Unified Argument Mining Pipeline as Text Generation with LLMs

Anonymous ACL submission

Abstract

An Argument Mining (AM) pipeline takes a text as input and outputs its argumentative structure by successively (i) classifying arguments within the text (ACC task), (ii) identifying argumentative relations between arguments (ARI task), and (iii) classifying these relations (ARC task). Modern LLM-based approaches reformulate each sub-task individually, converting them from text classification to text generation tasks using prompting techniques. In this work, we propose a novel decoder-only LLM approach that addresses the AM pipeline as a unified text generation task, rather than handling the sub-tasks independently. Our unified pipeline approach obviates the need of separate task modeling, mitigates error propagation inherent in successive task modeling, and enhances usability and deployment efficiency. We demonstrate the scalability and robustness of our approach across eight state-of-the-art LLMs and three benchmark datasets (AAEC, AbsRCT, CDCP), respectively. The pipeline achieves competitive or state-of-the-art performance compared to the best separate-task methods in which LLMs are fine-tuned independently on each sub-tasks and error propagation is not taken into account. Overall, our work establishes 'AM pipeline as text generation' as a rigorous, efficient, and scalable paradigm, providing strong baselines for future research.

1 Introduction

In Natural Language Processing (NLP), Argument Mining (AM) focuses on extracting the underlying argumentative structure of a text, enabling downstream reasoning systems, such as legal decision support tools or policy analysis engines, to interpret and act on the extracted arguments. AM has found widespread applications in diverse fields such as legal reasoning, social media discourse and scientific reasoning (Palau and Moens, 2009; Cabrio and Villata, 2018).

The core elements of a discursive text are argument components and the relations between them. Argument components are self-contained assertions that an author makes to substantiate his/her opinion on a contentious topic. Argument relations link pairs of argument components within a text, forming discursive connections that express either support or opposition. A complete argument mining pipeline consists of four inter-connected sub-tasks: identifying argument components in the text (ACI) followed by classification of identified argument components according to their argumentative role (ACC). The third task involves classifying all possible pairs of argument components in text as either related or unrelated (ARI). Finally, each pair of related arguments is classified according to their discursive relation, support or attack (ARC) (Stab and Gurevych, 2017). The AM pipeline outputs a fully parsed argumentative structure, capturing discursive intent and reasoning patterns to reveal how ideas are developed and justified in the text.

LLMs have become ubiquitous in modern NLP, replacing traditional task-specific systems with general-purpose, pre-trained architectures that exhibit strong zero-shot, few-shot, and transfer learning capabilities (Zhao et al., 2023; Wei et al., 2022; Minaee et al., 2025). Built on transformer-based architectures, LLMs are characterized by their deep and wide neural network structures, consisting of hundreds of layers and billions to trillions of parameters. The core innovation underlying LLMs is the self-attention mechanism, which enables the model to capture long-range dependencies in text by dynamically weighting the relevance of each token in a sequence with respect to all others (Vaswani et al., 2017). LLMs are pre-trained on massive corpora of text data using unsupervised learning objectives that allow the models to acquire rich syntactic, semantic, and world knowledge representations (Radford and Narasimhan, 2018; Devlin et al., 2019). The subsequent fine-tuning or

adaptation phase allows these models to specialize in domain-specific downstream tasks, such as machine translation, summarization, question answering, and sentiment analysis.

In this paper, we formulate the Argument Mining (AM) pipeline as a simple, compact, and unified task using fine-tuned decoder-only LLMs (see Figure 1). Our main contributions are as follows:

- We address the three core sub-tasks of the AM pipeline – ACC, ARI, and ARC – within a unified framework that models them as a single text generation task. Our framework employs fine-tuned decoder-only LLMs, rather than encoder–decoder architectures (Bao et al., 2022).
- Our unified formulation employs a single model for all AM sub-tasks and produces structured outputs. This eliminates separate task modeling stages, reduces error propagation across tasks, and enhances usability and deployment efficiency.
- Rather than prompting each AM sub-task independently within a meta-prompt, we propose a novel 2-step prompting strategy that pairs an ACC prompt with a joint ARI–ARC (ARIC) prompt within a meta-prompt. This strategy yields more coherent results in a single inference pass.
- To assess the robustness of our approach across diverse domains – essays, biomedical abstracts, and policy documents – we implement the AM pipeline on three widely used benchmark datasets: AAEC, AbsRCT, and CDCP. For the AAEC dataset, we evaluate our pipeline at both the paragraph and full-essay levels.
- To evaluate the scalability and generalizability of our approach, we conduct experiments with a range of state-of-the-art decoder-only LLMs, including LLaMA (3, 3.1, 3.2), Qwen (2, 2.5), Phi, Falcon and Mistral.

Overall, our unified AM pipeline achieves competitive or state-of-the-art performance on the AAEC, AbsRCT, and CDCP datasets, outperforming or matching the best existing methods that handle AM sub-task separately and disregard error propagation effects across tasks (cf. Figure 1). To ensure reproducibility, our code, prompt templates and other material will be made available in the final version.

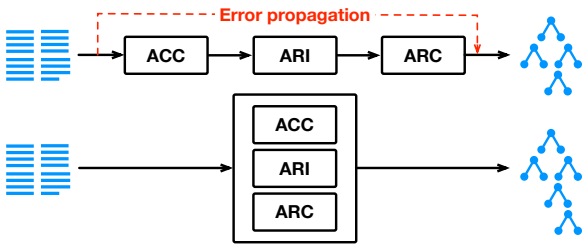

Figure 1: Sequential AM pipeline (top) vs unified AM pipeline (bottom).

2 Related Works

The use of LLMs for modeling Argument Mining sub-tasks as text generation has garnered increasing attention in recent research (Pojoni et al., 2023; Al Zubaer et al., 2023). The common methodology involves reformulating individual AM sub-tasks from text classification tasks into text generation tasks through prompt engineering techniques. For example, in the ACC task, the LLM is provided with the identified argument components (ACs) as part of its prompt and is instructed to generate the corresponding type labels (e.g., 'MajorClaim', 'Claim', or 'Premise').

Liu et al. (2023) frame argument mining as a generative machine reading comprehension task. They experiment with the AAEC dataset, modeling each task separately, and with the AbstRCT dataset, where they merge the ARI and ARC tasks into a single relation identification and classification task (ARIC). Their prompt template consists of a context and an input query concatenated together. The context element includes the complete text, with `'AC'` and `'non_AC'` tags used to delineate argument components from non-argumentative text. The input query consists of a single argument for the ACC task and pairs of arguments for the ARIC task. The generated output sequence consists of a class label and a path representation. The authors use BART-Base (Lewis et al., 2020) for the AAEC dataset and BioBART-Base (Yuan et al., 2022) for the AbstRCT dataset.

Bao et al. (2022) introduce a generative framework for end-to-end AM pipeline in the AAEC and CDCP datasets, reformulating its outputs as a target sequence. The authors use a pre-trained BART-based encoder-decoder architecture augmented with a constrained pointer mechanism. To mitigate order biases inherent to autoregressive gen-

eration, the authors further propose a reconstructed positional encoding method.

Fine-tuning LLMs involves enhancing their pre-trained knowledge with domain-specific supervised learning. To that end, Cabessa et al. (2025) fine-tune popular LLMs for argument mining as a text generation task. In addition to the AAEC and AbstRCT datasets, they also experiment with the CDCP dataset. For the ACC task, their prompt template consists of the complete text with argument components delineated using `<AC>` and `</AC>` tags, and the generated output is a list of predicted component types. For the ARI task, they generate a list of related argument pairs using the same prompt template. For the ARC task, given a list of related argument pairs, the LLM generates a list of predicted relation types. For the joint ARIC task, given the text with delineated argument components, the LLM is prompted to generate a list of triplets, where each triplet consists of a predicted related argument pair and its predicted relation type. For the AAEC dataset, the authors model the AM sub-tasks at both the essay and paragraph levels.

Gemechu et al. (2024) undertake an extensive study of the ARIC (ARI + ARC) task using three strategies: sequence classification, token classification and sequence alignment. Specifically, the authors use ROBERTA-large, DialogGPT (Zhang et al., 2020) and T5 models to address the joint ARIC task for a set of benchmark datasets. They experiment with three predominant model architectures: encoder-only, decoder-only and encoder-decoder. For rigorous evaluation, they undertake both in-dataset and cross-dataset task evaluation. Their results represent state-of-the-art for the ARIC joint task.

We position our work within the AM-as-text-generation paradigm. We improve upon existing approaches by modeling the entire AM pipeline as a unified text generation task. Our approach is more realistic and aligns more closely with real-world applications, as it mirrors the way argumentative discourse naturally unfolds in text – inter-connected and context-dependent – rather than through isolated sub-tasks. This unified generative framework allows the model to simultaneously capture argumentative structure, content, and relationships in a context-aware manner.

3 Materials and Methods

3.1 Datasets

We conduct our experiments on three benchmark datasets for Argument Mining (AM). The statistics of these datasets are provided in Appendix A.

(1) The *AAEC* dataset comprises 402 structured essays on a variety of topics (Stab and Gurevych, 2017). Argument components (ACs) in this dataset are annotated as one of three types: `'MajorClaim'`, `'Claim'`, or `'Premise'`. Argumentative relations (ARs) between these components are labeled as either `'Support'` or `'Attack'`. Notably, premises may only be related to claims or other premises within the same paragraph, whereas claims are related to major claims across different paragraphs.

(2) The *AbstRCT* dataset consists of 650 abstracts from randomized controlled trials (RCTs) selected from PubMed (Mayer, 2020). ACs are annotated using the same three labels as in AAEC dataset: `'MajorClaim'`, `'Claim'` and `'Premise'`, although in practice, major claims and claims are generally grouped into a single class. The ARs are also annotated as either `'Support'` or `'Attack'`.

(3) The *CDCP* dataset consists of 730 user comments on Consumer Debt Collection Practices (CDCP) rule by the Consumer Financial Protection Bureau (CFPB) (Park and Cardie, 2018). Here, ACs are annotated as one of five types, `'Fact'`, `'Policy'`, `'Reference'`, `'Testimony'`, or `'Value'`. ARs between components are labeled as either `'Evidence'` or `'Reason'`.

3.2 AM Pipeline

The AM pipeline consists of three text classification tasks: Argument Component Classification (ACC), Argument Relation Identification (ARI), and Argument Relation Classification (ARC).

ACC: This task involves classifying each argument component (AC) in a text into one of several component types, such as `'MajorClaim'(M)`, `'Claim'(C)`, and `'Premise'(P)`.

ARI: This task involves identifying argument relations (ARs) between pairs of ACs in a text. Specifically, this entails classifying each distinct pair `(i,j)` of ACs as either `'Related'(R)` or `'Non-related'(NR)`.

ARC: This task involves classifying each related pair `(i,j)` of ACs into a specific relation type, such as `'Support'(S)` or `'Attack'(A)`.

ARIC (ARI + ARC): In literature, ARI and ARC tasks have also been addressed jointly. In this joint task setting, each pair `(i,j)` of distinct ACs is classified as either `'Non-related'(NR)`, `'Support'(S)` or `'Attack'(A)`.

We model the AM pipeline as a *unified text generation task* using decoder-only LLMs (see Figure 1). Compared with traditional approaches that treat each sub-task independently (Cabessa et al., 2025), this unified formulation eliminates separate modeling stages and mitigates error propagation inherent in successive task modeling.

To implement the unified AM pipeline, the LLM is prompted to solve the ACC, ARI, and ARC subtasks within a single meta-prompt. A natural *3-step prompting strategy* guides the model to (i) identify argument component types, (ii) determine pairs of related components, and (iii) classify the relation type for each related pair, combining these instructions into a single meta-prompt. Details of the methodology and experimental results are provided in Appendix D.

Inspired by previous works where the ARI + ARC tasks are jointly addressed (Gemechu et al., 2024), we propose an alternative, more efficient *2-step prompting strategy*, in which the LLM addresses the ACC and ARIC (ARI + ARC) sub-tasks within a single meta-prompt. For each input text, the LLM generates two structured JSON outputs: (i) a list of argument component (AC) types, and (ii) a list of AR triplets of the form `(i, j, 'X')`, indicating that AC_i and AC_j are connected by the relation type `'X'` (e.g., support or attack). An illustrative example of the generated output for this unified AM pipeline is provided below. The classification metrics for ACC, ARI, ARC, and ARIC tasks are then computed as described in Section 3.4.

```
{"argument_types":
['M', 'C', 'P', 'P', 'P', 'P', 'C']}

{"argument_relations":
[(7,1,'S'), (2,7,'S'), (3,7,'S'),
 (5,7,'S'), (4,7,'S'), (6,1,'A')]}
```

In the AAEC dataset, premises can only relate to claims or other premises within the same paragraph, while claims connect to the major claim(s) across paragraphs at the essay level. To account for this dynamic, we design an additional prompting strategy in which the LLM is instructed to produce paragraph-level lists of AR triplets, along with an ordered list of stances indicating whether the identified claims support or attack the identified major claims in the text. The output format of this strategy is described in Appendix E.

3.3 Model Fine-Tuning

To address the full AM pipeline on the AAEC, AbstRCT, and CDCP datasets, we fine-tune the following 4-bit quantized LLMs:

- `Meta-Llama-3-70B-Instruct`
- `Meta-Llama-3-8B-Instruct`
- `Meta-Llama-3.1-8B-Instruct`
- `Qwen2.5-7B-Instruct`
- `Qwen2.5-14B-Instruct`
- `Mistral-Nemo-Instruct`
- `Phi-4-mini-instruct`
- `Falcon3-10B-Instruct`

These models reflect a diversity of size and architectural families, allowing us to assess the robustness of our approach.

Our prompt template is described in Appendix C. The prompts are composed of three elements: an instruction, an input, and an output. The *instruction* informs the model about the structure of its input text, defines the tasks to be performed (ACC and ARI + ARC), and specifies the expected format of the output in structured JSON. We also include a demonstration output as part of the instruction. The *input* consists of the text along with its corresponding ordered list of ACs. Finally, the *output* contains the output for the train sample, which consists of lists of AC types and AR triplets (see Section 3.2). During inference, the LLM is prompted to generate this output for a test sample based on the corresponding instruction and input elements.

3.4 Post-Processing

For all datasets, the output lists generated by LLMs are post-processed to compute the classification scores for the ACC, ARI, ARC, and ARIC tasks. The post-processing primarily involves harmonizing the ground truths and predicted argument components (ACs) and relation types to ensure the accurate computation of classification metrics.

ACC Metric: For each test sample, the LLM generates a list of predicted argument types, denoted as `preds`, which is then compared to the list of ground argument types, denoted as `grounds`. If `preds` contains more elements than `grounds`, the

extra elements in preds are removed. Conversely, if preds contains fewer elements than grounds, additional incorrect elements are added to match the length of grounds. Once the lengths of preds and grounds are aligned across all texts, the classification metric for ACC task is computed (F1 score for each class and overall macro-F1 score).

ARI Metric: For each test sample, the LLM generates a list of predicted triplets, where each triplet consists of a pair of related ACs along with its relation type. To compute the ARI metric, we first enumerate all possible pairs (i,j) of distinct ACs in the text. We then create two lists, grounds and preds, by labeling each pair of ACs as either related ('R') or non-related ('NR') based on ground truth annotations and LLM predictions, respectively. After aggregating over all texts, the classification metric over the 'R' and 'NR' classes can be computed.

ARC Metric: To compute ARC metric, we start with the list of ground triplets, grounds, and build the list of predicted triplets, preds, as follows: for each ground triplet (i,j,'X'), if a predicted triplet (i,j,'Y') (based on the same pair (i,j)) exists, it is appended to preds. Otherwise, a triplet with incorrect type, (i,j,'not X'), is appended to preds, where 'not X' denotes the opposite class of X (i.e., 'not X' = 'S' iff 'X' = 'A'). Finally, the labels of grounds and preds are aligned, and the classification metric over 'S' and 'A' classes is computed.

ARIC Metric: The ARI and ARC tasks are performed jointly. To evaluate the metric for this joint task, we proceed as follows. First, we enumerate all possible pairs (i,j) of distinct ACs in the text. Then, we create two lists, grounds and preds, by labeling each pair of ACs as either non-related ('NR'), support ('S'), or attack ('A') based on ground truth annotations and LLM predictions, respectively. Finally, we compute the classification metric for the 'NR', 'S', and 'A' classes.

Note that our unified text generation framework increases the complexity of the ARC task. While the original formulation assumes related AC pairs are given as input and focuses solely on relation type prediction, our approach requires the model to simultaneously generate both the related pairs and their corresponding relation types. Hence, any related pair of ACs missed by the model results in an incorrect relation prediction, leading to reduction in ARC scores compared to those in the literature.

4 Results

We compare our unified AM pipeline approach to the strongest existing methods where each sub-task is addressed separately and error propagation is not taken into account. Implementation details are given in Appendix B.

2-step vs 3-step prompting strategy: We compared the results of the 3-step and 2-step strategies to evaluate their relative effectiveness (Tables 1 and 2). The two strategies yield identical performance on the ACC task (91.2), indicating that the classification of argument components is robust to the prompting strategy. However, the 2-step strategy significantly outperforms the 3-step one on the ARI and ARIC tasks, achieving 85.7 and 72.6 respectively, compared to 79.4 and 67.8 for the 3-step variant. This demonstrates a clear advantage in relation prediction when ARI and ARC are jointly modeled. On ARC, the 3-step strategy outperforms the 2-step one (59.5 vs. 51.7), but these predictions are associated with incorrectly predicted pairs of related arguments, as confirmed by the lower ARI score. Overall, these results validate the effectiveness of the 2-step prompting strategy adopted in our approach.

AAEC dataset: The results for AAEC dataset at the essay and paragraph levels are presented in Tables 2 and 3, respectively. As mentioned in Section 3.4, the poor performance on the ARC task are explained by the fact that, in our joint task framework, the LLM must simultaneously predict both the pairs of ACs that are related and their corresponding relation types. In contrast, the original ARC task formulation assumes the related AC pairs to be provided as input and requires the model to predict only the relation types.

At the essay level, our unified pipeline achieves state-of-the-art (SOTA) performance on ACC and ARI tasks and competitive results on the ARIC task using the LLaMA-3-70B and LLaMA-3.1-8B models (see Table 2). The LLaMA-3-70B model achieves strong SOTA performance and the LLaMA-8B models consistently outperform non-LLaMA models with similar size. These results show that, for the AAEC dataset, joint task learning outperforms separate task learning, suggesting that LLMs can leverage information from one task to enhance performance on another.

At the paragraph level, the LLaMA models also achieve SOTA and competitive performance on

Model	ACC				ARI			ARC			ARIC			
	C	M	P	F1	NR	R	F1	A	S	F1	NR	A	S	F1
LLaMA-3-8B	80.1	95.4	93.4	89.6	98.2	56.7	77.5	20.6	94.4	57.5	98.2	29.7	55.9	61.3
LLaMA-3.1-8B	81.6	95.5	93.9	90.3	98.2	56.6	77.4	24.3	94.6	**59.5**	98.2	34.6	55.2	62.7
LLaMA-3.2-3B	75.7	92.8	92.1	86.8	97.9	48.4	73.1	14.1	94.4	54.3	97.9	23.5	47.3	56.2
Qwen2.5-7B	75.7	92.2	92.6	86.8	98.0	51.4	74.7	17.6	94.5	56.1	98.0	18.9	50.5	55.8
Qwen2.5-14B	81.8	95.0	94.0	90.3	98.3	59.7	79.0	22.4	95.0	58.7	98.3	31.7	59.6	63.2
Mistral-Nemo-2407	84.2	94.4	95.0	**91.2**	98.4	60.4	**79.4**	23.3	94.8	59.0	98.4	45.2	59.8	**67.8**

Table 1: Results of the AAEC dataset at essay level using the 3-step prompting strategy.

the ACC, ARI and ARIC tasks, respectively, although the scores are lower than those at the essay level (see Table 3). In this case, the LLaMA-3-70B model does not achieve top performance across all tasks, and the dominance of the LLaMA models is less pronounced overall. These results are surprising, as the AC and AR types are strongly influenced by their relative positions within paragraphs (Stab and Gurevych, 2017). This suggests that providing hand-crafted, linguistically-informed regularities may not necessarily aid LLMs and could, in fact, impede their effectiveness.

AbstRCT dataset: The results for the three test sets (neo, gla, and mix) of the AbsRCT dataset are reported in Table 4. For this dataset, ARI and ARC are typically addressed jointly (ARIC task), which explains the absence of results for these tasks in the literature. Here again, the poor results on the ARC task are due to its formulation in our framework. In this case, the LLaMA-3-70B model does not consistently outperform others across all tasks. For the neo and mix test sets, our approach achieves the second-best and competitive results on the ACC and ARIC tasks, respectively, while for the gla test set, our approach underperforms relative to the last reported results. The Mistral-Nemo-2407 model performs relatively well across different test sets and tasks, suggesting that this model may be more appropriate for this dataset.

CDCP dataset: The results for the CDCP dataset are presented in Table 5. As with the previous datasets, the results on the ARC task are limited due to the task formulation in our framework. On this dataset, we achieve the second-best almost-SOTA and SOTA results on the ACC and ARI tasks, respectively. On the whole, the unified pipeline approach achieves strong results on this dataset as well.

Discussion: Our LLM-based unified AM pipeline, formulated as a single text generation task, consistently delivers competitive or state-of-the-art performance across the core AM tasks (ACC, ARI and ARIC) and a variety of benchmark datasets representing diverse argumentative sources. For the AAEC dataset, our unified pipeline approach outperforms separate task fine-tuning methods. This improvement can be attributed to several factors. First, knowledge sharing across tasks plays a crucial role, as insights gained from one task provide valuable context that enhances performance on the others. Additionally, the interdependence of the ACC, ARI, and ARC tasks allows the model to leverage task-specific information and shared representations, enhancing overall performance. This joint setup also improves contextualization, enabling the model to better capture global argumentative structures.

Certain models prove more effective than others depending on the dataset type. This performance variability is not always driven by model size; rather, it suggests that the knowledge acquired during pre-training may transfer to downstream tasks with varying efficiency, depending on dataset characteristics.

Overall, the unified AM pipeline offers the practical advantage of relying on a single model and requiring only one inference pass to simultaneously perform all three AM sub-tasks. A structured graphical representation of the argumentative content can then be readily extracted from the LLM's output, as illustrated in Figure 2.

5 Conclusion

In this work, we propose a novel approach that models the AM pipeline as a unified text generation task using decoder-only LLMs. Our 2-step prompting strategy enables open-source and open-weight LLMs to jointly perform argument component classification, relation identification, and relation type classification efficiently, within a single inference pass. Our approach employs a single model for all AM sub-tasks, thus eliminating separate task modeling stages, reducing error propagation across

Model	ACC				ARI			ARC			ARIC			
	C	M	P	F1	NR	R	F1	A	S	F1	NR	A	S	F1
Liu et al. (2023)	-	-	-	89.2	-	-	82.7	-	-	81.0	-	-	-	-
Bao et al. (2021)	-	-	-	88.4	-	-	82.5	-	-	81.0	-	-	-	-
Cabessa et al. (2025)	-	-	-	89.5	-	-	83.5	-	-	**95.9**	-	-	-	-
Gemechu et al. (2024)	-	-	-	-	-	-	-	-	-	-	-	-	-	**78.0**
LLaMA-3-70B	84.9	97.1	94.8	**92.3**	98.2	76.1	**87.2**	25.1	82.6	53.8	98.2	52.4	73.5	74.7
LLaMA-3-8B	82.1	96.4	93.8	90.7	97.9	72.0	84.9	21.0	79.7	50.3	97.9	49.8	69.3	72.3
LLaMA-3.1-8B	82.4	97.0	94.0	91.2	98.0	73.4	85.7	23.0	80.4	51.7	98.0	48.9	70.8	72.6
Qwen2.5-7B	73.6	90.8	91.5	85.3	97.5	65.5	81.5	14.8	73.6	44.2	97.5	37.8	62.2	65.8
Qwen2.5-14B	80.5	97.1	93.2	90.2	97.8	71.0	84.4	18.0	79.1	48.5	97.8	42.3	68.6	69.6
Mistral-Nemo-2407	80.3	94.5	93.8	89.5	97.9	71.6	84.7	18.8	78.5	48.6	97.9	42.5	68.5	69.6
Falcon3-10B	76.9	94.7	92.3	88.0	97.5	66.3	81.9	15.4	74.2	44.8	97.5	37.6	62.7	65.9

Table 2: Results for the AAEC dataset at the essay level. The first four rows show the current SOTA results, where sub-tasks are modeled separately (rows 1–3) and error propagation is not considered – that is, the input to task $i+1$ is the ground truth rather than the output of task i. All models are 4-bit Instruct variants. F1 denotes the macro-averaged F1 score. Best results are boldfaced and second-best results are underlined.

Model	ACC				ARI			ARC			ARIC			
	C	M	P	F1	NR	R	F1	A	S	F1	NR	A	S	F1
Liu et al. (2023)	-	-	-	89.2	-	-	82.7	-	-	81.0	-	-	-	-
Bao et al. (2021)	-	-	-	88.4	-	-	82.5	-	-	81.0	-	-	-	-
Cabessa et al. (2025)	-	-	-	89.5	-	-	83.5	-	-	**95.9**	-	-	-	-
Gemechu et al. (2024)	-	-	-	-	-	-	-	-	-	-	-	-	-	**78.0**
LLaMA-3-70B	84.5	94.9	95.1	91.5	98.8	72.3	**85.6**	15.3	82.7	49.0	98.8	50.6	72.3	73.9
LLaMA-3-8B	82.4	93.8	94.4	90.2	98.6	66.8	82.7	12.1	78.8	45.5	98.6	47.5	66.8	71.0
LLaMA-3.1-8B	83.7	97.4	94.2	**91.8**	98.7	69.5	84.1	10.9	80.3	45.6	98.7	36.4	69.5	68.2
Qwen2.5-7B	75.1	92.8	91.6	86.5	98.3	58.5	78.4	-	-	-	-	-	-	-
Qwen2.5-14B	83.7	96.1	94.3	91.4	98.6	67.4	83.0	13.5	79.3	46.4	98.6	47.2	67.4	71.1
Mistral-Nemo-2407	80.3	96.4	93.3	90.0	98.6	66.1	82.3	13.0	78.2	45.6	98.6	47.7	65.8	70.7
Falcon3-10B	76.0	93.5	92.0	87.2	98.4	60.9	79.6	9.0	74.5	41.8	98.4	39.0	61.0	66.1
Phi-4-mini	76.0	93.5	91.9	87.1	98.2	57.0	77.6	7.1	72.6	39.9	98.3	31.0	58.8	62.7

Table 3: Results for the AAEC dataset at the paragraph level. The first four rows present the current SOTA results for comparison. All models are 4-bit Instruct variants. F1 denotes the macro-averaged F1 score.

```
{"argument_types":
['M', 'C', 'P', 'P', 'P', 'P', 'C', 'P', 'P', 'C', 'M']}

{"relation_types":
[(3,  2, 'S'), (4,  2, 'S'), (5,  2, 'S'), (6,   7, 'S'),
 (8, 10, 'S'), (9, 10, 'S'), (2,  1, 'S'), (2,  11, 'S'),
 (7,  1, 'A'), (7, 11, 'A'), (10, 1, 'S'), (10, 11, 'S')]}
```

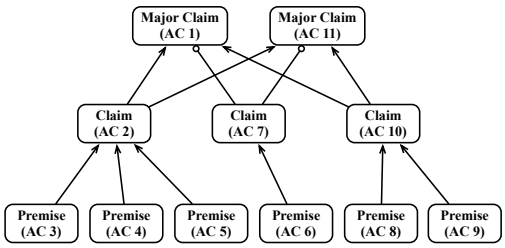

Figure 2: An LLM output and the corresponding graphical representation of the argumentative structure.

tasks, promoting more context-aware and coherent parsing of argumentative structures, and enhancing usability and deployment efficiency. Through extensive experiments on three benchmark datasets – AAEC, AbstRCT, and CDCP – and with popular LLMs such as LLaMA-3, LLaMA-3.1, Qwen-2.5, Mistral-Nemo, Phi-4, and Falcon-3, we achieve competitive or state-of-the-art performance compared to the best existing methods where each subtask is handled separately and error propagation effects are disregarded. Since error propagation inherently degrades performance in sequential task modeling (Bao et al., 2022), our unified formulation offers a streamlined and effective solution to mitigate its effects. Overall, our work lays the foundation for adopting unified, generative approaches to argument mining, offering a more compact, practical and realistic alternative to traditional AM pipeline architectures.

For future work, we are particularly interested in understanding how an LLM internally handles the unified task of juxtaposing three distinct yet interrelated sub-tasks. Ko et al. (2024) explore how LLMs leverage knowledge through a graph-based hierarchical deconstruction framework, which we believe is well-suited for the AM pipeline generation task. Additionally, advances in LLM reasoning, such as DeepSeek and GPT reasoning models, align naturally with this framework. We plan to explore this promising direction in future research.

We also plan to evaluate the current fine-tuning

Model	ACC			ARI			ARC			ARIC			
	C	P	F1	NR	R	F1	A	S	F1	NR	A	S	F1
Liu et al. (2023)	-	-	92.8	-	-	-	-	-	-	-	-	-	75.0
Si et al. (2023)	-	-	91.9	-	-	-	-	-	-	-	-	-	71.2
Cabessa et al. (2025)	-	-	**94.2**	-	-	-	-	-	-	-	-	-	77.1
Gemechu et al. (2024)	-	-	-	-	-	-	-	-	-	-	-	-	**84.0**
LLaMA-3-70B	90.9	95.0	93.0	96.8	70.7	83.8	36.3	79.8	**58.0**	96.8	63.8	68.4	76.3
LLaMA-3-8B	91.2	95.3	93.3	96.6	69.7	83.1	34.3	80.0	57.2	96.6	59.1	67.6	74.4
LLaMA-3.1-8B	91.2	95.1	93.2	96.7	70.0	83.4	33.5	79.1	56.3	96.7	61.3	68.4	75.5
Qwen2.5-7B	89.8	94.5	92.1	95.6	61.8	78.7	25.5	76.4	50.9	95.6	45.6	61.6	67.6
Qwen2.5-14B	91.2	95.1	93.2	96.8	71.0	**83.9**	32.8	81.5	57.1	96.8	53.1	71.3	73.7
Mistral-Nemo-2407	91.8	95.5	93.6	96.8	71.0	**83.9**	35.4	80.3	57.8	96.8	61.9	69.0	75.9
Falcon3-10B	91.6	95.4	93.5	96.1	64.4	80.2	21.9	75.7	48.8	96.1	40.4	64.9	67.1
Phi-4-mini	91.2	95.1	93.2	95.2	57.9	76.5	15.1	71.7	43.4	95.2	30.2	57.8	61.1
Liu et al. (2023)	-	-	92.6	-	-	-	-	-	-	-	-	-	74.3
Si et al. (2023)	-	-	92.4	-	-	-	-	-	-	-	-	-	73.3
Cabessa et al. (2025)	-	-	**93.7**	-	-	-	-	-	-	-	-	-	74.9
LLaMA-3-70B	88.4	94.6	91.5	96.2	71.0	83.6	22.0	83.7	**52.9**	96.2	49.1	71.7	72.3
LLaMA-3-8B	88.9	94.8	91.9	96.0	69.4	82.7	19.7	82.4	51.0	96.0	46.4	69.8	70.7
LLaMA-3.1-8B	89.7	95.2	92.4	95.9	68.5	82.2	15.4	81.8	48.6	95.9	35.1	70.1	67.0
Qwen2.5-7B	87.5	94.4	90.9	95.3	64.6	80.0	15.8	80.3	48.1	95.3	40.7	65.3	67.1
Qwen2.5-14B	88.5	94.7	91.6	95.8	67.8	81.8	16.5	81.5	49.0	95.8	42.3	68.8	69.0
Mistral-Nemo-2407	88.4	94.7	91.6	96.2	71.2	**83.7**	21.1	84.1	52.6	96.2	49.1	71.4	72.2
Falcon3-10B	87.7	94.5	91.1	95.3	64.0	79.6	8.8	79.3	44.0	95.3	24.5	65.8	61.9
Phi-4-mini	83.6	92.9	88.2	94.0	55.3	74.7	4.9	72.6	38.8	94.0	13.6	56.8	54.8
Liu et al. (2023)	-	-	94.0	-	-	-	-	-	-	-	-	-	73.9
Si et al. (2023)	-	-	92.2	-	-	-	-	-	-	-	-	-	72.7
Cabessa et al. (2025)	-	-	**95.9**	-	-	-	-	-	-	-	-	-	75.7
LLaMA-3-70B	92.2	95.7	94.0	96.0	67.1	81.6	25.0	83.3	**54.1**	96.0	52.6	67.2	71.9
LLaMA-3-8B	93.1	96.3	94.7	96.0	67.3	81.6	22.8	83.8	53.3	96.0	45.6	68.0	69.9
LLaMA-3.1-8B	93.0	96.1	94.5	95.9	66.5	81.2	25.4	82.3	53.9	95.9	62.7	65.6	74.8
Qwen2.5-7B	92.4	95.9	94.2	95.5	63.5	79.5	12.3	81.6	46.9	95.5	31.1	64.2	63.6
Qwen2.5-14B	91.6	95.5	93.6	95.7	64.5	80.1	12.4	81.8	47.1	95.7	29.8	65.7	63.7
Mistral-Nemo-2407	93.0	96.1	94.6	96.5	70.5	**83.5**	22.2	84.7	53.5	96.5	51.1	70.9	72.8
Falcon3-10B	91.2	95.3	93.2	95.8	64.9	80.3	12.0	81.0	46.5	95.8	30.4	65.8	64.0
Phi-4-mini	90.3	94.9	92.6	94.5	56.1	75.3	11.2	75.3	43.3	94.5	31.4	56.2	60.7

Table 4: Results for AbsRCT dataset (neo, gla, and mix test sets). The first four rows present the current SOTA results.

Model	ACC						ARI			ARC			ARIC			
	F	P	R	T	V	F1	NR	R	F1	E	R	F1	NR	E	R	F1
Bao et al. (2021)	-	-	-	-	-	82.5	-	-	67.8	-	-	-	-	-	-	-
Cabessa et al. (2025)	-	-	-	-	-	**87.3**	-	-	72.3	-	-	**80.4**	-	-	-	-
Gemechu et al. (2024)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**72.0**
LLaMA-3-70B	68.5	89.5	100.0	91.3	86.6	87.2	98.6	52.8	**75.7**	3.5	65.4	34.5	98.6	20.7	53.3	57.5
LLaMA-3-8B	61.9	91.9	66.7	88.1	85.9	78.9	98.1	45.3	71.7	0.0	62.4	31.2	98.1	0.0	45.4	47.8
LLaMA-3.1-8B	63.2	92.6	100.0	86.0	85.1	85.4	98.2	43.6	70.9	3.1	58.3	30.7	98.2	20.0	44.4	54.2
Qwen2.5-7B	58.2	83.7	66.7	86.1	83.3	75.6	98.0	41.3	69.7	1.0	57.4	29.2	98.0	6.7	40.8	48.5
Qwen2.5-14B	62.3	91.6	100.0	89.4	85.4	85.7	98.2	44.7	71.4	7.0	58.5	32.7	98.2	36.8	43.8	59.6
Mistral-Nemo-2407	67.6	92.1	66.7	88.0	87.4	80.4	98.3	45.6	71.9	0.0	60.3	30.2	98.3	0.0	45.7	48.0
Falcon3-10B	60.9	87.7	100.0	88.3	84.0	84.2	98.0	39.0	68.5	0.0	54.7	27.4	98.0	0.0	38.9	45.6
Phi-4-mini	60.9	85.1	100.0	84.8	82.3	82.6	97.3	25.9	61.6	0.0	42.7	21.4	97.3	0.0	25.5	41.0

Table 5: Results for CDCP dataset.

methodology against zero-shot and few-shot settings, particularly focusing on in-context learning strategies, where LLMs are provided with fully solved AM pipeline examples as demonstrations.

Finally, we envision the integration of chain-of-thought (CoT) reasoning mechanisms into end-to-end AM systems. By explicitly modeling intermediate reasoning steps, CoT methods can enhance interpretability and improve performance on complex argumentative structures. The incorporation of reasoning-optimized, state-of-the-art LLMs with CoT methods represents a natural evolution in AM pipeline design. These models, when guided by CoT prompts, may offer improved capability in tasks like component classification, relation prediction, and overall argument structure generation.

Limitations

Our study has the following limitations, which should be taken into account when interpreting the results. First, while the AAEC, AbsRCT, and CDCP datasets are relatively old, they remain widely adopted benchmarks in the argument mining literature, providing a standard and transparent basis for evaluating and comparing different methods. Also, following Gemechu et al. (2024), it would be valuable to evaluate the unified AM pipeline in both in-dataset and cross-dataset settings.

We did not perform formal statistical significance tests to quantify the variability of our results. We nevertheless note that the trends observed across eight LLMs and three benchmark datasets are consistent and robust.

Although we expect the comparative results between the 3-step and 2-step prompting strategies to generalize across datasets, a comprehensive empirical validation is still needed. Specifically, for the AAEC dataset, the LLaMA-3-70B model has not yet been evaluated under the 3-step strategy.

In addition, Cabessa et al. (2024) establish that LLM fine-tuning approach outperforms the less resource-intensive zero-shot and in-context learning strategies for separate AM sub-tasks. We intuit that this performance delta will be lower in the unified AM pipeline setting on account of enhanced knowledge sharing and improved contextualization. This, however, remains to be empirically validated.

In Tables 2–5, we compare our unified AM pipeline with the best separate-task modeling approaches reported in the literature, in which error propagation across tasks is *not* considered. Formally, in these approaches, the ground truths rather than the outputs of task i are given as inputs to task $i + 1$. To fully assess the impact of error propagation, these approaches would need to be re-evaluated with the outputs of task i fed as inputs to task $i + 1$, and then compared with our unified pipeline results. Since error propagation inevitably degrades performance in sequential task modeling, we expect this effect to favor our unified approach.

Finally, as our method relies on LLMs in a black-box setting, we have limited insight into the underlying reasoning process or failure cases of the models. Interpretability and explainability remains an open challenge in LLM-based argument mining.

References

Abdullah Al Zubaer, Michael Granitzer, and Jelena Mitrović. 2023. Performance analysis of large language models in the domain of legal argument mining. *Frontiers in Artificial Intelligence*, 6.

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

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, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 10437–10449. Association for Computational Linguistics.

Jérémie Cabessa, Hugo Hernault, and Umer Mushtaq. 2024. In-context learning and fine-tuning GPT for argument mining. *CoRR*, abs/2406.06699.

Jérémie Cabessa, Hugo Hernault, and Umer Mushtaq. 2025. Argument mining with fine-tuned large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6624–6635, Abu Dhabi, UAE. Association for Computational Linguistics.

Elena Cabrio and Serena Villata. 2018. Five years of argument mining: A data-driven analysis. In *Proceedings of IJCAI 2018*, IJCAI'18, pages 5427–5433. AAAI Press.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186. ACL.

Debela Gemechu, Ramon Ruiz-Dolz, and Chris Reed. 2024. ARIES: A general benchmark for argument relation identification. In *Proceedings of the 11th Workshop on Argument Mining (ArgMining 2024)*, pages 1–14, Bangkok, Thailand. Association for Computational Linguistics.

Miyoung Ko, Sue Hyun Park, Joonsuk Park, and Minjoon Seo. 2024. Hierarchical deconstruction of LLM reasoning: A graph-based framework for analyzing knowledge utilization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4995–5027, Miami, Florida, USA. Association for Computational Linguistics.

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

Boyang Liu, Viktor Schlegel, Riza Batista-Navarro, and Sophia Ananiadou. 2023. Argument mining as a multi-hop generative machine reading comprehension task. In *Findings of the ACL: EMNLP 2023, Singapore, December 6-10, 2023*, pages 10846–10858. ACL.

Tobias Mayer. 2020. *Argument Mining on Clinical Trials. (Fouille d'arguments à partir des essais cliniques)*. Ph.D. thesis, Université Côte d'Azur, France.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2025. Large language models: A survey. *Preprint*, arXiv:2402.06196.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of ICAIL 2019*, ICAIL '09, pages 98–107, New York, NY, USA. ACM.

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, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).

Mircea-Luchian Pojoni, Lorik Dumani, and Ralf Schenkel. 2023. Argument-mining from podcasts using chatgpt. In *Proceedings of ICCBR-WS 2023*, volume 3438 of *CEUR Workshop Proceedings*, pages 129–144. CEUR-WS.org.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Jiasheng Si, Liu Sun, Deyu Zhou, Jie Ren, and Lin Li. 2023. Biomedical argument mining based on sequential multi-task learning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 20(2):864–874.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS 2017*, pages 5998–6008.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Hongyi Yuan, Zheng Yuan, Ruyi Gan, Jiaxing Zhang, Yutao Xie, and Sheng Yu. 2022. BioBART: Pretraining and evaluation of a biomedical generative language model. In *Proceedings of the 21st Workshop on Biomedical Language Processing*, pages 97–109, Dublin, Ireland. Association for Computational Linguistics.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. Dialogpt: Large-scale generative pre-training for conversational response generation. *Preprint*, arXiv:1911.00536.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

A Datasets

Corpus Statistics		Component Statistics	
Tokens	147,271	major claims	751
Sentence	7,116	claims	1,506
Paragraphs	1,833	premises	3,832
Essays	402	Total	6,089

Table 6: Statistics for AAEC dataset (available here).

Split	Abstracts	Components
Neo-train	350	2,291
Neo-test	100	691
Gla-test	100	615
Mix-test	100	609

Table 7: Statistics of AbstRCT dataset (available here).

Components		Relations	
policy	815	reason	1174
value	2182	evidence	46
fact	785		
testimony	1117		
reference	32		
Total	4931		1220

Table 8: Statistics for CDCP dataset (available here).

B Implementation Details

All experiments were carried out using the LLaMA-Factory Python library (Zheng et al., 2024), with the QLoRa fine-tuning approach (Dettmers et al., 2023). All models employed are 4-bit quantized checkpoints freely available from Unsloth on Hugging Face. Hyperparameter details are provided in Appendix B.

In our experiments, we mainly used the default hyperparameter configuration of the LLaMA-Factory detailed in Table 9 below.

Parameter	Value
num_train_epochs	5 / 10 / 15
per_device_train_batch_size	2 / 4
gradient_accumulation_steps	4
learning_rate	5e-5
lr_scheduler_type	'cosine' / 'linear'
warmup_ratio	0.1
max_grad_norm	1.0
finetuning_type	"lora"
lora_target	"all"
quantization_bit	4
loraplus_lr_ratio	16.0 / 32.0
fp16	True

Table 9: Hyperparameters of the models.

C Prompts

The prompt template used in the 2-step strategy is illustrated in Figure 3.

```
### You are an expert in Argument Mining
tasked with analyzing argumentative structures in
essays.

INPUT:
You will receive:
- An essay title.
- The complete essay text.
- An enumerated list of identified arguments
extracted from the essay.

TASK 1: Argument Classification
... <task description> ...

TASK 2: Argument Relations Identification and
Classification
... <task description> ...

EXAMPLE:

### Output:
{"argument_types": ['M', 'M', 'C', 'P', 'P',
'P', 'C', 'P', 'P', 'P', 'C']}
{"argument_relations": [(4, 3, 'S'), (5, 3,
'A'), (6, 3, 'S'), (10, 11, 'S'), (9, 11,
'A'), (8, 7, 'S')]}

### Essay title: ...<title>...

### Essay text:
...<essay text>...

### List of arguments in the essay:
1.  ...<AC1>...
2.  ...<AC2>...
    ...
11. ...<AC11>...

### Output:
{"argument_types": ['M', 'C', 'P', 'P', 'P',
'P', 'C', 'P', 'P', 'C', 'M']}
{"relation_types": [(3, 2, 'S'), (4, 2, 'S'),
(5, 2, 'S'), (6, 7, 'S'), (8, 10, 'S'), (9,
10, 'S'), (2, 1, 'S'), (2, 11, 'S'), (7, 1,
'A'), (7, 11, 'A'), (10, 1, 'S'), (10, 11,
'S')]}
```

Figure 3: Prompt template. The instruction, input and output elements are represented in different colors.

Prompt 1. Example of prompt for the AAEC dataset at the essay level.

```
### You are an expert in Argument Mining tasked with analyzing
 argumentative structures in essays.

INPUT:
You will receive:
- An essay title.
- The complete essay text.
- An enumerated list of identified arguments extracted from
the essay.

TASK 1: Argument Classification
- Classify each argument in the essay into one of the
following categories: "MajorClaim"(M), "Claim"(C) or "Premise
"(P).
- MajorClaim (M): The main stance or position that the author
wants to prove in the essay. Usually appears in the
introduction and/or conclusion.
- Claim (C): A statement that directly supports or attacks the
 major claim. Claims are controversial assertions that require
 further evidence.
```

```
- Premise (P): A reason or evidence that directly supports or
attacks a claim or another premise.
- You must return a list of argument types in following JSON
format: {"argument_types": [argument_types (str),
argument_types (str), ..., argument_types (str)]}

TASK 2: Argument Relations Identification and Classification
- Identify relationships between arguments by determining
which arguments support or attack other arguments.
- For each related argument pair, classify the relationship as
 either: "Support"(S) or "Attack"(A).
- IMPORTANT: Only arguments that appear in the same line in
the essay can be related.
- You must return a list of triplets in the following JSON
format: {"argument_relations": [(target_index (int),
source_index (int), relation_type (str)), (target_index (int),
 source_index (int), relation_type (str)), ...]}
- Note: Indices are 1-based, referring to the position in the
provided arguments list.

Example:

### Output:
{"argument_types": ['M', 'M', 'C', 'P', 'P', 'P', 'C', 'P', 'P
', 'P', 'C']}
{"argument_relations": [(4, 3, 'S'), (5, 3, 'A'), (6, 3, 'S'),
 (10, 11, 'S'), (9, 11, 'A'), (8, 7, 'S')]}

### Essay title: Should students be taught to compete or to
cooperate?

### Essay text:
It is always said that competition can effectively promote the
 development of economy. In order to survive in the
competition, companies continue to improve their products and
service, and as a result, the whole society prospers. However,
 when we discuss the issue of competition or cooperation, what
 we are concerned about is not the whole society, but the
development of an individual's whole life. From this point of
view, I firmly believe that we should attach more importance
to cooperation during primary education.
First of all, through cooperation, children can learn about
interpersonal skills which are significant in the future life
of all students. What we acquired from team work is not only
how to achieve the same goal with others but more importantly,
 how to get along with others. During the process of
cooperation, children can learn about how to listen to
opinions of others, how to communicate with others, how to
think comprehensively, and even how to compromise with other
team members when conflicts occurred. All of these skills help
 them to get on well with other people and will benefit them
for the whole life.
On the other hand, the significance of competition is that how
 to become more excellence to gain the victory. Hence it is
always said that competition makes the society more effective.
 However, when we consider about the question that how to win
the game, we always find that we need the cooperation. The
greater our goal is, the more competition we need. Take
Olympic games which is a form of competition for instance, it
is hard to imagine how an athlete could win the game without
the training of his or her coach, and the help of other
professional staffs such as the people who take care of his
diet, and those who are in charge of the medical care. The
winner is the athlete but the success belongs to the whole
team. Therefore without the cooperation, there would be no
victory of competition.
Consequently, no matter from the view of individual
development or the relationship between competition and
cooperation we can receive the same conclusion that a more
cooperative attitudes towards life is more profitable in one's
 success.

### List of arguments in the essay:
1. we should attach more importance to cooperation during
primary education
2. through cooperation, children can learn about interpersonal
 skills which are significant in the future life of all
students
3. What we acquired from team work is not only how to achieve
the same goal with others but more importantly, how to get
along with others
4. During the process of cooperation, children can learn about
 how to listen to opinions of others, how to communicate with
others, how to think comprehensively, and even how to
compromise with other team members when conflicts occurred
5. All of these skills help them to get on well with other
people and will benefit them for the whole life
6. the significance of competition is that how to become more
excellence to gain the victory
7. competition makes the society more effective
8. when we consider about the question that how to win the
game, we always find that we need the cooperation
9. Take Olympic games which is a form of competition for
instance, it is hard to imagine how an athlete could win the
game without the training of his or her coach, and the help of
 other professional staffs such as the people who take care of
 his diet, and those who are in charge of the medical care
10. without the cooperation, there would be no victory of
competition
11. a more cooperative attitudes towards life is more
profitable in one's success


### Output:
{"argument_types": ['M', 'C', 'P', 'P', 'P', 'P', 'C', 'P', 'P
', 'C', 'M']}
{"relation_types": [(3, 2, 'S'), (4, 2, 'S'), (5, 2, 'S'), (6,
 7, 'S'), (8, 10, 'S'), (9, 10, 'S'), (2, 1, 'S'), (2, 11, 'S
'), (7, 1, 'A'), (7, 11, 'A'), (10, 1, 'S'), (10, 11, 'S')]}
```

D 3-Step Prompting

In the 3-step prompting strategy, the LLM is instructed to solve the successive ACC, ARI, and ARC tasks in a single prompt by generating three corresponding outputs: (i) a list of AC types, (ii) a list of pairs of related ACs, and (iii) a list of AR types. For this prompting strategy, the output is of the following format:

```
{"argument_types": ['M', 'M', 'C', 'P',
'P', 'P', 'C', 'P', 'P', 'P', 'C']}

{"related_arguments": [(4, 3), (5, 3), (6,
3), (10, 11), (9, 11), (8, 7)]}

{"relation_types": ['S', 'S', 'S', 'S',
'S', 'A']}
```

The prompt template used in this strategy is illustrated in Figure 4. The evaluation metrics for the ACC, ARI, ARC, and ARIC tasks are computed as follows.

ACC / ARC Metric: For each test sample, the LLM generates a list of predicted AC (resp. AR) types denoted as `preds`, which is then compared to the list of ground AC (resp. AR) types, denoted as `grounds`. If `preds` contains more elements than `grounds`, the extra elements in `preds` are removed. Conversely, if `preds` contains fewer elements than `grounds`, additional incorrect elements are added to match the length of `grounds`. Once the lengths of `preds` and `grounds` are aligned across all texts, the classification metric for ACC task is computed (F1 score for each class and overall macro-F1 score).

ARI Metric: For each test sample, the LLM generates a list of predicted pairs of related ACs. To compute the ARI metric, we first enumerate all

```
### You are an expert in Argument Mining
tasked with analyzing argumentative structures in
essays.

INPUT:
You will receive:
- An essay title.
- The complete essay text.
- An enumerated list of identified arguments
extracted from the essay.

TASK 1: Argument Classification
... <task description> ...

TASK 2: Argument Relations
... <task description> ...

TASK 3: Relation Classification
... <task description> ...

EXAMPLE:

### Output:
{"argument_types": ['M', 'M', 'C', 'P', 'P',
'P', 'C', 'P', 'P', 'P', 'C']}
{"related_arguments": [(4, 3), (5, 3), (6,
3), (10, 11), (9, 11), (8, 7)]}
{"relation_types": ['S', 'S', 'S', 'S', 'S',
'S']}

### Essay title: ...<title>...

### Essay text:
...<essay text>...

### List of arguments in the essay:
1.  ...<AC1>...
2.  ...<AC2>...
    ...
11. ...<AC11>...

### Output:
{"argument_types": ['M', 'C', 'P', 'P', 'P',
'P', 'C', 'P', 'P', 'C', 'M']}
{"relation_arguments: [(3, 2), (4, 2), (5,
2), (6, 7), (8, 10), (9, 10), (2, 1), (2,
11), (7, 1), (7, 11), (10, 1), (10, 11)]}
{"relation_types: ['S', 'S', 'S', 'S', 'S',
'S', 'S', 'S', 'A', 'A', 'S', 'S']}
```

Figure 4: Prompt template for the 3-step strategy. The instruction, input and output elements are represented in different colors.

possible pairs (i,j) of distinct ACs in the text. We then create two lists, grounds and preds, by labeling each pair of ACs as either related ('R') or non-related ('NR') based on ground truth annotations and LLM predictions, respectively. After aggregating over all texts, the classification metric over the 'R' and 'NR' classes can be computed.

ARIC Metric: To evaluate this joint task, we merge the list of AC pairs (from the ARI task) and the list of AR types (from the ARC task) into a list of triplets of the form (i,j,'X'), where 'X' = 'S' or 'X' = 'A'. The merging process is designed to be robust to cases where the lengths of the two lists do not match, by introducing wrong AR types when needed. Next, we enumerate all possible pairs (i,j) of distinct ACs in the text, and create two lists, grounds and preds, by labeling each pair of ACs as either non-related ('NR'), support ('S'), or attack ('A') based on ground truth annotations and LLM predictions, respectively. Finally, we compute the classification metric for 'NR', 'S', and 'A' classes.

E Paragraph-Level Prompting Strategy for AAEC Dataset

For the AAEC dataset, the output format of the paragraph-level prompting strategy is as follows:

```
{"argument_types":
['M', 'C', 'P', 'P', 'C', 'M']}

{"claim_stances": ['S', 'A']}

{"relation_types":
{'Paragraph 1': [],
 'Paragraph 2': [(3,2,'S'), (4,2,'S'),
                 (4,5,'A')],
 'Paragraph 3': []}}
```

Based on these generated lists, the set of AR triplets is computed as follows: 1) we take the predicted paragraph-level triplets, i.e., (3,2,'S'), (4,2,'S'), (4,5,'A') in this example; 2) we add the stance relations (['S','A']) between the claims (2 and 5) and all major claims (1 and 6), resulting in (2,1,'S'), (2,6,'S'), (5,1,'A'), (5,6,'S') in this example. The classification metrics for ACC, ARI, ARC, and ARIC tasks are then computed as described in Section 3.4.