# Reflective Agreement: Combining Self-Mixture of Agents with a Sequence Tagger for Robust Event Extraction

Anonymous ACL submission

## Abstract

Event Extraction (EE) involves automatically identifying and extracting structured information about events from unstructured text, including triggers, event types, and arguments. Traditional discriminative models demonstrate high precision but often exhibit limited recall, particularly for nuanced or infrequent events. Conversely, generative approaches leveraging Large Language Models (LLMs) provide higher semantic flexibility and recall but suffer from hallucinations and inconsistent predictions. To address these challenges, we propose Agreement-based Reflective Inference System (ARIS), a hybrid approach combining a Self Mixture of Agents with a discriminative sequence tagger. ARIS explicitly leverages structured model consensus, confidence-based filtering, and an LLM reflective inference module to reliably resolve ambiguities and enhance overall event prediction quality. We further investigate decomposed instruction fine-tuning for enhanced LLM event extraction understanding. Experiments demonstrate our approach outperforms existing state-of-the-art event extraction methods across three benchmark datasets.

## 1 Introduction

Event Extraction (EE) aims to identify structured event information from unstructured textual data, including event triggers, event types, and associated arguments with their roles (Doddington et al., 2004). Effective event extraction underpins critical applications in information retrieval, knowledge graph construction, and automated decision-making. Despite considerable advancements, robust event extraction remains challenging, primarily due to linguistic variability, semantic complexity, and limited generalization to infrequent or previously unseen events (Li et al., 2022).

There are two predominant methodologies in EE: discriminative approaches and generative methods leveraging LLMs. Discriminative methods, including transformer-based sequence taggers (e.g., RoBERTa) and structured prediction models, offer superior precision and structural consistency due to their explicit token-level training (Zeng et al., 2022; Liu et al., 2024). However, these methods often struggle with recall, especially for nuanced or rare events not extensively covered by training datasets. Conversely, generative LLM-based approaches (Zhu et al., 2024; Gao et al., 2024) demonstrate enhanced semantic flexibility and contextual understanding, achieving broader coverage and improved recall. Yet, these generative approaches frequently produce inconsistent predictions and hallucinations due to their inherent stochasticity, resulting in lower precision in their predictions (Meng et al., 2024).

Recently, hybrid multi-agent debate-based methods have emerged, leveraging multiple generative LLM agents to iteratively critique and refine predictions (Chan et al., 2024). Although promising, these debate approaches have critical limitations: they rely on iterative, often unstructured discussions without explicit grounding, leading to amplified hallucinations and inconsistent outputs; they lack principled mechanisms for systematically resolving persistent disagreements; and they introduce substantial computational overhead with unpredictable inference times. These shortcomings significantly limit their effectiveness and practical applicability.

In this paper, we introduce ARIS (Agreement-based Reflective Inference System), a hybrid event extraction framework explicitly designed to overcome these limitations. ARIS systematically integrates the complementary strengths of a generative Self Mixture-of-Agents, which uses multiple LLM instances decoding in parallel to promote output diversity, with a discriminative sequence tagger that provides essential structural grounding and precision. ARIS introduces the Reflective Agreement mechanism, a structured inference process that ex-

plicitly leverages model consensus and confidence-based filtering to select high-confidence event predictions, while employing reflective inference to resolve ambiguities systematically. Crucially, our reflective inference module relies on an LLM explicitly trained to understand the complete event extraction chain (trigger identification, trigger classification, argument identification, and argument classification) through decomposed instruction fine-tuning, significantly enhancing the accuracy and reliability of reflective reasoning.

Our contributions are as follows:

- We propose ARIS, a hybrid event extraction framework that systematically integrates generative flexibility and discriminative precision, explicitly addressing the limitations inherent to existing debate-based and standalone generative approaches.
- We introduce Reflective Agreement, a novel structured reflective inference mechanism that leverages explicit model agreement, confidence-based filtering, and contextual reflective reasoning to robustly resolve ambiguous predictions.
- We demonstrate empirically that ARIS achieves state-of-the-art performance across three event extraction benchmarks, consistently surpassing discriminative, generative, and existing hybrid debate-based methods. Beyond empirical results, ARIS advances theoretical understanding by providing new insights into structured reflective reasoning and hybrid model integration for complex NLP tasks.

## 2 Related Work

**Event Extraction with LLMs** LLMs have emerged as promising tools for Event Extraction tasks, offering strengths in contextual understanding and handling linguistic variation. Several studies have investigated zero-shot and few-shot prompting approaches for Event Extraction with LLMs (Chen et al., 2024). More sophisticated prompting frameworks like the Debate as Optimization (DAO) (Wang and Huang, 2024) employ multiple agent roles to iteratively refine event extraction predictions through structured debate. Other researchers have explored hybrid approaches combining task-specific models with LLMs. LC4EE (Zhu et al., 2024) uses task-specific models for initial Event Extraction, then employs manually defined rules to guide an LLM in verifying and correcting the output. Recent work has begun exploring fine-tuning approaches, with studies incorporating textual descriptions of event types into instruction tuning datasets (Srivastava et al., 2025) or combining Supervised Fine-Tuning with reinforcement learning (Gao et al., 2024). However, significant limitations persist across these approaches. Prompting-only methods typically underperform supervised fine-tuning of smaller discriminative models, such as RoBERTa-based models. Hybrid approaches rely on manual rule creation, which limit scalability. Importantly, there is also limited work on systematically designing instruction tuning datasets that address the distinct challenges of event extraction's core subtasks: trigger identification, trigger classification, argument identification, and argument classification. Consequently, many LLM-based methods still fail to outperform supervised fine-tuning of task-specific models.

**LLM Instruction Fine-Tuning** Instruction fine-tuning has emerged as a powerful approach for enhancing language models' capabilities on specific tasks. Recent advancements have focused on structuring the fine-tuning process to improve reasoning abilities and handle complex tasks more effectively. Chain-of-Thought (CoT) fine-tuning has gained significant attention, where instruction datasets are augmented with reasoning rationales. This enables the models to learn reasoning capabilities (Kim et al., 2023; Zelikman et al., 2022; Ho et al., 2023). Compositional Fine-Tuning addresses complex tasks by explicitly breaking them down into simpler component subtasks (Bursztyn et al., 2022). Rather than using end-to-end learning, CFT fine-tunes models on a set of component tasks, as well as the end-to-end task, enabling them to learn the end-to-end task more effectively.

**LLM Inference Time Improvement** Advanced inference strategies significantly enhance LLM performance without requiring model parameter updates. Chain-of-Thought prompting (Wei et al., 2022) enables LLMs to break down complex problems into intermediate reasoning steps, improving performance on tasks requiring compositional reasoning. Tree of Thoughts (Yao et al., 2023) extends this approach by allowing models to explore multiple reasoning paths simultaneously, evaluating alternatives and backtracking when necessary. Retrieval-Augmented Generation (Lewis

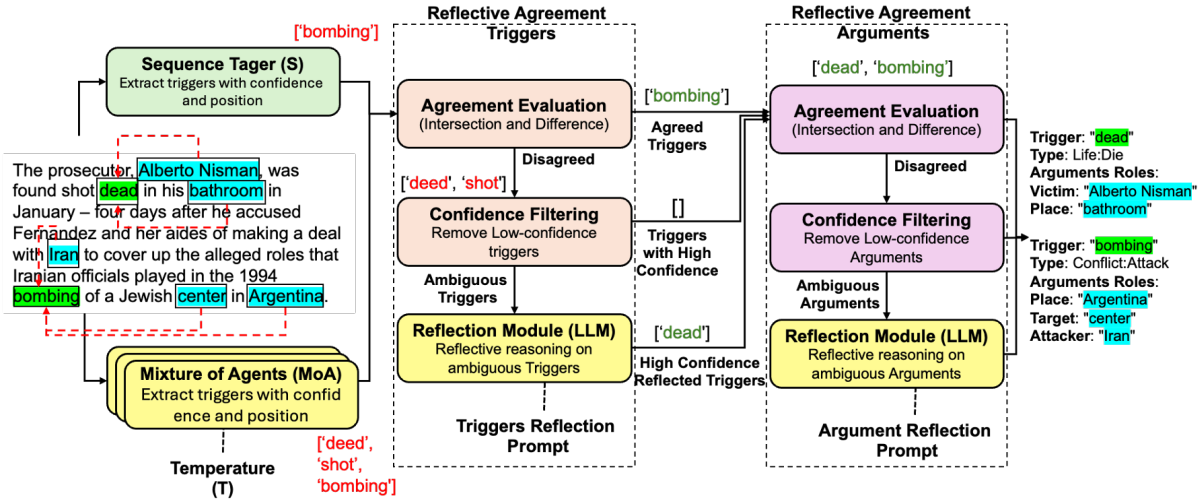


Figure 1: Overview of the proposed ARIS framework illustrating the Reflective Agreement process. ARIS systematically integrates predictions from a discriminative sequence tagger and a generative Self Mixture of Agents. Event triggers and arguments are extracted by each model with associated positions and confidence scores. The framework identifies consented predictions, filters out low-confidence disagreements, and employs a reflective inference module to resolve remaining ambiguities, ultimately producing robust, accurate, and structurally grounded event extraction results.

et al., 2020) incorporates external knowledge retrieval to improve factuality and reduce hallucination. Recent work has also explored self-correction mechanisms for LLMs, where LLMs can iteratively self-correct their outputs through interaction with external tools (Gou et al., 2024), or use episodic memory buffers to improve decision-making in subsequent attempts (Shinn et al., 2023).

Mixture of Agents (MoA) (Wang et al., 2025) is an ensemble approach that combines predictions from multiple different LLMs to improve performance through complementary strengths of diverse models. Self Mixture of Agents (Self-MoA) (Li et al., 2025) extends this concept by using multiple instances of the same model with different sampling parameters to generate diverse outputs. While these approaches have shown promise in general language generation tasks, their systematic application to structured prediction tasks like event extraction remains underexplored.

Our work builds upon these advances in LLM fine-tuning and self-correction mechanisms, specifically addressing the challenges of Event Extraction by developing a decomposed instruction fine-tuning approach combined with a structured self-reflection module that enables effective reasoning about event structures. Additionally, we are the first to systematically apply Self-MoA to event extraction, leveraging multiple instances of the same LLM to generate diverse candidate events that are then refined through agreement detection and confidence-based filtering with a discriminative sequence tagger.

## 3 Methodology

ARIS aims to enhance event extraction by integrating generative and discriminative approaches through structured model consensus and reflective reasoning. As illustrated in Figure 1, ARIS initiates with an explicit fine-tuning phase to equip an LLM with specialized capabilities for event subtasks, including trigger identification, event classification, and argument extraction. Following fine-tuning, ARIS implements a Self Mixture of Agents (Self-MoA), leveraging the fine-tuned LLM to generate diverse candidate events. Concurrently, a discriminative sequence tagging model independently predicts events. To consolidate predictions, ARIS employs structured consensus detection and confidence-based filtering, selectively retaining high-confidence agreements while discarding uncertain disagreements. To systematically address remaining ambiguities, ARIS utilizes a reflective inference module that capitalizes on the fine-tuned LLM's contextual reasoning capabilities. The subsequent sections detail the implementation and interactions of these key components. The detailed procedure of our approach is formalized as Algorithm 1, which can be found in Appendix E.

### 3.1 Self Mixture of Agents for Event Extraction

We formalize our approach with the following notation. Let $A = \{A_1, A_2, \ldots, A_n\}$ be the Self Mixture of Agents, where each agent $A_i$ is an LLM with temperature $T_i$.

**Event Decomposed Fine-Tuning.** To enhance the LLM's base understanding of the event extraction task, we develop a decomposed instruction fine-tuning dataset to explicitly guide the LLM to master distinct subtasks inherent to event extraction.

Given an event extraction dataset $D_{event} = \{(x_i, y_i)\}_{i=1}^{N}$ (input texts $x_i$ and corresponding event annotations $y_i$), we convert it into a decomposed instructional dataset $D_{decomp}$ structured around three primary subtasks:

First, we create holistic event structure modeling instructions that supervise complete event construction. These include full-structure construction tasks requiring the model to output a complete list of events in the passage (with triggers, types, and arguments with roles), and role-ablated construction variants that systematically mask one argument role per instance, requiring the model to infer the remainder while maintaining structural coherence.

Second, we develop trigger-focused reasoning instructions that isolate the foundational stages of event extraction: trigger detection focuses solely on identifying trigger spans; type classification assigns event types to known triggers (both individually and in batches); trigger discrimination provides binary classification supervision for distinguishing triggers from non-triggers; and joint trigger-type prediction unifies detection and classification into a single structured output.

Third, we create argument-level inference instructions that target post-trigger prediction. Argument extraction requires identifying argument spans for known triggers, role assignment classifies the role of each known argument, and joint argument-role prediction unifies extraction and classification into a single coherent operation.

We fine-tune the initial LLM $M_{LLM}^{init}$ on the decomposed dataset $D_{decomp}$, where the model first learns atomic subtasks (trigger identification, argument extraction) before progressing to intermediate compositional tasks (joint trigger-type prediction, role assignment) and finally to full event structure generation. Further details on this dataset construction can be found in Appendix F.

### 3.2 Hybrid Event Aggregation

Let $S$ be a pretrained sequence tagger (e.g., a fine-tuned transformer such as RoBERTa) trained for event extraction. For an input document $x$ (sentence or article), we define $G_x$ as the space of possible valid event spans in $x$. Each LLM agent $A_i$ produces event extraction predictions $E_{A_i}(x; T_i) \subset G_x$. Simultaneously, the sequence tagger produces its own set of event predictions $E_S(x) \subset G_x$. Initially, the predictions from all individual LLM agents are aggregated to create a preliminary combined prediction set $E_{SMoA}^{raw}(x) = \bigcup_{i=1}^{n} E_{A_i}(x; T_i)$.

Since the LLM agents generate multiple independent predictions through the self mixture of agents approach, these parallel predictions may identify the same trigger multiple times or reference non-existent spans due to hallucination. To ensure accurate alignment between model predictions and the source text, we apply a rule-based cleanup mechanism that consists of two sequential steps: span validation and positional sorting.

We first filter out predictions that reference non-existent text spans. Let $S_{text}(x) = \{s_1, s_2, \ldots, s_m\}$ be the set of all possible contiguous text spans present in the input document $x$. We retain only predictions whose spans exist in the source text, creating $E_{SMoA}^{valid}(x)$ which contains only events $e$ from $E_{SMoA}^{raw}(x)$ where $\text{span}(e) \in S_{text}(x)$.

We sort the valid predictions by their textual positions to establish a canonical ordering, producing $E_{SMoA}(x) = \text{sort}(E_{SMoA}^{valid}(x), \text{by position in } x)$.

The resulting cleaned prediction set $E_{SMoA}(x)$ serves as the foundation for subsequent agreement detection and disagreement handling steps.

### 3.3 Consensus Detection

We identify consensus predictions as events jointly extracted by both the Self-MoA and sequence tagger, defined as $E_{con}(x) = E_{SMoA}(x) \cap E_S(x)$. Events match when they have the same trigger identification, trigger classification, argument identification and argument classification predictions. These consensus predictions represent the most reliable predictions, where both generative and discriminative methods converge on the same result. Further details on the exact mechanism for consensus detection can be found in Appendix A.

### 3.4 Disagreement Handling via Confidence Filtering

For cases where the Self-MoA and sequence tagger disagree, we employ a confidence-based filtering strategy. We first define the complete set of predictions $E_{comb}(x) = E_{SMoA}(x) \cup E_S(x)$ and the disagreement set $E_{dis}(x) = E_{comb}(x) \setminus E_{con}(x)$.

We then compute confidence scores for predictions in the disagreement set. To calculate confidence for Self-MoA predictions, we need to track the origin of each prediction. Let $A_e = \{i : e \in E_{A_i}(x; T_i)\}$ be the set of agent indices that predicted event $e$. For Self-MoA predictions, confidence is calculated as the proportion of agents that made the same prediction:

$$C_{SMoA}(e) = \frac{|A_e|}{n} = \frac{|\{i : e \in E_{A_i}(x; T_i)\}|}{n}$$

For sequence tagger predictions, confidence is derived from the softmax score of the predicted token in the output layer. Let $T_{tags}$ be the set of all possible event tags in the sequence tagger's output vocabulary (including event types and argument roles), and let $\text{span}(e)$ denote the text span associated with event $e$. The confidence is calculated as the maximum probability over all tags for the given span: $C_S(e) = \max_{t \in T_{tags}} P_S(t|\text{span}(e), x)$, where $P_S(t|\text{span}(e), x)$ is the probability distribution over possible tags for the span of event $e$ given the input text $x$.

We define confidence thresholds $\theta_{SMoA}$ and $\theta_S$ to filter out low-confidence predictions. Given the sequence tagger's superior precision in predictions, high-confidence sequence tagger predictions that disagree with the Self-MoA are retained in the final prediction set. Specifically, for sequence tagger predictions with confidence exceeding the threshold ($C_S(e) \geq \theta_S$), we include them directly in the set $E^S_{hi\_conf}(x) = \{e \in E_S(x) \cap E_{dis}(x) \mid C_S(e) \geq \theta_S\}$.

For low-confidence predictions from both models, we apply confidence-based filtering. Let $E_{rem}(x)$ be the set of events to be removed due to low confidence, where $E^{SMoA}_{rem} = \{e \in E_{SMoA}(x) \cap E_{dis}(x) \mid C_{SMoA}(e) < \theta_{SMoA}\}$ represents low-confidence Self-MoA predictions and $E^S_{rem} = \{e \in E_S(x) \cap E_{dis}(x) \mid C_S(e) < \theta_S\}$ represents low-confidence sequence tagger predictions. The complete set of removed events is then $E_{rem}(x) = E^{SMoA}_{rem} \cup E^S_{rem}$.

### 3.5 Reflection on Ambiguous Predictions

The remaining disagreement predictions after confidence filtering represent ambiguous cases that require further analysis. We define this set as $E_{reflect}(x) = E_{dis}(x) \setminus E_{rem}(x)$, capturing all disagreements not removed during filtering.

We resolve these ambiguous cases through a reflection mechanism $R$ that formulates a structured query containing the original text context $x$ and the ambiguous predictions $E_{reflect}(x)$. This query presents each ambiguous prediction along with its surrounding context, asking the LLM to analyze and determine the correct prediction based on linguistic cues, event semantics, and contextual understanding. The reflection process leverages the LLM's reasoning capabilities to produce a refined set of resolved predictions $E_{reflected}(x) = R(E_{reflect}(x), x)$. Further details on this procedure can be found in Appendix C.

### 3.6 Final Prediction Set

The final prediction set combines high-confidence agreed predictions with those resolved through reflection, forming $E_{fin}(x) = E_{con}(x) \cup E^S_{hi\_conf}(x) \cup E_{reflected}(x)$. This approach leverages the complementary strengths of both generative and discriminative models: the structural consistency and precision of sequence taggers for straightforward cases, and the contextual reasoning capabilities of LLMs for resolving complex ambiguities.

As shown in Figure 1, our framework effectively handles hallucinations and disagreements between models. For instance, in the example text "The prosecutor, Alberto Nisman, was found shot dead in his bathroom in January - four days after he accused Fernandez and her aides of making a deal with Iran to cover up the alleged roles that Iranian officials played in the 1994 bombing of a Jewish center in Argentina.", the sequence tagger identifies only the trigger ['bombing'], while the MoA detects multiple candidates including ['dead', 'shot', 'bombing']. Through our reflective agreement process, the incorrect trigger 'shot' is filtered out, resulting in the accurate final triggers ['dead', 'bombing']. This demonstrates how ARIS combines discriminative precision with generative coverage while eliminating hallucinations.

| Base LLM | Approach | CASIE | | | | M2E2 | | | | MLEE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Trg-I | Trg-C | Arg-I | Arg-C | Trg-I | Trg-C | Arg-I | Arg-C | Trg-I | Trg-C | Arg-I | Arg-C |
| | TagPrime | **72.00** | **71.60** | 47.47 | 45.61 | 63.97 | 63.30 | 37.47 | 34.19 | 74.61 | 72.38 | 48.30 | 46.74 |
| | DEBATE-EE | – | 41.80 | – | 40.50 | – | – | – | – | – | – | – | – |
| | MMUTF | – | – | – | – | – | 55.50 | – | 38.20 | – | – | – | – |
| Llama-3.1 8B | One-Shot | 0.15 | 0.15 | 0.00 | 0.00 | 3.77 | 3.77 | 0.40 | 0.40 | 0.23 | 0.23 | 0.00 | 0.00 |
| | FineTuned-EE | 42.59 | 42.27 | 26.72 | 25.30 | 66.87 | 63.16 | 31.94 | 27.78 | 50.64 | 44.96 | 38.75 | 34.93 |
| | FineTuned-DEE | 65.89 | 65.34 | 44.60 | 42.56 | 67.43 | 64.00 | 36.33 | 32.80 | 55.40 | 52.14 | 50.98 | 48.33 |
| | ARIS | 70.78 | 70.27 | **48.79** | **46.84** | 73.49 | **71.39** | **41.41** | **38.84** | 73.80 | 70.33 | 54.30 | 50.86 |
| Phi-3 7B | One-Shot | 3.11 | 2.49 | 0.85 | 0.60 | 31.37 | 27.45 | 9.35 | 4.68 | 1.13 | 0.68 | 0.72 | 0.72 |
| | FineTuned-EE | 41.99 | 41.32 | 26.83 | 25.74 | 59.44 | 55.11 | 28.14 | 24.31 | 29.11 | 26.53 | 26.98 | 24.25 |
| | FineTuned-DEE | 62.26 | 61.53 | 42.00 | 40.55 | 67.26 | 64.31 | 32.30 | 29.57 | 37.89 | 35.52 | 45.93 | 44.31 |
| | ARIS | 69.08 | 68.39 | 46.63 | 44.99 | **74.22** | **71.39** | 41.11 | 36.61 | **74.78** | **72.45** | **59.19** | **56.98** |

Table 1: F1 score of Event Extraction performance (Trg=trigger, Arg=argument; I=identification, C=classification) across three benchmark datasets. Bold numbers indicate best performance on evaluation metric.

# 4 Experiments

## 4.1 Dataset and Evaluation Metrics

We evaluate our approach on three benchmark datasets for event extraction processed following the TextEE benchmark standardization process (Huang et al., 2024): CASIE (Satyapanich et al., 2020), M2E2 (Li et al., 2020), and MLEE (Pyysalo et al., 2012). We use the train/dev/test partitions defined in TextEE's "split1" for all three datasets. For M2E2, we used only the text, and did not include any image or video information. These datasets represent diverse domains and text structures: CASIE covers cybersecurity news with 5 event types in long paragraphs; M2E2 contains shorter news content with 8 event types primarily in 1-2 sentence format; and MLEE represents the biomedical domain with 29 event types across long paragraphs.

For evaluation, we report micro F1 scores for the following tasks: Trigger Identification, which evaluates the model's ability to correctly identify event trigger spans in text, regardless of event type; Trigger Classification, which measures performance in both identifying event triggers and correctly classifying their event types; Argument Identification, which assesses the model's capability in identifying argument entities associated with correctly identified event triggers; and Argument Classification, which requires correct identification of both the argument entity and its role assignment for a given event trigger. For all metrics, we employ exact match scoring.

## 4.2 Implementation

For the discriminative sequence tagger component of our proposed approach, we utilize TagPrime (Hsu et al., 2023), a unified framework for relational structure extraction that has demonstrated superior performance on event extraction tasks. We implement TagPrime with roberta-large from huggingface (FacebookAI, 2024) as the backbone encoder (Liu et al., 2019).

For the generative component, we employ Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and Phi-3-small-8k-instruct (Abdin et al., 2024). We access both models through their respective Hugging Face implementations (Meta-Llama, 2024; Microsoft, 2024). To train both LLMs for event extraction, we used LoRA (Hu et al., 2022) implemented with the Hugging Face PEFT library (Mangrulkar et al., 2022). Our LoRA configuration uses a rank of 32, a scaling factor ($\alpha$) of 128, and a dropout rate of 0.05. In all of our experiments we utilize 10 Self-MoA Agents that have a temperature of 0.9 unless stated otherwise. For our confidence-based filtering for our ARIS approach, we dynamically determined dataset-specific thresholds for each model and temperature setting to optimize system performance. These thresholds and details on how they were computed, are reported in Appendix B.

## 4.3 Overall Event Extraction Performance

We compare our proposed event extraction approach against several baseline approaches and state-of-the-art methods across multiple configurations. Our baselines include various LLM-based approaches using both zero-shot and few-shot prompting, as well as fine-tuned variants. The One-Shot baseline employs off-the-shelf LLMs with carefully designed prompts that explain the event extraction task and dataset structure, and provide a single

| Base LLM | Approach | Temp. | Trg-I | | | Trg-C | | | Arg-I | | | Arg-C | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| Llama-3.1 8B | Self-MoA | 0.9 | 46.99 | 71.17 | 56.51 | 44.98 | 68.29 | 54.15 | 28.03 | 63.15 | 38.76 | 26.28 | 59.09 | 36.32 |
| | Self-MoA | 0.6 | 53.12 | 68.42 | 59.72 | 50.99 | 65.79 | 57.37 | 30.73 | 58.79 | 40.33 | 28.81 | 55.02 | 37.80 |
| | Self-MoA | 0.1 | 63.18 | 61.31 | 62.07 | 60.79 | 59.02 | 59.75 | 39.25 | 52.83 | 44.98 | 37.19 | 50.03 | 42.61 |
| | ARIS | 0.9 | 69.16 | 76.85 | 72.69 | 67.20 | 74.75 | 70.66 | 42.94 | 55.27 | 48.17 | 40.60 | 52.19 | 45.51 |
| | ARIS | 0.6 | 68.26 | 76.59 | 72.15 | 66.37 | 74.52 | 70.17 | 48.56 | 50.52 | 49.37 | 46.30 | 48.04 | 46.94 |
| | ARIS | 0.1 | 66.72 | 77.19 | 71.56 | 64.72 | 74.92 | 69.43 | 44.78 | 53.06 | 48.38 | 42.61 | 50.53 | 46.59 |
| Phi-3 7B | Self-MoA | 0.9 | 39.02 | 73.50 | 50.47 | 37.15 | 70.34 | 48.13 | 24.80 | 60.23 | 35.08 | 23.30 | 56.55 | 32.95 |
| | Self-MoA | 0.6 | 46.29 | 69.46 | 55.02 | 44.27 | 66.75 | 52.72 | 32.35 | 58.62 | 41.54 | 30.69 | 55.43 | 39.37 |
| | Self-MoA | 0.1 | 59.77 | 60.80 | 59.82 | 57.15 | 58.35 | 57.30 | 41.40 | 49.14 | 44.62 | 39.52 | 46.73 | 42.52 |
| | ARIS | 0.9 | 71.55 | 74.81 | 72.69 | 69.58 | 72.86 | 70.74 | 47.16 | 51.65 | 48.79 | 44.52 | 48.64 | 46.00 |
| | ARIS | 0.6 | 73.49 | 74.52 | 73.78 | 71.46 | 72.54 | 71.78 | 50.01 | 49.33 | 49.43 | 47.68 | 46.94 | 47.06 |
| | ARIS | 0.1 | 68.84 | 76.03 | 72.04 | 66.70 | 73.77 | 69.85 | 46.64 | 50.76 | 48.38 | 44.63 | 48.45 | 46.24 |

Table 2: Impact of sampling temperature on event extraction performance. Results show precision (P), recall (R), and F1 scores across different sampling temperatures.

event extraction example without any task-specific training. For each test instance, we selected the most similar training example using TF-IDF vectorization (Salton and Buckley, 1988) with cosine similarity, ensuring that the provided examples are contextually relevant to the test cases.

For fine-tuned approaches, we implement two training strategies: FineTuned-EE represents standard end-to-end fine-tuning where the LLM learns to directly map input text to complete event structures (event triggers, event types, event arguments, and event argument roles) in a single step. In contrast, FineTuned-DEE leverages our proposed decomposed instruction fine-tuning approach (Section 3.1), where the model first learns individual subtasks before progressing to complete event extraction. For both FineTuned-EE and FineTuned-DEE, inference is performed using a single LLM with a temperature setting of 0.9. Our final proposed method, ARIS, combines the decomposed fine-tuning with our proposed inference framework that integrates the Self Mixture of Agents, consensus detection, confidence-based filtering, and reflection mechanisms.

We compare against several strong baselines including TagPrime (Hsu et al., 2023), a discriminative sequence tagging model that represents the current state-of-the-art for event extraction tasks, as well as recent LLM-based approaches: DEBATE-EE (Wang and Huang, 2024), which employs a multi-agent debate framework that iteratively refines event extraction predictions through discussions between debating agents, critics, and judges, enhanced with diverse retrieval-augmented generation and adaptive conformal prediction modules, and MMUTF (Seeberger et al., 2024), a unified template filling framework that extracts event arguments by matching candidates to argument roles using templates as queries. The results shown for DEBATE-EE and MMUTF rows are the F1 scores provided in their original papers.

The results in Table 1 demonstrate that our proposed approach achieves significant improvements over baseline methods across all three datasets. Our ARIS approach consistently outperforms competing methods, particularly in argument extraction tasks.

On CASIE, our method is competitive with TagPrime for trigger detection while surpassing it on argument tasks. For M2E2, we achieve the strongest performance across all metrics, with both Llama-3.1 and Phi-3 implementations of ARIS significantly outperforming TagPrime. On MLEE, our Phi-3 based ARIS implementation shows the strongest performance across all metrics. Notably, ARIS with Phi-3 improves argument classification F1 scores over the TagPrime model, surpassing the strong baseline by over 10 points.

The results of the one-shot experiment show that LLMs may struggle with event extraction when using basic prompting strategies, showing the need for fine-tuning on the task. The effectiveness of decomposed instruction fine-tuning is evident when comparing FineTuned-DEE with standard FineTuned-EE, showing consistent improvements across all datasets. The ARIS framework further enhances performance, particularly for argument-related tasks. These results validate our hypothesis that combining the complementary strengths of discriminative models and LLMs through our reflective agreement approach effectively addresses the limitations of individual approaches.

| Base LLM | Approach | Trg-I | | | Trg-C | | | Arg-I | | | Arg-C | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| Llama-3.1 8B | Self-MoA | 46.99 | 71.17 | 56.51 | 44.98 | 68.29 | 54.15 | 28.03 | **63.15** | 38.76 | 26.28 | **59.09** | 36.32 |
| | ARIS w/o TagPrime | 60.87 | 60.08 | 60.31 | 58.69 | 58.08 | 58.23 | 33.26 | 57.10 | 42.03 | 31.38 | 53.87 | 39.66 |
| | ARIS | 69.16 | **76.85** | **72.69** | 67.20 | **74.75** | 70.66 | 42.94 | 55.27 | 48.17 | 40.60 | 52.19 | 45.51 |
| Phi-3 7B | Self-MoA | 39.02 | 73.50 | 50.47 | 37.15 | 70.34 | 48.13 | 24.80 | 60.23 | 35.08 | 23.30 | 56.55 | 32.95 |
| | ARIS w/o TagPrime | 53.52 | 64.50 | 57.52 | 50.84 | 61.65 | 54.78 | 31.30 | 52.44 | 38.53 | 29.38 | 49.55 | 36.26 |
| | ARIS | 71.55 | 74.81 | **72.69** | 69.58 | 72.86 | **70.74** | 47.16 | 51.65 | **48.79** | 44.52 | 48.64 | **46.00** |
| | TagPrime | **76.64** | 65.33 | 70.19 | **75.43** | 64.32 | 69.09 | **49.34** | 40.88 | 44.41 | **46.76** | 38.88 | 42.18 |

Table 3: Ablation study demonstrating component contributions to event extraction performance. Results highlight the complementary strengths of discriminative (TagPrime) and generative approaches.

### 4.4 Impact of Temperature on Self-MoA LLMs

To understand how sampling diversity impacts performance, we evaluated our approach across three temperature settings (t=0.9, t=0.6, and t=0.1) during inference. Table 2 presents the averaged results across all datasets.

The results reveal that while temperature substantially affects standalone Self-MoA performance, the full ARIS approach maintains consistent performance across all settings. For Self-MoA alone, temperature variations produce dramatic differences in precision-recall trade-offs, where higher temperatures lead to higher recall and lower precision. We observe dramatic shifts in precision-recall balance for Phi-3, with precision increasing from 39.02% to 59.77% for trigger identification while recall drops from 73.50% to 60.80%.

In contrast, our full ARIS approach demonstrates stability across the tested temperatures, with F1 scores for all metrics showing variation of less than 2 points. This stability demonstrates that our confidence-based filtering, agreement detection, and reflection mechanisms effectively normalize the varying predictions produced at different temperatures.

#### 4.4.1 Ablation Study

To understand the contributions of different components in our approach, we conducted an ablation study focusing on the integration of the sequence tagger with the ARIS framework. Table 3 presents results averaged across all datasets, comparing our full ARIS approach against variants with components removed.

The results reveal clear complementary strengths between the discriminative and generative components. The RoBERTa-based TagPrime sequence tagger demonstrates superior precision across all tasks, but shows lower recall. Conversely, the ARIS approach without TagPrime or reflection exhibits higher recall, but show lower precision.

Our full ARIS approach effectively leverages these complementary strengths that results in higher F1 scores across all metrics. The improvements are even more pronounced for argument-related tasks. The practical impact of these complementary strengths is demonstrated through detailed pipeline examples in Appendix H. These examples trace the complete processing flow from initial predictions through agreement detection, confidence filtering, and reflection, providing concrete illustrations of how ARIS effectively leverages the precision of discriminative models and the semantic flexibility of generative approaches to improve event extraction performance.

## 5 Conclusion

In this paper, we introduced ARIS, a hybrid event extraction method that combines the complementary strengths of discriminative sequence taggers and generative LLMs through a structured reflective agreement mechanism. Our approach leverages a Self Mixture of Agents to generate diverse event predictions, employs agreement detection to identify high-confidence consensus predictions, applies confidence-based filtering to eliminate low-precision candidates, and utilizes a reflection mechanism powered by decomposed instruction fine-tuning to resolve ambiguous cases. Experiments across three benchmark datasets demonstrate that ARIS consistently outperforms existing state-of-the-art methods, with particularly notable improvements in argument extraction tasks. Beyond empirical performance gains, our work advances the theoretical understanding of hybrid model integration and structured reflective reasoning in complex NLP tasks.

## Limitations

While our proposed approach demonstrates improvements in event extraction performance, it comes with significant computational overhead. The approach requires running multiple LLM instances for the Self-MoA component, which substantially increases both inference time and computational resources compared to traditional discriminative models. Additionally, the training process for decomposed instruction fine-tuning demands considerable GPU resources and time, particularly when working with larger LLMs. A further limitation concerns the reflection mechanism itself, which handles the most ambiguous cases that neither the Self-MoA nor discriminative model could confidently resolve. While effective for some difficult instances, the reflection component may still struggle with highly challenging cases. This is an inherent ceiling to the reflection-based approach when confronted with the hardest examples.

## References


Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, and 1 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Victor Bursztyn, David Demeter, Doug Downey, and Larry Birnbaum. 2022. Learning to perform complex tasks through compositional fine-tuning of language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1676–1686.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards better LLM-based evaluators through multi-agent debate. In *The Twelfth International Conference on Learning Representations*.

Ruirui Chen, Chengwei Qin, Weifeng Jiang, and Dongkyu Choi. 2024. Is a large language model a good annotator for event extraction? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17772–17780.

George R Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ace) program–tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*.

FacebookAI. 2024. Facebookai-roberta-large. https://huggingface.co/FacebookAI/roberta-large.

Jun Gao, Huan Zhao, Wei Wang, Changlong Yu, and Ruifeng Xu. 2024. Eventrl: Enhancing event extraction with outcome supervision for large language models. *arXiv preprint arXiv:2402.11430*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14852–14882.

I-Hung Hsu, Kuan-Hao Huang, Shuning Zhang, Wenxin Cheng, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2023. Tagprime: A unified framework for relational structure extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12917–12932.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Kuan-Hao Huang, I-Hung Hsu, Tanmay Parekh, Zhiyu Xie, Zixuan Zhang, Prem Natarajan, Kai-Wei Chang, Nanyun Peng, and Heng Ji. 2024. Textee: Benchmark, reevaluation, reflections, and future challenges in event extraction. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 12804–12825.

Seungone Kim, Se Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 2023. The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12685–12708.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Manling Li, Alireza Zareian, Qi Zeng, Spencer Whitehead, Di Lu, Heng Ji, and Shih-Fu Chang. 2020. Cross-media structured common space for multimedia event extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational*

*Linguistics*, pages 2557–2568, Online. Association for Computational Linguistics.

Qian Li, Jianxin Li, Jiawei Sheng, Shiyao Cui, Jia Wu, Yiming Hei, Hao Peng, Shu Guo, Lihong Wang, Amin Beheshti, and 1 others. 2022. A survey on deep learning event extraction: Approaches and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 35(5):6301–6321.

Wenzhe Li, Yong Lin, Mengzhou Xia, and Chi Jin. 2025. Rethinking mixture-of-agents: Is mixing different large language models beneficial? *arXiv preprint arXiv:2502.00674*.

Wanlong Liu, Li Zhou, DingYi Zeng, Yichen Xiao, Shaohuan Cheng, Chen Zhang, Grandee Lee, Malu Zhang, and Wenyu Chen. 2024. Beyond single-event extraction: Towards efficient document-level multi-event argument extraction. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9470–9487, Bangkok, Thailand. Association for Computational Linguistics.

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*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Zihao Meng, Tao Liu, Heng Zhang, Kai Feng, and Peng Zhao. 2024. Cean: Contrastive event aggregation network with llm-based augmentation for event extraction. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 321–333.

Meta-Llama. 2024. Llama-3.1-8B-Instruct. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct.

Microsoft. 2024. Phi-3-small-8k-instruct. https://huggingface.co/microsoft/Phi-3-small-8k-instruct.

Sampo Pyysalo, Tomoko Ohta, Makoto Miwa, Han-Cheol Cho, Jun'ichi Tsujii, and Sophia Ananiadou. 2012. Event extraction across multiple levels of biological organization. *Bioinformatics*, 28(18):i575–i581.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Taneeya Satyapanich, Francis Ferraro, and Tim Finin. 2020. Casie: Extracting cybersecurity event information from text. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

Philipp Seeberger, Dominik Wagner, and Korbinian Riedhammer. 2024. MMUTF: Multimodal multimedia event argument extraction with unified template filling. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6539–6548, Miami, Florida, USA. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Saurabh Srivastava, Sweta Pati, and Ziyu Yao. 2025. Instruction-tuning llms for event extraction with annotation guidelines. *arXiv preprint arXiv:2502.16377*.

Junlin Wang, Jue WANG, Ben Athiwaratkun, Ce Zhang, and James Zou. 2025. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*.

Sijia Wang and Lifu Huang. 2024. Debate as optimization: Adaptive conformal prediction and diverse retrieval for event extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16422–16435, Miami, Florida, USA. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Qi Zeng, Qiusi Zhan, and Heng Ji. 2022. EA$^2$E: Improving consistency with event awareness for document-level argument extraction. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2649–2655, Seattle, United States. Association for Computational Linguistics.

Mengna Zhu, Kaisheng Zeng, JibingWu JibingWu, Lihua Liu, Hongbin Huang, Lei Hou, and Juanzi Li. 2024. LC4EE: LLMs as good corrector for event extraction. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12028–12038, Bangkok, Thailand. Association for Computational Linguistics.

# A Agreement Detection

Agreement detection reconciles event predictions from the Self-MoA ensemble and the sequence tagger by identifying cases where both systems refer to the same underlying event mention. This process involves separate matching criteria for triggers and arguments, as detailed below.

## A.1 Trigger Span Agreement

Trigger predictions from both models are considered to be in agreement if their textual spans overlap beyond a predefined threshold, indicating they refer to the same underlying event mention. This criterion effectively handles partial overlaps, such as when one span is a substring of another—for example, "`attached`" versus "`was attached`"—as they semantically represent the same trigger. In these scenarios, the span predicted by the sequence tagger is retained due to its higher precision in determining exact span boundaries.

## A.2 Argument Span Agreement

To determine agreement between argument predictions, we first align them based on their associated trigger and event type. For each matched trigger between the two systems, its candidate arguments are evaluated independently for span-level overlap. This enables partial agreement at the argument level: a single trigger may have some arguments in agreement and others not, depending on their span overlap. For example, arguments like "`the government officials`" and "`government officials`" linked to the same trigger and event type are considered to be in agreement. In such cases, we retain the span predicted by the sequence tagger due to its higher precision in boundary identification.

# B Confidence Threshold Selection

Predictions that are not in agreement between the two systems enter this phase. Confidence thresholds for filtering event predictions were dataset-specific, determined by analyzing the distribution of confidence scores on validation sets. For each dataset, we:

1. Computed confidence score distributions separately for correct (found in gold annotations) and incorrect predictions.
2. Used descriptive statistics (mean, median, quartiles) to guide a targeted search range for optimal thresholds.
3. Conducted a search within this range, selecting thresholds that maximized the validation set $F_1$ score.

This threshold selection procedure was repeated individually for trigger and argument predictions, ensuring dataset-specific tuning that improved overall performance. After the agreement detection phase, predictions identified as disagreements are handled based on finalized confidence thresholds: high-confidence disagreements are retained directly, low-confidence disagreements are discarded immediately, and intermediate-confidence cases are forwarded to the reflection mechanism for further analysis.

## B.1 Dataset and Temperature Specific Confidence Thresholds

These two tables present the per-dataset, per-temperature confidence thresholds applied during disagreement handling. Table 4 gives the trigger-level thresholds, while Table 5 lists the argument-level thresholds. In both tables, $\theta_S$ is the sequence-tagger (TagPrime) retention threshold, $\theta^{+}_{\text{SMoA}}$ is the high-confidence Self-MoA keep threshold, and $\theta^{-}_{\text{SMoA}}$ is the low-confidence Self-MoA drop threshold.

| Model | Dataset | Temp | $\theta_S$ | $\theta^{+}_{\text{SMoA}}$ | $\theta^{-}_{\text{SMoA}}$ |
|---|---|---|---|---|---|
| Phi-3 | M2E2 | 0.1 | 0.90 | 0.90 | 0.20 |
| | | 0.6 | 0.89 | 0.95 | 0.60 |
| | | 0.9 | 0.89 | 0.90 | 0.50 |
| | CASIE | 0.1 | 0.035 | 1.10 | 0.90 |
| | | 0.6 | 0.008 | 1.10 | 0.80 |
| | | 0.9 | 0.007 | 0.95 | 0.40 |
| | MLEE | 0.1 | 0.99 | 1.10 | 0.90 |
| | | 0.6 | 0.99 | 1.10 | 0.90 |
| | | 0.9 | 0.99 | 1.10 | 0.80 |
| Llama-3.1 | M2E2 | 0.1 | 0.80 | 1.00 | 0.70 |
| | | 0.6 | 0.80 | 0.90 | 0.50 |
| | | 0.9 | 0.80 | 0.85 | 0.30 |
| | CASIE | 0.1 | 0.004 | 1.00 | 0.90 |
| | | 0.6 | 0.004 | 0.95 | 0.50 |
| | | 0.9 | 0.004 | 0.90 | 0.50 |
| | MLEE | 0.1 | 1.00 | 1.00 | 0.10 |
| | | 0.6 | 1.00 | 0.95 | 0.40 |
| | | 0.9 | 0.00 | 0.80 | 0.55 |

Table 4: Trigger-level confidence thresholds

# C Reflection Mechanism

Our reflection mechanism addresses ambiguous predictions, cases of disagreement between the Self-MoA ensemble and the sequence tagger that

| Model | Dataset | Temp | $\theta_S$ | $\theta^+_{\text{SMoA}}$ | $\theta^-_{\text{SMoA}}$ |
|---|---|---|---|---|---|
| Phi-3 | M2E2 | 0.1 | 1.00 | 1.00 | 0.70 |
| | | 0.6 | 0.99 | 0.85 | 0.60 |
| | | 0.9 | 0.99 | 0.75 | 0.30 |
| | CASIE | 0.1 | 0.07 | 1.10 | 0.95 |
| | | 0.6 | 1.10 | 0.95 | 0.50 |
| | | 0.9 | 0.07 | 0.81 | 0.30 |
| | MLEE | 0.1 | 1.10 | 0.90 | 0.30 |
| | | 0.6 | 1.10 | 0.90 | 0.40 |
| | | 0.9 | 1.10 | 0.60 | 0.30 |
| Llama-3.1 | M2E2 | 0.1 | 0.99 | 0.99 | 0.50 |
| | | 0.6 | 0.99 | 1.00 | 0.90 |
| | | 0.9 | 0.90 | 0.99 | 0.50 |
| | CASIE | 0.1 | 0.05 | 1.00 | 0.70 |
| | | 0.6 | 0.03 | 0.90 | 0.60 |
| | | 0.9 | 0.04 | 0.90 | 0.60 |
| | MLEE | 0.1 | 1.00 | 0.70 | 0.50 |
| | | 0.6 | 1.00 | 0.80 | 0.50 |
| | | 0.9 | 1.00 | 0.50 | 0.10 |

Table 5: Argument-level confidence thresholds

remain unresolved after confidence-based filtering. This section details the structured reflection procedure, including prompt design, parsing strategies, and LLM configuration.

### C.1 Prompt Format

Our reflection mechanism employs carefully designed structured prompts to elicit precise responses from the LLM. Each prompt follows a format comprising:

1. **Role specification**: Defines the LLM's precise role (e.g., argument validator).
2. **Task description**: Provides explicit instructions for classifying candidates.
3. **Generation rules**: Sets strict output constraints to avoid hallucinations and ensure structured responses.
4. **Context**: Supplies the complete passage and candidate triggers or arguments for accurate contextual evaluation.
5. **Example output**: Demonstrates the required structured output format.

### C.2 LLM Configuration

For both trigger and argument reflection, we use our fine-tuned LLMs with the following settings to ensure deterministic and accurate outputs:

- **Temperature**: 0.1 (to ensure consistent, deterministic outputs)
- **Max tokens**: 4096

```
You previously identified the following
candidate triggers:

<CANDIDATE_TRIGGERS_TO_VERIFY>

Your task is to decide for each whether it truly
 signals an event trigger.

Generation Rules:
1. Classify each phrase as either 'Trigger' or '
Non-Trigger'.
2. Output strictly in the required format-no
extra text.

Output Format (strict):
- Wrap the answer in triple backticks (```)
- Write: ClassificationMap = {"phrase1": "
Trigger", "phrase2": "Non-Trigger", ...}

Example:
```ClassificationMap = {"therapy": "Trigger", "
increase dose": "Non-Trigger"}```

Passage:
<FULL_PASSAGE_TEXT>

Candidates:
<TRIGGER_CANDIDATE_LIST>

Q: For each candidate above, decide whether it
is a 'Trigger' or 'Non-Trigger'.
```

Figure 2: Structured prompt for binary trigger verification via reflection.

- **Length penalty**: 1.05 (to maintain concise, focused responses)

### C.3 Trigger Reflection

Triggers requiring reflection are presented within structured prompts (see Figure 2). The LLM classifies each candidate as `"Trigger"` or `"Non-Trigger"`. Parsed reflection results update the final trigger set by retaining only confirmed triggers for subsequent argument extraction.

### C.4 Argument Reflection

Ambiguous arguments undergo similar reflection prompts (Figure 3), explicitly linking each argument to its trigger. The LLM assigns a binary `is_correct` flag, enabling precise filtering and integration into final event representations.

## D Final Integration of Predictions

After performing agreement detection, confidence-based filtering, and reflection, we consolidate predictions into a unified output representation.
```

```
You are an argument validator.
Given a single trigger and its candidate
arguments, decide which arguments are valid.

Generation Rules:
1. An argument is valid only if the passage
supports its role for this trigger.
2. Preserve the input order-do not add, remove,
or reorder.
3. Output exactly three fields per argument: `
text`, `role`, `is_correct`.
4. Wrap the entire response in triple backticks
(```).

Passage:
"<FULL_PASSAGE_TEXT>"

Trigger:
"<TRIGGER_TEXT>" (type: "<EVENT_TYPE>")

Candidate Arguments to verify:
<CANDIDATE_ARGUMENTS_TO_VERIFY>

Q: For each candidate above, set `is_correct` to
 `true` or `false`.
```

Figure 3: Structured prompt for binary argument verification via reflection.

### D.1 Triggers

Trigger predictions fall into one of three categories:

- **Agreed triggers:** Identified by both the Self-MoA ensemble and the sequence tagger.
- **High-confidence single-source triggers:** Produced by only one model but retained due to exceeding the confidence threshold.
- **Reflected triggers:** Ambiguous cases resolved by the LLM reflection mechanism.

Each group is maintained as a separate list during processing. In the final stage, all triggers are merged to form the complete trigger set for each document.

### D.2 Arguments

Arguments are integrated per trigger, preserving the provenance of each prediction. For a given trigger, its associated arguments may come from any of the following sources:

- **Agreed arguments:** Confirmed by both systems for a shared trigger.
- **High-confidence disagreements:** Provided by one system with sufficient confidence.
- **Reflected arguments:** Verified after reflection over ambiguous trigger-argument pairs.

Throughout processing, argument predictions carry the identifier of their associated trigger, allowing us to correctly reassemble arguments under their originating triggers during the final merge. This ensures that each trigger in the final output is paired with the full set of validated and reconciled arguments, regardless of their source path in the pipeline.

## E ARIS Algorithm

The Reflective Agreement algorithm integrates predictions from a discriminative model (TagPrime) and a generative Mixture-of-Agents (Self-MoA), systematically leveraging model consensus, confidence-based filtering, and reflective inference to enhance event extraction accuracy. The ARIS Algorithm can be found in Algorithm 1.

## F Decomposed Instruction Dataset Construction

This appendix describes the construction of the Decomposed Instruction Dataset that is used for the instruction fine-tuning stage in ARIS (Section 3.1). The goal of this dataset is to teach the LLM the complete reasoning chain of event extraction through a curriculum of thirteen task variants. To equip the LLMs with a rich understanding of each event extraction subtask. We curated instruction datasets from MLEE, M2E2, and CASIE, converting each into a unified JSON schema following the TextEE split 1 configuration.

**Holistic Event-Structure Modeling** These variants require the model to generate an end-to-end representation of every event in a passage, enforcing coherence across triggers, types, and arguments:

- **Full-Structure Construction**: extract all triggers in passage order, assign each the correct event type, and list every argument with its role.
- **Role-Ablated Construction**: as above, but systematically mask exactly one argument role per instance, compelling the model to infer missing components.

**Trigger-Focused Reasoning** By isolating the foundational stage of event extraction, these variants sharpen the model's precision in identifying and classifying triggers:

- **Trigger Detection Only**: list every trigger span in passage order, without type information.

**Algorithm 1:** Reflective Agreement for Event Extraction (ARIS)

**Data:** Ordered trigger sets $E_{tagger}$ and $E_{SMoA}$, confidence threshold $\tau$, Reflection Module $R$, input text $x$
**Result:** Final event trigger set $E_{final}(x)$
$E_{agree}(x) \leftarrow \{(t,p) \mid (t,p,\_) \in E_{tagger}, (t,p,\_) \in E_{SMoA}\}$
$E_{disagree}(x) \leftarrow \{(t,p) \mid (t,p,\_) \in (E_{tagger} \cup E_{SMoA}) \setminus E_{agree}(x)\}$
$E_{high\_conf}(x), E_{ambiguous}(x) \leftarrow \emptyset, \emptyset$
**foreach** $(t,p) \in E_{disagree}(x)$ **do**
  $conf_{tagger} \leftarrow$ confidence of $(t,p)$ in $E_{tagger}$ (0 if missing)
  $conf_{SMoA} \leftarrow$ confidence of $(t,p)$ in $E_{SMoA}$ (0 if missing)
  $combined\_conf \leftarrow \frac{conf_{tagger}+conf_{SMoA}}{\text{number of models predicting } (t,p)}$
  **if** $combined\_conf \geq \tau$ **then**
    $E_{high\_conf}(x) \leftarrow E_{high\_conf}(x) \cup \{(t,p)\}$
  **else**
    $E_{ambiguous}(x) \leftarrow E_{ambiguous}(x) \cup \{(t,p)\}$
$E_{reflected}(x) \leftarrow R(E_{ambiguous}(x), x)$
$E_{final}(x) \leftarrow \text{SortByPosition}(E_{agree}(x) \cup E_{high\_conf}(x) \cup E_{reflected}(x))$
**return** $E_{final}(x)$

- **Trigger Type Classification – Single**: given one trigger, choose its event type.
- **Trigger Type Classification – Multi**: batch-classify the types of all triggers.
- **Trigger vs. Non-Trigger Discrimination**: binary classification of candidate n-grams as triggers or non-triggers, using hard negatives drawn from the local context.
- **Event Detection (joint)**: detect all triggers and assign types within a single structured output.

**Argument-Level Inference** Focusing on post-trigger reasoning, these variants train the model to extract and label arguments conditioned on known triggers:

- **Argument Extraction – Single**: list all argument spans for one specified trigger (roles omitted).
- **Argument Extraction – Multi**: for each trigger in passage order, list its arguments (roles omitted).
- **Role Assignment – Single**: given one trigger–argument pair, assign the correct role.
- **Role Assignment – Multi**: for a specified trigger, assign roles to all its candidate arguments in order.
- **Argument Extraction (Joint)**: Given all triggers, extract all associated arguments for each trigger and assign a semantic role to each.

Table 6 summarizes the number of instruction examples per variant and dataset, illustrating the scale and balance of our decomposed curriculum. Together, these thirteen variants provide a curriculum that progresses from atomic subtasks (e.g., isolated classification) to full event construction.

All prompts follow a six-part canonical structure (role → task → rules → format → example → query) and answers are serialized as fenced code blocks under a single top-level key (e.g., `EventArguments`, `Triggers`, `RoleAssignments`).

### F.1 Negative Sampling for Trigger Discrimination

To generate `trigger vs. non-trigger` examples, we sample negative n-grams that (i) occur exactly once in the passage, (ii) share no substring with any gold trigger, (iii) lie within a three-token window of any trigger, and (iv) satisfy a POS constraint (verbs, nouns, or determiners). We draw up to three negatives per document to ensure sufficient coverage of hard negatives.

### F.2 Example Instruction

Figure 4 presents a complete *Argument Extraction – Single* instruction.

## G Training and Hyperparameter Details

This section outlines the hardware setup and key hyperparameters used to train the RoBERTa sequence tagger and fine-tune the LLM-based components

| Task Variant | Casie | M2E2 | MLEE |
|---|---|---|---|
| Full-Structure Construction | 1,047 | 640 | 199 |
| Role-Ablated Construction | 930 | 606 | 194 |
| Trigger Detection Only | 1,047 | 640 | 199 |
| Trigger Type Classification (Single) | 5,181 | 736 | 1,793 |
| Trigger Type Classification (Multi) | 1,047 | 640 | 199 |
| Trigger vs. Non-Trigger Discrimination (Multi) | 931 | 526 | 193 |
| Trigger vs. Non-Trigger Discrimination (Single) | 4,184 | 1,381 | 938 |
| Event Detection (Joint) | 1,047 | 640 | 199 |
| Argument Extraction (Single Trigger) | 5,183 | 736 | 1,839 |
| Argument Extraction (Multi Triggers) | 1,047 | 640 | 199 |
| Argument Extraction (Joint) | 1,047 | 640 | 199 |
| Role Assignment (Single Argument) | 15,466 | 1,108 | 2,760 |
| Role Assignment (Multi Arguments) | 5,980 | 748 | 4,705 |
| **Total** | **44,137** | **9,681** | **13,616** |

Table 6: Number of examples per decomposed instruction variant and dataset.

```
You are an argument extractor.
Extract all arguments for the specific trigger
shown below.

Generation Rules:
1. List arguments in the exact order they appear
 in the passage.
2. Ignore argument roles and include only the
argument texts.

Output Format (strict):
- Wrap the answer in triple backticks (```).
- Write: Arguments = ["arg1", "arg2", ...].

Example:
```
Arguments = ["insulin", "VEGF"]
```

Passage:
"US Needs Broad Coalition to Fight IS Militants,
 Analysts Say-With President Barack Obama
setting a new strategy to combat Islamic State
militants (also known as ISIL or ISIS) in Iraq
and Syria, analysts say he will need to build a
broad-based coalition of international and
regional players to support those efforts"

Q: What are the arguments of the trigger "combat
" (event type: "Conflict:Attack")?
A: ```\nArguments = ["militants"]\n```
```

Figure 4: Argument Extraction – Single instruction example

in ARIS.

### G.1 Infrastructure for LLM Fine-Tuning

Experiments were conducted on a single GPU per run:

- CASIE/ MLEE: NVIDIA H200 (140GB)
- M2E2: NVIDIA A100 (80GB)

Average fine-tuning times: CASIE (8h), MLEE (3h), M2E2 (<1 h).

**Training** We fine-tune two instruction models: `microsoft/Phi-3-small-8k-instruct` and `meta-llama/Llama-3.1-8B-Instruct`, each using LoRA-based parameter-efficient adaptation. Both models are trained for 2 epochs with context length 4096, a batch size of 4. For Phi-3, we apply LoRA with $r$=32, $\alpha$=128, dropout 0.05, and target modules {`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `down_proj`, `up_proj`}. For LLaMA-3.1, LoRA is applied to `q_proj` and `v_proj` with the same rank and scaling settings.

### G.2 RoBERTa Sequence Tagger

We use `roberta-large` as the backbone encoder for all sequence tagging experiments.

**Training** Batch sizes and epochs per dataset are summarized in below table.

| Dataset | Task | Batch Size | Epochs |
|---|---|---|---|
| CASIE | ED / EAE | 16 / 4 | 10 / 90 |
| MLEE | ED / EAE | 16 / 4 | 60 / 90 |
| M2E2 | ED / EAE | 32 / 6 | 10 / 90 |

| | |
|---|---|
| **Input Text** | Moments after the revered activist was escorted through a crowd, the assassin walked towards Gandhi and, at a range of just one meter, fired his gun three times, **killing** the man who led India's historic revolt against British rule. |
| **Self–MoA Triggers** | `fired`, `killing` |
| **TagPrime Triggers** | `killing` |
| **Agreement Set** | `killing` |
| **Disagreement (high-conf.)** | – |
| **Disagreement (low-conf.)** | `fired` *(discarded)* ✓ |
| **Disagreement (need reflection)** | – |
| **Final Trigger List** | `killing` |
| **Argument Pipeline for "killing"** | |
| **Self–MoA Arguments** | assassin $\xrightarrow{Agent}$ killing; Gandhi $\xrightarrow{Victim}$ killing |
| **TagPrime Arguments** | assassin $\xrightarrow{Agent}$ killing; man $\xrightarrow{Victim}$ killing |
| ... | |
| **Final Event Representation** | assassin $\xrightarrow{Agent}$ killing; Gandhi $\xrightarrow{Victim}$ killing |
| **Gold Reference** | assassin $\xrightarrow{Agent}$ killing; Gandhi $\xrightarrow{Victim}$ killing; gun $\xrightarrow{Instrument}$ killing |

Figure 5: Illustrative walk-through of the ARIS pipeline on an M2E2 document. *Step 1:* the Self–MoA ensemble suggests two triggers (`killing`, `fired`) while the TagPrime outputs (`killing`). *Step 2:* the agreement module keeps the shared trigger `killing` and flags the disagreement `fired`. *Step 3:* confidence filtering rejects the low-confidence `fired`. *Step 4:* reflection resolves argument-level mismatches. *Outcome:* the final event representation matches gold except for the still-missing *Instrument* (gun), revealing an open error category.

## H Examples

**ARIS Refinement on M2E2 sample input** Figure 5 illustrates the three-stage ARIS pipeline: agreement detection, confidence-based filtering, and reflection-based resolution.