPM since it yields the best R-F1 scores on both AAE

and CDCP.

7 Conclusion

In this paper, we transform the end-to-end AM task into a generation task and apply the pre-trained BART model with a pointer mechanism to solve it. To better adapt this generative model to the end-to-end AM task, we replace the original positional encoding of the BART decoder with our proposed reconstructed positional encoding. On the other hand, we present a constrained pointer mechanism to further improve our model, which is achieved by multi-task learning during training and constrained decoding during inference. The extensive experimental results and detailed analysis demonstrate the superiority of our proposed method.

Limitations

Although our proposed reconstructed positional encoding can alleviate the order biases problem, it may not be eliminated completely because the order of the target sequence is still fixed during training. Therefore, for future work, we plan to explore better approaches to address the order biases issue.

In addition, our model has the problem of generating repetitive tuples. Although this does not affect the performance, it can increase the inference time. Therefore, we will also investigate methods to mitigate the repetitive generation problem in future work.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China 62006062 and 62176076, Shenzhen Foundational Research Funding JCYJ20200109113441941, JCYJ20210324115614039, the Major Key Project of PCL2021A06, Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies 2022B1212010005.

References

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

Jianzhu Bao, Chuang Fan, Jipeng Wu, Yixue Dang, Jiachen Du, and Ruifeng Xu. 2021. A neural transition-based model for argumentation mining. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 6354–6364. Association for Computational Linguistics.

Roy Bar-Haim, Yoav Kantor, Lilach Eden, Roni Friedman, Dan Lahav, and Noam Slonim. 2020. Quantitative argument summarization and beyond: Cross-domain key point analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 39–49. Association for Computational Linguistics.

Filip Boltuzic and Jan Snajder. 2014. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argument Mining, hosted by the 52nd Annual Meeting of the Association for Computational Linguistics, ArgMining@ACL 2014, June 26, 2014, Baltimore, Maryland, USA*, pages 49–58. The Association for Computer Linguistics.

Artem N. Chernodub, Oleksiy Oliynyk, Philipp Heidenreich, Alexander Bondarenko, Matthias Hagen, Chris Biemann, and Alexander Panchenko. 2019. TARGER: neural argument mining at your fingertips. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 3: System Demonstrations*, pages 195–200. Association for Computational Linguistics.

Oana Cocarascu and Francesca Toni. 2017. Identifying attack and support argumentative relations using deep learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1374–1379. Association for Computational Linguistics.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 1835–1845. Association for Computational Linguistics.

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

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, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 484–490. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 334–343. The Association for Computer 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, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 11–22. Association for Computational Linguistics.

Alexander R. Fabbri, Faiaz Rahman, Imad Rizvi, Borui Wang, Haoran Li, Yashar Mehdad, and Dragomir R. Radev. 2021. Convosumm: Conversation summarization benchmark and improved abstractive summarization with argument mining. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 6866–6880. Association for Computational Linguistics.

Hao Fei, Donghong Ji, Bobo Li, Yijiang Liu, Yafeng Ren, and Fei Li. 2021. Rethinking boundaries: End-to-end recognition of discontinuous mentions with pointer networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12785–12793. AAAI Press.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Transition-based semantic dependency parsing with pointer networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7035–7046. Association for Computational Linguistics.

Eirini Florou, Stasinos Konstantopoulos, Antonis Koukourikos, and Pythagoras Karampiperis. 2013. Argument extraction for supporting public policy formulation. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities, LaTeCH@ACL 2013, August 8, 2013, Sofia, Bulgaria*, pages 49–54. The Association for Computer Linguistics.

Debanjan Ghosh, Aquila Khanam, Yubo Han, and Smaranda Muresan. 2016. Coarse-grained argumentation features for scoring persuasive essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics.

Debanjan Ghosh, Smaranda Muresan, Nina Wacholder, Mark Aakhus, and Matthew Mitsui. 2014. Analyzing argumentative discourse units in online interactions. In *Proceedings of the First Workshop on Argument Mining, hosted by the 52nd Annual Meeting of the Association for Computational Linguistics, ArgMining@ACL 2014, June 26, 2014, Baltimore, Maryland, USA*, pages 39–48. The Association for Computer Linguistics.

Theodosios Goudas, Christos Louizos, Georgios Petasis, and Vangelis Karkaletsis. 2014. Argument extraction from news, blogs, and social media. In *Artificial Intelligence: Methods and Applications - 8th Hellenic Conference on AI, SETN 2014, Ioannina, Greece, May 15-17, 2014. Proceedings*, volume 8445 of *Lecture Notes in Computer Science*, pages 287–299. Springer.

Xinyu Hua, Zhe Hu, and Lu Wang. 2019. Argument generation with retrieval, planning, and realization. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2661–2672. Association for Computational Linguistics.

Kuo Yu Huang, Hen-Hsen Huang, and Hsin-Hsi Chen. 2021. HARGAN: heterogeneous argument attention network for persuasiveness prediction. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13045–13054. AAAI Press.

Yohan Jo, Jacky Visser, Chris Reed, and Eduard H. Hovy. 2019. A cascade model for proposition extraction in argumentation. In *Proceedings of the 6th Workshop on Argument Mining, ArgMining@ACL 2019, Florence, Italy, August 1, 2019*, pages 11–24. Association for Computational Linguistics.

Khalid Al Khatib, Lukas Trautner, Henning Wachsmuth, Yufang Hou, and Benno Stein. 2021. Employing argumentation knowledge graphs for neural argument generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4744–4754. Association for Computational Linguistics.

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 Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4691–4698. Association for Computational Linguistics.

John Lawrence and Chris Reed. 2019. Argument mining: A survey. *Comput. Linguistics*, 45(4):765–818.

Mirko Lenz, Premtim Sahitaj, Sean Kallenberg, Christopher Coors, Lorik Dumani, Ralf Schenkel, and Ralph Bergmann. 2020. Towards an argument mining pipeline transforming texts to argument graphs. In *Computational Models of Argument - Proceedings of COMMA 2020, Perugia, Italy, September 4-11, 2020*, volume 326 of *Frontiers in Artificial Intelligence and Applications*, pages 263–270. IOS Press.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Jialu Li, Esin Durmus, and Claire Cardie. 2020. Exploring the role of argument structure in online debate persuasion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8905–8912. Association for Computational Linguistics.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 894–908. Association for Computational Linguistics.

Marco Lippi and Paolo Torroni. 2015. Context-independent claim detection for argument mining. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 185–191. AAAI Press.

Linlin Liu, Xiang Lin, Shafiq R. Joty, Simeng Han, and Lidong Bing. 2019. Hierarchical pointer net parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1007–1017. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard H. Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1403–1414. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, pages 55–60. The Association for Computer Linguistics.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 319–328. The Association for Computational Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. 2007. Automatic detection of arguments in legal texts. In *The Eleventh International Conference on Artificial Intelligence and Law, Proceedings of the Conference, June 4-8, 2007, Stanford Law School, Stanford, California, USA*, pages 225–230. ACM.

Gaku Morio, Hiroaki Ozaki, Terufumi Morishita, Yuta Koreeda, and Kohsuke Yanai. 2020. Towards better non-tree argument mining: Proposition-level biaffine parsing with task-specific parameterization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3259–3266. Association for Computational Linguistics.

Huy Nguyen and Diane J. Litman. 2015. Extracting argument and domain words for identifying argument components in texts. In *Proceedings of the 2nd Workshop on Argumentation Mining, ArgMining@HLT-NAACL 2015, June 4, 2015, Denver, Colorado, USA*, pages 22–28. The Association for Computational Linguistics.

Huy V. Nguyen and Diane J. Litman. 2018. Argument mining for improving the automated scoring of persuasive essays. In *Proceedings of the Thirty-Second*

AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 5892–5899. AAAI Press.

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, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 985–995. Association for Computational Linguistics.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *The 12th International Conference on Artificial Intelligence and Law, Proceedings of the Conference, June 8-12, 2009, Barcelona, Spain*, pages 98–107. ACM.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Andreas Peldszus. 2014. Towards segment-based recognition of argumentation structure in short texts. In *Proceedings of the First Workshop on Argument Mining, hosted by the 52nd Annual Meeting of the Association for Computational Linguistics, ArgMining@ACL 2014, June 26, 2014, Baltimore, Maryland, USA*, pages 88–97. The Association for Computer Linguistics.

Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1384–1394. The 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, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1364–1373. Association for Computational Linguistics.

Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2019. Classification and clustering of arguments with contextualized word embeddings. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 567–578. Association for Computational Linguistics.

Liliang Ren, Chenkai Sun, Heng Ji, and Julia Hockenmaier. 2021. Hyspa: Hybrid span generation for scalable text-to-graph extraction. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 4066–4078. Association for Computational Linguistics.

Ramon Ruiz-Dolz, José Alemany, Stella Heras Barberá, and Ana García-Fornes. 2021. Transformer-based models for automatic identification of argument relations: A cross-domain evaluation. *IEEE Intell. Syst.*, 36(6):62–70.

Robin Schaefer and Manfred Stede. 2021. Argument mining on twitter: A survey. *it Inf. Technol.*, 63(1):45–58.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083. Association for Computational Linguistics.

Noam Slonim, Yonatan Bilu, Carlos Alzate, Roy Bar-Haim, Ben Bogin, Francesca Bonin, Leshem Choshen, Edo Cohen-Karlik, Lena Dankin, Lilach Edelstein, et al. 2021. An autonomous debating system. *Nature*, 591(7850):379–384.

Wei Song, Ziyao Song, Lizhen Liu, and Ruiji Fu. 2020. Hierarchical multi-task learning for organization evaluation of argumentative student essays. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3875–3881. ijcai.org.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 46–56. ACL.

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

Eva Maria Vecchi, Neele Falk, Iman Jundi, and Gabriella Lapesa. 2021. Towards argument mining for social good: A survey. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 1338–1352. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Hao Wang, Zhen Huang, Yong Dou, and Yu Hong. 2020. Argumentation mining on essays at multi scales. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 5480–5493. International Committee on Computational Linguistics.

Hang Yan, Junqi Dai, Tuo Ji, Xipeng Qiu, and Zheng Zhang. 2021a. A unified generative framework for aspect-based sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2416–2429. Association for Computational Linguistics.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021b. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 5808–5822. Association for Computational Linguistics.

Songlin Yang and Kewei Tu. 2022. Bottom-up constituency parsing and nested named entity recognition with pointer networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 2403–2416. Association for Computational Linguistics.

Yuxiao Ye and Simone Teufel. 2021. End-to-end argument mining as biaffine dependency parsing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 669–678. Association for Computational Linguistics.

Shuai Zhang, Yongliang Shen, Zeqi Tan, Yiquan Wu, and Weiming Lu. 2022. De-bias for generative extraction in unified NER task. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 808–818. Association for Computational Linguistics.

Wenxuan Zhang, Yang Deng, Xin Li, Yifei Yuan, Lidong Bing, and Wai Lam. 2021a. Aspect sentiment quad prediction as paraphrase generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 9209–9219. Association for Computational Linguistics.

Wenxuan Zhang, Xin Li, Yang Deng, Lidong Bing, and Wai Lam. 2021b. Towards generative aspect-based sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021*, pages 504–510. Association for Computational Linguistics.