CU-MAM: Coherence-Driven Unified Macro-Structures for Argument Mining

Anonymous ACL submission

Abstract

Argument Mining (AM) involves the automatic identification of argument structure in natural language. Traditional AM methods rely on micro-structural features derived from the internal properties of individual Argumentative Discourse Units (ADUs). However, argument structure is shaped by a macro-structure capturing the functional interdependence among ADUs. This macro-structure consists of segments, where each segment contains ADUs that fulfill specific roles to maintain coherence within the segment (**local coherence**) and across segments (**global coherence**). This paper presents an approach that models macro-structure, capturing both local and global coherence to identify argument structures. Experiments on heterogeneous datasets demonstrate superior performance in both in-dataset and cross-dataset evaluations. The cross-dataset evaluation shows that macro-structure enhances transferability to unseen datasets.

1 Introduction

Argument Mining (AM), a Natural Language Processing (NLP) task, involves identifying and analysing argument structures within natural language (Persing and Ng, 2016; Stab and Gurevych, 2017; Eger et al., 2017; Potash et al., 2016; Lawrence and Reed, 2020). It involves argument component segmentation (ACS), argument component type classification (ACTC), argument relation (AR) identification (ARI), and AR type classification (ARTC) (Peldszus and Stede, 2015a; Eger et al., 2017; Lawrence and Reed, 2020). This study focuses on ACTC, ARI, and ARTC.

The identification of argument structures requires modeling the roles of ADUs and ARs as functions of a global structure, governing coherent arrangement of these components to fulfill the overarching Discourse Purpose (DP) (Grosz and Sidner, 1986; Freeman, 2011). The global structure is decomposed into local structures, each aligned with a specific Discourse Segment Purpose (DSP). These localised structures ensure segment-level coherence by organising ADUs and ARs into functional units, much like how words combine into phrases to convey meaning (Grosz and Sidner, 1986). For instance, Figure 1 illustrates four localised structures in a COVID-19 contact tracing argument: (1) the effectiveness of South Korea's contact tracing, (2) government preparedness, (3) non-app-based tracing, and (4) advancements in testing. In each local structure, the ADUs fulfill the DSP of that segment. For example, the ADUs in segment (3) address the DSP of non-app-based tracing.

The arrangement of ADUs and the ARs within the local structures is shaped by the intentions of the arguer and the sequential ordering of ADUs ensuring a natural flow for maintaining coherence (Travis, 1984; Freeman, 2011; Wang et al., 2019; Kazemnejad et al., 2024). The intentional structure captures the logical flow of ADUs and can extend beyond proximity to connect ADUs based on their roles and contributions to the DSP of the segment (Grosz and Sidner, 1986; Freeman, 2011). Figure 1 illustrates this interplay, demonstrating the sequential flow of arguments (e.g., ADU1 $\rightarrow$ ADU2 $\rightarrow$ ADU3 $\rightarrow$ ADU4 $\rightarrow$ ADU5) alongside logical relationships transcending proximity (e.g., A14 $\rightarrow$ A17, A13 $\rightarrow$ A19, or A20 $\rightarrow$ A23). This underscores the importance of modeling macro-structure governing the intentional and the sequential flow of ADUs and their ARs. See details in Appendix C and more examples in Figure 3.

However, most previous works focus on features derived from the internal structure of ADUs, often referred to as the micro-structure (Freeman, 2011), while overlooking the broader macro-structure. They frame AM tasks as either dependency parsing (Peldszus and Stede, 2015b), sequence tagging (Eger et al., 2017), or sequence classification (Reimers et al., 2019; Ruiz-Dolz et al., 2021), concentrating primarily on isolated ADU pairs. End-to-

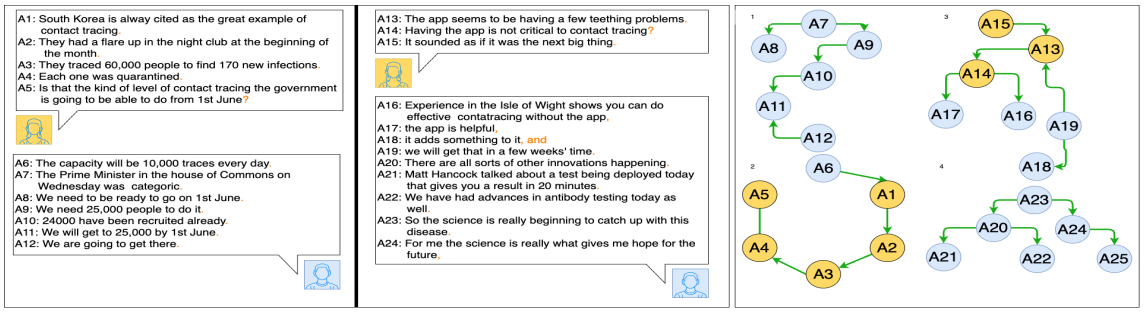


Figure 1: An example of the argument structure decomposed into four local structures (1 to 4). It shows how ADUs and AR are shaped by the intentional structure, where consecutive ADUs may span different segments, and non-consecutive ADUs can share the same segment. The argument is taken from QT30, illustrating a dialog between two participants, highlighted in light blue and yellow (Hautli-Janisz et al., 2022).

end AM approaches model dependencies between tasks, employing various techniques, including biaffine operations for learning non-tree AM structures (Morio et al., 2020), a transition-based model for constructing both tree and non-tree argument graphs (Bao et al., 2021), and positional encodings in generative AM frameworks to mitigate order biases (Bao et al., 2022). Recent advancements in language models (LM) have enhanced their ability to capture higher-level structures for processing long documents by encoding document sections, hierarchies, and global contexts, resulting in notable improvements in NLP tasks like document classification and summarisation (He et al., 2024; Cao and Wang, 2022; Liu et al., 2022; Bai et al., 2021). However, in the domain of AM, aside from position-aware discourse self-attention for identifying discourse elements (Song et al., 2020) and end-to-end approaches focusing on capturing dependencies between tasks, there has been limited progress in addressing the critical aspect of modeling macro-structures, which often remains implicit. To our knowledge, no AM work has proposed a unified architecture that models coherence by integrating macro-structure encoding logical and sequential ADU flows at both local and global levels, while anchoring AM tasks to this coherence.

In this study, we propose **CU-MAM**: *Coherence-Driven Unified Macro-Structures for Argument Mining*, an approach that anchors ACTC, ARI, and ARTC tasks to a unified macro-structure. Given a pair of ADUs and the entire argument as context, the model predicts the ADU types (ACTC) and the AR between them (ARI and ARTC), considering both local and global structural information. This is accomplished through a multi-task learning approach that jointly models ACTC, ARI, and ARTC as primary tasks while treating local and global structure learning as auxiliary tasks. The argument is represented as a graph, where ADUs are nodes and ARs are edges, capturing the complete argument structure. Local and global structures learning is achieved by classifying graph edges into their respective local or global categories. A self-attention layer attends to the graph's nodes and edges to encode the local and global structures relevant to the ADU pair under consideration. To contextualise ADU type and AR predictions within these macro-structures, the self-attention layer's output is fused with the ADU-pair representation via a cross-attention mechanism. Additionally, the sequential flow of the argument is modeled by incorporating ADU-level positional encodings into the ADU embeddings. These positional encodings are derived from the order of ADUs and discourse participant transitions (e.g., proponent-opponent shifts) (Freeman, 2011; Budzynska and Reed, 2011).

This paper makes the following contributions: (a) We propose a macro-structure to capture the coherent arrangement of ADUs. (b) We introduce an architecture combining a graph-based neural model with a dual attention mechanism to capture local and global argument structures. A multi-task learning framework anchors ACTC, ARI, and ARTC to these macro-structures. (c) We achieve state-of-the-art (SOTA) results across multiple datasets, including a cross-dataset evaluation where previous SOTA models struggle to surpass random chance.

2 Related Work

2.1 Argument Mining

Argument Mining has been studied through diverse paradigms, emphasising the micro-structure of arguments. One common approach frames AM as

a dependency parsing task (Peldszus and Stede, 2015b), leveraging discourse parsing techniques (Muller et al., 2012). Peldszus and Stede (2016) extend this by mapping Rhetorical Structure Theory (RST) trees (Taboada and Mann, 2006) to argument structures. Other works model AM as token-based sequence tagging (Eger et al., 2017), classifying tokens into argument components and AR types using the BIO tagging scheme. Gemechu and Reed (2019) decompose ADUs into fine-grained components, predicting ARs based on their interactions. Recent studies fine-tune pre-trained LMs, employing sequence-pair classification setups (Reimers et al., 2019; Ruiz-Dolz et al., 2021). These configurations focus on the internal structure of ADUs, while neglecting the broader macro-structure.

Efforts toward end-to-end AM have largely focused on leveraging task interdependencies. Pipeline architectures train independent models for sub-tasks, integrating global constraints through Integer Linear Programming (ILP) (Persing and Ng, 2016; Stab and Gurevych, 2017). Neural approaches adopt joint multi-task setups to model interdependencies across tasks (Eger et al., 2017). Morio et al. (2022) introduce a cross-corpus training strategy, while Bao et al. (2022) propose a generative framework incorporating constrained pointer mechanisms and reconstructed positional encodings into an end-to-end AM setup. Despite these advancements, these methods emphasize task-level dependencies, offering limited or no explicit modeling of the macro-structure.

2.2 Structural Encoding in Language Models

Recent advancements in LMs have improved the capacity to encode long texts and represent document structures (He et al., 2024; Cao and Wang, 2022; Liu et al., 2022; Bai et al., 2021; Zaheer et al., 2020; Beltagy et al., 2020). For instance, He et al. (2024) and Cao and Wang (2022) utilise section structures to encode document hierarchies, while Liu et al. (2022) employ hierarchical sparse attention and specialised tokens to capture local and global information within a document. Similarly, Bai et al. (2021) use positional encoding at various linguistic segments to capture hierarchies. Beltagy et al. (2020) introduces Longformer, which combines local windowed attention with global for long-document. Zaheer et al. (2020) propose BigBird, a model leveraging sparse attention mechanisms that integrate global, local, and random attention patterns to handle long sequences. Although these models provide avenues for encoding macro-structures, their effectiveness in addressing the unique challenges of argumentation's macro-structure remains limited (see Tables 2 and 3), primarily due to their reliance on static document structural features, which fail to capture the distinct characteristics of argumentation—such as logical relationships, argumentative flows, and the interplay between ADUs.

Data	No_arg	No_ADU	No_AR	No_LOC	Dist_ARs
AAEC	402	6089	5338	3.3	2.6
US2016	499	8610	3772	5.1	3.2
QT30	724	11266	3314	7	4.8
CDCP	731	4779	1353	5.6	3.4

Table 1: Dataset summary of argument counts (No_arg), average number of ADUs (No_ADU), ARs (No_AR), local structures (No_LOC), and ADU distance in ARs (Dist_ARs) per argument.

3 Method

3.1 Data

Heterogeneous datasets encompassing various domains and genres are utilised, including student persuasive essay corpora (AAEC) (Stab and Gurevych, 2017), Consumer Debt Collection Practices (CDCP) (Park and Cardie, 2018), the US 2016 presidential debate corpus (US2016) (Visser et al., 2019), and a corpus of argument and conflict in broadcast debate (QT30) (Hautli-Janisz et al., 2022). The AAEC and CDCP, are monological, while the US2026 and QT30 are dialogical. The datasets CDCP, AAE, QT30, and US2016 employ different annotation standards for ADUs and ARs. CDCP defines five ADU types—Reference, Fact, Testimony, Value, and Policy—and two AR types—Reason and Evidence. AAE uses three ADU types—MajorClaim, Claim, and Premise—and four AR types—Support, Attack, For, and Against. In the QT30 and US2016 datasets, ADUs are not explicitly labeled; instead, their types are inferred from the direction of the ARs (premise to conclusion), resulting in two ADU types: Premise and Conclusion.

3.1.1 Global Structure

The global structure consists of all valid ARs within the argument structure, which are essential for achieving the DP. These valid ARs represent a subset of the possible permutations of connections between the ADUs in the argument.

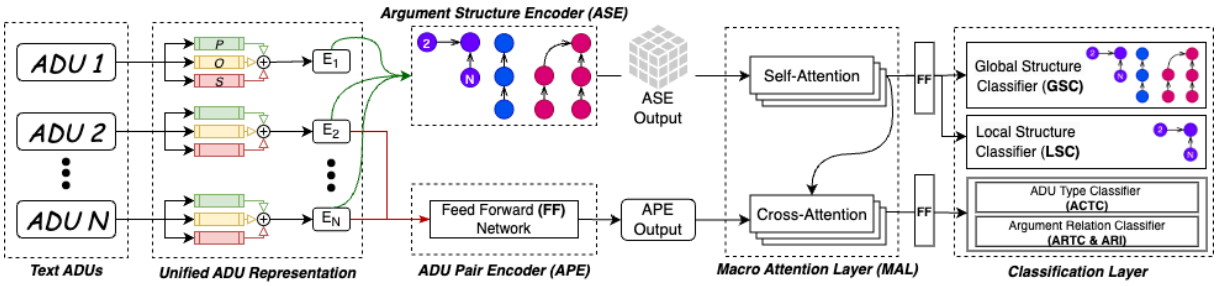


Figure 2: CU-MAM Architecture.

3.1.2 Local Structure

An argument structure is represented as graphs where ADUs and ARs are nodes connected by edges. Local Structure identification involves upward and downward traversals of the graph from each AR node. The upward traversal identifies chains of ADUs leading to the AR, capturing the local structure that establishes its context. The downward traversal, traces the chain of ADUs following the AR, ensuring the continuity of the argument segment. The beginning of a local structure is identified by a node with no inward connections (start ADU), marking the segment's starting point, while its end is defined by a node without successors (end ADU), indicating the segment's conclusion. In cases where the start ADU involves multiple downward chains (divergent structures), all such chains are included. Furthermore, every subgraph—whether serial, divergent, convergent, or linked—between the start and end ADUs is incorporated to ensure a complete and coherent segment (see Appendix D for details). Table 1 provides a summary of the dataset statistics. Among the argument structures, 73% involve more than one local structure, with 67% containing between 2 to 7 local structures. Additionally, 64% of ARs occur between ADUs positioned 1 to 5 apart, and 17% involve ADUs that are within a distance of 1.

3.2 Model

This section provides an overview of the task definition, model architecture, and baseline configurations used in the experiment.

3.2.1 Task Definition

Given an argument $\mathbf{A}$ comprising a sequence of ADUs and a specific ADU pair (ADU_i, ADU_j), the model's primary task is to predict the types of ADU_i, ADU_j, and the AR between them, one pair at a time, within the context of the argument's macro-structure (local and global structure). To achieve this, the model is trained on auxiliary tasks that predict local and global structures, anchoring the primary task to these macro-structures in a multi-task setting. During inference, only the primary task is used. See Section A.2 for input details.

3.2.2 Architecture

As shown in Figure 2, the model consists of five key components: (A) Unified ADU Representation, which combines ADU embeddings with positional information; (B) Argument Structure Encoder, which employs a graph network where ADUs are nodes and ARs between them are edges; (C) ADU-Pair Encoder, which encodes the specific pair of ADUs under consideration; (D) Macro-Attention Layer, which attends to the graph's nodes and edges to capture the ADUs and ARs that constitute the local and global structures relevant to the ADU pair; and (E) Classification Layers, which predict ADU types, ARs, and classify graph edges as local, global, or none, in a multi-task setting.

(A) Unified ADU Representation (UAR)

ADU representation combines a pre-trained LM embedding, with two ADU-level positional embeddings capturing sequential argument flow. Given an argument $A = \{ADU_1, ADU_2, \ldots, ADU_n\}$, the unified representation, $\mathbf{ADU}'_i$ is given by:

$$\mathbf{ADU}'_i = \mathbf{ADU}_i \oplus \mathbf{O}_i \oplus \mathbf{P}_i \quad (1)$$

where $\mathbf{ADU}_i$ is the sentence embedding (S) of the ADU obtained by mean pooling token embeddings from LM. $\mathbf{O}_i$ is the order-based positional embedding indicating the ADU's sequential index, and $\mathbf{P}_i$ is the participant transition embedding, capturing participant shifts in multi-participant dialogues, with all ADUs assigned the same index in monologues. We experiment with sinusoidal absolute positional encodings (Vaswani et al., 2017) and relative positional embeddings (Shaw et al., 2018).

Absolute positional embeddings are added to ADU embeddings, while relative embeddings are fused during attention computation (See Section B.2).

(B) Argument Structure Encoder (ASE)

A Graph Neural Network (GNN) (Brody et al., 2021) models the argument structure as a graph $G = (V, E)$, where nodes $V = \{v_1, v_2, \ldots, v_n\}$ represent ADUs, and edges $E \subseteq V \times V$ capture ARs between ADUs, facilitating local and global structure predictions relevant to a given ADU pair. The graph is constructed using the unified ADU embeddings from Equation 1. At each layer, node states are updated as:

$$\mathbf{h}_v^{(k+1)} = \sigma\left(\mathbf{W}_k \mathbf{h}_v^{(k)} + \sum_{u \in \mathcal{N}(v)} \mathbf{W}'_k \mathbf{h}_u^{(k)}\right) \quad (2)$$

where $\mathbf{h}_v^{(k)}$ is the node state at layer k, and $\mathcal{N}(v)$ denotes neighboring ADUs. Each AR between ADUs is represented by the concatenation of their node embeddings.

(C) ADU-Pair Encoder (APE)

Encodes the relationship between the pair of ADUs (ADU_i, ADU_j) under consideration. A feedforward layer is applied to the unified embeddings of ADU_i and ADU_j to capture their interaction.

(D) Macro-Attention Layer (MAL)

Two self-attention layers attends to the argument graph from step B, learning the local and global structures relevant to the ADU pair from step C. The self-attentions are applied to the edge embeddings of the argument graph to capture the relationships between nodes (i.e., the AR between ADUs). The outputs from both attention layers pass through fully connected layers, and used to classify the edges into their respective macro-structures (see Section 3.2.2).

To contextualise the ADU type and AR predictions within the macro-structural context, the outputs of the self-attention layers are fused with the ADU-pair representation from step C using a cross-attention mechanism. The queries are derived from the ADU-pair encoder's output, while the keys and values are projections of the summation of the self-attention layers. The final representation of the ADU pair, denoted as $\mathbf{R}_{\text{ADU-pair}}$, is obtained by adding the cross-attention's output to the original ADU-pair encoder's output. This final representation combining both the structural context and the ADU pair representation is used to predict both the ADU types and the ARs between them.

(E) Classification Layers

Linear classifiers are used for predicting the ADU pair types (ACTC) and AR between them (ARI and ARTC), using the contextualised ADU-pair representations, $\mathbf{R}_{\text{ADU-pair}}$. We jointly model ARI and ARTC, following the approach in (Bao et al., 2021), while also modeling ARI independently for comparison with studies treating them separately. The local and global structures are learned by predicting argument graph edges as binary outputs, trained on gold labels (prepared in Sections 3.1.1 and 3.1.2) that indicate the presence or absence of connections between nodes. These tasks are framed as auxiliary within a multi-task setup. The model trains using a loss function (L) that integrates task-specific losses and a regularisation term, $L = L_0 + L_1 + L_2 + L_3 + R$, where L_0 is the loss for ADU type prediction, L_1 for AR classification, L_2 for global-structure prediction, L_3 for local-structure prediction, and R for the regularization term.

Due to the significantly higher number of non-AR edges compared to AR edges, L_2 is computed exclusively for AR edges, excluding non-AR edges. Similarly, L_3 ignores both AR edges and non-AR edges outside the local structure. However, this approach leads to model overfitting on AR edges. To mitigate this issue, a regularisation term (R) based on the distance editing score is introduced to penalise deviations from the gold argument structure, encouraging correct classification of both AR and non-AR edges.

3.2.3 Baselines

We establish two baselines using RoBERTa (Liu et al., 2019), reportedly achieving strong performance in AM and BigBird (Zaheer et al., 2020), for its architecture in capturing global context. The first baseline, Vanilla Sequence-Pair Classification (V-SeqCls), fine-tunes the LMs on concatenated ADU pairs. The second, Vanilla Argument Context (V-ArgC), includes the entire argument as context alongside ADU pairs for direct comparison to CU-MAM. Since both LMs are used in CU-MAM for ADU embeddings, evaluating them as standalone baselines ensures robust comparisons. BigBird serves as a strong baseline due to its global context modeling (see Appendix A.3 for more details).

LM	Model	ACTC				ARI				ARTC			
		AAEC	CDCP	US2016	QT30	AAEC	CDCP	US2016	QT30	AAEC	CDCP	US2016	QT30
RoBERTa	***V-SeqCls*** (Baseline)	69.4	77.6	69.7	71.1	56.6	62.1	72.5	71.7	50.1	14.2	67.1	68.3
	V-ArgC (Baseline)	66.4	73.3	65.0	66.4	54.4	59.2	68.5	69.4	49.3	13.4	64.8	67.3
	CU-MAM^{rel} (ours)	**77.5**	83.1	**75.9**	75.5	68.1	70.4	78.7	77.1	58.1	30.6	75.8	**76.6**
BigBird	***V-SeqCls*** (Baseline)	69.2	77.4	68.4	70.3	57.8	64.3	69.2	71.1	50.1	15.2	67.4	68.2
	V-ArgC (Baseline)	70.7	78.3	70.3	71.6	60.9	64.8	74.2	74.1	49.4	16.7	68.9	68.4
	CU-MAM^{rel} (ours)	77.2	**84.6**	75.4	**76.8**	**70.4**	**72.3**	**80.7**	**78.4**	**58.4**	**31.4**	**76.6**	75.2

Table 2: In-dataset evaluation performance of CU-MAM and baseline models (***V-SeqCls*** and ***V-ArgC***).

4 Experiment

4.1 Training setup

The models are trained for six epochs with a batch size of 16 using Adam optimiser (Kingma and Ba, 2014) with a learning rate of 2×10^{-5}. The primary tasks use categorical cross-entropy, while auxiliary tasks for graph edge prediction use binary cross-entropy. Results are averaged over three runs with different random seeds. Additional experimental details are in Appendix A. The code and dataset are available at `https://github.com/ANONYMOUS` (anonymised).

4.2 Implementation Details

For the AAEC and CDCP, we use the provided training and test data, sampling 10% of the training set for validation. We split the US2016 and QT30 datasets into 70% training, 20% testing, and 10% validation. For AAEC, results are reported at the essay level, with two paragraph-level results for comparison with related works. The AAEC-P^{+} merges the *Claim:Against* and *Claim:For* labels into a single *Claim* label, while AAEC-P uses the original annotations. For cross-dataset evaluations, we use AAEC-P^{+} since US2016 and QT30 do not include *Major Claim* as an argument component type. Similarly, we merge 'For' and 'Against' into Support and Attack, respectively in the AAEC, while the 'Rephrase' ARs in QT30 and US2016 are merged into Support relation.

ADU embeddings in the CU-MAM configurations are obtained from RoBERTa and BigBird.

4.3 Evaluation Setup

The models are evaluated using two setups: In-Dataset Evaluation (ID) and Cross-Dataset Evaluation (CD). For ID, the models are trained and evaluated on the same dataset using the provided training-test split. In CD, the models are trained on one dataset and evaluated on the remaining $n-1$ datasets to assess their performance on unseen data. CDCP is excluded from the CD setup due to differences in ADU and AR type annotations. Across both setups, average macro F-scores (F) are reported for the test dataset. We also report the F1 score for comparison with related works.

4.4 Comparison Systems

CU-MAM is benchmarked against related works, including Bao et al. (2021), Morio et al. (2020), Ruiz-Dolz et al. (2021), Gemechu and Reed (2019), (Potash et al., 2017), (Kikteva et al., 2023) and GPT-4o (OpenAI, 2023). GPT-4o is evaluated using few-shot prompting, the detail is provided in Section B.3. We also make indirect comparisons with Eger et al. (2017), Morio et al. (2022), and Bao et al. (2022), which combine argument segmentation with ACT, ARI, and ARTC.

4.5 Results

Tables 2 and 3 compare the performance of CU-MAM and the two baselines (Section 3.2.3) across the datasets in ID and CD setups. CU-MAM refers to full configuration with local and global structure prediction and MAcro-Attention (unless specified). The main results are reported for CU-MAM with relative positional encoding (**CU-MAM^{rel}**) due to its superiority over absolute encoding. Tables 2 and 3 clearly show that incorporating macro-structural features improves performance across all tasks in both ID and CD, as described below.

In-Dataset (ID) Evaluation

As can be seen in Tables 2, CU-MAM consistently outperforms the two baseline methods that rely solely on fine-tuning LMs across all tasks. In average, for **ACTC**, CU-MAM improves by 6.6% over V-SeqCls and 8% over V-ArgC. For **ARI**, the improvements are 8.9% and 9%, respectively, while for **ARTC**, CU-MAM shows 10.9% and 11.3% improvements over V-SeqCls and V-ArgC. These results highlight the effectiveness of macro-structural features, leading to superior performance compared to the baselines. To calculate the improvement over

LLM	Model	ACTC			ARI			ARTC		
		AAEC	US2016	QT30	AAEC	US2016	QT30	AAEC	US2016	QT30
RoBERTa	***V-SeqCls*** (Baseline)	52.1	55.4	48.9	46.2	48.2	47.8	38.9	45.1	44.4
	V-ArgC (Baseline)	47.5	52.4	48.6	38.9	46.9	47.5	36.9	44.2	41.6
	CU-MAM^{rel} (ours)	64.6	66.1	**66.4**	56.2	62.0	60.5	50.9	58.5	58.0
BigBird	***V-SeqCls*** (Baseline)	55.5	51.4	50.2	43.9	53.6	54.3	40.6	45.4	45.3
	V-ArgC (Baseline)	56.6	53.5	55.5	47.3	56.7	56.5	43.6	46.5	47.1
	CU-MAM^{rel} (ours)	**65.7**	**67.4**	66.3	**57.7**	**66.0**	**64.5**	**51.8**	**61.1**	**60.5**

Table 3: Cross-dataset evaluation performance of CU-MAM and baseline models (***V-SeqCls*** and ***V-ArgC***).

V-SeqCls, we subtract the average performance of BigBird and RoBERTa V-SeqCls from the average performance of BigBird and RoBERTa CU-MAM. Similar approach is used for calculating the improvements over V-ArgC.

Cross-Dataset (CD) Evaluation

Similarly, Table 3 shows that CU-MAM excels in CD evaluations. For **ACTC**, CU-MAM improves by 13.8% over V-SeqCls and 15.2% over V-ArgC. For **ARI**, the improvements are 12.4% and 12.1%, respectively. For **ARTC**, CU-MAM shows improvements of 13.5% and 14.4%. Notably, CU-MAM consistently surpasses the baselines, often achieving cross-dataset performance that is comparable to in-dataset evaluations. For example, when trained on the QT30 dataset and tested on US2016, the BigBird-based CU-MAM matches in-dataset performance. In contrast, baseline models perform near-random chance, showcasing CU-MAM's strong cross-dataset generalisation.

Comparison Systems

As can be seen in Table 4, CU-MAM outperforms all comparison systems, including the indirect comparison approaches that combine argument segmentation with ACT, ARI, and ARTC. The indirect comparisons should be interpreted cautiously due to differences in task setups. BigBird-based CU-MAM outperforms RoBERTa-based CU-MAM configurations in both ID and CD, highlighting its strength in capturing global contexts.

Dataset	Model	ACTC		ARI		ARTC	
		F1	Macro	F1	Macro	F1	Macro
AAEC-E	**Eger et al. (2017)**	66.2	-	-	-	34.8	-
	Morio et al. (2022)	76.6	-	-	-	54.7	-
	GPT-4o	61.4	59.7	54.6	56.4	51.1	49.3
	CU-MAM^{rel}	**79.3**	77.2	62.6	70.4	**60.1**	**58.6**
AAEC-P	**Bao et al. (2022)**	75.9	-	-	-	50.1	-
	Eger et al. (2017)	70.8	-	-	-	45.5	-
	Morio et al. (2022)	76.5	-	-	-	59.6	-
	GPT-4o	62.1	63.3	54.9	60.2	52.3	50.4
	CU-MAM^{rel}	**79.8**	78.4	66.3	81.2	**64.4**	63.1
AAEC-P^{+}	**Potash et al. (2017)**	-	84.9	60.8	76.7	-	-
	Morio et al. (2022)	88.4	86.8	69.3	-	68.1	57.1
	Bao et al. (2021)	-	**88.4**	70.6	82.5	-	81
	GPT-4o	66.4	65.6	58.2	67.7	56.5	58.2
	CU-MAM^{rel}	**88.7**	87.1	**75.4**	**85.4**	**73.1**	**82.7**
CDCP	**Bao et al. (2022)**	57.7	-	-	-	16.6	-
	Morio et al. (2022)	81.0	82.3	40.2	-	40.1	20.4
	Bao et al. (2021)	-	82.5	37.3	67.8	-	-
	Morio et al. (2020)	-	78.9	34.0	-	-	-
	GPT-4o	58.5	68.4	30.1	61.3	32.2	23.4
	CU-MAM^{rel}	**83.4**	**84.6**	**44.8**	**72.3**	**45.1**	**31.4**
US2016	**Ruiz-Dolz et al. (2021)**	-	-	-	-	-	70
	Gemechu and Reed (2019)	-	-	-	64		62
	GPT-4o	64.9	62.3	52.6	56.7	58.4	54.6
	CU-MAM^{rel}	**77.3**	**75.4**	**63.8**	**80.7**	79.8	**76.6**
QT30	**Kikteva et al. (2023)**	-	-	-	-	-	56.0
	GPT-4o	65.1	64.6	51.8	57.1	58.1	52.8
	CU-MAM^{rel}	**75.8**	**76.8**	**62.5**	**78.4**	**77.8**	**75.2**

Table 4: CU-MAM performance and comparison approaches.

4.6 Error Analysis

We categorise the error types in CU-MAM versus the baselines as "Jump-to-Conclusion," "Reversed Connection," and "Non-relational ADU Link" errors, all affecting sequential and logical coherence (see examples in Appendix E). The "Jump-to-Conclusion" Error occurs, when an ADU A is linked directly to ADU C bypassing the intermediary ADU B. The "Reversed Connection" error happens when the direction of an AR is incorrect, and the "Non-relational ADU Link" error arises when ARs are formed between unrelated ADUs.

Misclassification errors are reduced in both local and global structures, highlighting CU-MAM's effectiveness in maintaining coherence at both levels. Analysis of 50 argument maps shows that 61% of the baseline's misclassifications occur within the same local structure, compared to just 12% for CU-MAM, resulting in a 77.4% reduction in errors. While CU-MAM is more effective at reducing local structure errors, it also reduces global misclassifications, though most errors still arise from global structure issues, such as incorrect connections between cross-local structures or ADUs outside the argument graph. Moreover, CU-MAM reduces "Jump-to-Conclusion" errors to 16% (down from 56% in the baseline) and cuts "Reversed Connection" errors by 32%, demonstrating its ability to preserve logical and sequential flow.

4.7 Ablation study

Config	ACTC		ARTC	
	ID	CD	ID	CD
Baseline	72.1	53.7	50.5	44.7
CU-MAM^{-G}	76.7	63.5	58.7	54.5
CU-MAM^{-L}	74.8	60.5	56.2	52.1
CU-MAM$^{\text{-Att}}$	75.2	64.2	60.1	53.1
CU-MAM$^{\text{Att-only}}$	73.5	59.8	56.5	50.8
CU-MAM^{Full}	78.5	66.5	60.3	57.8

Table 5: F1-scores of baseline and CU-MAM configurations in ID and CD, averaged across the dataset for ACTC and ARTC.

We analyse the contributions of local-structure prediction and global-structure prediction by evaluating CU-MAM without local-structure prediction (CU-MAM$^{\text{-L}}$) and without global-structure prediction (CU-MAM$^{\text{-G}}$). We also analyse the impact of Macro-Attention by evaluating CU-MAM without the Macro-Attention (CU-MAM$^{\text{-Att}}$). These configurations are compared against: (a) the baseline performance, computed as the average of V-SeqCls and V-ArgC, and (b) the full CU-MAM (CU-MAM$^{\text{Full}}$), which includes local-structure prediction, global-structure prediction and Macro-Attention. These configurations of CU-MAM are based on BigBird and absolute positional embeddings, since it achieved the highest performance (See Section F.1 for more details). We also assess the impacts of the two types of positional information and the effectiveness of absolute versus relative embeddings (see Section F.2 for more details).

Local vs. Global Structure Prediction. As shown in Table 5, both CU-MAM$^{\text{-L}}$ and CU-MAM$^{\text{-G}}$ outperform the baseline, highlighting the effectiveness of each structure prediction on its own. However, either local-structure or global-structure prediction can not achieve the performance level of CU-MAM$^{\text{Full}}$ individually, confirming their complementary benefits. CU-MAM^{-L} performs worse than CU-MAM^{-G}, suggesting that local-structure prediction has a greater impact than global-structure prediction.

Macro-Attention Layer (MAL). In CU-MAM$^{\text{-Att}}$ configuration, local and global structures are predicted using a stack of feedforward layers instead of MAL, applied to the output of argument structure encoder, ensuring the same parameter count for a fair comparison with CU-MAM$^{\text{Full}}$. We also evaluate CU-MAM with the MAL but without the auxiliary tasks (CU-MAM$^{\text{Att-only}}$), to isolate the effect of MAL. In CU-MAM$^{\text{Att-only}}$, the MAL is used only for computing the cross-attention that contextualises the ADU-pair. As shown in Table 5, ablating MAL results in a performance drop compared to CU-MAM$^{\text{Full}}$, although it still outperforms the baseline. However, CU-MAM$^{\text{Att-only}}$ performs worse than CU-MAM$^{\text{-Att}}$, emphasising MAL is more effective when combined with auxiliary tasks.

Positional Encoding: We evaluate the impact of order embedding (O) and participant transition embedding (P) individually using absolute (Abs) and relative (Rel) positional encodings. Table 6 shows the performance with each positional feature ablated, highlighting the reduction in performance when each feature is removed (e.g., P^- represents CU-MAM without P). As can be seen from the Table, ablating O results in a greater performance drop compared to ablating P on both monological and dialogical datasets, with P showing no impact on the monological dataset. In average, Rel based configurations outperform Abs configurations.

Config	Monologue	Dialogue
Full (Abs)	43.3	74.4
Full (Rel)	44.5	76.1
O^- (Abs)	42.1	71.3
O^- (Rel)	42.4	71.6
P^- (Abs)	43.1	72.4
P^- (Rel)	44.3	72.7

Table 6: F1-scores for configurations without P, O on monological and dialogical datasets in the ID evaluation. The results are averaged across the dataset on ARTC.

5 Conclusion

This work introduces CU-MAM, the first approach modeling AM tasks as a function of macro-structure to capture coherence. By leveraging structural representations, it models logical and sequential argument flow, capturing local and global dependencies. CU-MAM achieves significant performance gains over baselines and comparison approaches, setting new SOTA results across datasets. Its strong cross-dataset adaptability overcomes domain adaptation challenges where existing SOTA models struggle, demonstrating its ability to generalise across diverse argumentation structures.

Limitations

Despite its merits, the CU-MAM approach has the following limitations:

Limited Applicability to Other NLP Tasks: The participants transitions features and local-structure encoding are specifically designed for argumentation tasks. As such, their applicability to other NLP tasks that do not involve argumentative structures is limited.

Pre-Training Objectives Not Addressed: Although the evaluation focuses on fine-tuning for leveraging macro-structural features, it does not address the training objectives that could be employed during the pre-training phase of LLMs to better integrate these features.

Interpretability and Explainability: The explanations for the model's performance are based on empirical results, ablation studies, and error analysis. While these analyses are valuable, additional techniques such as attention mechanism analysis could provide a more comprehensive understanding of model behavior.

References

He Bai, Peng Shi, Jimmy Lin, Yuqing Xie, Luchen Tan, Kun Xiong, Wen Gao, and Ming Li. 2021. Segatron: Segment-aware transformer for language modeling and understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12526–12534.

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 (Volume 1: Long Papers)*, pages 6354–6364.

Jianzhu Bao, Yuhang He, Yang Sun, Bin Liang, Jiachen Du, Bing Qin, Min Yang, and Ruifeng Xu. 2022. A generative model for end-to-end argument mining with reconstructed positional encoding and constrained pointer mechanism. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10437–10449.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Shaked Brody, Uri Alon, and Eran Yahav. 2021. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*.

Katarzyna Budzynska and Chris Reed. 2011. Whence inference. *University of Dundee Technical Report*.

Shuyang Cao and Lu Wang. 2022. Hibrids: Attention with hierarchical biases for structure-aware long document summarization. *arXiv preprint arXiv:2203.10741*.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. *arXiv preprint arXiv:1704.06104*.

James B Freeman. 2011. *Dialectics and the macrostructure of arguments: A theory of argument structure*, volume 10. Walter de Gruyter.

Debela Gemechu and Chris Reed. 2019. Decompositional argument mining: A general purpose approach for argument graph construction. In *Proceedings of the 57st Annual Meeting of the Association for Computational Linguistics*, pages 1341–1351.

HP Grice. 1975. Logic and conversation. *Syntax and Semantics*, 3:43–58.

Barbara J Grosz, Aravind K Joshi, and Scott Weinstein. 1995. Centering: A framework for modelling the local coherence of discourse.

Barbara J Grosz and Candace L Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3):175–204.

Annette Hautli-Janisz, Zlata Kikteva, Wassiliki Siskou, Kamila Gorska, Ray Becker, and Chris Reed. 2022. Qt30: A corpus of argument and conflict in broadcast debate. In *Proceedings of the 13th Language Resources and Evaluation Conference*, pages 3291–3300. European Language Resources Association (ELRA).

Haoyu He, Markus Flicke, Jan Buchmann, Iryna Gurevych, and Andreas Geiger. 2024. Hdt: Hierarchical document transformer. *arXiv preprint arXiv:2407.08330*.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2024. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36.

Zlata Kikteva, Alexander Trautsch, Patrick Katzer, Mirko Oest, Steffen Herbold, and Annette Hautli. 2023. On the impact of reconstruction and context for argument prediction in natural debate. In *Proceedings of the 10th Workshop on Argument Mining*, pages 100–106.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

John Lawrence and Chris Reed. 2020. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818.

Yang Liu, Jiaxiang Liu, Li Chen, Yuxiang Lu, Shikun Feng, Zhida Feng, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2022. Ernie-sparse: Learning hierarchical efficient transformer through regularized self-attention. *arXiv preprint arXiv:2203.12276*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Karen Elizabeth Lochbaum. 1994. *Using collaborative plans to model the intentional structure of discourse*. Harvard University.

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*, pages 3259–3266.

Gaku Morio, Hiroaki Ozaki, Terufumi Morishita, and Kohsuke Yanai. 2022. End-to-end argument mining with cross-corpora multi-task learning. *Transactions of the Association for Computational Linguistics*, 10:639–658.

Philippe Muller, Stergos Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained decoding for text-level discourse parsing. In *Proceedings of COLING 2012*, pages 1883–1900.

R OpenAI. 2023. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2:13.

Joonsuk Park and Claire Cardie. 2018. A corpus of erulemaking user comments for measuring evaluability of arguments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Andreas Peldszus and Manfred Stede. 2015a. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948.

Andreas Peldszus and Manfred Stede. 2015b. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948.

Andreas Peldszus and Manfred Stede. 2016. Rhetorical structure and argumentation structure in monologue text. In *Proceedings of the Third Workshop on Argument Mining*, pages 103–112.

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

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2016. Here's my point: Joint pointer architecture for argument mining. *arXiv preprint arXiv:1612.08994*.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2019. Classification and clustering of arguments with contextualized word embeddings. *arXiv preprint arXiv:1906.09821*.

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

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Wei Song, Ziyao Song, Ruiji Fu, Lizhen Liu, Miaomiao Cheng, and Ting Liu. 2020. Discourse self-attention for discourse element identification in argumentative student essays. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2820–2830.

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

Maite Taboada and William Mann. 2006. Rhetorical structure theory: Looking back and moving ahead. *Discourse studies*, 8(3):423–459.

Stephen Toulmin. 1958. The uses of argument, cambridge univ.

Lisa deMena Travis. 1984. *Parameters and effects of word order variation*. Ph.D. thesis, Massachusetts Institute of Technology.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jacky Visser, Barbara Konat, Rory Duthie, Marcin Koszowy, Katarzyna Budzynska, and Chris Reed. 2019. Argumentation in the 2016 us presidential elections: annotated corpora of television debates and social media reaction. *Language Resources and Evaluation*, pages 1–32.

Benyou Wang, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. 2019. Encoding word order in complex embeddings. *arXiv preprint arXiv:1912.12333*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B Dolan. 2020. Dialogpt: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278.

A Experiment Setup

A.1 Training Procedure

Hyper-parameters: We employ Adam optimisation (Kingma and Ba, 2014) to minimise the cost function, using a learning rate of 2×10^{-5} and categorical cross-entropy loss and a batch size of 16. Experimental results are reported based on the average of three runs with different random seeds.

Gradient Clipping: To prevent exploding gradients during training, we applied gradient clipping. We used a maximum gradient norm (`max_grad_norm`) parameter to determine the threshold for gradient clipping.

Warm-up and Learning Rate Schedule: We employ a linear warm-up strategy for the learning rate. The number of warm-up steps is set to 10% of the total training steps. Following the warm-up phase, the learning rate schedule is determined by a lambda function. This function linearly increases the learning rate during the warm-up phase and decreases it linearly thereafter.

A.2 Input Setup

Except the V-SeqClas configurations, the entire argument along with the pair of ADUs is provided to the model.

The **Input Format:**"{Argument} [EG] {premise} [SEP] {conclusion}", where Argument = {ADU1 [SEP] ADU2 [SEP] ... ADUn}, with n representing the number of ADUs in the argument.

Extracting Relevant Argument: When the entire argument exceeds the maximum sequence length allowed by the underlying LM, a relevant span of the argument is extracted that includes both the premise and conclusion while staying within the length limit. This process is carried out as follows:

1. **Length Calculation:** The argument, premise, and conclusion are tokenized using the model's tokenizer. The total length is then calculated by summing the tokens for the premise, conclusion, argument, and special tokens (`[CLS]` and `[SEP]`).

2. **Span Selection:**
   - If the total length is within the model's maximum sequence limit, the entire argument is concatenated with the premise and conclusion.
   - If the total length exceeds the limit:
     - The positions of the premise and conclusion within the argument are identified, and a span is selected that includes both, along with additional surrounding context, ensuring the total length fits within the limit.
     - If including the span involving both the premise and conclusion exceed the maximum limit, start with the premise, expand the span towards the conclusion until the size constraint is met, and append the conclusion to the argument span.

Maximum Number of ADUs in an Argument: We set the maximum number of ADUs to 128 for computational efficiency.

A.3 Base LM

Both for the baselines and the CU-MAM configurations, we utilise the HuggingFace implementation of RoBERTa[1], BigBird [2]. In the baseline setup (both with and without argument context), we fine-tune the models based on the output of the [CLS] token from the final output layer. Similarly, ADU embeddings in the CU-MAM configurations are obtained from RoBERTa and BigBird.

B CU-MAM Architecture

B.1 ADU Embedding

We utilise pre-trained LMs (Liu et al., 2019; Radford et al., 2019; Zhang et al., 2020) to obtain contextualised token embeddings $\mathbf{H} \in R^{n \times d}$ for the

[1] https://huggingface.co/docs/transformers/en/model_doc/roberta

[2] https://huggingface.co/docs/transformers/en/model_doc/big_bird

entire input where n is the input length and d is the hidden size of the model. ADUs are identified within the sequence using the special separator token ([SEP]). To obtain embeddings for each ADU, we apply mean pooling over the token embeddings within each ADU. Let $\mathbf{H}_i \in R^{l_i \times d}$ represent the token embeddings for the i-th ADU, where l_i is the length of the i-th ADU. The ADU embedding $\mathbf{ADU}_i \in R^d$ is computed as:

$$\mathbf{ADU}_i = \frac{1}{l_i} \sum_{j=1}^{l_i} \mathbf{H}_{i,j}$$

The resulting set of ADU embeddings forms a matrix $\mathbf{A} \in R^{m \times d}$, where m is the number of ADUs.

B.2 Positional Encoding

We experiment with both fixed and relative positional embeddings. For absolute positional embeddings, we employ the sinusoidal position signal, following the approach introduced by the Transformer model (Vaswani et al., 2017). For relative positional embeddings, we adopt the method proposed by Shaw et al. (2018), which encodes the relative distances between ADU in the argument, $a_{ij} = e_{j-i}$, where e represents the learnable embeddings and $j - i$ indicates the relative distance between ADU j and ADU i. We leverage dual positional embeddings to incorporate the two types of positional information: the index representing the order of each ADUs within the argument (ADU order embedding) and the participant transition embedding. Both approaches are further explained below.

Absolute Positional Encoding. The embedding of an ADU, denoted as $\mathbf{ADU}_i$, is enhanced with absolute positional information by incorporating both order embeddings and participant transition embeddings. This process involves the following steps:

1. **Sinusoidal Function for Embeddings:** Consistent with the approach used in standard Transformers, sinusoidal functions are employed to generate embeddings for argument flow ($\mathbf{T}_i$) based on both ADU order ($\mathbf{O}_i$) and proponent-opponent transitions ($\mathbf{P}_i$):

$$\mathrm{T}_{(index,2i)} = \sin\left(\frac{index}{10000^{2i/d_{\text{model}}}}\right)$$

$$\mathrm{T}_{(index,2i+1)} = \cos\left(\frac{index}{10000^{2i/d_{\text{model}}}}\right)$$

where $index$ denotes the position of the ADU and d_{model} is the dimensionality of the model. This method applies to both ADU order and participant transition embeddings, providing a unified approach for incorporating positional information.

Each ADU is represented by fusing its ADU embedding ($\mathbf{ADU}_i$), order embedding ($\mathbf{O}_i$), and participant transition embedding ($\mathbf{P}_i$) to form a unified representation of ADU ($\mathbf{ADU}'_{\text{abs}}$) . A matrix $\mathbf{A}_{\text{abs}}$ of size $n \times d$ is formed, where n is the number of ADUs in the argument and d is the embedding dimension:

$$\mathbf{ADU}_{i,j}{}^{\text{abs}} = \text{ADU}_i + \mathbf{O}_i + \mathbf{P}_i$$

2. **Relative Positional Encoding.** The attention mechanism adjusts the attention scores $\mathbf{A}'_{i,j}$ to integrate relative distances on the fly:

$$\mathbf{ADU}_{i,j}{}^{\text{rel}} = \text{softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_j^\top}{\sqrt{d_k}} + \mathbf{R}^{\text{O}}_{i,j} + \mathbf{R}^{\text{P}}_{i,j}\right)$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query, key, and value matrices, respectively, derived from the ADUs embeddings. $\mathbf{R}^{\text{O}}_{i,j}$ represents the embeddings of the relative order information and is given by, $\mathrm{R}^{\text{O}}_{i,j} = \mathbf{W}^{\text{O}}(\text{pos}_i - \text{pos}_j).\mathbf{W}^{\text{O}}$ is the learnable weight matrix for ADU positions, and O_i and O_j are the index reflecting the order of the ADUs i and j within the argument. $\mathbf{R}^{\text{P}}_{i,j}$ represents the relative embeddings for participant transition and is given by, $\mathrm{R}^{\text{P}}_{i,j} = \mathbf{W}^{\text{P}}(\mathrm{P}_i - \mathrm{P}_j).\mathbf{W}^{\text{P}}$ is the learnable weight matrix for turn number, and P_i and P_j are the transition numbers of ADUs i and j within the argument.

B.3 GPT for AR Prediction

B.3.1 Experimental Settings

We utilise the chat completion configuration of GPT-4o for the three tasks.

(a) **Configurations:** We use GPT-4o and set a maximum token limit of 2048, a temperature of 0.7, a top-p probability of 0.9.

(b) **Prompts Strategy:** We employ few-shot prompts, where specific examples are provided as part of the instruction. We

create prompt templates that include instructions and two examples randomly selected from a list of examples. An example of a prompt tamplate for the ARTC task is shown below.

```
You are a 3-class classifier model tasked
with assigning a label to the argument
relation between two argument units
(argument 1 and argument 2).
Classify the following  pair of arguments,
argument 1: {ADU_1}
argument 2: {ADU_2},
into:
"support" (if argument 1 supports
argument 2),
"contradict" (if argument 1 attacks
argument 2),
and "None" (if no argument relation exists
between argument 1 and argument 2).
Please enter:
1 - for support,
2 - for contradict,
0 - for None relation.
Examples from each argument
relation types are provided below:
Example 1: the argument relation between
the argument "people feel, when they have
been voicing opinions on different
matters, that they have been not
listened to", and
the argument "people feel that they have
been treated disrespectfully on all
sides of the different arguments and
disputes going on"
is support, and  hence prediction label is 1.
Example 2: The argument relation between
"there would be no non-tariff barriers
with the deal  done with the EU" and
the argument "there are lots of
non-tariff barriers
with the deal done with the EU"
is contradiction, and
hence prediction label is 2.
```

Note: We use the actual examples to show support and contradiction relations, which should be a placeholder variable in the final prompt template.

C Macro-Structure

An argument is a coherent arrangement of utterances organised in a specific order (Grosz and Sidner, 1986; Toulmin, 1958; Freeman, 2011). Freeman (2011) propose a framework describing how these utterances collectively contribute to natural language argumentation, particularly focusing on their supportive roles and structural patterns, termed as "macro-structure". This framework encompasses techniques such as divergent, convergent, linked, and serial reasoning, which illustrate how reasons combine to support conclusions. It underscores the significance of understanding the entire sequence of ideas within an argument, including claims, challenges, responses, and counter-responses, to establish coherent structure.

Coherence within discourse can be viewed at two levels: **local coherence** and **global coherence**. Local coherence refers to coherence among the utterances in a segment of an argument, while global coherence refers to the coherence spanning segments (Grosz and Sidner, 1986; Grosz et al., 1995). Grosz and Sidner argue that the coherence depends on the intentional structure of discourse addressed via the overall DP and DSP (Grosz and Sidner, 1986; Grosz et al., 1995). These intentions are reflective of the speaker's goals, akin to Gricean conversational implicatures (Grice, 1975). In a multi-party discourse, the DSP for a given segment aligns with the intention of the conversational participant initiating that segment (Lochbaum, 1994). Freeman (2011) models these interactions as the interplay between the proponents and opponents, showing how proponents assert and address opponents' challenges, forming a chain of reasoning and highlighting the importance of tracing these transitions for understanding the argument.

IAT (Budzynska and Reed, 2011) offers a framework representing how argument structure is linked to the intentional structure and the dynamics within dialogue structure. In essence, IAT offers a macro-structural analysis by representing the intentional structure and illocutionary dynamics within argumentative discourse, by linking dialogical moves to their communicative intentions and illo-

Algorithm 1 Extract Local-Structures from Argument Map

Require: Argument map represented as nodes and edges, with each node categorised as ADU, and AR
Ensure: List of local-structures
Initialise an empty list to store local-structures: $local\_structures$
Identify nodes corresponding to AR Nodes in the argument map
for each ADU Node in the argument map **do**
Perform an upward traversal to identify the chain of ADUs leading to the AR
Perform a downward traversal to identify the chain of ADUs following the AR Node
Mark the start of each local-structure in the upward traversal by identifying nodes without inward connections
Mark the end of each local-structure in the downward traversal by identifying nodes without successors
Include all chains of ADUs between the start and end node
Add the identified local-structure to $local\_structures$
end for
return $local\_structures$

cutionary forces. For example, Figure (1b) illustrates participant interactions alongside argument structures, showcasing diverse dialogue moves such as "Asserting", "Arguing", "Questioning", "Illocuting", and "Restating" (Budzynska and Reed, 2011). Annotated corpora, such as the corpus of US presidential debate 2016 (Visser et al., 2019) annotated following such framework, exemplify how dialogical interactions unfold as a series of moves, each mapped to a structural element within the argument graph. Although these dynamics are common in dialogue, similar conceptualisations apply to monologue, where a speaker delivers multiple utterances to an audience (Grosz et al., 1995).

D Local Structures Extraction from Argument Map

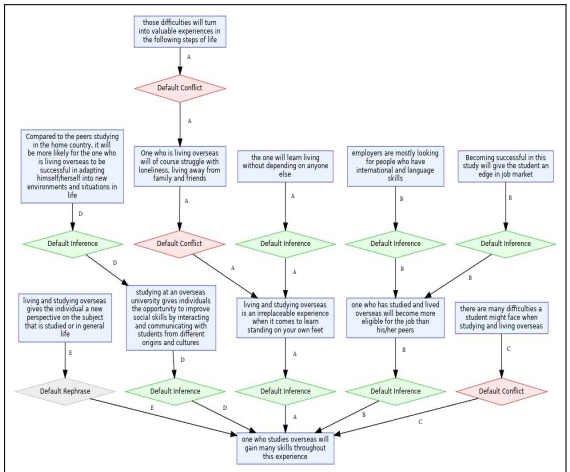
We navigate through argument following an upward traversal to identify the chain of ADUs leading to the AR node and a downward traversal to identify the chain of ADUs following the AR node. The algorithm marks the end of each local-structure in the upward traversal by identifying nodes without inward connections and in the downward traversal by identifying nodes without successors. It includes all chains of ADUs that end at the same node to form the local-structure.

Local-structures are segments of the argument map that represent coherent chains of ADUs leading to and following an AR. We present Algorithm 1 to outline the procedure for extracting local-structures from a global argument map. The algorithm takes as input the argument map represented as nodes and edges, where each node represents ADUs and the ARs. The relations between ADUs are presented based on the edges between the ADU and AR nodes. The algorithm generates a list of local-structures that are pertinent to the respective ARs within the argument map. To evaluate the correctness of local structures, two annotators assessed each local structure produced by the algorithm as either correct or incorrect. A local structure is considered correct if it aligns with the expected argument segment. The annotators are provided with a guideline that describes an argument segment. The inter-annotator agreement, measured with the Kappa statistic, was 0.78, indicating substantial agreement.

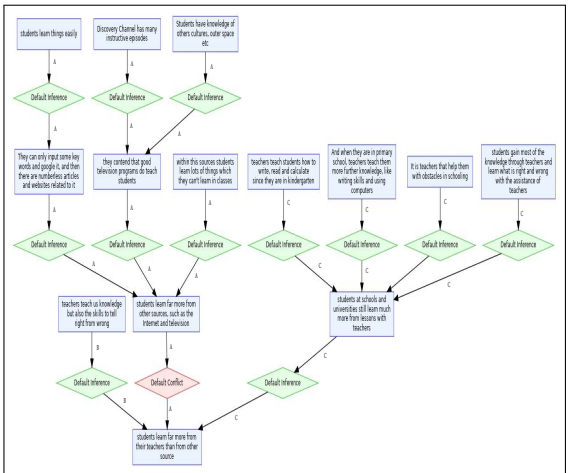
For illustrative purposes, Figure 3 presents several examples showcasing argument maps that feature multiple local-structures. In these examples, the local-structures are annotated with numerical labels. Each number used for annotation corresponds to a distinct local-structure. ARs that share the same numerical label are part of the same local-structure.

E Error Analysis

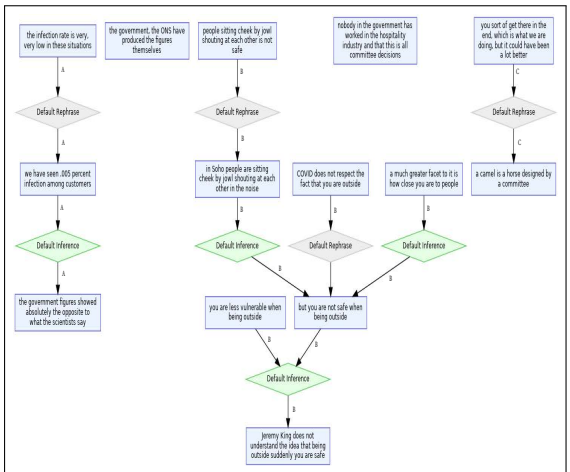
Figure 4 presents an example of an argument map generated by the baseline model. In this map, argument relations are labeled with



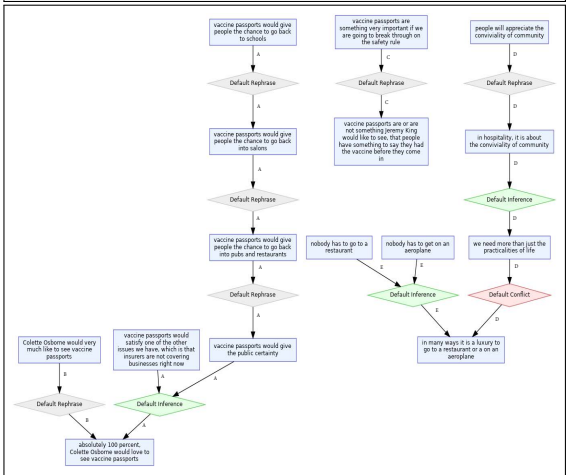
(a) An argument structure with 5 segments (A,B,C,D and E)



(b) An argument structure with 3 segments (A,B and C)



(c) An argument structure with 3 segments (A,B and C



(d) An argument structure with 5 segments (A,B,C,D and E)

Figure 3: An example of argument structures involving multiple segments. ADUs are logically interconnected via AR to form coherent argument structure. Figure (a) and (b) are taken from AAEC, while (c) and (d) are taken from QT30. As can be seen from the figure, (a) and (b) forms one complete graph while (c) and (d) are scattered into multiple disconnected graphs forming islands of argument segments.

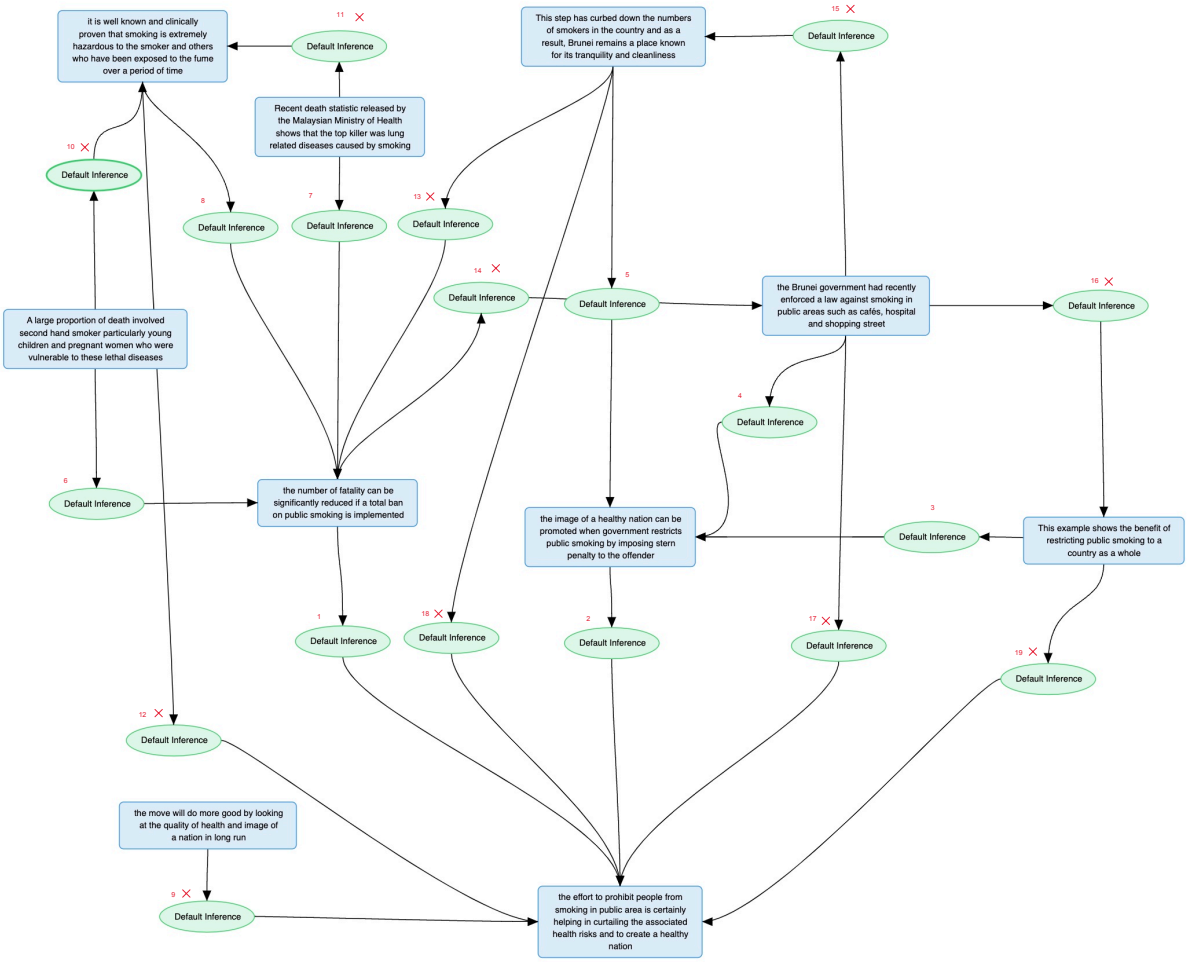


Figure 4: Example of error analysis. The argument map displays relations with arbitrary numbering, where incorrect predictions are marked with an (x) symbol.

numbers, and incorrect AR predictions are highlighted with an (x) symbol. The figure provides a visual representation of the errors made by the baseline model, allowing for a clearer understanding of the error types in AR predictions.

F Ablation Study Setup

In this section, we outline the setup for the ablation study, which aims to assess the impact of different components within the CU-MAM architecture. By systematically removing specific components or features, we compare the resulting configurations against both the complete CU-MAM model (upper-bound performance) and the baseline (lower-bound performance). The study evaluates the effect of key components, including local-structure prediction, global-structure prediction, the Macro-Attention Layer (MAL), and positional information in various configurations of CU-MAM.

F.1 Local and Global Structure prediction and MAL Ablation

We evaluate several configurations of the CU-MAM architecture, each of which targets the ablation of a specific component:

- **CU-MAM^{-L}**: This configuration removes the local-structure prediction, retaining only the global-structure prediction and the Macro-Attention Layer (MAL).
- **CU-MAM^{-G}**: This configuration removes the global-structure prediction, retaining the local-structure prediction and MAL.
- **CU-MAM^{-Att}**: This configuration removes the MAL and replaces it with a

stack of feedforward layers (with a comparable number of parameters to MAL) for local and global structure predictions. This substitution allows a fair comparison of model size between the two configurations.

- **CU-MAM$^{\text{Att-only}}$**: This configuration isolates the effect of MAL by using it only to compute the cross-attention between the pair of ADUs under consideration, without involving any auxiliary tasks. In this setup, the two classifier layers predicting the local and global structures are removed, allowing for an evaluation of MAL's impact without the auxiliary tasks.
- **CU-MAM$^{\text{Full}}$**: This configuration incorporates all components: local-structure prediction, global-structure prediction, and the Macro-Attention Layer, representing the full CU-MAM model.

For Simplicity, the following points outline the setup for the local and global structure prediction and MAL ablation study:

- we consider ACTS and ARTC task.
- The study is based on BigBird based CU-MAM configuration with relative positional encoding since it achieve the highest performance.
- For each configuration an average performance across the dataset is reported for both ID and CD to provide a single performance value for each configuration.

The baseline performance is calculated by averaging the F1-scores of the two baseline, V-SeqCls and V-ArgC, across the entire dataset for each task. Specifically, the baseline for each task is calculated as follows:

$$\text{Baseline}_{\text{ACTC}} = \text{avg}(\text{V-SeqCls}, \text{V-ArgC})_{\text{ACTC}}$$

$$\text{Baseline}_{\text{ARTC}} = \text{avg}(\text{V-SeqCls}, \text{V-ArgC})_{\text{ARTC}}$$

This average is computed over the complete dataset for each respective task.

F.2 Positional Encoding Ablation

We investigate the impact of different positional encoding strategies through an ablation study, where each positional feature is removed individually to assess its effect on model performance. Specifically, we evaluate the use of two types of positional information:

- **Order embedding** (O)
- **Participant transition embedding** (P)

For each type of positional encoding, we test both **absolute** (Abs) and **relative** (Rel) encodings. The ablation study is conducted by systematically removing each of these positional features and comparing the resulting performance against the full configuration, which includes both O and P embeddings. A drop in performance after removing a feature highlights its contribution to the overall model's effectiveness. By comparing the performance of each ablated configuration to the full model, we isolate and quantify the impact of each positional feature. The configurations for ablating the positional information are as follows:

- **Full (Abs)**: The model uses both order and participant transition embeddings with absolute positional encoding.
- **Full (Rel)**: The model uses both order and participant transition embeddings with relative positional encoding.
- O^- **(Abs)**: The model is ablated by removing the order embedding with absolute positional encoding.
- O^- **(Rel)**: The model is ablated by removing the order embedding with relative positional encoding.
- P^- **(Abs)**: The model is ablated by removing the participant transition embedding with absolute positional encoding.
- P^- **(Rel)**: The model is ablated by removing the participant transition embedding with relative positional encoding.

For simplicity, the following points outline the setup for the positional encoding ablation study:

- The study is conducted on the ARTC task.
- The ablation study is based on the BigBird-based CU-MAM configuration,

as it achieved the highest performance in previous experiments.

- For each configuration, the average performance across the dataset is reported for CD evaluations on monologue and dialogue datasets separately, providing a single performance value for each configuration and comparing their effectiveness with respect to the dataset nature.