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 localized structures ensure segment-level coherence by organizing 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 localized 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 immediate proximity to connect ADUs based on their underlying roles and contributions to the DSP of the argument segment (Grosz and Sidner, 1986; Freeman, 2011). Figure 1 illustrates this interplay, showing sequential progressions reflecting the natural flow of the argument (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, which governs the intentional and the sequential flow of ADUs and their ARs. Additional examples of such local structures are presented in Figure 2, with further details on the macro-structure provided in Appendix C.

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 tag-

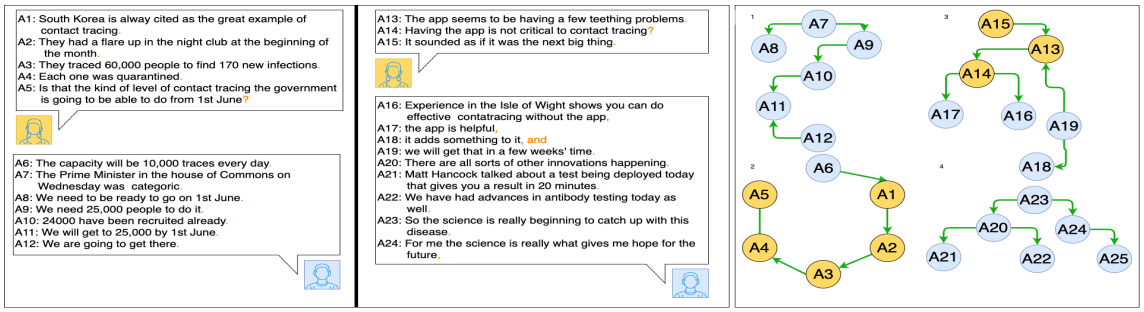


Figure 1: An example of the argument structure decomposed into four local structures (1 to 4). It shows how ADUs and AR are shaped by the intentional structure, where consecutive ADUs may span different segments, and non-consecutive ADUs can share the same segment. The argument is taken from QT30, illustrating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

ging (Eger et al., 2017), or sequence classification (Reimers et al., 2019; Ruiz-Dolz et al., 2021), concentrating primarily on isolated ADU pairs. End-to-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). Position-aware discourse self-attention has also been utilized for identifying discourse elements (Song et al., 2020). While some of these works focus on capturing dependencies between tasks and others incorporate positional information, they fail to address the critical aspect of modeling macro-structures encoding coherence, 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 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 types of the ADUs and the AR between them, contextualized within the relevant local and global structures. This is achieved through a multi-task learning that jointly model auxiliary tasks for classifying graph edges into the relevant local or global. An argument is represented as a graph, with ADUs as nodes and ARs as edges, capturing the complete argument structure. 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 the macro-structural context, the output of this layer 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 dual attention mechanism to capture the local and global argument structures. A multi-task learning framework is used for anchoring ACTC, ARI, 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 struggled to surpass random chance.

2 Related Work

2.1 Argument Mining

Argument Mining has been studied through diverse paradigms, emphasizing 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 using sub-graph matching and evidence

graph models. 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 language models (LMs), employing sequence-pair classification setups (Reimers et al., 2019; Ruiz-Dolz et al., 2021). These configurations primarily focus on the internal structure of ADUs, akin to the "logical form" central to deductive logic, while neglecting the broader macro-structure. Freeman (Freeman, 2011) refers to this as the "micro-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) employs 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 extended sequences.

Although these models provide avenues for encoding macro-structures, their effectiveness in addressing the unique challenges of argumentation's macro-structure remains limited. Fine-tuning these models on Argument Mining (AM) tasks yields suboptimal performance compared to our macro-structure-aware architecture (see Section 4.5 for empirical comparisons). This limitation primarily stems from their reliance on generic structural patterns, which fail to capture the distinct characteristics of argumentation—such as logical relationships, argumentative flows, and the nuanced interplay between ADUs. This underscores the need for architectures that explicitly encode argumentation-specific macro-structures, moving beyond the generic positional and hierarchical encodings commonly employed in existing LMs.

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 (US16) (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 monolingual, while the US2026 and QT30 are dialogical. The datasets CDCP, AAE, QT30, and US16 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 US16 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.

3.1.2 Local Structure

An argument structure is represented as graphs where ADUs and ARs serve as nodes connected by edges. Local Structure identification involves

Data	No_arg	No_ADU	No_AR	No_LOC	Dist_ARs
AAEC	402	6089	5338	3.3	2.6
US16	499	8610	3772	5.1	3.2
QT30	724	11266	3314	7	4.8
CDCP	731	4779	1353	5.6	3.4

Table 1: Summary of dataset showing the number of arguments (No_arg), the average number of ADUs within each argument (No_ADU), the number of support (No_RA), attack (No_CA), the average number of local structures (No_LOC), and distance between ADUs involving AR (Dist_ARs).

both 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, on the other hand, 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 sub-graph—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).

To evaluate the correctness of local structures, two annotators assessed each as either correct or incorrect, yielding a binary evaluation. The inter-annotator agreement, measured with the Kappa statistic, was 0.78, indicating substantial agreement. Table 1 provides a summary of the dataset statistics. Of the argument structures, 73% involve more than one local structure, with 67% containing 2 to 7 local structures. Additionally, 64% of ARs occur between ADUs 1 to 5 positions apart, and 17% involve ADUs 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. 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

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, capturing the full argument structure; (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. Further details of these components are provided in the following sections.

(A) Unified ADU Representation

The representation of each ADU is derived by combining the ADU embedding from a pre-trained with two types of ADU-level positional embeddings for capturing sequential argument flow. Formally, given an argument $A = \{ADU_1, ADU_2, \ldots, ADU_n\}$, the unified representation of $\mathbf{ADU}'_i$ is:

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

where $\mathbf{ADU}_i$ is the ADU embedding obtained by mean pooling over token embeddings from the LM, yielding a fixed-size vector of dimension d. $\mathbf{O}_i$ 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 sine-cosine-based absolute positional encodings (Vaswani et al., 2017) and relative positional embeddings (Shaw et al., 2018). Absolute positional embeddings are added to the ADU embedding, while relative embeddings are incorporated during attention computation. See Section B.2 for details.

(B) Argument Structure Encoder

A Graph Neural Network (GNN) (Brody et al., 2021) is employed to represent the argument structure as a graph $G = (V, E)$, where vertices $V = \{v_1, v_2, \ldots, v_n\}$ correspond to ADUs, and edges $E \subseteq V \times V$ represent ARs between ADUs. The GNN captures relationships between ADUs and encodes the overall argument structure, facilitating the prediction of both local and global structures relevant to a given ADU pair. The graph is constructed using the unified embeddings of ADUs obtained from Equation 1. At each layer of the GNN, the hidden state of node v is updated based on the feature information from its neighboring nodes as follows:

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

where $\mathbf{h}_v^{(k)}$ represents the hidden state of node v at layer k, and $\mathcal{N}(v)$ denotes the neighboring ADUs connected to node v. Each edge $(i, j) \in E$, representing an AR between ADUs v_i and v_j, is encoded as a concatenation of their respective node embeddings.

(C) ADU-Pair Encoder

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

Self-attention with two heads attends to the argument graph from step B, learning the local and global structures relevant to the ADU pair in step C. The attention mechanism is applied to the edge embeddings from the graph network to capture relationships between edges. The outputs from both self-attention heads are summed, passed through a fully connected layer, and used to classify the edges into their respective macro-structural categories (see Section 3.2.2).

To contextualise the ADU type and AR predictions within the broader macro-structural context, the output of the self-attention layer (which encodes the local and global structures from Step D) is fused with the ADU-pair representation from Step C using a cross-attention layer. The query ($\mathbf{Q}_c$) is derived from the ADU-pair encoder output, while the key ($\mathbf{K}_{\text{self-attn}}$) and value ($\mathbf{V}_{\text{self-attn}}$) are projections of the self-attention layer's output. The final representation of the ADU pair, denoted as $\mathbf{R}_{\text{ADU-pair}}$, is obtained by adding the cross-attention output to the original ADU-pair encoder 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 as described below.

(E) Classification Layers

Linear classifiers are used for predicting the types of the ADU pair and AR between them, using the contextualised ADU-pair, $\mathbf{R}_{\text{ADU-pair}}$. We jointly model ARI and ARTC, as in (Bao et al., 2021), while also modeling ARI independently for comparison with studies that treat them separately. The relevant local and global structures are learned through predicting the entire graph edges from step B into local and global. This is achieved through a multi-task setup that treats the structure prediction as an auxiliary tasks. The model trains on the loss function combines task-specific losses and a regularization term:

$$L = L_0 + L_1 + L_2 + L_3 + L_4,$$

where L_0, L_1, L_2, and L_3 represent the losses for ADU type prediction, AR classification, global-structure prediction, and local-structure prediction, respectively. L_4 serves as the regularization term. Since the number of non-AR edges is significantly higher than AR edges, L_2 is computed only for AR edges, excluding non-AR edges. For L_3, we disregard both non-AR edges and AR edges outside the local structure. However, this approach may cause the model to overfit to AR edges. To mitigate this, the regularization term (L_4) is introduced, which uses edge distance editing to penalize deviations from the gold argument structure for both AR and non-AR edges.

3.2.3 Baselines

We establish two baselines using pre-trained LMs: RoBERTa (Liu et al., 2019), reportedly achieving strong performance in AM tasks, and BigBird (Zaheer et al., 2020), for its architectural advantage for capturing long-range dependencies and global context. The first baseline, Vanilla Sequence-Pair Classification (V-SeqCls), fine-tunes the models on concatenated ADU pairs. The second, Vanilla Argument Context (V-ArgC), incorporates the entire argument as context alongside ADU pairs, allowing a direct comparison to CU-MAM. Since both LMs

LLM	Model	ACTC				ARI				ARTC			
		AAEC	CDCP	US16	QT30	AAEC	CDCP	US16	QT30	AAEC	CDCP	US16	QT30
RoBERTa	**V-SeqCls**	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	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}	**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**	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	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}	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 baselines.

are also utilised in CU-MAM to generate ADU embeddings, evaluating them as standalone baselines and within the CU-MAM framework ensures comprehensive and robust comparisons. BigBird, in particular, serves as a strong baseline due to its architecture for modeling global context (see Appendix A.3 for more details).

4 Experiment

4.1 Training setup

The models are trained for six epochs with a batch size of 16. Optimization is performed using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 2×10^{-5}. The primary tasks employ categorical cross-entropy loss, while binary cross-entropy loss is used for the auxiliary tasks predicting the graph edges. Results are averaged over three runs with different random seeds to ensure robustness. Additional details on the experimental setup are provided in Appendix A. The code and dataset used in this work are publicly available at `https://github.com/ANONYMOUS` (link redacted for anonymity).

4.2 Implementation Details

For the AAEC and CDCP datasets, we use the provided training and test data, with 10% of the training set randomly sampled as the validation set. For US16 and QT30, we split the datasets into 70% training, 20% testing, and 10% validation. For AAEC, results are primarily reported at the essay level. Additionally, two paragraph-level results are included for comparison with related work. 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 US16 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 US16 are merged with "Inference" (Support) relation.

ADU embeddings in the CU-MAM configurations are derived from RoBERTa and BigBird to evaluate the robustness of CU-MAM across different LMs.

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. In addition to the average macro score, we report the F1 score for each AC and AR type for the ACTS, ARI, and ARTC tasks 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 component segmentation with ACT, ARI, and ARTC as end-to-am AM setup.

4.5 Results

Tables 3 and 2 compare the performance of CU-MAM, baseline systems, and related approaches across the datasets in both evaluation setups. The main results are reported for the CU-MAM configuration utilising relative positional encoding (**CU-**

LLM	Model	ACTC			ARI			ARTC		
		AAEC	US16	QT30	AAEC	US16	QT30	AAEC	US16	QT30
RoBERTa	**V-SeqCls**	52.1	55.4	48.9	46.2	48.2	47.8	38.9	45.1	44.4
	V-ArgC	47.5	52.4	48.6	38.9	46.9	47.5	36.9	44.2	41.6
	CU-MAM^{rel}	64.6	66.1	**66.4**	56.2	62.0	60.5	50.9	58.5	58.0
BigBird	**V-SeqCls**	55.5	51.4	50.2	43.9	53.6	54.3	40.6	45.4	45.3
	V-ArgC	56.6	53.5	55.5	47.3	56.7	56.5	43.6	46.5	47.1
	CU-MAM^{rel}	**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 baselines.

MAM^{rel}), as it consistently outperforms the configuration with absolute positional encoding. The results in Tables 3 and 2 clearly highlight the effectiveness of incorporating macro-structural features, as evidenced by the performance improvements across all three tasks in both the ID and CD settings.

In-Dataset (ID) Evaluation

In the ID evaluations, CU-MAM consistently outperforms baseline methods that rely solely on fine-tuning pre-trained language models, showing significant improvements across all evaluated tasks. For **ACTC**, CU-MAM achieves an average improvement of 6.6% over V-SeqCls and 8% over V-ArgC. In the case of **ARI**, the average gains are 8.9% and 9%, respectively. Similarly, for **ARTC**, CU-MAM demonstrates gains of 10.9% over Vanilla-S and 11.3% over Vanilla-C. These averages for CU-MAM and baseline methods are computed across BigBird and RoBERTa configurations. These results underscore CU-MAM's ability to effectively leverage macro-structural dependencies, leading to substantial performance improvements when compared to traditional fine-tuning approaches.

Cross-Dataset (CD) Evaluation

As shown in Table 3, CU-MAM demonstrates strong generalisation performance in CD evaluations. For **ACTC**, CU-MAM achieves improvements of 13.8% over V-SeqCls and 15.2% over V-ArgC. For **ARI**, the gains are 12.4% and 12.1%, respectively. Similarly, for **ARTC**, CU-MAM delivers improvements of 13.5% and 14.4%. Notably, CU-MAM consistently surpasses baseline methods, often achieving cross-dataset performance that is comparable to in-dataset evaluations. For instance, when trained on the QT30 dataset and evaluated on the US16 dataset, the BigBird-based CU-MAM model matches the performance of models trained and tested on the same dataset. In contrast, baseline models show near-random performance, highlighting CU-MAM's robust ability to transfer knowledge across different datasets.

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	-
	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	-
	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
	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	-	-	-
	CU-MAM^{rel}	**83.4**	**84.6**	**44.8**	**72.3**	**45.1**	**31.4**
US16	**Ruiz-Dolz et al. (2021)**	-	-	-	-	-	70
	Gemechu and Reed (2019)	-	-	-	64		62
	CU-MAM^{rel}	77.3	75.4	63.8	**80.7**	79.8	**76.6**
QT30	**Kikteva et al. (2023)**	-	-	-	-	-	56.0
	CU-MAM^{rel}	75.8	76.8	62.5	78.4	77.8	**75.2**

Table 4: Performance of CU-MAM against comparison approaches.

CU-MAM outperforms all comparison systems, including the indirect comparison approaches that combine argument segmentation with ACT, ARI, and ARTC identification. The indirect comparisons, should be interpreted cautiously due to differences in task setups. In both ID and CD, the BigBird-based CU-MAM configurations outperform the RoBERTa-based configurations, suggesting BigBird's strength in capturing global contexts.

4.6 Error Analysis

We analyse the error types observed in CU-MAM versus the baseline, categorising them into "Jump-to-Conclusion", "Reversed Connection" Error or "Incorrect Connection" Error (see examples in Appendix 3). The "Jump-to-Conclusion" Error occurs, when an ADU A is incorrectly linked directly to

ADU C without passing through the intermediary ADU B, while the "Reversed Connection" Error occurs when an AR is reversed (when the direction of the relation is wrong), and "Incorrect Connection" Error occurs when ARs are incorrectly established between unrelated ADUs. The analysis of 50 argument maps shows that 61% of the baseline's misclassifications occur within the same local structure, whereas only 12% of CU-MAM's misclassifications are within the same local structure, resulting in a 77.4% reduction in errors. Jump-to-Conclusion errors are reduced significantly in CU-MAM, accounting for 16% of errors compared to 56% in the baseline. Furthermore, CU-MAM reduces errors related to "Reversed Connection" by 32%.

4.7 Ablation study

Config	ACTC		ARTC	
	ID	CD	ID	CD
Baseline	72.1	53.7	50.5	44.7
CU-MAM^{+L}	76.7	63.5	58.7	54.5
CU-MAM^{+G}	74.8	60.5	56.2	52.1
CU-MAM^{+L+G}	78.5	66.5	60.3	57.8

Table 5: Average F1-scores of baseline, CU-MAM with local structure only, global structure only, and their combination on ACTC and ARTC in both ID and CD evaluation setups.

The impact of each macro-structural feature is analysed using the BigBird configuration with relative positional encoding, which achieves the highest performance. Performance gain is calculated as the difference in average F1-scores between CU-MAM and the baselines V-SeqCls and V-ArgC.

Local vs. Global Structure Prediction: Table 5 shows that local structure prediction ($^{+L}$) outperforms global structure prediction ($^{+G}$) across all metrics. Combining both ($^{+L+G}$) achieves the best performance, highlighting their complementary benefits.

Auxiliary Task vs. Attention Layer: Furthermore, we compare the effect of local and global structure prediction as an auxiliary task (Aux^{+}), without the attention layer, and the use of the attention layer only (Att^{+}) without the auxiliary task, specifically for the ARTC task. We use a fully connected feedforward network instead of the attention layer with the same parameter count for a fair comparison. As shown in Table 6, both the auxiliary task ($^{\text{Aux}+}$) and the attention layer ($^{\text{Attn}+}$) improve performance, with the auxiliary task yielding better results. Their combination achieves the highest performance, highlighting their complementary strengths.

Model	Configuration	ACTC-ID	ACTC-CD	ARTC-ID	ARTC-CD
CU-MAM	Aux^{+}	75.2	64.2	60.1	53.1
CU-MAM	Attn^{+}	73.5	59.8	56.5	50.8

Table 6: Ablation study comparing CU-MAM configurations on ACTC and ARTC tasks. ID and CD denote in-domain and cross-domain performance, respectively.

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

Table 7: Average F-1 scores for CU-MAM configurations using absolute (Abs) and relative (Rel) positional embeddings on monological and dialogical datasets in the ID evaluation setup for ARTC.

Positional Encoding: As shown in Table 7, we evaluate the performance of order embedding (O) and participant transition embedding (P) with absolute (Abs) and relative (Rel) positional encodings. On average, relative positional encoding outperforms absolute encoding across both dialogical and monological datasets. The fusion of O and P consistently yields the best results, with O driving stronger improvements, particularly in the dialogical dataset, while P provides no improvement in the monological dataset.

5 Conclusion

This work introduces CU-MAM, the first approach to 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 exceptional cross-dataset adaptability overcomes domain adaptation challenges, where existing SOTA models often fail, highlighting 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.

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.

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 entire input where n is the input length and d is

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

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

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.

participant transition

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, $\mathrm{R}^{\text{O}}_{i,j} = \mathbf{W}^{\text{O}}(\mathrm{pos}_i - \mathrm{pos}_j).\mathbf{W}^{\text{O}}$ 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

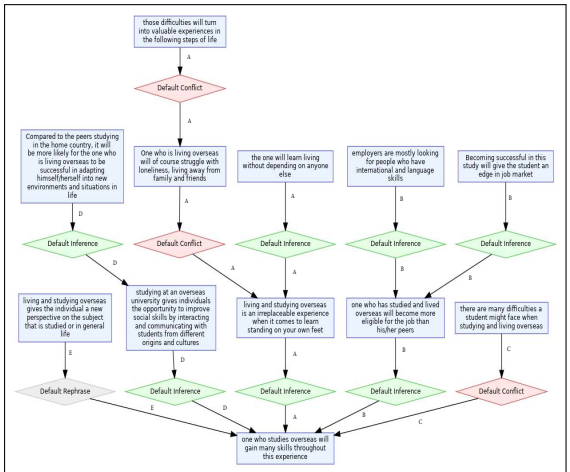
We navigate through argument following an upward traversal to identify the chain of ADUs leading to the AR node and a downward traversal to identify the chain of ADUs following the AR node. The algorithm marks the end of each local-structure in the upward traversal by identifying nodes without inward 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.

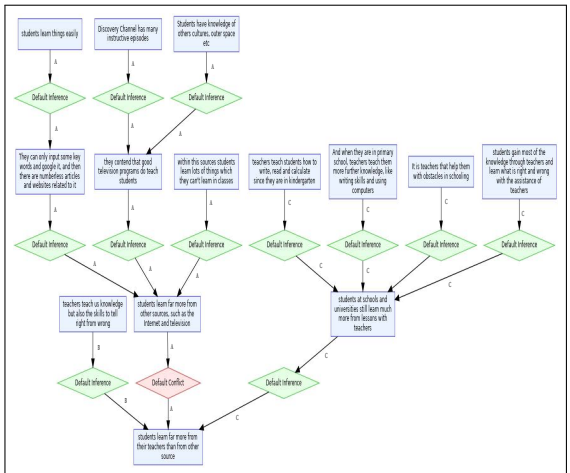
The algorithm generates a comprehensive list of local-structures that are pertinent to the respective ARs within the overarching argument map. Each of these local-structures is identified and cataloged according to their relevance to specific AR in the argument map. For illustrative purposes, Figure 2 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

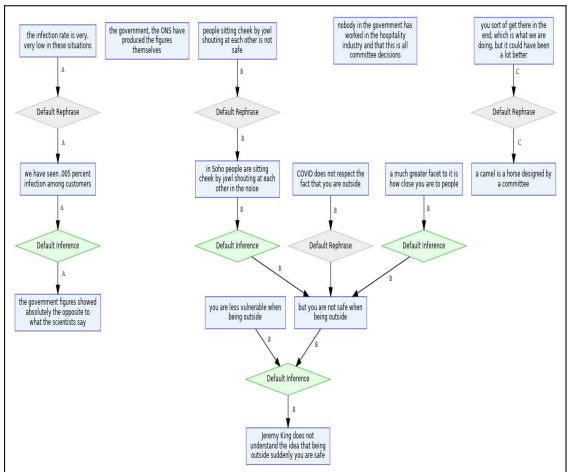
Figure 3 presents an example of an argument map generated by the baseline model. In this map, argument relations are labeled with numbers, and incorrect AR predictions are highlighted with an (x) symbol. The figure provides a visual representation of the errors made by the baseline model, allowing for a clearer understanding of the error types in AR predictions.



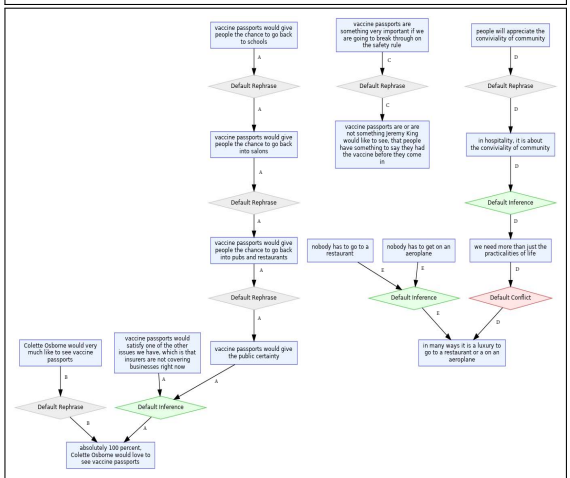
(a) An argument structure with 5 segments (A,B,C,D and E)



(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 2: 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.

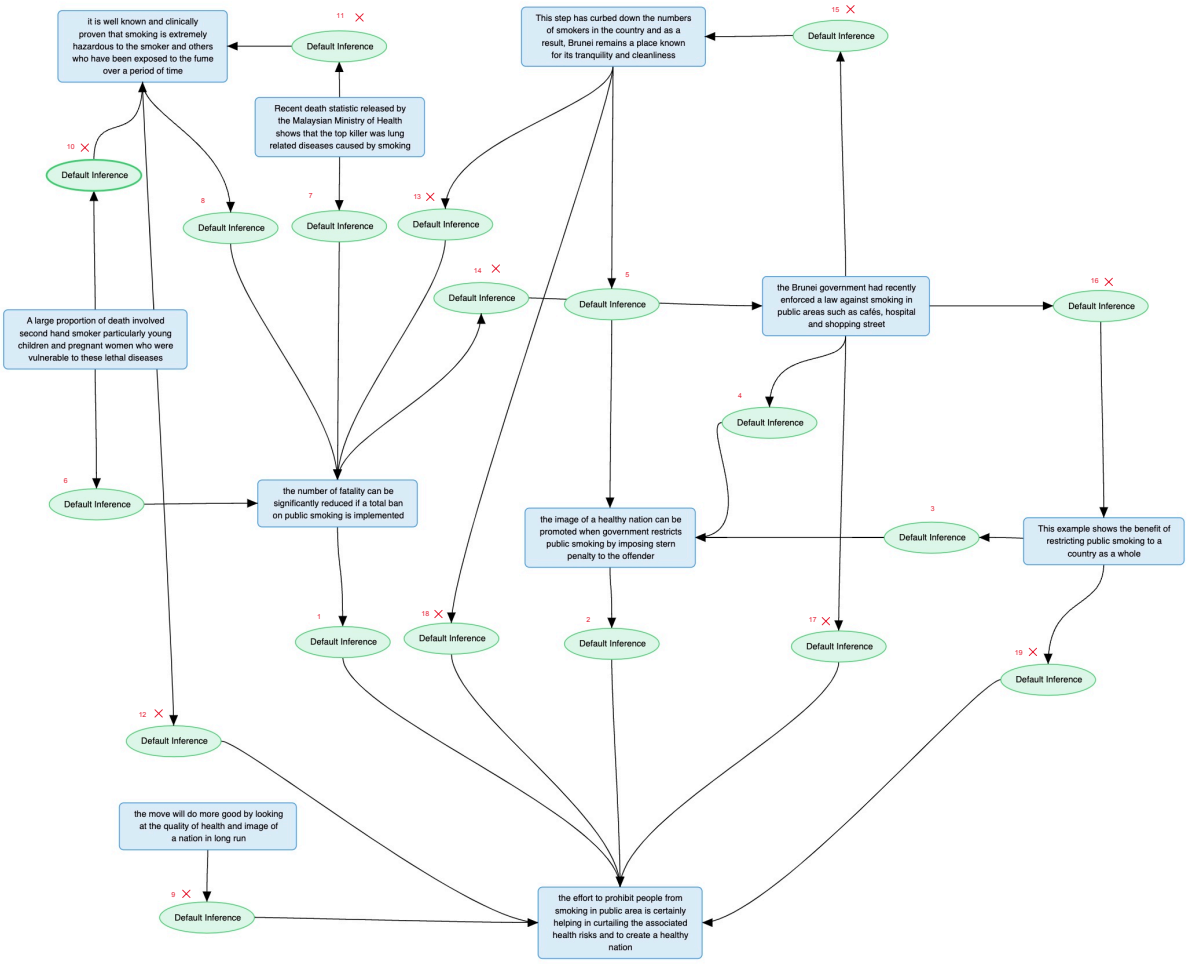


Figure 3: Example of error analysis. The argument map displays relations with arbitrary numbering, where incorrect predictions are marked with an (x) symbol.