End-to-End Argument Mining through Autoregressive Argumentative Structure Prediction

Nilmadhab Das, Vishal Vaibhav*, Yash Sunil Choudhary, V. Vijaya Saradhi, Ashish Anand
Applied Machine Learning (AMaL) Lab
Department of Computer Science and Engineering, Indian Institute of Technology (IIT), Guwahati, India
North Eastern Regional Institute of Science and Technology (NERIST), Arunachal Pradesh, India*
{nilmadhabdas, y.choudhary, saradhi, anand.ashish}@iitg.ac.in, 424115@nerist.ac.in*

***Abstract*—Argument Mining (AM) helps in automating the extraction of complex argumentative structures such as Argument Components (ACs) like *Premise, Claim* etc. and Argumentative Relations (ARs) like *Support, Attack* etc. in an argumentative text. Due to the inherent complexity of reasoning involved with this task, modelling dependencies between ACs and ARs is challenging. Most of the recent approaches formulate this task through a generative paradigm by flattening the argumentative structures. In contrast to that, this study jointly formulates the key tasks of AM in an end-to-end fashion using Autoregressive Argumentative Structure Prediction (AASP) framework. The proposed AASP framework is based on the autoregressive structure prediction framework that has given good performance for several NLP tasks. *AASP* framework models the argumentative structures as constrained pre-defined sets of actions with the help of a conditional pre-trained language model. These actions build the argumentative structures step-by-step in an autoregressive manner to capture the flow of argumentative reasoning in an efficient way. Extensive experiments conducted on three standard AM benchmarks demonstrate that AASP achieves state-of-the-art (SoTA) results across all AM tasks in two benchmarks and delivers strong results in one benchmark.**

***Index Terms*—Argument Mining, Computational Argumentation, End-to-End System**

I. Introduction

Argument mining (AM) focuses on extracting argumentative structures from discourse dynamics. It has applications in automated essay scoring [1], legal decision support [2], healthcare [3], etc. Early studies modelled AM as dependency parsing [4], [5]. This approach involved intricate post-processing steps to match gold dependency graphs. Later in [6], the authors proposed single-task and multi-task settings with a *biaffine-attention* module [7] to model relational structures efficiently. However, comparing each AC with every other AC led to class imbalance issues due to the majority of mappings being unrelated. Generative methods like [8] used flattened sequences but struggled to handle relational dependencies efficiently. In [9], the authors reformulated AM through a *Structured Prediction* task-adaptation to perform text-to-text generation using *Augmented Natural Language (ANL)* [10]. However, this approach required repeated AC spans, making it less efficient for complex structures with many relations.

Modelling argumentative relations is more complex than handling components, as structural representations vary across domains. Some datasets use tree structures for ARs [11], while others use non-tree-structured representations [12]. In [13], the authors noted that argumentative structures resemble answers to *why* questions, forming reasoning chains *(a series of premises/claims)* leading to conclusions. Sequence-like approaches [8] or flattened-text approaches [9] often fail to capture these complexities. This limitation affects performance in modeling relational structures when tackling all AM tasks jointly.

There are four key tasks of AM: *(i) Argument Component Identification (**ACI**)*, identifying argumentative text spans; *(ii) Argument Component Classification (**ACC**)*, categorizing spans as claims or premises; *(iii) Argumentative Relation Identification (**ARI**)*, detecting links between spans; and *(iv) Argumentative Relation Classification (**ARC**)*, determining the type of these links (e.g., support or attack). Prior work approached these tasks independently [14] or partially [15]. Recently, *"End-to-End"* systems emerged to solve all tasks jointly [16].

Recently, the *Autoregressive Structured Prediction (ASP) framework* [17] has achieved strong performance in several structured prediction tasks like NER, co-reference resolution, and relation extraction. It does so through a constrained set of structure-building actions with a conditional language model. The ASP framework has been tested on foundational NLP tasks only. This framework, though achieved convincing performance, has not been applied in complex reasoning tasks such as AM.

ASP framework is applicable in AM tasks as the standard datasets of AM are having tree or graph structures. We note the following differences between foundational structured prediction tasks and AM tasks: **(I)** The target output spans in these foundational tasks are relatively shorter. In contrast, AM tasks involve longer spans, where each AC often spans across extended sections of text. **(II)** In relation extraction or co-reference resolution, the distance between two related tokens (or entities) is typically smaller. On the other hand, AM tasks often involve related arguments that can span large distances. For example, one argument may be located at the beginning

This paper has been accepted for publication at the IEEE/INNS International Joint Conference on Neural Networks (IJCNN) 2025.

*The author carried out this research during an internship at IIT Guwahati.

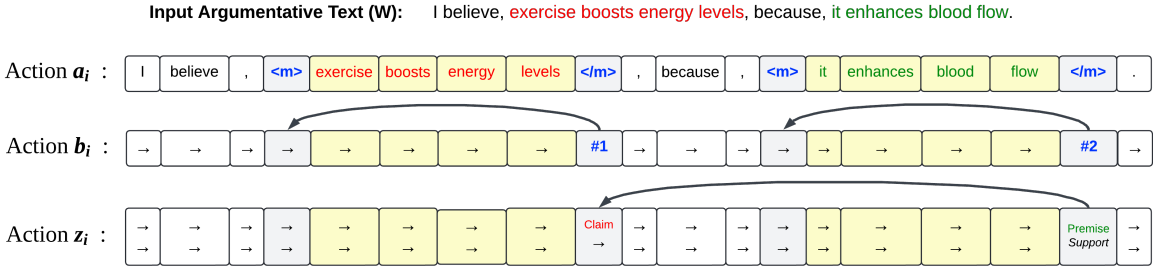

Fig. 1: Illustration of the decoding process in the AASP framework for end-to-end argument mining. The process involves three main actions step-by-step: (1) Action a_i constructs the target sequence for span-identifying actions; (2) Action b_i finalizes the AC span boundaries by matching the <m> and </m> tokens; and (3) Action z_i connects related ACs by linking their $< /m >$ tokens and classifies both the AC types and the types of their ARs.

of a paragraph, while the other is at the end, with multiple intermediate ACs. These distinctions make it a meaningful and promising avenue to investigate the capability of the ASP framework in addressing AM tasks.

In this paper, we propose to use the ASP framework in solving all four key AM tasks and demonstrate its success for the ***End-to-End*** AM framework. We name it as ***Autoregressive Argumentative Structure Prediction (AASP)***. To accommodate longer spans and distantly related ACs, we modify the feed-forward networks of the original ASP framework to a larger size. This modification is the key component in adaption the ASP framework to handle varying argumentative relational structures present in different AM corpora. Through AASP, we exploit the autoregressive phenomenon of a conditional language model for the AM task-specific constrained set of actions, including *(i) Span-Identifying Actions*, which involve the identification of AC spans; *(ii) Boundary-Pairing Actions*, which finalize the span boundaries; and *(iii) Type-Labeling Actions*, which classify AC types, connect related ACs, and determine their AR types.

Training a conditional language model to predict these structure-building actions exposes the argumentative structure such a way that it allows the model to capture the dependencies within the ACs better. This explicit modelling contributes to the learning of intra-structure relationships between ACs by utilizing the semantic and contextual knowledge of the language model. Experimenting with three standard structurally distinct AM benchmarks, AASP achieves SoTA results in two benchmarks and produces higher Micro-F1 scores on relational AM tasks in the rest of the benchmark. In summary, the key contributions of this paper are:

1) We propose *Autoregressive Argumentative Structure Prediction* **(AASP)**, an end-to-end framework that jointly solves all four key tasks of AM using a conditional language model.
2) By adapting the ASP framework for the AM tasks, we solve the ACI task using *Span-Identifying* and *Boundary-Pairing* actions, and the ACC, ARI and ARC tasks using the *Type-Labeling* actions.
3) Utilizing larger FFNs to execute all these actions enhances the ability to handle longer AC spans and distantly related ACs, leading to performance improvements, as demonstrated in the *Ablation Study*.
4) We conduct extensive experiments across three diverse AM benchmarks, demonstrating the robust performance of AASP in all four key AM tasks. Our results further highlight that AASP handles relational dependencies better, thereby capturing the *chain of reasoning* more effectively as a byproduct.

II. Related Work

While most previous studies have concentrated on the single [18] or the subset of the four key tasks of AM in a pipelined setup [19], recent efforts have shifted toward joint modelling of these tasks in an end-to-end manner. Among the earlier joint approaches, *Persing et al.* [20] first employed joint modelling of ACC and ARC tasks using Integer Linear Programming (ILP) for joint inference to mitigate error propagation between tasks. Later, *Eger et al.* [4] explored various joint reformulations of ACC and ARC tasks using LSTM architecture, including sequence tagging, dependency parsing, multi-task tagging, and relation extraction. With the advancement of the NLP landscape, *Ye et al.* [5] first proposed a biaffine dependency parsing-based approach coupled with the rich representations of pre-trained language models to solve ACC and ARC tasks jointly. A significant limitation in the AM literature is the scarcity of datasets, which was highlighted in [6], where the authors proposed a cross-corpora multi-task approach using a span-biaffine architecture with Longformer [21] to first solve all four key tasks in an end-to-end setup. In this formulation, a span classifier generates BIO tags for spans, and average pooling is used to create span representations. Later, several generative frameworks also emerged to model argumentative structures as a generation of predefined targets. Within this paradigm, *Bao et al.* [8] framed the AM task as a *text-to-sequence* generation problem, producing a structured sequence comprising AC and AR types along with the start and end indices of AC spans. Later, the TANL framework [10] was recently adopted by *Kawarada et al.* [9], framing end-to-end AM as a text-to-text generation task to model ACC and ARC tasks jointly. More recently, *Sun et al.* [16] also proposed a generative approach with discourse structure-aware prefix

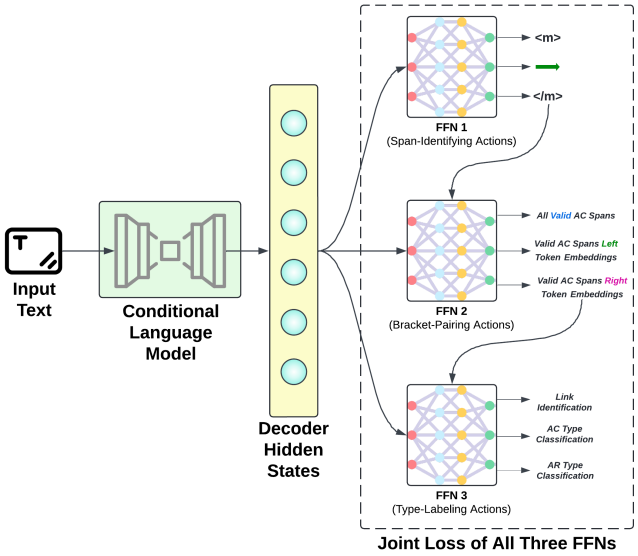

Fig. 2: Model Architecture of AASP

using BART [22] to model all four key tasks with task-specific prompts.

III. Proposed Method

We define the argumentative paragraph as, $W = w_1, w_2, w_3, \ldots, w_n$, where n is the total number of tokens in W. We denote a text span from w_i to w_j in W as $w_{i:j}$. Among the four key AM tasks, *ACI* aims to identify the set of argumentative spans, $C_{ACI} = \{(s_i, e_i)\}_{i=1}^{n}$, where each span $w_{s_i:e_i}$ corresponds to a potential AC. Second, *ACC* assigns a type t_i (e.g., Premise, Claim) to each identified span $(s_i, e_i) \in C_{ACI}$, resulting in $C_{ACC} = \{(t_i, s_i, e_i)\}_{i=1}^{n}$. Third, *ARI* detects the set of relations $R_{ARI} = \{(h_j, t_j)\}_{j=1}^{m}$, where (h_j, t_j) represents a connection between two ACs, with h_j denoting the head AC (the source of the relation) and t_j denoting the tail AC (the target of the relation), where $h_j, t_j \in C_{ACC}$. Finally, *ARC* categorizes each identified relation $(h_j, t_j) \in R_{ARI}$ into a specific type r_j (e.g., Support, Attack), yielding $R_{ARC} = \{(h_j, t_j, r_j)\}_{j=1}^{m}$.

Through the *AASP framework*, we reformulate all these AM tasks as the prediction of a sequence of actions, where we construct the argumentative structure step-by-step with a conditional language model as given in Fig. 1. Building on the original formulation proposed in [17], ASP formulation is adapted upon the four key tasks of AM for *(i) Construction of action sequences* and *(ii) Conditional modelling of actions*. We discuss the details in subsequent sections to maintain the coherency.

A. Construction of Action Sequences

Given W, the goal is to predict a sequence of tuples $\langle span_i, boundary_i, type\text{-}label_i \rangle$.

Tuples denote action space. If the tuple $\langle \text{span}_i, \text{boundary}_i, \text{type-label}_i \rangle$ is denoted as y_i, then the goal is to predict the sequence $y_1, y_2, \ldots, y_N$ and the action space Y_i is given as,

$$Y_i = A \times B_i \times Z_i \tag{1}$$

Components of action spaces are described below.

Span-Identifying Actions:

These actions take symbols from the set $A = \{\langle m \rangle, \rightarrow, \langle /m \rangle\}$, where:

- $\langle m \rangle$ marks the beginning of an AC span,
- $\langle /m \rangle$ marks the end of the span, and
- $\rightarrow$ copies the input text as it appears from left to right, irrespective of whether it is an AC or Non-AC span.

These actions are performed using FFN 1 of Fig. 2.

Boundary-Pairing Actions: These actions denoted with set B_n given as,

$$B_n = \{i \mid i < j \wedge a_i = \langle m \rangle \wedge a_j = \langle /m \rangle \ \& \ i, j \in \{1, \ldots, n\}\} \tag{2}$$

Boundary-pairing actions connect the beginning of an AC span with the end of that AC span; that is, they connect the start action $\langle m \rangle$ with the end action $\langle /m \rangle$.

Specifically, the sequence produced by the *Span-Identifying Actions*, is scanned from left to right, identifying each $\langle /m \rangle$ and immediately pairing it with the leftmost unmatched $\langle m \rangle$ before continuing. This way, the spans are resolved incrementally without crossing boundaries, and the unmatched symbols are removed as erroneous predictions. These actions are performed using both FFN 1 & FFN 2 jointly, as shown in Fig. 2.

With the completion of the *Span-Identifying* and *Boundary-Pairing* actions, the *ACI task* is completed.

Type-Labeling Actions: These actions, denoted as Z_n, perform the following: *(i) labelling AC spans into predefined AC types (ACC task)*, and *(ii) jointly establishing the relationships between related ACs and classifying their AR types (ARI & ARC tasks)*. Here, the sequence produced by the *Span-Identifying Actions* is traversed from left to right to find the $\langle /m \rangle$ symbol. Once found, it connects with the previously most related already identified $\langle /m \rangle$ to build the intra-span relationship. Additionally, both the spans corresponding to the $\langle /m \rangle$ symbols are classified into predefined AC types, and the connected link is classified into a predefined AR type. This whole process is performed using both FFN 2 & FFN 3 jointly in Fig. 2. Thus, if we define a set L that combines all AC and AR types, then:

$$Z_n = \{p \mid p < q \wedge a_p = \langle /m \rangle\} \times L \tag{3}$$

B. Conditional Modeling of Actions

Given an input argumentative text W, we transform it into a sequence of tuples denoted by $y = (y_1, y_2, \ldots, y_N)$. This sequence of tuples is modelled as the conditional probability given below:

$$p_\theta(y \mid W) = \prod_{n=1}^{N} p_\theta(y_n \mid y_{<n}, W) \tag{4}$$

Here, each $p_\theta(y_n \mid y_{<n}, W)$ corresponds to the probability of the n-th action given the preceding actions and the input argumentative text W.

Hence, the log-likelihood of the model is then given by:

$$\log p_\theta(y \mid W) = \sum_{n=1}^{N} \log p_\theta(y_n \mid y_{<n}, W) \tag{5}$$

We then calculate the probability value of $p_\theta(y_n \mid y_{<n}, W)$ over softmax of the dynamic set Y_n that changes as a function of the history $y_{<n}$, i.e.,

$$p_\theta(y_n \mid y_{<n}, W) = \frac{\exp s_\theta(y_n)}{\sum_{y'_n \in Y_n} \exp s_\theta(y'_n)} \tag{6}$$

where s_θ is a parameterized score function and is implemented using two independent feed-forward networks (FFN 2 & FFN 3) as:

$$s_\theta(y_n) = \begin{cases} \text{FFN}^{z_n}_{a_n}(p_n) & \text{if } a_n = \langle /m \rangle \\ \text{FFN}_{a_n}(h_n) & \text{otherwise} \end{cases} \tag{7}$$

Here, h_n is the decoder hidden state at step n, represented as a column vector, and $p_n = [h_n^\top; h_{b_n}^\top]^\top$ represents the AC span mention corresponding to y_n. Considering the longer text spans of ACs and the presence of distantly related ACs, we choose to select larger FFNs as compared to the ASP framework to better capture the relational dependencies.

The dynamic vocabulary is used at each time step, and the best-decoded sequence is finalized as y^* using the greedy decoding strategy. At decoding step n, y^*_n is calculated as:

$$y^*_n = \arg\max_{y'_n} p_\theta(y'_n \mid y_{<n}, W) \tag{8}$$

This resultant y^*_n is the best sequence of actions used to construct the argumentative structure for the given W.

IV. Experimental Setup

A. Datasets

We evaluate our approach on three structurally distinct standard AM benchmarks. A brief overview of these datasets is provided below.

Argument Annotated Essay (AAE) [11]: This dataset employs a tree-structured annotation scheme where each AC can have at most one outgoing AR. It includes 402 student essays annotated at the segment (span) level, with each essay comprising multiple paragraphs. In total, there are 1,833 paragraphs, annotated with three types of ACs: *Claim*, *MajorClaim*, and *Premise*, resulting in 6,089 ACs and 3,832 ARs.

Fine-Grained Argument Annotated Essay (AAE-FG) [23]: An extended version of the `AAE` dataset with finer categorization of ACs. The *Claim* and *MajorClaim* categories are divided into *Fact*, *Value*, and *Policy*, while the *Premise* category is split into *Common Ground*, *Testimony*, *Hypothetical Instance*, *Statistics*, *Real Example*, and *Others*. Hence, this version defines nine AC types, with the ARs unchanged from `AAE`.

Consumer Debt Collection Practices (CDCP) [12]: This dataset utilizes a non-tree argumentation scheme, permitting an AC to have multiple outgoing ARs. It consists of 731 user comments sourced from the Consumer Financial Protection Bureau (CFPB) website. The data includes five types of ACs: *Fact*, *Testimony*, *Reference*, *Policy*, and *Value* with two AR types: *Reason* and *Evidence*. In total, the dataset comprises 4,931 ACs and 1,220 ARs.

B. Implementation Details

We employ the *Flan-T5-Large* [24] as the conditional language model across all experiments. For the `AAE` and `AAE-FG` datasets, we use a batch size of 64, while for `CDCP`, the batch size is set to 16. The maximum output sequence length is capped at 256 for `AAE` and `AAE-FG` and at 1024 for `CDCP`. We utilize the AdamW optimizer with a learning rate of 5×10^{-5}. All experiments are conducted on a single NVIDIA A100 GPU. Each experiment is run for 200 epochs, with checkpoints saved every 200 steps. The best model is selected based on performance on the validation set. Results are averaged over three independent runs. Both the feed-forward neural networks include one hidden layer of size 1500 and are trained with a learning rate of 3×10^{-4}. We use the following libraries: (i) ASP framework[1] [17], and (ii) HuggingFace's Transformers[2].

C. Evaluation

Following the prior studies [6], we evaluate all four AM tasks using the *Micro-F1* score. We consider only exact matches of AC spans as correct, disregarding any partial matches to maintain strict alignment with previous works.

D. Baselines

We select the following SoTA models as baselines, comparing them to the datasets where applicable: **BiPAM** [5]: Dependency parsing for end-to-end AM using biaffine operations and BERT-base [25]. **BiPAM-syn** [5]: Extends BiPAM with syntactic information from the Stanford dependency parser. **BART-B** [26]: A generative model for aspect-based sentiment analysis, adapted for AM by [8]. **GMAM** [8]: Frames AM as sequence generation with positional encoding and pointer mechanisms. **ST** [6]: A dependency parser for AM using Longformer [21] and biaffine attention [7]. **ANL-AM** [9]: Adapts the TANL framework [10] for AM with text-to-text generation. **DENIM** [16]: A sequence generation-based AM framework with discourse-aware multi-task prompting.

V. Results and Discussion

A. Main Results and Comparison with Baselines

Table I compares the overall performance of AASP compared to recent baselines across all four AM tasks on the three standard datasets. Our method consistently outperforms existing baselines on the `AAE` and `AAE-FG` datasets, achieving average Micro-F1 score improvements of 3.3% and 3.81%, respectively. These results underscore the effectiveness of

[1]https://github.com/lyutyuh/ASP

[2]https://github.com/huggingface

Dataset	Frameworks	Micro-F1 Scores of Key AM Tasks				AVG
		ACI	ACC	ARI	ARC	
AAE	BiPAM	–	72.90	–	45.90	–
	BiPAM-syn	–	73.50	–	46.40	–
	BART-B	81.71	73.61	49.75	47.93	63.25
	GMAM	84.10	75.94	50.40	50.08	65.13
	ST	85.02	75.43	55.75	55.19	67.84
	ANL-AM	–	76.93	–	58.57	–
	DENIM	85.75	76.50	59.55	58.51	70.08
	AASP (Ours)	**87.80**	**79.29**	**63.72**	**62.69**	**73.38 (+3.3)**
AAE-FG	GMAM*	84.20	57.32	30.81	25.95	49.57
	ST*	85.75	60.06	57.53	57.03	65.09
	AASP (Ours)	**87.20**	**62.96**	**63.27**	**62.15**	**68.90 (+3.81)**
CDCP	BART-B	–	56.15	–	13.76	–
	GMAM	–	57.72	–	16.57	–
	ANL-AM	–	**68.94**	–	28.42	–
	ST	**82.88**	68.90	31.94	31.94	**53.92**
	AASP (Ours)	73.82	62.14	**35.77**	**35.77**	51.88

TABLE I: Performance comparison on the AAE, AAE−FG and CDCP datasets. The Micro-F1 scores of ACI, ACC, ARI and ARC tasks are reported with AVG, indicating the average score of all four tasks. Apart from AAE−FG, all the reported results are directly taken from the corresponding baselines. For AAE−FG, baselines with an asterisk (*) indicate results obtained using the corresponding open-sourced code with original hyperparameters used for AAE dataset. "–" indicates the results are not available for the corresponding baselines.

Dataset	Model	ACI	ACC	ARI	ARC	AVG
AAE	AASP	**87.80**	**79.29**	**63.72**	**62.69**	**73.38**
	AASP*(-rhs)*	87.61	78.23	61.49	60.19	71.88 (-1.50)
AAE-FG	AASP	**87.20**	62.96	**63.27**	**62.15**	**68.90**
	AASP*(-rhs)*	86.94	**63.43**	62.10	60.78	68.31 (-0.59)
CDCP	AASP	73.82	62.14	**35.77**	**35.77**	**51.88**
	AASP*(-rhs)*	73.82	**62.85**	33.94	33.94	51.13 (-0.74)

TABLE II: Results of ablation experiments across datasets. AASP*(-rhs)* refers to the AASP framework with *Reduced Hidden States* of both the Feed-Forward Networks corresponding to the score function s_θ described in Section III-B. Best scores are in **Bold**.

AASP in handling tree-structured argumentation. Notably, the improvements are most pronounced in the complex relational tasks (ARI and ARC), where AASP surpasses the baselines by 4.17% and 4.18% on AAE, and 5.74% and 5.12% on AAE−FG, compared to the gains in the component tasks (ACI and ACC): 2.05% and 2.79% on AAE, and 1.45% and 2.90% on AAE−FG. Specifically, in AAE dataset, the most recent generative SoTA baseline DENIM uses an additional discourse structure information through *prefix* to better capture the relational dependency of argumentative structures. In contrast, AASP produces SoTA results without using any additional customized information. On the non-tree-structured CDCP dataset, AASP demonstrates significant gains in the relational tasks, achieving a substantial improvement of 3.83% in ARI and ARC over the current SoTA. However, in component tasks such as ACI and ACC, AASP shows limited performances in the CDCP dataset. While most baselines struggle with complex relational tasks, AASP demonstrates improved performance in both ARI and ARC tasks in both tree-structured and non-tree-structured argumentation. This highlights AASP's ability to model relational dependencies effectively across diverse datasets.

Given that the ST baseline has been widely applied to most standard datasets, it stands as a robust baseline for comparison. Therefore, in the subsequent sections, we compare all analyses with ST to better compare the performance of AASP.

B. Ablation Study

We conduct an ablation study to assess the impact of FFNs in the score function s_θ by altering their size. In AASP, the hidden state size is 1500, while the original ASP framework uses 150. To analyze this, we run experiments on all datasets with a hidden state size of 150 instead of 1500. We denote this variant as AASP*(-rhs)*, where *rhs* stands for *reduced hidden states*. The results, as shown in Table II, indicate that AASP*(-rhs)* underperforms in terms of average scores across all four AM tasks compared to AASP. The ACI task performance is dropped in AAE and AAE−FG datasets with smaller FFNs. It means that the *Boundary-Pairing* actions require larger FFNs to finalise the boundaries efficiently. Although for AAE−FG and CDCP, the ACC task performance improves in AASP*(-rhs)*, the performance decline in ARI and ARC tasks is significant across all datasets. It suggests that the size of the FFNs plays a crucial role in effectively modelling complex relational dependencies. Specifically, the score function s_θ is better equipped to handle distantly related ACs when the *Type-Labeling* actions connect the $\langle/m\rangle$ symbols of both the ACs. As a by-product, the *chains of reasoning* is also handled efficiently when larger FFNs are used.

C. Performance of the ACI task on Varied Length Paragraphs

In this section, we compare the performance of AASP with ST for the ACI task across datasets (See Fig. 3). In the AAE

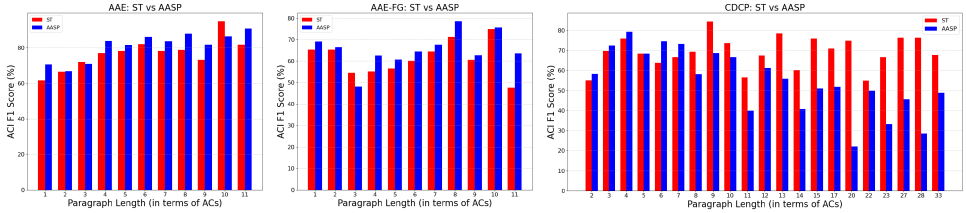

Fig. 3: Comparison of ACI Micro-F1 scores (%) between AASP and ST across paragraphs of varying lengths (in terms of the number of ACs present in that paragraph) for the AAE, AAE−FG, and CDCP datasets.

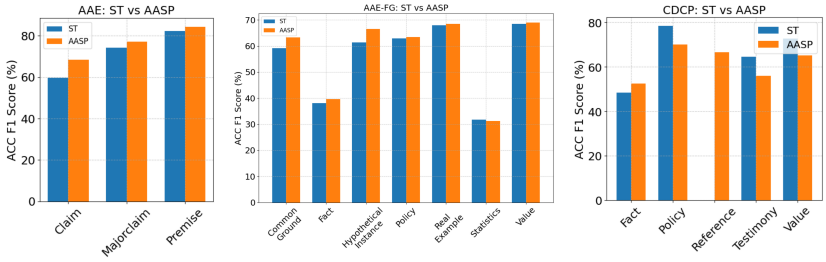

Fig. 4: Comparison of ACC Micro-F1 scores (%) for AASP and ST across different AC categories in the AAE, AAE−FG, and CDCP datasets.

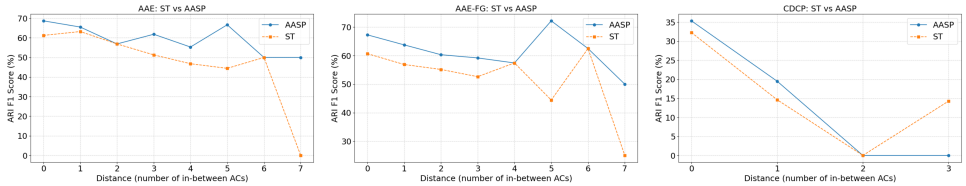

Fig. 5: Performance analysis of ARI task Micro-F1 scores (%) for AASP and ST across varying distances (in terms of number of intermediate ACs) in the AAE, AAE-FG, and CDCP datasets.

dataset, the Micro-F1 score for the ACI task improves as paragraph length (in ACs) increases, peaking at 9–11 ACs. The ST baseline starts at around 60% Micro-F1 for single-AC paragraphs and rises to 80% for longer ones. AASP begins slightly higher, at 65%, and consistently outperforms ST, exceeding 80% for the longest paragraphs. For the AAE−FG dataset, a similar trend is observed, but the overall performance is slightly lower than in AAE. Here, both ST and AASP show a more gradual improvement, beginning at around 65% and reaching approximately 70–75% for 8–10 ACs. In the CDCP dataset, the relationship between paragraph length and Micro-F1 score becomes interesting. The ST baseline performs well across a wide range of lengths, achieving high scores (70–80%) for most configurations. However, while AASP remains competitive, it underperforms compared to ST for longer paragraphs, especially beyond 8 ACs. But it surpasses ST by a noticeable margin for paragraphs with fewer than 8 ACs. This indicates that the AASP framework struggles with the ACI task for longer paragraphs in the complex, non-tree-structured CDCP dataset. However, it achieves superior performance across all three datasets for paragraphs of moderate length.

D. Analysis of the ACC task for Different AC Categories

In this section, we compare the performance of AASP and ST on different AC categories across datasets (see Fig. 4). In the AAE dataset, AASP outperforms ST across all AC types. The PREMISE category achieves the highest Micro-F1 score, making it the most predictable for both frameworks. Strong performance is also observed for the MAJORCLAIM type. Notably, AASP shows a significant improvement of nearly 10% over ST for the CLAIM category, making it the better choice for CLAIM classification. The AAE−FG dataset, with its more fine-grained AC labels, poses a greater challenge for both frameworks. In the AAE−FG dataset, the original AC categories from AAE are divided into finer-grained classes, reducing the training instances per class. Despite this, both frameworks perform robustly. AASP shows notable gains in categories like HYPOTHETICAL INSTANCE and COMMON GROUND, demonstrating its strength in handling finer-grained argumentation. The CDCP dataset presents mixed results. While ST performs better overall in the AC classification task, AASP outperforms ST in certain categories. For example, AASP achieves better performance in FACT classification. Surprisingly, ST fails to detect any instances of the REFERENCE category, whereas AASP achieves over 65%

Dataset	Framework	Lengthwise Distributions of Chains			Accuracy (%)	
		Ground Truth	Predicted	Correct	2-length	3-length
AAE	ST	{2: 130, 3: 15, 4: 2}	{2: 109, 3: 5}	{2: 39}	30.00	0.00
	AASP	{2: 130, 3: 15, 4: 2}	{2: 124, 3: 16, 4: 1}	{2: 46, 3: 2}	**35.38**	**13.33**
AAE-FG	ST	{2: 130, 3: 15, 4: 2}	{2: 122, 3: 9, 4: 1}	{2: 37, 3: 2}	28.46	13.33
	AASP	{2: 130, 3: 15, 4: 2}	{2: 125, 3: 10}	{2: 45, 3: 3}	**34.62**	**20.00**
CDCP	ST	{2: 23, 3: 9, 4: 1}	{2: 18, 3: 10}	{2: 1}	4.35	0.00
	AASP	{2: 23, 3: 9, 4: 1}	{2: 13, 3: 1}	{2: 4}	**17.39**	0.00

TABLE III: Performance Comparison of ST and AASP across datasets on Relation Chain Detection for Varied Length Chains.

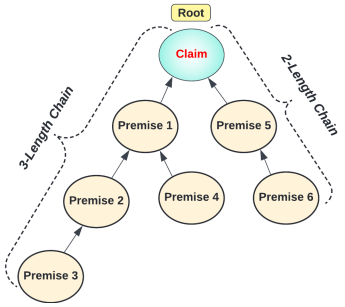

Fig. 6: Example of chains present in an argumentative structure

Micro-F1 score in this category. These empirical results emphasize the effectiveness of AASP for ACC task, particularly for fine-grained or underrepresented categories.

E. Analysis of ARI Task for Long-Range Relations

In argumentative paragraphs, the relationships between related ACs exhibit a fascinating variety of patterns. Some related ACs are directly linked with no intermediate ACs, forming straightforward connections, while others are moderately far apart, involving a few intermediate ACs. The most challenging cases arise when ACs are distantly related, with one appearing near the beginning of the paragraph and the other near the end. Identifying such varied-distant relations is critical for understanding complex argumentative structures. To assess this capability, we analyze the performance of both ST and AASP across varying distantly related ACs, as shown in Fig. 5. In the AAE dataset, AASP consistently outperforms ST across all distances. For direct relations (no intermediate ACs), AASP achieves a Micro-F1 score of nearly 70%, while ST lags behind at 60%. Although performance declines with increasing distance, AASP proves more robust. It maintains a Micro-F1 score above 50% even at the maximum distance of 7 intermediate ACs, where ST experiences a sharp drop. In the AAE-FG dataset, a similar trend is observed. The CDCP dataset presents a more challenging scenario due to the complex, non-tree-structured nature of its argumentative relations. Both frameworks show similar patterns in this dataset up to 2 intermediate ACs. However, for the maximum distance (3 intermediate ACs), AASP's performance drops sharply to 0%, highlighting the challenges of identifying long-range relations in this dataset.

F. Performance Analysis of Relational Chains

Relational chains (e.g. a series of premises/claims as shown in Fig. 6) are the building blocks of the *Chain of Reasoning* present in any argumentative discourse. Currently, none of the existing works has focused on this aspect of analysis. As shown in Fig. 6, an argumentative paragraph comprising seven ACs: $\{Claim, Premise_1, Premise_2, \ldots, Premise_6\}$. With these ACs, a 3-length sequential chain is formed as: $Premise_3 \rightarrow Premise_2 \rightarrow Premise_1 \rightarrow Claim$, with $Claim$ as the root AC. Here, each AC is directly connected to the preceding one. In such configurations, identifying relations between adjacent ACs poses a significant challenge. This difficulty arises because all ACs in the chain are (in-)directly connected through the root AC, increasing the likelihood of misidentifying relations. For example, one might incorrectly predict $Premise_3 \rightarrow Premise_1$ instead of the correct $Premise_3 \rightarrow Premise_2$. In order to assess the performance in detecting these relational chains, we present an in-depth analysis in Table III, where we compare ST and AASP frameworks across datasets. AASP achieves 35.38% accuracy for 2-length chains in AAE and 34.62% in AAE-FG, compared to ST's 30.00% and 28.46%, respectively. However, both frameworks struggle to accurately identify longer chains (3-length and above). The CDCP dataset, with its complex non-tree argumentative structures, poses challenges in both frameworks. However, AASP still surpasses ST with 17.39% accuracy for 2-length chains compared to ST's 4.35%. This comparative analysis highlights the effectiveness of AASP in detecting chains of ACs over existing baselines. Moreover, it also emphasizes the need for further advancements in modelling complex relational structures to capture the chain of reasoning more accurately.

G. Error Analysis

We conducted error analysis across all three datasets and identified the following errors (see Fig. 7): *(i) AC Boundary Mismatches*, where predicted ACs differ in length from the ground truth; *(ii) Missed ACs*, where valid ACs are not detected; *(iii) False Positive ACs*, where non-existent ACs are predicted; *(iv) AC Misclassifications*, where ACs are assigned incorrect types; *(v) AR Misclassifications*, where ARs are incorrectly labeled; *(vi) Split ACs*, where true ACs are divided into smaller segments; and *(vii) Merged ACs*, where multiple ACs are wrongly combined into one.

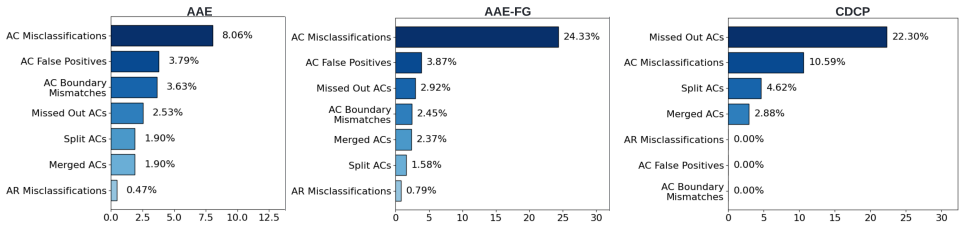

Fig. 7: Error distribution across three datasets with AASP framework, highlighting the percentage of various error categories.

Analyzing the errors across the datasets, *AC Misclassifications* emerge as the most significant error type overall, particularly in AAE-FG (24.33%) and CDCP (10.59%), indicating a consistent challenge in correctly classifying ACs, where the number of AC types is more. *Missed Out ACs*, another major source of error for the CDCP dataset with 22.30% error, suggests that the AASP struggles more with identifying all potential ACs in this complex non-tree-structured dataset than other datasets. *AC False Positives* and *AC Boundary Mismatches* show relatively similar proportions in AAE (3.79% and 3.63%, respectively) and AAE-FG (3.87% and 2.45%, respectively). Surprisingly, the CDCP dataset has no such errors due to the fact that in CDCP, each AC spans over a full sentence, unlike AAE or AAE-FG, where the ACs are further segmented inside sentences. Detecting full-sentence ACs is relatively easier than the segmented ACs. Upon manually analyzing the predictions of both versions of AAE, we find spans such as *"I believe that"* or *"People might argue that"* are included inside the AC spans, alleviating the incorrect boundaries. *Split ACs* and *Merged ACs*, which reflect structural prediction errors, remain minor across datasets but still noticeable, with AAE and AAE-FG showing similar patterns (1.9%-2.4% for both error types). However, the CDCP dataset shows a slightly higher occurrence of Split ACs (4.62%) compared to Merged ACs (2.88%) due to its ACs spanning over a full sentence. Whenever the sentence is longer, the split sometimes happens when connectives such as *"and"* or *"because"* are present inside sentences. Surprisingly, *AR Misclassifications* are the least frequent errors across all datasets, staying below 1%, indicating that the AASP performs relatively well in recognizing argumentative relations effectively.

VI. Conclusion

In this work, we introduced **AASP**, an end-to-end framework designed to address all four key tasks of AM jointly. By extending the ASP framework, we effectively tackled the ACI task through span identification and boundary-pairing actions, while ACC, ARI, and ARC tasks were addressed using type-labeling actions. Our approach leverages larger FFNs to enhance the execution of these actions, improving the model's ability to process longer AC spans and distantly related ACs. This contributes to overall performance gains, as verified through our ablation study. To gain deeper insights into the relational performance, we conduct an in-depth relational chain analysis. The results highlight the effectiveness of AASP in outperforming competitive baselines. Out of the three benchmarks, AASP achieved SoTA results across all four AM tasks in two benchmarks while demonstrating SoTA performance for relational tasks in the remaining benchmark.

References

[1] Z. Ke, W. Carlile, N. Gurrapadi, and V. Ng, "Learning to give feedback: Modeling attributes affecting argument persuasiveness in student essays," in *IJCAI*, 2018.

[2] V. R. Walker, D. Foerster, J. M. Ponce, and M. Rosen, "Evidence types, credibility factors, and patterns or soft rules for weighing conflicting evidence: Argument mining in the context of legal rules governing evidence assessment," in *Proceedings of the 5th Workshop on Argument Mining*, 2018.

[3] T. Mayer, E. Cabrio, and S. Villata, "Transformer-based argument mining for healthcare applications," in *ECAI*, 2020.

[4] S. Eger, J. Daxenberger, and I. Gurevych, "Neural End-to-End Learning for Computational Argumentation Mining," in *ACL*, 2017.

[5] Y. Ye and S. Teufel, "End-to-end argument mining as biaffine dependency parsing," in *EACL*, 2021.

[6] G. Morio, H. Ozaki, T. Morishita, and K. Yanai, "End-to-end argument mining with cross-corpora multi-task learning," *TACL*, 2022.

[7] T. Dozat and C. D. Manning, "Simpler but more accurate semantic dependency parsing," in *ACL*, 2018.

[8] J. Bao, Y. He, Y. Sun, B. Liang, J. Du, B. Qin, M. Yang, and R. Xu, "A Generative Model for End-to-End Argument Mining with Reconstructed Positional Encoding and Constrained Pointer Mechanism," in *EMNLP*, 2022.

[9] M. Kawarada, T. Hirao, W. Uchida, and M. Nagata, "Argument mining as a text-to-text generation task," in *EACL*, 2024.

[10] G. Paolini, B. Athiwaratkun, J. Krone, J. Ma, A. Achille, R. Anubhai, C. N. dos Santos, B. Xiang, and S. Soatto, "Structured prediction as translation between augmented natural languages," *ArXiv*, 2021.

[11] C. Stab and I. Gurevych, "Parsing Argumentation Structures in Persuasive Essays," *Computational Linguistics*, 2017.

[12] V. Niculae, J. Park, and C. Cardie, "Argument Mining with Structured SVMs and RNNs," in *ACL*, 2017.

[13] B. Liu, V. Schlegel, R. Batista-Navarro, and S. Ananiadou, "Argument mining as a multi-hop generative machine reading comprehension task," in *EMNLP*, 2023.

[14] G. Morio, H. Ozaki, T. Morishita, Y. Koreeda, and K. Yanai, "Towards better non-tree argument mining: Proposition-level biaffine parsing with task-specific parameterization," in *ACL*, 2020.

[15] J. Bao, C. Fan, J. Wu, Y. Dang, J. Du, and R. Xu, "A neural transition-based model for argumentation mining," in *ACL-IJCNLP*, 2021.

[16] Y. Sun, G. Chen, C. Yang, J. Bao, B. Liang, X. Zeng, M. Yang, and R. Xu, "Discourse structure-aware prefix for generation-based end-to-end argumentation mining," in *ACL*, 2024.

[17] T. Liu, Y. Jiang, N. Monath, R. Cotterell, and M. Sachan, "Autoregressive structured prediction with language models," in *EMNLP*, 2022.

[18] T. Kuribayashi, H. Ouchi, N. Inoue, P. Reisert, T. Miyoshi, J. Suzuki, and K. Inui, "An empirical study of span representations in argumentation structure parsing," in *ACL*, 2019.

[19] J. Bao, C. Fan, J. Wu, Y. Dang, J. Du, and R. Xu, "A Neural Transition-based Model for Argumentation Mining," in *ACL-IJCNLP*, 2021.

[20] I. Persing and V. Ng, "End-to-End Argumentation Mining in Student Essays," in *NAACL-HLT*, 2016.

[21] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *ArXiv*, 2020.

[22] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *ACL*, 2020.

[23] R. Schaefer, R. Knaebel, and M. Stede, "Towards fine-grained argumentation strategy analysis in persuasive essays," in *Proceedings of the 10th Workshop on Argument Mining*, 2023.

[24] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma *et al.*, "Scaling instruction-finetuned language models," *JMLR*, 2024.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[26] H. Yan, J. Dai, T. Ji, X. Qiu, and Z. Zhang, "A unified generative framework for aspect-based sentiment analysis," in *ACL-IJCNLP*, 2021.