Improving End-to-End Argument Mining with LLMs via Multi-Candidate Reranking and Bounded Repair

Chen Tang[1], Min Peng[1](✉), and Gang Tian[1](✉)

[1] Wuhan University, Wuhan, China
tctc_whu@163.com, {pengm, tiang2008}@whu.edu.cn

Abstract. End-to-end Argument Mining (AM) aims to extract structured argument graphs directly from raw text by jointly identifying components and their rhetorical relations. Large Language Models (LLMs) have shown promise for this complex task by formulating it as direct text-to-graph generation. However, since most current LLM-based approaches rely on a single-pass generation paradigm, they struggle to handle the tight coupling of AM subtasks. Consequently, early extraction errors easily cascade, resulting in fragmented relations and globally inconsistent structures. Critically, these single-pass models lack mechanisms to backtrack and correct upstream mistakes when downstream inconsistencies arise. To address this problem, we propose the Multi-Candidate Reranking and Bounded Repair (MCR-BR) framework for LLM-based end-to-end AM. Instead of committing to a single output, our pipeline first generates diverse component candidates to preserve alternative hypotheses. It then constructs full argument graphs for these candidates and ranks them using a consistency-guided scoring strategy based on structural completeness, semantic coherence, and validity. For any remaining structural anomalies, a bounded local correction loop actively repairs the graph. We further introduce a targeted optimization module for conclusion-like components, which are common extraction bottlenecks. Experiments on the AAEC and CDCP benchmarks demonstrate that the MCR-BR framework consistently improves in-domain extraction and exhibits potential for cross-domain generalization.

Keywords: Argument Mining, Large Language Models, Self-Correction, Error Propagation, Structured Prediction.

1 Introduction

End-to-end Argument Mining (AM) involves extracting structured argument graphs directly from raw text. This task underpins applications such as educational feedback [1], legal analysis [2], and debate organization. Generative Large Language Models (LLMs) are increasingly central to computational argumentation [3,4]. Real-world scenarios require systems that output coherent argument graphs directly, rather than disconnected intermediate results. This makes the end-to-end formulation of AM particularly important.

Existing research has explored several main directions for end-to-end AM. Early systems often relied on pipeline designs that first identify components and then predict relations [5,7]. Later work adopted joint modeling and neural structured prediction to better capture subtask interactions [8,9,10,11,12]. Recently, generative text-to-text formulations have simplified the task interface. Large language models (LLMs) further enhance this direction via strong instruction-following and reasoning capabilities [13,14]. These lines of work have established direct structure generation as a practical approach.

Despite these notable advancements, end-to-end AM remains challenging due to the tight coupling of its subtasks. A major issue is error propagation: upstream mistakes in component extraction negatively affect downstream relation identification. For instance, if a model misses a key "Claim" during the initial extraction phase, any "Premise" or "Evidence" supporting it becomes structurally orphaned (see Fig. **1**(a)). In standard single-pass systems, such a local omission triggers a domino effect that fragments the global argument graph. This issue is particularly pronounced in LLM-based structured generation. While LLMs excel at semantic reasoning, forcing them to output complex structures in a single pass frequently introduces isolated clusters, cyclic dependencies, or hallucinated links. Furthermore, current paradigms lack a structural recovery mechanism. When an initial mistake creates structural incoherence (such as the dangling "Evidence" in Fig. **1**(b)), the model cannot use this downstream anomaly as feedback to reconsider its upstream choices.

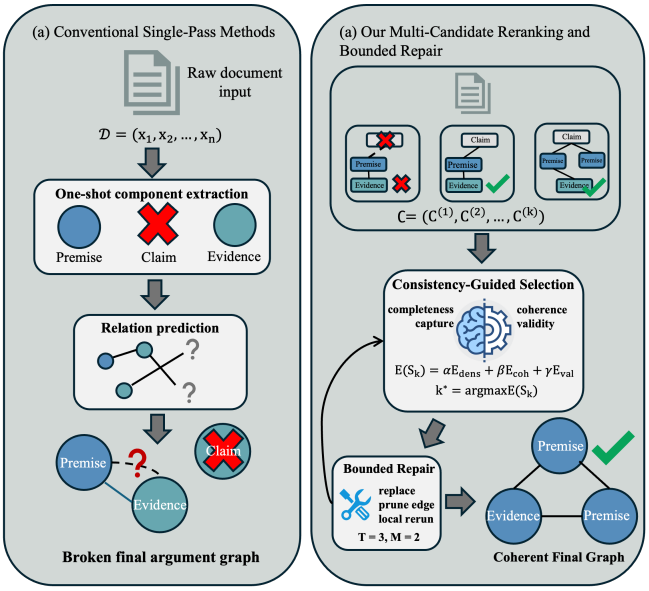

Fig. 1. Error propagation in pipelines vs. structural recovery through the MCR-BR framework. As shown in (a) Conventional Single-Pass Methods, a missed component (Claim) leads to orphaned evidence and broken relations. In contrast, (b) the MCR-BR framework uses this structural inconsistency as feedback to trigger a bounded correction procedure, retrieving an alternative component hypothesis from the generated candidate pool to restore a valid argument graph.

We address this gap by proposing the Multi-Candidate Reranking and Bounded Repair (MCR-BR) framework, an inference-time structural recovery approach for LLM-based end-to-end AM. Moving away from single-pass decoding, MCR-BR leverages candidate preservation, consistency scoring, and local repair. Initially, the model generates diverse candidate sets to maintain competing component hypotheses. We evaluate full argument structures for each set based on completeness, semantic coherence, and validity. When structural violations occur, a bounded correction loop executes targeted repairs—swapping problematic components with alternatives from the candidate pool, removing incoherent edges, or re-predicting specific relations. This mechanism effectively turns downstream consistency checks into active feedback to refine the argument graph.

Evaluations on the AAEC [5] and CDCP [6] datasets confirm that this verify-and-repair mechanism yields substantial improvements in both component extraction and relation prediction over single-pass baselines. We also observe encouraging robustness in coarse-schema cross-domain transfer settings.
Our main contributions are:

- We propose the Multi-Candidate Reranking and Bounded Repair (MCR-BR) framework, an inference-time structural recovery approach for end-to-end AM. By using consistency-guided selection and bounded local repair, MCR-BR actively corrects early extraction mistakes and mitigates error propagation.
- We introduce an optimization module designed to enhance the extraction of conclusion-like components. Through a claim-specific adapter and cue-aware soft prompting, this module explicitly targets and resolves a major structural bottleneck in relation prediction.
- We demonstrate the effectiveness of our approach through extensive experiments on the AAEC and CDCP benchmarks. MCR-BR consistently outperforms strong generative baselines in both in-domain evaluation and shows promising adaptability in cross-domain coarse-schema transfer.

2 Related Work

2.1 End-to-End Argument Mining

Early studies on AM typically decomposed the task into separate stages such as component identification and relation prediction [5,7]. To reduce the fragmentation between subtasks, subsequent work explored more integrated formulations, including joint modeling, neural structured prediction, and text-to-text generative architectures [8,9,10,11,12,13]. These methods established end-to-end AM as a practical benchmark setting by mapping raw text directly to structured argument graphs. Recent advances have formulated argument mining as a text-to-text generation task, where Kawarada et al. [14] provided a systematic evaluation of generative approaches, highlighting the strong performance of large seq2seq models. Other works have investigated whether

LLMs can perform relation-based argument mining [15], and extended these methods to Chinese argumentative essays [16].

While the aforementioned AM methods can effectively generate argument structures, they are largely designed as single-pass systems. Once an upstream component is missed or mis-typed, downstream relation prediction often becomes unreliable, which limits their ability to recover from early structural mistakes.

2.2 LLM-Based Reasoning for Structured Generation

LLMs have recently shown strong potential in complex structured generation tasks by combining semantic understanding with flexible instruction following. Prior studies have demonstrated that prompting strategies, such as chain-of-thought and self-consistency, can improve multi-step inference quality [17,18]. Iterative refinement and self-correction mechanisms have also been explored to improve LLM outputs after an initial draft [19,20]. These ideas embed LLMs into multi-stage reasoning pipelines rather than treating them as one-shot predictors.

When applying LLMs to end-to-end AM, predicting complex relations can lead to malformed outputs or globally inconsistent structures. Existing LLM-based AM approaches often lack specific feedback mechanisms to detect and repair these structural anomalies. Recent work has also explored fact-driven logical reasoning [21] and LLM-based logical fallacy detection [22]. Our work addresses this by explicitly using downstream structural consistency as a feedback signal to select, diagnose, and repair end-to-end argument graphs.

3 Problem Definition

Following prior work [14], we formulate end-to-end AM as a structured prediction task mapping a raw document $D = (x_1, \ldots, x_n)$ of n tokens to a directed argument graph $G = (\mathcal{U}, \mathcal{R})$.

The node set (component set) is defined as:

$$\mathcal{U} = \{u_k\}_{k=1}^{|\mathcal{U}|} \qquad (1)$$

where $u_k = (b_k, e_k, t_k)$, with $1 \le b_k \le e_k \le n$ denoting the start and end token indices of the k-th argumentative span, and $t_k \in \mathcal{T}$ representing its component type.

The edge set (relation set) is defined as:

$$\mathcal{R} = \{(u_i, u_j, r_{ij}) \mid u_i, u_j \in \mathcal{U}, u_i \neq u_j, r_{ij} \in \mathcal{Y}\} \qquad (2)$$

where r_{ij} is the relation type. The directed edge flows from a supporting or attacking unit u_i to its target u_j. To ensure logical validity, G is constrained to be a Directed Acyclic Graph (DAG).

We preserve the native schema for each dataset. For AAEC, $\mathcal{T} = \{MajorClaim, Claim, Premise\}$ and $\mathcal{Y} = \{Support, Attack\}$. For CDCP, $\mathcal{T} = \{Fact, Testimony, Value, Policy, Reference\}$ and $\mathcal{Y} = \{Reasons, Evidence\}$.

Ultimately, the model must jointly resolve three coupled subtasks: boundary detection (where), component classification (what type), and relation identification (how components connect).

4 Method

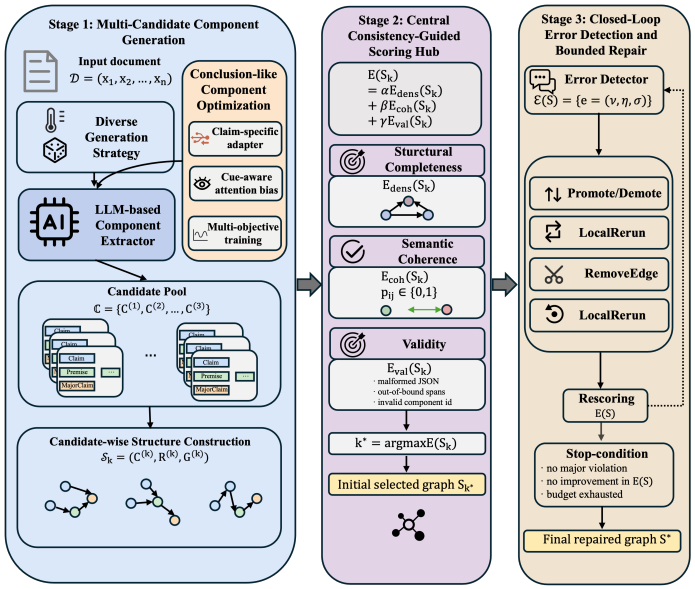

Fig. 2. The overall architecture of the MCR-BR framework for end-to-end argument mining. The pipeline encompasses candidate generation, structure construction, consistency-guided scoring, and a closed-loop bounded repair mechanism, supported by a targeted conclusion-like component optimizer.

We introduce the Multi-Candidate Reranking and Bounded Repair (MCR-BR) framework designed to mitigate error propagation in LLM-based argument mining (Fig. **2**). Given a raw document $D = (x_1, x_2, \ldots, x_n)$, the model first generates multiple candidate component sets to preserve structural diversity. It then maps each candidate set into a complete argument graph and evaluates them through a consistency-guided scoring hub. The highest-scoring graph serves as the initial solution. If this structure exhibits anomalies—such as isolated conclusion nodes or invalid references—a bounded correction loop takes over, executing local repairs until the graph achieves structural integrity or the computational budget runs out. A targeted optimization module further enhances the extraction of crucial conclusion-like components.

4.1 Multi-Candidate Component Generation

To reduce the brittleness of downstream relation prediction to early extraction errors, given the input document D, we avoid committing to a single output. Instead, we generate a diverse candidate set

$$\mathcal{C} = \{C^{(1)}, C^{(2)}, \dots, C^{(K)}\} \quad (3)$$

where each $C^{(k)}$ is a structured set of argumentative components extracted from the same document. Candidate diversity is introduced through controlled decoding variation, including temperature sampling and prompt paraphrasing.

For each component, the model outputs token-index spans and component labels in structured JSON. Overlapping spans are resolved using a deterministic priority strategy based on confidence, span length, and position. During generation, the model may also produce auxiliary diagnostic fields for internal validation, which are discarded before evaluation.

4.2 Consistency-Guided Structure Scoring

Building upon the diverse component candidate sets $\mathcal{C}$ obtained from the previous stage, we next construct and evaluate full argument graphs. Specifically, for every candidate $C^{(k)}$, we invoke the relation identification module to produce a complete argument structure $S_k = (C^{(k)}, R^{(k)}, G^{(k)})$. To determine the most reliable structure, we then score each candidate with a composite scoring function:

$$E(S_k) = \alpha E_{\text{dens}}(S_k) + \beta E_{\text{coh}}(S_k) + \gamma E_{\text{val}}(S_k) \quad (4)$$

where E_{dens}, E_{coh}, and E_{val} correspond to structural completeness, semantic coherence, and validity, respectively.

Structural Completeness. Let $\mathcal{Q}(S_k) = \{u_j \in C^{(k)} \mid t_j \in \mathcal{T}_{\text{conc}}\}$ be the set of conclusion-like components, where $\mathcal{T}_{\text{conc}}$ is defined by the dataset's argumentation schema. For each $u_j \in \mathcal{Q}(S_k)$, its coherence-weighted in-degree is $d^{-}_{\text{coh}}(u_j) = \sum_{(i,j,r_{ij}) \in R^{(k)}} p_{ij}$, where $p_{ij} \in \{0,1\}$ is a verifier judgment. We compute

$$E_{\text{dens}}(S_k) = \begin{cases} \frac{1}{|\mathcal{Q}(S_k)|} \sum_{u_j \in \mathcal{Q}(S_k)} \min\left(\frac{d^{-}_{\text{coh}}(u_j)}{\tau(t_j)}, 1\right), & |\mathcal{Q}(S_k)| > 0, \\ 0, & |\mathcal{Q}(S_k)| = 0, \end{cases} \quad (5)$$

where $\tau(t_j)$ is the expected minimum support for type t_j. The specific schema definitions and τ values are detailed in Section 5.1.

Semantic Coherence. For each edge (i, j, r_{ij}), a verifier returns a binary judgment $p_{ij} \in \{0,1\}$ indicating whether the relation is supported in context. We compute

$$E_{\mathrm{coh}}(S_k) = \begin{cases} \frac{1}{|R^{(k)}|}\sum_{(i,j,r_{ij})\in R^{(k)}} p_{ij}, & |R^{(k)}| > 0, \\ 0, & |R^{(k)}| = 0. \end{cases} \quad (6)$$

Validity and Anomaly Detection. We set $E_{\mathrm{val}}(S_k) = 1$ if the output satisfies all deterministic checks; otherwise $E_{\mathrm{val}}(S_k) = 0$. Violations include malformed JSON, out-of-bound token spans, references to non-existent component ids, structural cycles that violate the DAG constraint, and diagnostic anomalies.

Finally, we select $k^* = \backslash\mathrm{argmax}_k E(S_k)$ as the initial candidate structure.

4.3 Bounded Local Correction

Although the consistency-guided scoring strategy successfully selects the most globally coherent candidate graph (denoted as S_k*), this initial structure may still exhibit local anomalies. To refine this output, we introduce a bounded local correction stage. Given the currently selected structure S=(C,R,G), a DetectErrors function first returns a sorted violation list

$$\mathcal{E}(S) = \{e = (\nu, \eta, \sigma)\} \quad (7)$$

where $\nu \in \{VAL, COH, DENS\}$ denotes the violation category (validity, incoherence, or density), η specifies the affected locus (a node j or an edge (i,j)), and $\sigma \in [0,1]$ is the severity score, computed as:

$$\sigma(e) = \begin{cases} 1, & \nu = VAL, \\ 1 - p_{ij}, & \nu = COH,\ \eta = (i,j), \\ 1 - \min\left(\frac{\bar{d}^{-}_{\mathrm{coh}}(u_j)}{\tau(t_j)}, 1\right), & \nu = DENS,\ \eta = j. \end{cases} \quad (8)$$

We sort $\mathcal{E}(S)$ in descending order of σ, breaking ties by priority $VAL \succ COH \succ DENS$. We treat structural validity (VAL) and verifier-rejected incoherent edges (COH) as deterministic triggers that always prompt correction actions. Thus, only the density anomaly requires a tunable threshold, triggering repair when its severity indicates a drop below the acceptable threshold (i.e., density below δ_{dens}).

Each correction iteration applies at most M actions in total. The rule-based repair operations include: (1) Promote/Demote: type conversion within conclusion-like categories (e.g., MajorClaim ↔ Claim) if it improves E_{dens} without validity errors; (2) Replace: substituting a problematic component with an alternative span from another candidate that maximizes $E(S)$; (3) RemoveEdge: deleting an edge (i,j,r_{ij}) if $p_{ij} = 0$ and no new invalid reference is created; (4) LocalRerun: re-predicting relations for a source component constrained to valid target ids. Every repair action is conditionally accepted only if the resulting graph maintains the DAG property and improves or maintains the overall score ($E(S') \geq E(S)$); actions introducing cycles or degrading the score are immediately reverted.

After each repair, the modified graph is re-scored. The process safely terminates and finalizes the output when no remaining violations trigger the thresholds, no action improves the overall score $E(S)$, or the predefined correction budget (maximum iterations and actions per iteration) is exhausted.

4.4 Conclusion-like Component Optimization

While the multi-candidate generation and bounded correction stages provide robust structural recovery at the global graph level, their effectiveness is fundamentally bottlenecked by the quality of key component predictions. Since conclusion-like components (e.g., MajorClaim, Policy) are disproportionately influential for downstream relations, we introduce a targeted optimization module for the component extractor to enhance their identification, consisting of three parts:

First, a claim-specific adapter (a parallel LoRA module) is injected into the component extractor's attention projection layers. It is dynamically activated during JSON generation when decoding the value field of a "type" key matching a conclusion-like label.

Second, cue-aware soft prompting appends learnable continuous prefix tokens to the input sequence when predefined argumentative cues (e.g., "therefore", "in conclusion") are detected, guiding the model's focus on these structural signals.

Third, multi-objective training is used with the loss:

$$\mathcal{L}_{comp} = \mathcal{L}_{gen} + \lambda \mathcal{L}_{sal} (9)$$

where $\mathcal{L}_{gen}$ is the standard negative log-likelihood for JSON generation, $\mathcal{L}_{sal}$ is an auxiliary binary cross-entropy loss predicting if each sentence contains a conclusion-like component, and λ is a balancing hyperparameter.

5 Experiments and Analysis

5.1 Experimental Setup

Datasets and Metrics. We evaluate our method on two public argument mining benchmarks: AAEC [5] (essay-level, 3 component types) and CDCP [6] (online comments, 5 types with complex multi-parent relations). For cross-domain evaluation (AAEC→CDCP), we apply a coarse-schema protocol mapping specific components to general claim/premise categories. For in-domain evaluation, we report Component-F1 (exact boundary and type match) and Relation-F1 (correct directed edge and relation label), following prior work [9]. For cross-domain transfer, we report Coarse Component-F1 and Unlabeled Link-F1 to account for schema incompatibilities.

Baselines and Implementation Details. We compare our framework against strong generation-based methods (FLAN-T5, GPT-4) [14] and an encoder-based multi-task

baseline (RoBERTa MTL) [9]. To rigorously isolate and validate the contributions of our closed-loop design, we also evaluate several budget-matched internal variants using our backbone: (1) Single-Pass, which represents the standard direct generation paradigm without any structural recovery; (2) DAG Post-process, which applies rule-based cycle removal to the single-pass output; (3) Self-Refine, which prompts the LLM to iteratively critique and revise its own output; and (4) Verifier-only Rerank, which selects the best candidate using our consistency scoring but omits the bounded repair loop. Our backbone is Qwen2.5-7B-Instruct, fine-tuned via LoRA. For consistency scoring, we set $\tau=2$ for top-level claims and $\tau=1$ for others, with scoring weights $\alpha=0.4$, $\beta=0.4$, $\gamma=0.2$, and density threshold $\delta_{\text{dens}}=1.0$. Correction is bounded to $T=3$ iterations and $M=2$ actions per iteration. The auxiliary loss weight for conclusion optimization is $\lambda=0.5$. Models are trained using AdamW with early stopping. We use the dataset splits and evaluation scripts from Kawarada et al. [14].

5.2 Main In-Domain Results

Table **1** details the main in-domain results on AAEC and CDCP.

Table 1. In-domain main results on AAEC and CDCP.

Method	Comp-F1	Rel-F1 (Sup)	Rel-F1 (Atk)	Rel-F1
FLAN T5-XXL [14]	80.15	62.34	48.72	61.19
GPT-4-turbo 20-shot [14]	55.51	29.86	18.45	28.38
RoBERTa-large MTL [9]	76.43±0.55	54.21±0.52	41.83±0.67	52.15±0.54
LLM Single-Pass	77.82±0.51	58.43±0.45	44.21±0.62	56.87±0.48
LLM w/ Constrained Decoding	78.54±0.46	59.12±0.42	45.33±0.58	57.65±0.45
LLM w/ DAG Post-process	78.91±0.49	59.45±0.48	45.87±0.61	58.12±0.47
LLM Multi-Sample (LogProb)	79.45±0.38	60.12±0.41	46.58±0.55	58.94±0.42
LLM w/ Self-Refine (1-round)	79.82±0.45	60.75±0.43	47.12±0.56	59.63±0.45
Ours (K=1, w/o correction)	81.24±0.44	61.87±0.39	48.34±0.58	60.73±0.41
Ours (Verifier-only Rerank, K=3)	82.18±0.40	63.15±0.36	49.82±0.52	62.21±0.38

Method	Comp-F1	Rel-F1 (Sup)	Rel-F1 (Atk)	Rel-F1
Ours (Full, K=3)	83.71±0.42	65.92±0.38	51.47±0.54	64.83±0.38
Method	Comp-F1	Rel-F1 (Reasons)	Rel-F1 (Evidence)	Rel-F1
FLAN T5-XXL [14]	72.68	35.24	28.41	33.96
RoBERTa-large MTL [9]	68.52±0.58	31.45±0.61	25.72±0.73	30.23±0.64
LLM Single-Pass	71.35±0.48	33.82±0.52	27.15±0.68	32.64±0.55
LLM w/ DAG Post-process	72.14±0.50	34.65±0.49	28.02±0.63	33.51±0.52
LLM w/ Self-Refine (1-round)	72.95±0.47	35.58±0.48	28.94±0.60	34.42±0.50
Ours (K=1, w/o correction)	74.56±0.45	36.94±0.46	30.27±0.58	35.78±0.48
Ours (Verifier-only Rerank, K=3)	75.12±0.41	37.85±0.44	31.42±0.55	36.64±0.45
Ours (Full, K=3)	76.45±0.43	39.12±0.45	32.86±0.57	38.17±0.46

Table **1** demonstrates that our full framework (Full, K=3) consistently boosts both component and relation extraction. On AAEC, it reaches 83.71 Component-F1 and 64.83 Relation-F1, surpassing the strong FLAN T5-XXL baseline by +3.56 and +3.64 points, respectively. Internal variant comparisons reveal that while DAG Post-process and Self-Refine marginally improve the Single-Pass baseline, they fail to recover missing upstream components. Combining structure-aware selection (Verifier-only Rerank) with bounded correction effectively resolves these global anomalies. This performance trend extends to CDCP (+4.21 Relation-F1 over FLAN T5-XXL), proving the framework handles complex, multi-parent graph structures efficiently.

5.3 Ablation Study

Table **2** presents the ablation results on AAEC.

Table 2. Ablation results on AAEC.

Configuration	Comp-F1	Rel-F1	ΔComp	ΔRel
Full Method	83.71±0.42	64.83±0.38	—	—
Ours (K=1, w/o correction)	81.24±0.44	60.73±0.41	-2.47	-4.10
w/o Claim-specific Adapter	81.96±0.48	63.08±0.44	-1.75	-1.75
w/o Cue-Aware Prompting	82.34±0.45	62.87±0.40	-1.37	-1.96

Configuration	Comp-F1	Rel-F1	ΔComp	ΔRel
w/o Multi-Objective	82.89±0.43	63.54±0.39	-0.82	-1.29
w/o Validity Signal (E_val)	81.56±0.46	61.92±0.43	-2.15	-2.91

Table **2** breaks down the individual contributions of each module within our framework. Moving from the single-pass baseline (K=1, w/o correction) to multi-candidate reranking (Verifier-only Rerank) yields a +1.48 improvement in Relation-F1. This indicates that generating and preserving diverse candidate pools effectively mitigates early extraction errors. Building upon this, the introduction of the bounded repair loop (Full Method) provides an additional +2.62 Relation-F1 boost, demonstrating that active, localized correction is crucial for resolving residual structural anomalies. Furthermore, the conclusion-like optimization module contributes a targeted +1.75 Relation-F1 gain. As detailed in Table **2**(c), this improvement is almost entirely driven by the enhanced extraction of MajorClaim and Claim nodes.

Signal ablation (Table **2**b) further proves that the consistency scoring components—E_dens, E_coh, and E_val—are complementary. Removing any single signal degrades overall performance, as structural density, semantic plausibility, and formatting constraints each target distinct and non-overlapping error types during the reranking and repair phases.

Independent Verifier Analysis. To rule out homogeneous self-evaluation bias, we substituted the verifier with Llama-3-8B-Instruct. As shown in Table **3**, this yielded minimal performance fluctuation (ΔRel-F1 = −0.28), with a 94.2% top-1 agreement on candidate selection. This confirms our consistency signals capture objective semantic compatibility rather than model-specific artifacts.

5.4 Cross-Domain Transfer

Table **4** reports the cross-domain transfer results from AAEC to CDCP under the coarse-schema, unlabeled-link protocol.

Table 3. Impact of using an independent verifier on AAEC.

Verifier Model	Comp-F1	Rel-F1	Top-1 Agreement	Trigger Rate
Qwen2.5-7B (Default)	83.71	64.83	100.0%	42.5%
Llama-3-8B (Independent)	83.71	64.55	94.2%	43.8%

Table 4. Cross-domain transfer results (AAEC→CDCP).

Method	Coarse Comp-F1	Unlabeled Link-F1
FLAN T5-XXL (zero-shot)	52.34±0.65	31.25±0.72

Method	Coarse Comp-F1	Unlabeled Link-F1
LLM Single-Pass (zero-shot)	54.87±0.58	33.42±0.68
Ours (K=1, w/o correction, zero-shot)	57.23±0.52	38.76±0.61
Ours (Full, zero-shot)	60.78±0.48	43.21±0.55

Under the coarse-schema transfer protocol, our full method shows promising improvements over the zero-shot FLAN T5-XXL baseline, particularly on unlabeled link prediction (+11.96 F1). Because single-pass generation is highly fragile under domain shift—frequently producing broken graphs and orphaned references—introducing candidate-pool structural recovery proves especially valuable. The observed cross-domain robustness stems from the framework's abstraction of the problem: by relying on universal principles of graph completeness and semantic validity rather than memorizing dataset-specific surface patterns, the model can effectively evaluate and repair structures even when the underlying text distribution and argumentation schema change.

5.5 Oracle Analysis

Table 5. Oracle upper bound analysis on AAEC.

Condition	Comp-F1	Rel-F1	ΔRel
Predicted Components	83.71±0.42	64.83±0.38	—
Candidate-Oracle (Best of K=3)	85.12±0.36	67.24±0.34	+2.41
Condition-Oracle (gold input)	N/A	78.42±0.35	+13.59
Output-Oracle (gold replacement)	100.00	83.35±0.32	+18.52

The oracle analysis in Table **5** reveals the remaining headroom of the system and validates the multi-candidate strategy. The Candidate-Oracle (Best of K=3) reveals a theoretical upper bound of 67.24 Relation-F1, confirming that multiple candidates effectively preserve valid hypotheses rather than merely oversampling. Our consistency scoring captures much of this potential, achieving a Top-1 Oracle Agreement of 78.4%. Furthermore, substituting predicted components with gold components boosts Relation-F1 to 78.42 (+13.59 points), highlighting that component extraction remains the primary bottleneck in end-to-end AM.

5.6 Bounded Correction Mechanism Analysis

Table 6. Bounded correction behavior and repair statistics on AAEC.

Metric	Value
Correction Trigger Rate	42.5%
Avg. Iterations (when triggered)	1.8

Metric	Value
Avg. Actions (when triggered)	2.4
LocalRerun	45.2%
RemoveEdge	24.8%
Replace	19.5%
Promote/Demote	10.5%
Invalid ID Ratio	18.5% → 2.1%
Malformed JSON Ratio	6.2% → 0.0%
Incoherent Edge Ratio	22.4% → 5.8%
Overall E_val Pass Rate	75.3% → 97.9%

The correction procedure triggers on 42.5% of documents, highlighting the frequency of structural anomalies in single-pass generation. Repair is efficient, requiring an average of 1.8 iterations and 2.4 actions to resolve violations. Before-and-after comparisons confirm the mechanism's success: hard structural errors (malformed JSON, invalid references) are virtually eliminated, and semantically incoherent edges drop sharply from 22.4% to 5.8%.

5.7 Error Analysis and Limitations

To better understand the boundaries of the MCR-BR framework, we conducted a detailed error analysis on both successfully repaired cases and residual failures.

Successfully Repaired Cases. The majority of successful repairs involve resolving isolated components and invalid references. For example, when the initial generation produces a "Premise" pointing to a non-existent "Claim" ID, the bounded repair loop effectively retrieves an alternative "Claim" from the candidate pool and re-links the premise. Similarly, cyclic dependencies are reliably broken by the RemoveEdge action guided by the verifier's coherence score.

Unrepaired Cases and Limitations. Despite these successes, the framework exhibits certain limitations. First, deep logical errors remain challenging. When a generated argument graph is structurally valid and the individual edges are superficially plausible, the verifier may fail to detect subtle logical fallacies or contradictory reasoning spanning multiple hops. Second, the framework is constrained by the candidate pool quality. If all generated candidates miss a highly implicit component, the repair mechanism cannot recover it. Finally, the reliance on an external verifier introduces a potential bottleneck; if the verifier's semantic judgment is flawed, it may incorrectly reject valid edges or accept incoherent ones.

Computational Cost Trade-offs. A notable limitation of the MCR-BR framework is the increased computational overhead during inference. Generating multiple candidates

(K=3) and running the verifier for consistency scoring inherently multiplies the required FLOPs compared to a single-pass approach. Furthermore, the bounded repair loop introduces sequential decoding steps when anomalies are detected. While we mitigate this by enforcing strict budgets (maximum T=3 iterations and M=2 actions), the framework trades inference speed for structural reliability.

6 Conclusion

We address the error propagation bottleneck in LLM-based end-to-end argument mining by designing the Multi-Candidate Reranking and Bounded Repair (MCR-BR) framework. Relying on a single-pass generation makes models vulnerable to cascading errors. Conversely, MCR-BR preserves multiple structural hypotheses and actively refines them using downstream consistency signals. By integrating candidate preservation, structure-aware scoring, and bounded local repair, our method strongly mitigates global inconsistencies. Evaluations across the AAEC and CDCP benchmarks prove that transforming passive structural constraints into active repair mechanisms significantly enhances both component extraction and relation linking, while facilitating better cross-domain adaptability. Future work will investigate boundary-aware decoding strategies and efficient candidate pruning, extending this closed-loop reasoning to denser argument graphs.

References

1. Wambsganss, T., Janson, A., Söllner, M.: Improving Students' Argumentation Skills Using Dynamic Machine-Learning-Based Modeling. Information Systems Research (2025)
2. Habernal, I. et al.: Mining Legal Arguments in Court Decisions. Artificial Intelligence and Law (2024)
3. Chen, G. et al.: Exploring the Potential of Large Language Models in Computational Argumentation. In: Proc. ACL (2024)
4. Castagna, F. et al.: Computational Argumentation-Based Chatbots: A Survey. Journal of Artificial Intelligence Research (2024)
5. Stab, C., Gurevych, I.: Parsing Argumentation Structures in Persuasive Essays. Computational Linguistics (2017)
6. Park, J., Cardie, C.: Identifying Appropriate Support for Propositions in Online User Comments. In: Proc. ArgMining (2014)
7. Habernal, I., Gurevych, I.: Argumentation Mining in User-Generated Web Discourse. Computational Linguistics (2017)
8. Peldszus, A., Stede, M.: Joint Prediction in MST-style Discourse Parsing for Argumentation Mining. In: Proc. EMNLP (2015)
9. Eger, S. et al.: Neural End-to-End Learning for Computational Argumentation Mining. In: Proc. ACL (2017)
10. Potash, P. et al.: Here's My Point: Joint Pointer Architecture for Argument Mining. In: Proc. EMNLP (2017)
11. Kuribayashi, T. et al.: An Empirical Study of Span Representations in Argumentation Structure Parsing. In: Proc. ACL (2019)

12. Morio, G. et al.: Towards Better Non-Tree Argument Mining: Proposition-Level Biaffine Parsing. In: Proc. ACL (2020)
13. Bao, J. et al.: A Generative Model for End-to-End Argument Mining with Reconstructed Positional Encoding. In: Proc. EMNLP (2022)
14. Kawarada, M., Hirao, T., Uchida, W.: Argument Mining as a Text-to-Text Generation Task. In: Proc. EACL (2024)
15. Gorur, D., Rago, A., Toni, F.: Can Large Language Models Perform Relation-Based Argument Mining? In: Proc. COLING (2025)
16. Song, Y. et al.: Comprehensive Argument Mining for Chinese Argumentative Essays Using Large Language Models. In: Proc. NLPCC (2025)
17. Wei, J. et al.: Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In: Proc. NeurIPS (2022)
18. Wang, X. et al.: Self-Consistency Improves Chain of Thought Reasoning in Language Models. In: Proc. ICLR (2023)
19. Madaan, A. et al.: Self-Refine: Iterative Refinement with Self-Feedback. In: Proc. NeurIPS (2023)
20. Pan, L. et al.: Automatically Correcting Large Language Models: Surveying the Landscape. TACL (2024)
21. Ouyang, S., Zhang, Z., Zhao, H.: Fact-Driven Logical Reasoning for Machine Reading Comprehension. In: Proc. AAAI (2024)
22. Teo, N. et al.: Large Language Models for Logical Fallacy Detection. In: Proc. PAKDD (2025)
23. Roush, A., Shabazz, Y., Balaji, A.: OpenDebateEvidence: A Massive-Scale Argument Mining and Summarization Dataset. In: Proc. NeurIPS (2024)