Multi-Document Event Extraction Using Large and Small Language Models

Qingkai Min[1,2], **Zitian Qu**[3], **Qipeng Guo**[4], **Xiangkun Hu**[5], **Zheng Zhang**[6], **and Yue Zhang**[2*]
[1] Zhejiang University [2] School of Engineering, Westlake University
[3] Tsinghua University [4] Shanghai AI Lab [5] Fudan University [6] NYU Shanghai
{minqingkai, zhangyue}@westlake.edu.cn

Abstract

Multi-document event extraction aims to aggregate event information from diverse sources for a comprehensive understanding of complex events. Despite its practical significance, this task has received limited attention in existing research. The inherent challenges include handling complex reasoning over long contexts and intricate event structures. In this paper, we propose a novel collaborative framework that integrates large language models for multi-step reasoning and fine-tuned small language models to handle key subtasks, guiding the overall reasoning process. We introduce a new benchmark for multi-document event extraction and propose an evaluation metric designed for comprehensive assessment of multiple aggregated events. Experimental results demonstrate that our approach significantly outperforms existing methods, providing new insights into collaborative reasoning to tackle the complexities of multi-document event extraction.[1]

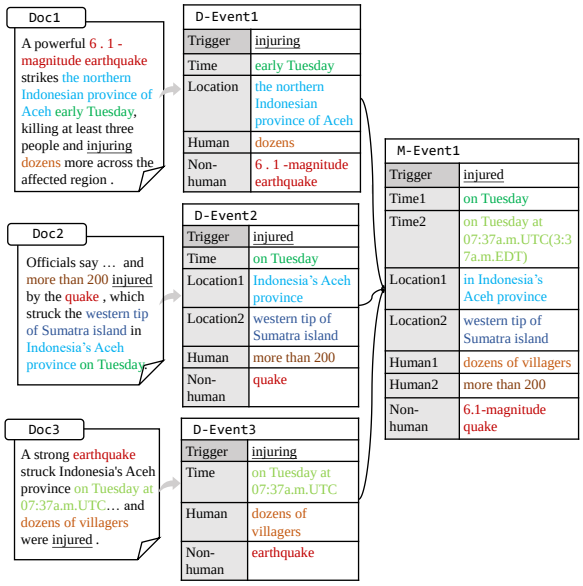

Figure 1: An example of multi-document event extraction, showing how injury-related information is aggregated from multiple sources, with overlapping arguments highlighted in the same color.

1 Introduction

Event extraction refers to the process of identifying and structuring event-related information from unstructured text. This involves detecting specific triggers, participants, and associated temporal and spatial attributes, thereby converting free-form text into structured data (Li et al., 2013). Event extraction supports downstream tasks like summarization, knowledge base construction, and question answering. The task has evolved from focusing on events within individual sentences to handling events at the document level (Yang et al., 2018; Xu et al., 2021; Yang et al., 2021). This shift breaks sentence boundaries to collect information across multiple parts of a document, enabling more comprehensive and coherent event representations.

In real-world scenarios, users typically acquire information from multiple documents originating from different sources. This multi-document perspective is critical because different documents may provide unique, yet essential, details that complement each other to offer a more complete picture of an event. As shown in Figure 1, the three documents discuss the same earthquake, with Document 3 providing a more specific timestamp—"*07:37 a.m. UTC*"—compared to the other two. Meanwhile, Document 2 provides the exact number of injured individuals, stating "*more than 200*". These varying details demonstrate how information from different sources can come together to offer a more comprehensive understanding of the event.

Despite its practical significance, multi-document event extraction remains an underexplored area of research (Ji and Grishman, 2008;

*Corresponding author
[1]The code and dataset are publicly available at https://github.com/taolusi/MEET.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 19266–19297
November 4-9, 2025

Ji et al., 2009; Gao et al., 2024). Compared to document-level event extraction, this task presents considerable challenges, primarily due to the longer input contexts, increased inconsistencies in expression, terminology, and detail introduced by multiple documents, as well as the higher volume of events, which complicates the coreference aggregation process. At the core of these challenges is the high demand placed on models' reasoning capabilities, as they must integrate and align information across a large, complex space.

We tackle the challenges of multi-document event extraction by introducing ECB++, a new benchmark dataset built on ECB+ corpus (Cybulska and Vossen, 2014), which provides separate annotations for cross-document entity and event coreference. To facilitate the consolidation of event-specific information, ECB++ augments the dataset with fine-grained, document-level argument annotations. Notably, for events mentioned across multiple sources in ECB++, arguments extracted from a single document account for only 53.4% of the total on average. To address the new evaluation challenge of matching multiple events, we propose CEAF-MEE, a novel metric that extends existing document-level event extraction metrics, which typically focus on single-event evaluation. It aligns predicted and reference event structures at both the event and argument levels through maximum bipartite matching.

Multi-document event extraction challenges for both small language model (SLM) supervised fine-tuning pipeline[2] and large language model (LLM) sequence-to-sequence generation. SLM pipeline suffers from error propagation across subtasks; for example, document-level argument extraction has an F1 score below 70%, and cross-document event coreference resolution remains below 80%. Sequence-to-sequence LLMs, on the other hand, often lack explicit rationales, which affects both performance and interpretability (Wei et al., 2022). This is further complicated by the nature of our task, which requires reasoning across potentially hundreds of events spread across documents, resulting in long and intricate chains of aggregation that cannot be easily captured in a single sequence. We provide illustrative examples in Appendix A.

Recent work has increasingly explored LLMs for information extraction tasks, often combining their general abilities with the task-specific adaptability of SLMs (Wan et al., 2023; Ma et al., 2023; Chen et al., 2024; Min et al., 2024; Zhu et al., 2024). Following this line, we propose a collaborative framework that leverages the complementary strengths of LLMs and SLMs: a pipeline of fine-tuned SLMs handle core subtasks and provide structured intermediate outputs, while LLM agents perform multi-step reasoning with a holistic understanding of these outputs and the global context.

Specifically, our framework consists of two primary reasoning stages. In the *event composition* stage, the system creates cross-document clusters by aligning events and their associated arguments. This stage is formulated as a heuristic search over a large combinatorial space, where an LLM agent incrementally performs the composition by integrating SLM-generated structures with a broader contextual understanding. In the subsequent *event consolidation* stage, a separate LLM agent iteratively conducts cluster-level reasoning to consolidate each cluster into a single, coherent event representation—selecting a representative trigger and resolving redundant, or irrelevant arguments.

Experimental results show that our collaborative framework significantly outperforms both pipeline-based methods and LLM sequence-to-sequence approaches. Analysis reveals that our model excels at resolving both event coreference and argument consolidation, highlighting its effectiveness in tackling the complex reasoning demands of the task. The relatively modest scores underscore the inherent complexity and challenges of the task.

To our knowledge, this is the first work to integrate the large and small language models for multi-step reasoning in event extraction. We hope our framework and benchmark will support further progress in information extraction research.

2 Task and Benchmark

2.1 Terminology and Task Definition

Key Event Concepts

Event Trigger: A word or phrase in the text that signals the occurrence of an event, e.g., *injuring* and *injured* in Figure 1.

Event Argument: An entity or event fulfilling a specific semantic role, such as time, location, or participants (human or non-human). To support nested structures, events can also serve as arguments of other events—for example, in Figure 1, *quake* is the non-human agent of the *injured* event.

[2]In this work, SLM refers to smaller pre-trained language models, like BERT and RoBERTa, which are more cost-effective for fine-tuning on specific tasks.

Document-Level Event: A structured event table from a single document, consisting of a trigger and several arguments. This is illustrated by `D-Event1`, `D-Event2`, and `D-Event3` in Figure 1.

Multi-Document Event: A structured table combining event data from multiple documents about the same occurrence, unifying diverse triggers and overlapping arguments into a canonical form. This is illustrated by `M-Event1` in Figure 1.

Task Definition

Input: A set of documents $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$, possibly thematically related.

Output: A set of multi-document events $\hat{\mathcal{E}} = \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_m\}$, where each event $\hat{e}_i = (\hat{t}_i, \hat{A}_i)$ consists of a canonical trigger $\hat{t}_i$ and a set of arguments $\hat{A}_i = \{\hat{a}_{i,1}, \hat{a}_{i,2}, \ldots, \hat{a}_{i,k_i}\}$.

2.2 Benchmark Dataset ECB++

Dataset Creation We build our dataset upon the Event Coreference Bank Plus (ECB+) (Cybulska and Vossen, 2014), a widely-used public benchmark for cross-document event and entity coreference resolution. ECB+ provides event triggers, entity mentions, and both event and entity coreference chains. To enable event argument consolidation, we extend ECB+ by annotating event arguments at document level, resulting in structured events. By integrating these document-level events with the original cross-document event and entity coreference chains, we create **ECB++**, a comprehensive multi-document event representation.

We adopt FrameNet (Baker et al., 1998) as the guiding schema for annotating event arguments. FrameNet is a lexical resource that defines semantic roles for participants in various event types, ensuring both consistent and semantically precise annotation of event arguments. Following Li et al. (2021), the annotation was first performed by three linguistics students and subsequently reviewed by an instructor to ensure high-quality output. The Inter-Annotator Agreement (IAA) reached 83.4%, confirming the quality of the dataset. The annotation guidelines can be found in Appendix B.1.

Dataset Analysis The ECB++ dataset consists of 982 news articles spanning 43 topics, such as earthquakes, criminal cases, and corporate acquisitions. Each topic contains 23–25 documents, with an average of 60 events per topic. Notably, one-quarter of the events are mentioned in at least three documents, highlighting the need for both event aggregation and the ability to handle event-dense scenarios.[3] Each topic is divided into two subtopics, each focusing on a distinct yet related seminal event. This design allows documents from both subtopics to be presented together as input, facilitating event aggregation across documents in realistic and complex conditions. More detailed statistics are shown in Table 1.

Dataset Split	Train	Dev	Test
# Topics	25	8	10
Avg. # Docs per Topic	23.0	24.5	20.6
Avg. # Events per Topic (M/S)	16.44/44.6	16.1/35.0	18.2/62.3
Avg. # Docs per Event (M/S)	3.7/1	4.6/1	3.4/1
Avg. # Args per Event (M/S)	4.2/2.4	4.0/2.2	3.7/2.1

Table 1: Dataset statistics. "M/S" denotes values for events mentioned in multiple vs. single documents.

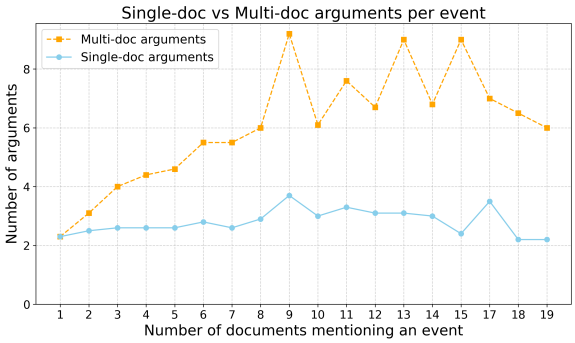

Figure 2: Distribution of single-document arguments (annotated) and multi-document arguments (aggregated) per event by number of documents mentioning the event.

We further examine the importance of event consolidation by comparing the number of arguments in single-document settings with those in multi-document settings, as shown in Figure 2. As more documents mention the same event, the proportion of arguments derived from multi-document aggregation increases, emphasizing the need for event aggregation. We also conduct a quantitative analysis of argument distribution across documents, revealing considerable redundancy in events mentioned in multiple documents. While 1,230 arguments (≈40%) appear in only a single document, the majority—1,866 arguments (≈60%)—are mentioned in two or more documents (e.g., 632 in 2 docs, 350 in 3, 222 in 4, etc.), further emphasizing the need for advanced aggregation techniques, rather than simple merging. Detailed statistics are in provided Table 6 in Appendix B.2.

[3] Events mentioned in a single document are more frequent than in multiple documents due to ECB+'s original annotations. This pattern reflects the inherent nature of news reporting, where each article typically contains unique content.

2.3 Evaluation Metric

Multi-document event extraction poses a unique challenge: unlike document-level tasks, which typically evaluate a single event, our task requires matching multiple events. We propose **CEAF-MEE**, a metric specifically designed for multi-event evaluation. CEAF-MEE is inspired by CEAF-REE and its variants (Du et al., 2021; Chen et al., 2023), developed to evaluate argument alignment within individual events.

CEAF-MEE tackles the multi-event matching problem using a two-level bipartite matching. At the first level, predicted events are aligned with reference events. For predicted events $p \in \mathcal{P}$ and reference events $r \in \mathcal{R}$, CEAF-MEE finds a one-to-one alignment g^* that maximizes the sum of similarity scores $\phi(p, r)$ across all matched event pairs. The similarity function ϕ measures the match quality of an event pair by averaging trigger and argument scores. The trigger score quantifies the span overlap of the two event triggers. The argument score is computed via a second-level bipartite matching over multiple arguments, where the edge weights correspond to the span overlap between predicted and reference arguments. The final argument score is obtained by averaging the scores of matched argument pairs.

Given the optimal alignment g^*, precision and recall are computed as:

$$\text{Precision} = \frac{\sum_{(p,r)\in g^*} \phi(p,r)}{\sum_{p\in\mathcal{P}} \phi(p,p)},$$

$$\text{Recall} = \frac{\sum_{(p,r)\in g^*} \phi(p,r)}{\sum_{r\in\mathcal{R}} \phi(r,r)}.$$

3 Method

We first introduce the key sub-tasks and their pipeline organization (Section 3.1), followed by the SLM- and LLM-based implementations (Section 3.2). We then explore LLM sequence-to-sequence approaches (Section 3.3), and finally present a collaborative framework integrating LLM reasoning with SLM specialization (Section 3.4).

3.1 Task Decomposition and Pipeline Design

We design a modular pipeline that builds upon the standard event extraction task and extends it to better suit the multi-document setting.

1. *Event Trigger Detection*
Input: A document $d_i \in \mathcal{D}$.
Output: A set of triggers $T_i = \{t_{i,1}, t_{i,2}, \ldots, t_{i,p_i}\}$, each $t_{i,j}$ is a text span.

2. *Cross-Document Event Coreference Resolution*
Input: The complete set of event triggers $T = \{t_1, t_2, \ldots, t_p\}$ extracted from all documents.
Output: A set of event clusters $C_T = \{C_{T,1}, C_{T,2}, \ldots, C_{T,m}\}$, each cluster $C_{T,k}$ contains triggers referring to the same real-world event.

3. *Document-Level Event Argument Extraction*
Input: A document $d_i \in \mathcal{D}$ and its corresponding triggers $T_i = \{t_{i,1}, \ldots, t_{i,p_i}\}$.
Output: A set of document-level events $E_{d_i} = \{e_{i,1}, \ldots, e_{i,p_i}\}$, where each event $e_{i,j} = (t_{i,j}, A_{i,j})$ consists of a trigger $t_{i,j} \in T_{d_i}$ and its associated arguments $A_{i,j} = \{a_{i,j,1}, a_{i,j,2}, \ldots\}$.

4. *Cross-Document Argument Coreference Resolution*
Input: The full set of event arguments $A = \bigcup_{i,j} A_{i,j}$ extracted from all documents in $\mathcal{D}$.
Output: A set of argument clusters $C_A = \{C_1^A, C_2^A, \ldots, C_l^A\}$, each cluster C_k^A contains arguments referring to the same real-world referent.

5. *Event Canonicalization*
Input: A set of event clusters C_T and a set of argument clusters C_A.
Output: A set of multi-document events $\hat{\mathcal{E}} = \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_m\}$, where each event $\hat{e}_i = (\hat{t}_i, \hat{A}_i)$ consists of a canonical trigger $\hat{t}_i$ representing cluster $C_{T,i}$, and a set of canonical arguments $\hat{A}_i$ aggregated from the relevant argument clusters in C_A.

The overall structure can be summarized as:[4]

$$\mathcal{D} \rightarrow T_d \rightarrow \begin{cases} C_T \\ E_d = (t, A) \rightarrow C_A \end{cases} \rightarrow \hat{\mathcal{E}}$$

3.2 Pipeline Approaches

SLM SFT-based Pipeline We build a modular pipeline using state-of-the-art SLMs, each fine-tuned for a specific subtask. Document-level modules operate on individual documents, while cross-document modules process concatenated document pairs. Event canonicalization, which requires integrating information across multiple documents and exceeds SLM input limits, is approximated with a frequency-based heuristic that selects the most common surface form for triggers and arguments. Model architectures are detailed in Appendix C.1.

[4]The presented pipeline is one reasonable task flow, though alternative orders may be valid.

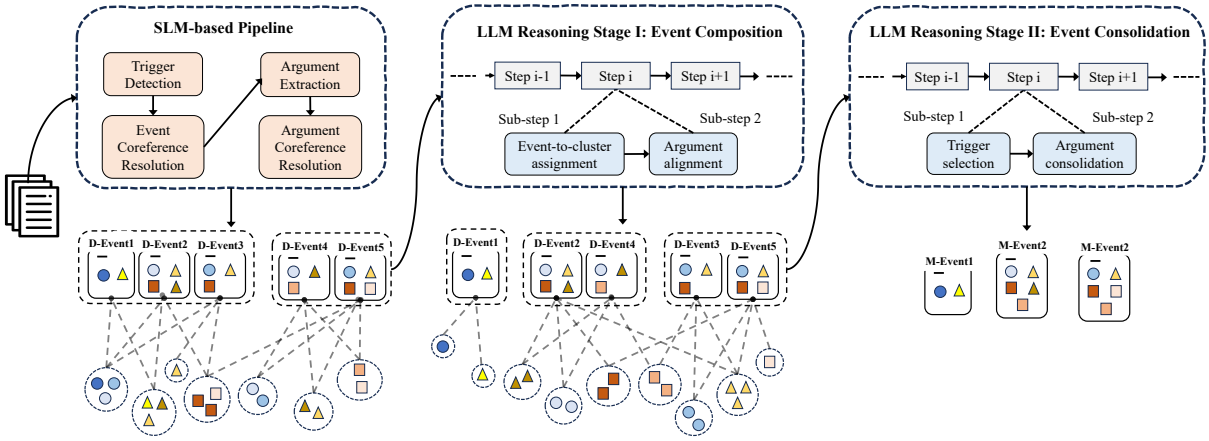

Figure 3: Our collaborative framework. D-Event denotes document-level events, and M-Event denotes aggregated multi-document events. In each event table, – represents the trigger; other icons represent arguments. Icons with the same shape indicate the same role, while those with the same color refer to the same underlying referent.

LLM ICL-based Pipeline In parallel, we implement a pipeline variant that replaces fine-tuned SLM modules with large language models driven via in-context learning. Each subtask is handled by a dedicated prompt, avoiding the substantial computational cost that would be required to fine-tune the large models. Prompting details are provided in Appendix C.2.

3.3 LLM Sequence-to-Sequence Generation

Unlike the modular pipeline above, this approach treats multi-document event extraction as a direct sequence-to-sequence generation task. It bypasses intermediate steps such as trigger detection, argument extraction, and coreference resolution, relying on the LLM to produce final event outputs directly. We explore two variants: **in-context learning (ICL)**, where the model is prompted with instructions and a few examples, and **supervised fine-tuning (SFT)**, where the model is trained with instructions and labeled data. Both variants generate canonicalized events without requiring explicit intermediate processing. Prompting details are provided in Appendix C.3.

3.4 Collaborative Framework with LLM Reasoning

Our framework, as illustrated in Figure 3, starts with a fine-tuned SLM-based pipeline that generates initial event clusters—each corresponding to multiple document-level events—and their associated argument clusters. This is followed by two primary LLM-driven reasoning stages: event composition and event consolidation.

3.4.1 Reasoning Stage I: Event Composition

In the first reasoning stage, an LLM agent conducts a multi-step heuristic search over possible event compositions, jointly clustering document-level events and aligning their arguments. Guided by the initial clusters from the SLM pipeline and the global context, the agent iterates over all document-level events $e \in \bigcup_{d \in \mathcal{D}} E_d$, assigning each event to an event cluster C'_T and simultaneously iterating over its associated arguments to update the corresponding argument clusters C'_A.

Event-to-cluster assignment For each event e, the LLM evaluates its compatibility with each candidate event cluster $C'_i \in C'_T$, where $C'_i \subseteq \bigcup_{d \in \mathcal{D}} E_d$ contains multiple document-level events. The LLM assigns a discrete relevance score $s_i \in \{1, 2, 3, 4, 5\}$ to reflect contextual alignment, trigger synonymy, and argument overlap. This decision is framed as a comparative ranking task, where the LLM integrates global context with structural cues from the SLM predictions. If the highest score exceeds a threshold θ, e is merged into the corresponding cluster C'_{i^*}; otherwise, a new cluster is initialized for e.

To improve LLM efficiency, we adopt two strategies to reduce the candidate cluster search space. First, in **core event–based cluster initialization**, core events are selected as seeds based on their connectivity within the SLM-predicted cluster, applied to clusters with at least three document-level events. Second, in **trigger-based cluster grouping**, event clusters are organized into coarse buckets based on trigger synonymy, identified by the LLM agent. The LLM then restricts its reasoning to the relevant

bucket, avoiding unnecessary comparisons with unrelated clusters.

Argument alignment Once an event e is assigned to an event cluster C'_{i^*}, each of its arguments $a_j^k \in \mathcal{A}_e$ is evaluated for alignment with each candidate argument cluster $C_j^{A'} \in C'_{A,i^*}$, where $C'_{A,i^*} \subseteq C'_A$ denotes the set of argument clusters associated with C'_{i^*}. If a suitable argument cluster is identified, the argument is merged into it; otherwise, a new argument cluster is created. The LLM considers both role-level semantic compatibility and coreference cues derived from the initial SLM predictions. Crucially, the LLM is encouraged to override SLM predictions when contextually justified—for example, aligning arguments with different SLM-assigned roles but semantically equivalent referents, or rejecting SLM-suggested links that conflict with the broader discourse context.

Through iterative reasoning, both the event cluster structure C'_T and argument clusters C'_A gradually evolve toward greater internal coherence.

3.5 Reasoning Stage II: Event Consolidation

In the second reasoning stage, a separate LLM agent builds upon the intermediate event and argument clusters C'_T and C'_A from the previous stage to iteratively perform cluster-level reasoning. The agent consolidates these clusters into a set of canonical events $\hat{\mathcal{E}} = \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_m\}$. Each reasoning step comprises two subtasks:

Trigger selection The LLM agent selects a canonical trigger $\hat{t}_i$ from the set of candidate triggers $t_{i,1}, t_{i,2}, \ldots, t_{i,m}$, which are extracted from the constituent document-level events $e_{i,1}, e_{i,2}, \ldots, e_{i,m} \subseteq C'_i$.

Argument consolidation The LLM agent jointly assesses all argument clusters $C'_{A,i} \subseteq C'_A$ linked to C'_i. It first evaluates the relevance of each argument cluster to the event, filtering out those that lack semantic coherence. Subsequently, the LLM iteratively selects a canonical argument from each retained cluster, which together compose the final argument set $\hat{A}_i$. The relevance evaluation primarily focuses on two types of errors propagated from earlier stages: argument extraction errors, which involve filtering out arguments that are semantically or factually inconsistent with the event's core meaning; and argument coreference errors, which require removing redundant arguments that refer to the same underlying referent.

This process jointly analyzes all argument clusters, leveraging the LLM's ability to generalize from the majority of trustworthy arguments while revising outliers, ensuring semantic consistency and eliminating redundancy without relying on its internal standards.

All prompting details are in Appendix C.4. To further clarify the components of our method, we provide a running example in Appendix C.5.

4 Experiments

4.1 Experimental Setup

Model Details All LLM ICL experiments are conducted using the DeepSeek-V3 model (DeepSeek-AI, 2024) with the temperature set to 0 to reduce randomness. Each ICL experiment is run three times, and we report the average results. LLM SFT experiments are conducted with the Qwen-3-14B model (Qwen-Team, 2025). For SLMs in the pipeline, we use the default configurations from the original papers, with details in Appendix C.1.

Evaluation Metric For CEAF-MEE, we evaluate trigger and argument matches using both exact and head match. Exact match requires full span overlap, while head match follows Li et al. (2021), allowing partial matches by comparing the head words of predicted and gold spans. If the head words align, the prediction is considered correct. We extract head words using the spaCy NLP toolkit (Honnibal et al., 2020) with the `en_core_web_sm` model.

4.2 Main Results

The overall performance comparison is shown in Table 2. Our collaborative method significantly outperforms all baselines in both exact and head match metrics, demonstrating its effectiveness for the task. Nevertheless, the relatively low absolute scores underscore the inherent difficulty of the task. Specifically, we observe the following:

First, when comparing the pipeline and seq2seq approaches, we observe that, regardless of whether SFT or ICL is employed, the pipeline consistently outperforms seq2seq. In terms of exact match F1, the pipeline achieves 10% higher performance with SFT and 4.3% higher with ICL. This difference is primarily driven by recall, with the SFT pipeline achieving 14.8% higher recall and the ICL pipeline achieving 8.2% higher recall. These results suggest that the pipeline method we designed is better suited to capture the full range of requirements

Method		Exact match			Head match		
		P	R	F1	P	R	F1
Pipeline	SLM SFT	31.4	45.0	35.6	36.6	52.5	41.6
	LLM ICL	14.2	13.0	12.5	23.1	21.0	20.2
Seq2seq	LLM SFT	27.8	30.2	26.6	34.0	36.9	32.7
	LLM ICL	32.9	4.8	8.2	42.5	6.0	10.2
Our collaborative		**37.7**	**48.4**	**41.2**	**43.6**	**55.8**	**47.6**

Table 2: Multi-document event extraction results on ECB++. Our Collaborative method outperforms the baselines with statistically significant gains in both exact and head match F1 (paired t-test, $p < 0.01$).

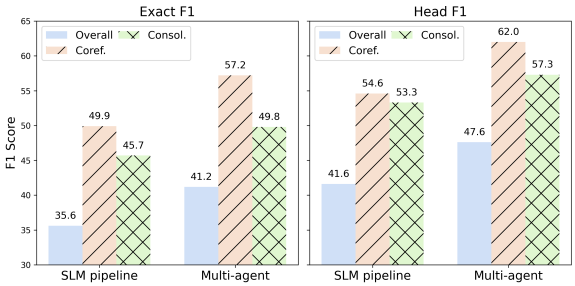

Figure 4: Disentangled evaluation. "Overall" denotes the original evaluation, "Coref." measures event coreference, and "Consol." assesses argument consolidation.

for this complex task, while seq2seq-based methods face significant challenges when generating directly. These trends hold true for both exact match and head match metrics.

Next, when comparing SFT with ICL, SFT clearly outperforms ICL, with a 23.1% higher exact match F1 in the pipeline setting and an 18.4% improvement in seq2seq. This improvement is primarily due to the significantly higher recall scores achieved by SFT (by 20–30%), highlighting the importance of supervised learning, which enables the model to better accommodate the task's comprehensive requirements. In contrast, the instruction-following and example-driven nature of ICL is insufficient to meet the task's complex and nuanced requirements. These trends remain consistent for both exact match and head match metrics.

Additionally, SFT achieves 17.2% higher precision in the pipeline setting compared to ICL, whereas in the seq2seq case, ICL outperforms SFT in precision. This further suggests that seq2seq SFT methods face significant difficulties in handling the task's complexity, which can potentially undermine the effectiveness of the underlying LLM.

Finally, our collaborative method demonstrates improvement over both the pipeline and seq2seq approaches. Compared to the SLM pipeline method, our approach boosts exact match F1 by 5.6% and head match F1 by 6%. Notably, recall improves by 3.5% over the SLM pipeline, suggesting that our collaborative framework enhances the initial SLM outputs by learning from these predictions and incorporating more comprehensive global reasoning. Moreover, precision increases by 3.8% over LLM ICL, highlighting that our multi-step effectively leverages the LLM's capabilities while preserving its inherent strengths. This allows our method to improve the SLM predictions without diminishing the LLM's core performance.

4.3 Disentangled Analysis

Multi-document event extraction involves both linking coreferential events and consolidating information across events, particularly focusing on arguments. To disentangle these components and assess them independently, we perform an evaluation-stage decomposition. In the standard evaluation, predicted events are aligned with gold events using both triggers and arguments as matching criteria. Under this decomposition, event coreference resolution is evaluated by removing all arguments and considering only triggers. Event argument consolidation is assessed on matched event pairs—excluding unmatched events that would otherwise receive a zero score—thereby isolating the quality of argument integration while minimizing the influence of coreference errors.

As shown in Figure 4, the overall score is substantially lower than the separate coreference and consolidation scores, highlighting the inherent complexity of the task. The consolidation score remains lower than the coreference score, emphasizing the difficulty of integrating argument information across multiple documents. Our collaborative approach outperforms the fine-tuned SLM pipeline by 8% F1 points on coreference evaluation and 4% F1 points on consolidation evaluation (under both exact and head match settings), demonstrating its effectiveness in addressing the key challenges of multi-document event extraction.

4.4 Ablation Study

To better understand the contribution of each component, we start with the complete reasoning process and progressively ablate key modules. To enable more fine-grained analysis, we treat the substeps of event composition—event assignment and argument alignment—as stage 1.1 and stage 1.2,

Model Variant	Exact F1	Head F1
SLM pipeline	35.6	41.6
Full collaborative model	41.2	47.6
- stage 1.1	36.5	42.6
- stage 1.2 & stage 2	39.3	45.2
- stage 2	39.9	46.0
- stage 1.2	41.0	47.1

Table 3: Step-by-step ablation of reasoning modules in the collaborative framework.

respectively, and ablate them individually.

From Table 3, we observe that event assignment (stage 1.1) is of primary importance, as its removal leads to the most drastic performance decline, resulting in a 5% drop in the head F1 score. This step not only corrects event coreference but also lays the foundation for subsequent event consolidation. Next, argument alignment across events (stage 1.2) and cluster-level argument consolidation (stage 2) play crucial roles in final event canonicalization, with their removal causing a 2.6% decrease overall. More specifically, removing stage 1.2 alone leads to a 1.6% drop, while removing stage 2 alone results in a 0.5% decrease. These results indicate that the two components contribute complementarily, each having a positive impact on the overall performance.

A detailed case study illustrating the impact of these components is provided in Appendix D.

4.5 Extended Analysis of LLM Capabilities

LLM Self-Consistency We adopt a self-consistency strategy within our method, inspired by Yao et al. (2023), sampling each reasoning step three times, and using a dedicated LLM agent to select the most plausible output via a voting prompt. As shown in Table 4, unlike prior studies where self-consistency yields substantial gains, our approach yields only marginal improvements. We attribute this to two main factors. First, our carefully designed prompts help reduce sampling variance, producing relatively stable outputs. Second, a relatively simple voting prompt design—such as "analyze the choices below, conclude which better follows the instruction"—lacks clear evaluative criteria in our task, making it less effective at consistently resolving subtle differences between outputs.

Chat Model vs. Reasoning Model We compare two LLM backbones, DeepSeek-V3 (chat) and DeepSeek-R1 (reasoning) (DeepSeek-AI, 2025),

Method	Exact match			Head match		
	P	R	F1	P	R	F1
Collaborative	37.7	**48.4**	41.2	**43.6**	55.8	47.6
+ self-consistency	**38.1**	48.3	**41.3**	**43.6**	**55.9**	**47.7**
+ reasoning-backbone	36.0	47.2	39.6	42.0	55.2	46.3

Table 4: Effect of LLM self-consistency and reasoning backbone on model performance.

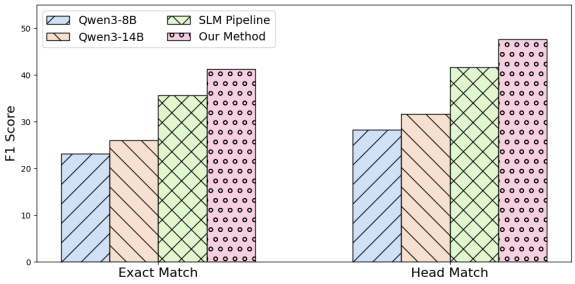

Figure 5: Impact of LLM model size on seq2seq SFT performance.

on our method. As shown in Table 4, R1 performs consistently worse. This may be attributed to a misalignment between the model's internal reasoning and our task-specific logic. A basic observation is that, even when explicitly instructed to return "none" if no suitable cluster is found, R1 frequently returns a similar cluster instead.

Impact of Model Size on LLM seq2seq SFT We evaluate different Qwen3 LLMs (8B and 14B) as backbones for the seq2seq SFT approach, with results illustrated in Figure 5. The performance gap between Qwen3-8B and Qwen3-14B is relatively small, especially when compared to the gap with the SLM pipeline and our method. This suggests that simply increasing model size may be insufficient to address the structural and contextual complexity of the task. Due to GPU memory limitations, we were unable to fine-tune larger models such as Qwen3-32B.

4.6 Computational Efficiency

We assess the efficiency of different methods in terms of computational resources and time. The SLM-based pipeline operates efficiently on a single GPU (e.g., NVIDIA V100), with each module training in a few hours and the entire process completing within a dozen hours.

Our collaborative method, which augments the SLM pipeline with LLMs for multi-step reasoning, is divided into two stages. The first stage requires a large search space, but we optimize it as detailed in

our method, significantly reducing both time and computational cost. In total, the combined time for both stages is under 12 hours, depending on the performance of the DeepSeek API.

For other LLM-based methods, the LLM seq2seq SFT method, training a 14B model on 8 NVIDIA 80GB A100 GPUs, takes several hours. The LLM seq2seq ICL and LLM pipeline ICL methods, both utilizing the DeepSeek API, require a few hours and about a dozen hours, respectively.

5 Related Work

Event Extraction In recent years, deep learning has significantly advanced the field of event extraction. Most existing approaches rely on fine-tuning pretrained language models such as BERT or RoBERTa to model event-related semantics and dependencies within text (Wadden et al., 2019; Yang et al., 2019; Li et al., 2020; Sheng et al., 2021). Some methods further incorporate components like graph neural networks to better model the interactions among events and their arguments (Cao et al., 2021; Sun et al., 2022). Nevertheless, event extraction remains a challenging task due to the complexity of its underlying schema—requiring accurate identification of event types, triggers, and multiple argument roles within intricate contexts. For example, document-level role-filler extraction on MUC-4 yields only around 50% F1 (Du and Cardie, 2020), while argument extraction on WikiEvents barely exceeds 60% (Lin et al., 2025).

Combination of LLM and SLM for IE Large language models have shown strong capabilities in understanding and reasoning across a wide range of NLP tasks—e.g., translation, summarization, and even math problem solving—often with just a few examples (Wang et al., 2023; Zhang et al., 2023; Xu et al., 2024; Laban et al., 2023). However, LLM-based in-context learning still generally underperforms compared to supervised SLMs such as RoBERTa in the domain of information extraction (Han et al., 2023; Li et al., 2023a; Huang et al., 2024). This gap largely stems from the inherent complexity of IE tasks, whose definitions often vary across different settings and datasets, making it difficult to fully specify the task using only a natural language instruction and a few examples (Peng et al., 2023). To address this limitation, recent work has explored combining LLMs with SLMs to leverage their complementary strengths. Specifically, LLMs offer broad generalization capabilities and strong contextual understanding across diverse tasks, while SLMs provide better efficiency and can be fine-tuned for high accuracy on specific tasks. By integrating these advantages, such hybrid approaches aim to achieve more robust and adaptable performance (Wan et al., 2023; Ma et al., 2023; Xu et al., 2023; Li et al., 2023b; Min et al., 2024; Chen et al., 2024; Nath et al., 2024; Ding et al., 2024; Zhu et al., 2024; Ye et al., 2024).

Cross-document Event Reasoning Early work has explored extending the scope of event beyond single documents. Ji and Grishman (2008) leverages clusters of topically related documents to enhance ACE event extraction by combining global cross-document evidence with local predictions. Ji et al. (2009) further introduces the task of cross-document event extraction and tracking, linking events involving the same centroid entity along a timeline. While exploring cross-document reasoning, they do not fully address event-level aggregation—consolidating partial event information across sources into coherent representations.

Recently, Gao et al. (2024) introduced a Chinese dataset, CLES, for cross-document event extraction. While it shares the high-level objective of event aggregation with our work, the two differ significantly in task formulation and dataset characteristics. Specifically, they merge related events into broad conceptual units, abstracting from trigger-level details, whereas we maintain fine-grained trigger annotations to model sub-events like "*authorized*", "*launched*", and "*killed*" within an airstrike. Additionally, CLES is constructed from Chinese Wikipedia and leverages entity linking to a knowledge base for argument aggregation, whereas our ECB++ dataset is based on English news articles and uses coreference resolution to capture non-canonical mentions such as "*the child*" or "*she*".

6 Conclusion

We investigate the task of multi-document event extraction, focusing on the key challenge of event aggregation. To support this, we establish a new benchmark and design a tailored evaluation metric. To address the complexity of event alignment and consolidation, we propose a collaborative framework: SLM components manage key local subtasks, while LLM components perform global reasoning. The multi-step reasoning process explicitly generates rationales by integrating outputs from the SLMs and leveraging long-context understanding.

Limitations

While our framework yields notable gains over existing baselines, it still depends considerably on the performance of fine-tuned SLMs for handling local subtasks. Errors from SLM agents can propagate to later stages, limiting the overall effectiveness of the system. Moreover, the current setup enforces a relatively fixed division of roles: SLMs are responsible for local decisions, while LLMs perform global reasoning. This separation may limit the adaptability of the system. Future work could explore more dynamic interaction between LLMs and SLMs—not only allowing LLMs to intervene in local decisions when needed, but also using LLM rationales and predictions to guide and refine the fine-tuning of SLMs.

Additionally, due to GPU memory limitations, we were unable to fine-tune larger models such as Qwen3-32B. Fine-tuning LLMs is an important research direction. A potential avenue for improvement lies in fine-tuning decomposed intermediate steps rather than fine-tuning sequence-to-sequence generation. This approach could help alleviate the resource challenges associated with training LLMs, while still benefiting from the refined reasoning at each step.

Ethical Considerations

We adhere to the ACL Code of Ethics and ensure that no private or non-public data is used in this work. Our research relies on the widely-used, publicly available dataset ECB+ (Cybulska and Vossen, 2014) and focuses on objectively extracting event information.

Acknowledgments

We would like to thank anonymous reviewers for their insightful comments to help improve the paper. This work has been supported by the National Natural Science Foundation of China (NSFC) Key Project under Grant Number 62336006 and the National Natural Science Foundation of China (NSFC) Key Project under Grant Number 62161160339.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal, Quebec, Canada. Association for Computational Linguistics.

Yuwei Cao, Hao Peng, Jia Wu, Yingtong Dou, Jianxin Li, and Philip S. Yu. 2021. Knowledge-preserving incremental social event detection via heterogeneous gnns. In *Proceedings of the Web Conference 2021*, WWW '21, page 3383–3395, New York, NY, USA. Association for Computing Machinery.

Arie Cattan, Alon Eirew, Gabriel Stanovsky, Mandar Joshi, and Ido Dagan. 2021. Cross-document coreference resolution over predicted mentions. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5100–5107, Online. Association for Computational Linguistics.

Wei Chen, Lili Zhao, Zhi Zheng, Tong Xu, Yang Wang, and Enhong Chen. 2024. Double-checker: Large language model as a checker for few-shot named entity recognition. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3172–3181, Miami, Florida, USA. Association for Computational Linguistics.

Yunmo Chen, William Gantt, Weiwei Gu, Tongfei Chen, Aaron White, and Benjamin Van Durme. 2023. Iterative document-level information extraction via imitation learning. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1858–1874, Dubrovnik, Croatia. Association for Computational Linguistics.

Agata Cybulska and Piek Vossen. 2014. Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4545–4552, Reykjavik, Iceland. European Language Resources Association (ELRA).

DeepSeek-AI. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bowen Ding, Qingkai Min, Shengkun Ma, Yingjie Li, Linyi Yang, and Yue Zhang. 2024. A rationale-centric counterfactual data augmentation method for cross-document event coreference resolution. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1112–1140, Mexico City, Mexico. Association for Computational Linguistics.

Xinya Du and Claire Cardie. 2020. Document-level event role filler extraction using multi-granularity contextualized encoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8010–8020, Online. Association for Computational Linguistics.

Xinya Du, Alexander Rush, and Claire Cardie. 2021. GRIT: Generative role-filler transformers for document-level event entity extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 634–644, Online. Association for Computational Linguistics.

Qiang Gao, Zixiang Meng, Bobo Li, Jun Zhou, Fei Li, Chong Teng, and Donghong Ji. 2024. Harvesting events from multiple sources: Towards a cross-document event extraction paradigm. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1913–1927, Bangkok, Thailand. Association for Computational Linguistics.

Ridong Han, Tao Peng, Chaohao Yang, Benyou Wang, Lu Liu, and Xiang Wan. 2023. Is information extraction solved by chatgpt? an analysis of performance, evaluation criteria, robustness and errors. *arXiv preprint arXiv:2305.14450*, page 48.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

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, Bangkok, Thailand. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.

Heng Ji, Ralph Grishman, Zheng Chen, and Prashant Gupta. 2009. Cross-document event extraction and tracking: Task, evaluation, techniques and challenges. In *Proceedings of the International Conference RANLP-2009*, pages 166–172, Borovets, Bulgaria. Association for Computational Linguistics.

Philippe Laban, Wojciech Kryscinski, Divyansh Agarwal, Alexander Fabbri, Caiming Xiong, Shafiq Joty, and Chien-Sheng Wu. 2023. SummEdits: Measuring LLM ability at factual reasoning through the lens of summarization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9662–9676, Singapore. Association for Computational Linguistics.

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

Bo Li, Gexiang Fang, Yang Yang, Quansen Wang, Wei Ye, Wen Zhao, and Shikun Zhang. 2023a. Evaluating chatgpt's information extraction capabilities: An assessment of performance, explainability, calibration, and faithfulness. *arXiv preprint arXiv:2304.11633*.

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event extraction as multi-turn question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 829–838, Online. Association for Computational Linguistics.

Junpeng Li, Zixia Jia, and Zilong Zheng. 2023b. Semi-automatic data enhancement for document-level relation extraction with distant supervision from large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5495–5505, Singapore. Association for Computational Linguistics.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria. Association for Computational Linguistics.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online. Association for Computational Linguistics.

Xingjian Lin, Shengfei Lyu, Xin Wang, Qiuju Chen, and Huanhuan Chen. 2025. Generation-augmented and embedding fusion in document-level event argument extraction. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4078–4084, Abu Dhabi, UAE. Association for Computational Linguistics.

Yubo Ma, Yixin Cao, Yong Hong, and Aixin Sun. 2023. Large language model is not a good few-shot information extractor, but a good reranker for hard samples! In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10572–10601, Singapore. Association for Computational Linguistics.

Yubo Ma, Zehao Wang, Yixin Cao, Mukai Li, Meiqi Chen, Kun Wang, and Jing Shao. 2022. Prompt for extraction? PAIE: Prompting argument interaction for event argument extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages

6759–6774, Dublin, Ireland. Association for Computational Linguistics.

Qingkai Min, Qipeng Guo, Xiangkun Hu, Songfang Huang, Zheng Zhang, and Yue Zhang. 2024. Synergetic event understanding: A collaborative approach to cross-document event coreference resolution with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2985–3002, Bangkok, Thailand. Association for Computational Linguistics.

Abhijnan Nath, Shadi Manafi Avari, Avyakta Chelle, and Nikhil Krishnaswamy. 2024. Okay, let's do this! modeling event coreference with generated rationales and knowledge distillation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3931–3946, Mexico City, Mexico. Association for Computational Linguistics.

Hao Peng, Xiaozhi Wang, Jianhui Chen, Weikai Li, Yunjia Qi, Zimu Wang, Zhili Wu, Kaisheng Zeng, Bin Xu, Lei Hou, and 1 others. 2023. When does in-context learning fall short and why? a study on specification-heavy tasks. *arXiv preprint arXiv:2311.08993*.

Qwen-Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Jiawei Sheng, Shu Guo, Bowen Yu, Qian Li, Yiming Hei, Lihong Wang, Tingwen Liu, and Hongbo Xu. 2021. CasEE: A joint learning framework with cascade decoding for overlapping event extraction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 164–174, Online. Association for Computational Linguistics.

Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and Philip S Yu. 2022. Graph structure learning with variational information bottleneck. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4):4165–4174.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. GPT-RE: In-context learning for relation extraction using large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3534–3547, Singapore. Association for Computational Linguistics.

Longyue Wang, Chenyang Lyu, Tianbo Ji, Zhirui Zhang, Dian Yu, Shuming Shi, and Zhaopeng Tu. 2023. Document-level machine translation with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16646–16661, Singapore. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Runxin Xu, Tianyu Liu, Lei Li, and Baobao Chang. 2021. Document-level event extraction via heterogeneous graph-based interaction model with a tracker. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3533–3546, Online. Association for Computational Linguistics.

Xin Xu, Yuqi Zhu, Xiaohan Wang, and Ningyu Zhang. 2023. How to unleash the power of large language models for few-shot relation extraction? In *Proceedings of the Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, pages 190–200, Toronto, Canada (Hybrid). Association for Computational Linguistics.

Yifan Xu, Xiao Liu, Xinghan Liu, Zhenyu Hou, Yueyan Li, Xiaohan Zhang, Zihan Wang, Aohan Zeng, Zhengxiao Du, Zhao Wenyi, Jie Tang, and Yuxiao Dong. 2024. ChatGLM-math: Improving math problem-solving in large language models with a self-critique pipeline. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9733–9760, Miami, Florida, USA. Association for Computational Linguistics.

Hang Yang, Yubo Chen, Kang Liu, Yang Xiao, and Jun Zhao. 2018. DCFEE: A document-level Chinese financial event extraction system based on automatically labeled training data. In *Proceedings of ACL 2018, System Demonstrations*, pages 50–55, Melbourne, Australia. Association for Computational Linguistics.

Hang Yang, Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, and Taifeng Wang. 2021. Document-level event extraction via parallel prediction networks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6298–6308, Online. Association for Computational Linguistics.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5284–

5294, Florence, Italy. Association for Computational Linguistics.

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.

Junjie Ye, Nuo Xu, Yikun Wang, Jie Zhou, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. Llm-da: Data augmentation via large language models for few-shot named entity recognition. *arXiv preprint arXiv:2402.14568*.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023. Extractive summarization via ChatGPT for faithful summary generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3270–3278, Singapore. 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 Illustrative Examples of SLM Errors and LLM Rationales

As noted in the introduction, multi-document event extraction challenges for both small language model supervised fine-tuning pipeline and large language model sequence-to-sequence generation. This appendix provides concrete illustrations to clarify (i) representative errors arising in SLM pipeline subtasks and (ii) what kinds of explicit rationales are required for LLM reasoning to support performance and interpretability but are not easily supplied through prompts.

A.1 SLM Pipeline Errors

We illustrate two representative sources of errors in SLM pipeline:

(a) Document-level argument extraction. This component has an F1 score below 70%. For example, given the following source sentence:

> "A strong earthquake rattled Indonesia's West Papua province Wednesday just days after a powerful quake levelled buildings and **killed** one person... The 6.1-magnitude quake was the latest in a series of dozens of powerful tremors to have hit the region since 7.6 and 7.5 magnitude quakes that struck off the provincial capital Manokwari on Sunday... Wednesday's shallow quake hit at 7:48 am (2248 GMT Tuesday)..."

The SLM system outputs are shown in Table 5.

(b) Cross-document event coreference. This component has an F1 score below 80%. Consider the following text:

> "Earlier this morning, an **earthquake**(1) with a preliminary magnitude of 2.0 struck near The Geysers, according to the USGS. The **earthquake**(2) struck at about 7:30 a.m. and had a depth of 1.4 miles, according to the USGS. The 4.2-magnitude **earthquake**(3) was recorded at approximately 9:27 a.m. Sunday, according to the US Geological Survey, which originally had rated the **quake**(4) as 4.4 in magnitude."

All four mentions (1–4) refer to the same seismic event trigger, but the SLM pipeline linked only (1) and (2), leaving (3) and (4) unclustered. In a separate report on the same topic:

> "A 4.2-magnitude **earthquake**(5) shook a remote area of eastern Sonoma County on Sunday morning, the largest in a flurry over the weekend, according to the U.S. Geological Survey."

Here, SLM correctly linked (5) to quake (4) but still failed to reconnect mentions (3) and (4) back into the main event cluster, resulting in a fragmented representation.

A.2 Explicit Rationales Required for LLMs

To illustrate the structured reasoning process, we present one representative iteration from Reasoning Stage I of Event Composition.

Input: Event Clusters and Candidate Event (AXXX refers to the argument cluster id predicted by SLM):

Event Cluster 1 (EC1)

Trigger: *injured*
Human Participant: dozens (A777)
Non-Human Participant: quake (A768)
Time: Tuesday (A784)
Location: in a region (A783)

Event Cluster 2 (EC2)

Trigger: *injured*
Human Participant: others (A781)
Non-Human Participant: quakes (A768)
Time: Sunday (A764)
Location: Papua, eastern Indonesia (A848)

Candidate Event

Trigger: *injuring*
Human Participant: dozens (A777)
Non-Human Participant: quake (A768)
Location: across the affected region (A798)

Argument Role	Prediction	Gold Standard	Match?
Human Participant	person	one person	No (incomplete)
Non-Human Participant	7.6	quake	No (role confusion)
Time	Wednesday	on Sunday	No (misaligned)
Location	Indonesia's West Papua province	Indonesia's West Papua province	Yes

Table 5: Examples of argument extraction errors in the SLM pipeline.

Sub-step 1: Event-to-cluster assignment

Feature Comparison

Context
EC1: Same event: 6.1 quake in Aceh
EC2: Different event: Quake in Papua
Decision: EC1 aligned
Trigger
EC1: *injuring* $\approx$ *injured*
EC2: *injuring* $\approx$ *injured*
Decision: Both match
Human Participant
EC1: dozens (A777)
EC2: others (A781)
Decision: Exact match with EC1
Non-Human Participant
EC1: quake (A768)
EC2: quakes (A768)
Decision: Shared across both
Time
EC1: (missing) vs. Tuesday (A784)
EC2: (missing) vs. Sunday (A764)
Decision: No contradiction with EC1
Location
EC1: region (A798) vs. region (A783)
EC2: region (A798) vs. Papua (A848)
Decision: Consistent with EC1

LLM Decision

Relevance Level: e (Same news, synonymous triggers, consistent arguments)
Action: Merge the candidate event into EC1

Sub-step 2: Argument alignment

Argument Alignment

Human Participant: dozens (A777)
LLM Reasoning: Exact match confirms identity
Non-Human Participant: quake (A768)
LLM Reasoning: Exact match confirms same entity
Location: in a region, across the affected region (A783, A798)
LLM Reasoning: Semantically similar, inferred same location
Time: Tuesday (A784)
LLM Reasoning: Candidate lacks explicit time but does not contradict Tuesday; inferred same

B Dataset Details

B.1 Annotation Guidelines

B.1.1 Event Definition

In line with the TimeML specification, we define an event as a situation that "happens or occurs". Following the ECB+ guidelines, we categorize events into several fine-grained types:

1. **Occurrence**: These events describe things that happen in the world, such as *die, crash, build, merge, sell, land, arrive, distribute, eruption, explosion*.
2. **Perception**: Actions involving the perception of another event, e.g., *see, hear, watch, feel, glimpse, behold, listen, overhear*.
3. **Reporting**: Actions related to reporting by people or organizations, such as *say, report, tell, announce, explain, cite, state*.
4. **Aspectual**: These events focus on different facets of event development, such as *begin, finish, stop, continue*, capturing initiation, culmination, and continuation.
5. **State**: Denotes states of being or circumstances, including expressions like *be on board, hope, live, shortage, was an actor, the crisis, peace*. Often appears in predicative constructions (e.g., *to be* + nominal/adjectival phrase).
6. **Causative**: Describes causal actions, such as *cause, lead to, result, facilitate, induce, produce, bring about*.
7. **Generic**: Used for generic events not anchored in specific time or space.

Each event is represented as a **trigger–arguments structure**, where:

- **Trigger**: A word or phrase signaling the occurrence of the event.
- **Arguments**: Entities or events fulfilling specific semantic roles.

The arguments typically include:

- **Time**: When something happens or holds true.
- **Location**: Where something happens or holds true.
- **Participants**: Who or what is involved, divided into:
  - **Human Participants**
  - **Non-Human Participants**

The example below shows a structured event representation:

Event Components	Value
Trigger	injured
Time	on Tuesday
Location 1	in Indonesia's Aceh province
Location 2	western tip of Sumatra island
Human Participant	dozens of villagers
Non-Human Participant	6.1-magnitude quake

B.1.2 Annotation of Event Arguments

Our annotation process builds on the Event Coreference Bank Plus (ECB+), a widely used public benchmark for cross-document event coreference resolution. The ECB+ corpus contains 982 news articles spanning 43 topics, including earthquakes, criminal cases, and corporate acquisitions.

The objective of our annotation is to identify arguments for each event mention in a document. The event triggers are inherited directly from the ECB+ dataset, ensuring consistency with the original event coreference annotations.

To streamline the process, we developed a custom web-based annotation tool. During each annotation session, annotators were presented with:

- A target event pre-annotated from ECB+.
- A full document containing the event.
- A list of candidate arguments, which included:
  - Entities (from ECB+).
  - Other event mentions from the same document (from ECB+).

Annotators are instructed to select the arguments that semantically fit the target event from the provided candidate set. To ensure consistency and semantic accuracy, the following procedures were followed:

- Annotators were required to consult the FrameNet database (https://framenet.icsi.berkeley.edu/framenet_search) to identify the appropriate semantic frame corresponding to each target event.

- Based on the selected frame, annotators determined which roles needed to be filled and matched these with candidate arguments from the document.
- Annotators are encouraged to select all candidates that satisfied the semantic constraints of the identified frame.

This two-tiered approach—anchoring event semantics in FrameNet and constraining annotation to document-internal candidate sets—ensures high annotation quality and interpretability.

About FrameNet

FrameNet is a lexicographic database based on frame semantics, a linguistic theory that describes meaning through conceptual structures called *frames*. Each frame represents a typical situation (e.g., an injury event) and defines the semantic roles (called *frame elements*) that participants play in that situation.

For example, the "injured" event corresponds to the "Injury" frame, which describes an event where an **Agent** causes harm to a **Victim**. The corresponding semantic roles are outlined as below:

Semantic Frame

Agent: 6.1-magnitude quake
Explanation: The event causing the injury.

Victim: Dozens of villagers
Explanation: The entities that are injured.

Time: On Tuesday
Explanation: The time when the injury occurred.

Place 1: In Indonesia's Aceh province
Explanation: The location where the injury took place.

Place 2: Western tip of Sumatra island
Explanation: Another location where the injury took place.

Using FrameNet to identify semantic frames and their elements ensures annotators consistently interpret the roles of event arguments, enhancing the clarity and quality of the annotations.

B.1.3 Annotation Review

To ensure annotation quality, a review process is conducted. In this stage, a senior annotator validates and refines the initial annotations. For each target event, the senior annotator is provided with the golden trigger and its annotated arguments from the first stage.

Following the same procedure as the initial annotation, senior annotators are asked to consult FrameNet to identify the semantic frame evoked by each trigger. Based on the frame structure, they verify whether the selected arguments correctly filled the expected semantic roles (e.g., **Agent**, **Victim**, **Time**, **Place**) and whether any required arguments are missing.

To illustrate the correction process, we provide the following example:

Original Document

> Indian ship thwarts **piracy** attempt in Gulf of Aden, 26 pirates arrested. Updated: November 11, 2011, 18:50 IST. Indian Naval ship, INS Sukanya, thwarted a piracy attack in the Gulf of Aden and captured three boats of the pirates. A statement from the Defence PRO says the incident happened yesterday when INS Sukanya was escorting a group of merchant vessels.

Annotation Correction

Semantic Role	Argument
Perpetrator	pirates
Victim	vessels
Instrument	**boats** *(added)*
Time	yesterday
Place	Gulf of Aden

Correction Explanation

> "Boats" was initially overlooked in the annotations as a potential **Instrument**, representing the object that the pirates used to carry out the attack. This correction ensures that all semantic roles are properly filled.

This protocol allowed us to improve both recall and precision of argument annotations while maintaining consistency across the dataset.

B.1.4 Annotation Examples Based on Web UI

To assist with the annotation process, we developed a custom web-based interface that simplifies the task of annotating event arguments within a document. The web UI allows annotators to view the

full document and select relevant arguments from a pre-defined list. The interface is designed for efficiency and ease of use, enabling high-quality annotations.

To clarify how the annotation process works, let's examine examples of a piracy event. As shown in the Figure 6 and Figure 7, the upper-left box displays all occurrences of the same "piracy" event across different documents. By clicking on each event, the corresponding document appears on the right side of the interface. The event mentions in the document are highlighted in blue, while the candidate arguments are shown in red. On the lower-left corner, there is a list of candidate arguments, which includes both event mentions and entities that might be relevant to the event.

In the first example (Figure 6), the annotator selects the following relevant arguments:

Argument	**Role**
ships	Vehicle
arms	Instrument
attempt	Purpose
international waters	Location

In the second example (Figure 7), the selected arguments are:

Argument	**Role**
vessels	Victim
in the Gulf of Aden	Location
attack	Purpose
on 10 Nov 11	Time

Additionally, we provide the FrameNet semantic frame for the "piracy" event, as shown in Figure 8, which annotators can reference. This frame helps annotators correctly identify the semantic roles of the arguments, such as **Perpetrator**, **Victim**, **Instrument**, and others. Annotators are instructed to consult the FrameNet database to select arguments for each event trigger.

B.2 Dataset Statistic

Table 6 provides a breakdown of argument distribution across documents. In the case of multi-document events, a significant portion of the arguments (1,230, or approximately 40%) are unique to a single document, while the remainder, 1,866 arguments (approximately 60%), appear in multiple documents. Specifically, a large number of events are mentioned in two or more documents, with 632 arguments occurring in two documents, 350 in three, and 222 in four, reflecting considerable redundancy. This redundancy indicates a substantial overlap in the events reported across different sources, highlighting the need for more sophisticated aggregation techniques to handle repeated arguments, rather than relying on simple merging strategies.

In contrast, arguments in single-document events occur only within one document by definition, with 4,669 such arguments. This reflects the inherent uniqueness of content in individual news articles, in stark contrast to the redundancy found in multi-document events.

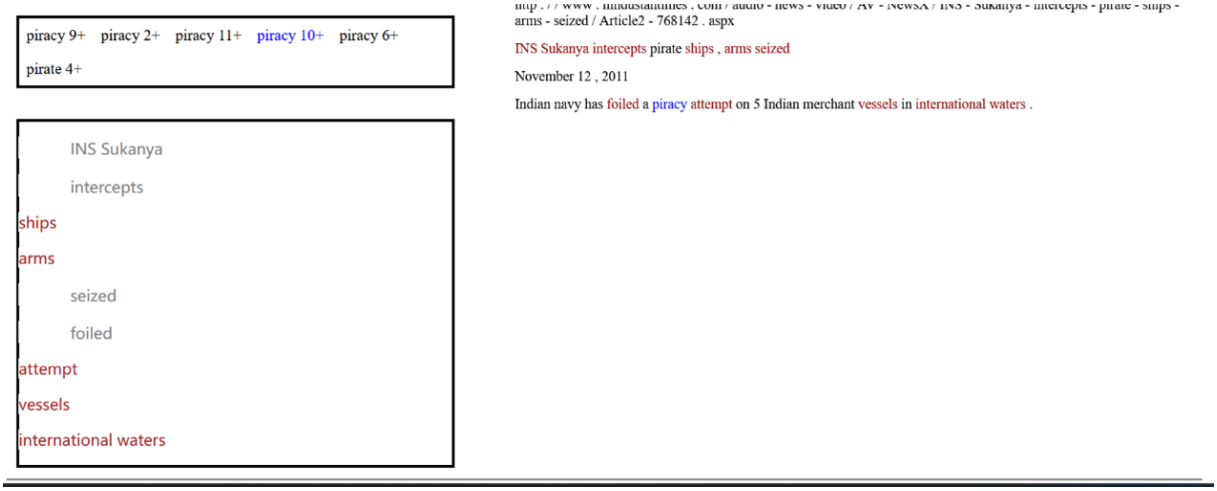

Figure 6: Example 1 for the "piracy" event.

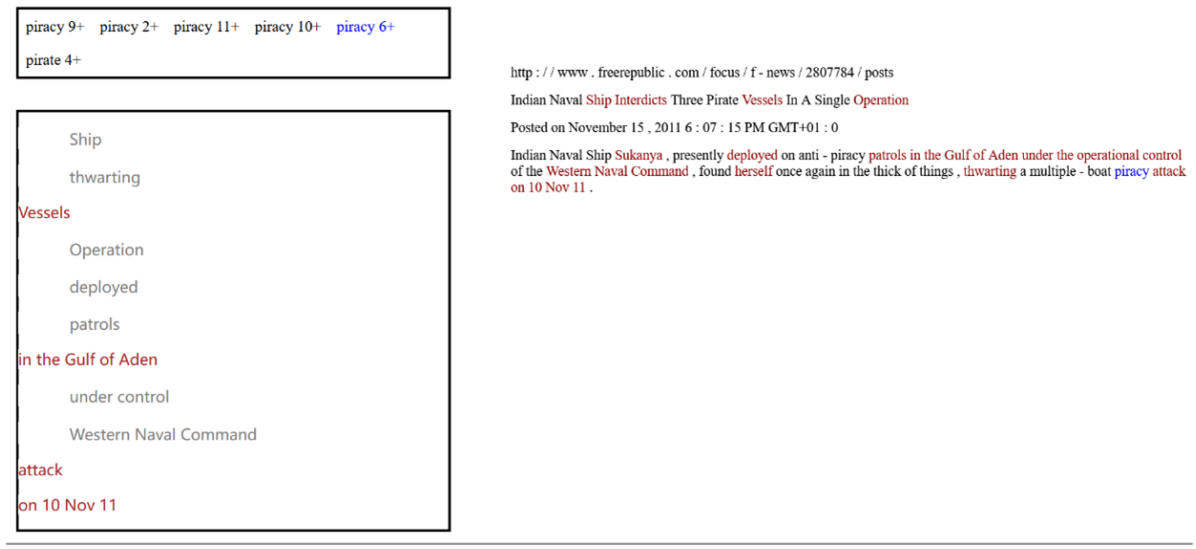

Figure 7: Example 2 for the "piracy" event.

# Docs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Multi-doc Events	1230	632	350	222	188	130	80	70	58	54	30	20	18	3	4	1	4	2
Single-doc Events	4669	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

Table 6: Argument distribution across documents.

Piracy

Definition:

The words in this frame describe situations in which a Perpetrator forcibly seizes control over a Victim's Vehicle to gain some end.

French farmers have HIJACKED a lorry and destroyed its load of strawberries.

But attackers have HIJACKED cars waiting at traffic lights and road junctions by simply opening the passenger door, jumping in and forcing the driver to a more deserted area.

FEs:

Core:

Perpetrator [Perp] **Semantic Type:** Sentient	This is the person (or other agent) who forcibly seizes control over a vehicle to gain some end.
Vehicle [Veh]	This is the means of transportation which gets under the control of the perpetrator. Example: The terrorists hijacked the plane.
Victim [Vic] **Semantic Type:** Sentient	This FE describes the people that are suffering as a result of the Perpetrator's action. The rebels skyjacked a plane with 89 passangers.

Non-Core:

Degree [Degr] **Semantic Type:** Degree	Degree to which event occurs
Duration [Duration] **Semantic Type:** Duration	The amount of time for which a state holds or a process is ongoing.
Event [Evnt] **Semantic Type:** State_of_affairs	The unlawful seizure of a Vehicle by a Perpetrator.
Explanation [exp] **Semantic Type:** State_of_affairs	The situation that the Perpetrator had in mind which lead to the decision to perpetrate the particular act of piracy. He CARJACKED the Lincoln Towncar instead because it looked ritzier.
Frequency [Freq]	The number of times that the VEHICLE is seized in acts of piracy. The ferry that Kenneth was on was HIJACKED twice in three days
Goal [Goal] **Semantic Type:** Goal	This is the place to which the vehicle is directed.
Instrument [Inst] **Semantic Type:** Physical_entity	An object that the Perpetrator uses in taking control of the Vehicle. The plane was HIJACKED with knives
Manner [Manr] **Semantic Type:** Manner	Manner of performing an action
Means [Means] **Semantic Type:** State_of_affairs	An act performed by the Perpetrator by which the seizure of the Vehicle is accomplished.
Place [Place] **Semantic Type:** Locative_relation	Where the event takes place.
Purpose [] **Semantic Type:** State_of_affairs	The action that the Perpetrator is trying to accomplish by piracy. We have HIJACKED this plane just to show how easy it is.
Source [Src] **Semantic Type:** Source	This is the place from which the vehicle is taken away.
Time [] **Semantic Type:** Time	When the event occurs.

Figure 8: The semantic frame for the piracy event as found in FrameNet, illustrating key roles such as **Perpetrator**, **Victim**, **Instrument**, and **Location**.

C Model Implementations

C.1 SLM-based Pipeline Models

Event Trigger Detection We follow the approach in Cattan et al. (2021), which uses a BERT-based (Devlin et al., 2019) span scorer with pruning to keep only high-quality candidate triggers on ECB+ dataset. For each candidate span $s \in \mathcal{S}_d$ within document d, a scoring function $f(s, d)$ evaluates its likelihood of being an event trigger. The final trigger set T_d is formed by selecting spans whose scores exceed a predefined threshold τ:

$$T_d = \{s \in \mathcal{S}_d \mid f(s, d) > \tau\}.$$

Cross-Document Event Coreference Resolution We adopt the method from Cattan et al. (2021), performing coreference resolution over predicted events rather than gold-standard annotations. Coreference scores are computed for all pairs of candidate events (e_i, e_j) both within and across documents, where $e_i, e_j \in \mathcal{E}_{\text{all}}$:

$$\text{Score}(e_i, e_j) = f_\theta(e_i, e_j),$$

with f_θ denoting a BERT-based pairwise scorer.

Cross-document event clusters are then formed via agglomerative clustering on $\mathcal{E}_{\text{all}}$, using the similarity scores and a threshold τ:

$$C_T = \text{AgglomerativeCluster}(\mathcal{E}_{\text{all}}, \text{Score}, \tau).$$

Document-Level Event Argument Extraction We adopt a trigger-based approach for document-level argument extraction Ma et al. (2022). Given a trigger $t \in T_d$ in document d, and a predefined set of roles $\mathcal{R} = \{r_1, r_2, \ldots, r_k\}$, the model selects the most likely argument span for each role from a candidate set $\mathcal{S}_d \cup \{\epsilon\}$, where ϵ denotes a null span indicating no argument. Formally, for each role $r \in \mathcal{R}$, the predicted argument is:

$$a_r^* = \arg \max_{s \in \mathcal{S}_d \cup \{\epsilon\}} \text{Score}(s \mid d, t, r),$$

where $\text{Score}(s \mid d, t, r)$ is computed by a BART-based (Lewis et al., 2020) model conditioned on d, t, and r via prompt-based input formatting.

The final argument set for trigger t is:

$$A_t = \{(r, a_r^*) \mid a_r^* \neq \epsilon\}.$$

Cross-Document Argument Coreference Resolution We use a model similar to event coreference to separately handle argument coreference, addressing the complexity of cases where events serve as arguments for other events in nested structures. A BERT-based pairwise scorer followed by agglomerative clustering groups argument mentions into clusters $C_A = \{C_1^A, C_2^A, \ldots\}$.

Event Canonicalization We apply a frequency-based heuristic to select canonical representatives within each event cluster C_i. Specifically, given the set of triggers $T_i = \{t_{i1}, t_{i2}, \ldots, t_{im}\}$ and the set of argument mentions $A_i = \{a_{i1}, a_{i2}, \ldots, a_{in}\}$ extracted from the constituent document-level events in cluster C_i, we define the canonical trigger and arguments as:

$$\hat{t}_i = \arg \max_{t \in T_i} \text{freq}(t),$$

$$\hat{A}_i = \left\{ \arg \max_{a \in C} \text{freq}(a) \mid C \in \mathcal{C}(A_i) \right\},$$

Module	Backbone Model	Learning Rate	Batch Size	Epochs / Steps	Other Hyperparameters
Event Trigger Detection	RoBERTa-large	1e-4	16	10 epochs	Max span length: 10 Score threshold: 0.25
Cross-Document Event Coreference Resolution	RoBERTa-large	1e-4	32	10 epochs	Clustering linkage: average Clustering merge threshold: 0.5
Document-Level Event Argument Extraction	BART-base	2e-5	16	10,000 steps	Max span length: 10 Prompt template: concatenate Max prompt length: 80
Cross-Document Argument Coreference Resolution	RoBERTa-large	1e-4	32	10 epochs	Clustering linkage: average Clustering merge threshold: 0.8

Table 7: Hyperparameter settings for SLMs in the pipeline.

where freq$(\cdot)$ denotes the frequency of occurrence, and $\mathcal{C}(A_i)$ represents the set of argument clusters associated with C_i. The canonical event representation is thus $\hat{e}_i = (\hat{t}_i, \hat{A}_i)$.

Hyperparameters For models that require training, the hyperparameter settings are provided in Table 7.

C.2 LLM-based Pipeline Prompts

We have adapted the standard SLM-based pipeline by leveraging the extended context handling ability and flexibility of LLMs, resulting in a modified pipeline tailored for multi-document processing.

Step 1: Event trigger detection

Task Objective:
You are given a natural language document composed of multiple sentences. Your task is to identify **event triggers** in each sentence. A trigger is the **head word or phrase** that most strongly conveys the event's meaning. Wrap each identified trigger with ##.

---

1. Trigger Extent
- Mark only the **head** of the event phrase: - For **verbal expressions**, annotate only the main verb (e.g., `##shooting##`, not `may have been shooting`). - For **nominal expressions**, annotate the main noun that expresses the event (e.g., `##earthquake##`). - **Do not annotate**: - Auxiliary verbs (e.g., `may`, `have`, `did`). - Negation words (e.g., `not`, `never`). - Determiners and adjectives (e.g., `this terrible` in `this terrible war`). - In **light-verb constructions**, annotate both the verb and the noun **separately**: - e.g., "make an offer" → `##make##`, `##offer##`.

---

2. Valid Parts of Speech for Triggers
Triggers can be: - **Verbs**: e.g., `##fights##`, `##arrives##` - **Nouns** (including nominalizations and proper nouns): e.g., `##eruption##`, `##World War II##` - **Attributive participles**: e.g., `##crying##` in "the crying baby" - **Predicative complements** (after copular verbs): e.g., `##marine##` in "was a marine" - **Pronouns** referring to prior events: e.g., `##It##` in "It did not trigger a tsunami"

---

3. Output Format
- Assign a sentence ID (Sentence1, Sentence2, etc.) for each sentence. - Wrap each trigger with ##. - Preserve the **original sentence wording** in full. - Return one labeled sentence per line. - If multiple triggers appear, wrap each independently.

Instructions to the model:
- Read the paragraph sentence by sentence. - Identify each event trigger and wrap it with ##. - Return one labeled sentence per line using the format above.
The document to process is as following:
`{}`

Step 2: Event argument extraction

Task Objective:
You are given a document formatted as:
```
Sentence1: ...
Sentence2: ...
```
Your task is to extract all events. An event consists of:

- **trigger**: the head word or phrase already annotated with `##`.
- **arguments**: four types—
  - `TIME`: answers "when" (dates, times, relative times).
  - `LOCATION`: answers "where" (places, venues, regions).
  - `HUMAN_PART`: answers "who" (individuals, groups, pronouns).

- NON_HUMAN_PART: answers "what" (objects, institutions, abstract entities).
- If multiple mentions of the same type occur, assign numbered keys (e.g. HUMAN_PART_1, HUMAN_PART_2).

Output Format:
Return only a JSON array of event objects. No explanatory text. Each object must have:

```
{
  "event_id": "E1",
  "trigger": "...",
  "arguments": {
    "TIME": "...",
    "LOCATION": "...",
    "HUMAN_PART": "...",
    "NON_HUMAN_PART": "..."
  }
}
```

Example

```
Sentence1: He ##arrived## in Paris
    on January 1.
Sentence2: They ##celebrated## the
    victory.
```

Expected output:

```
[
{
  "event_id": "E1",
  "trigger": "arrived",
  "arguments": {
    "TIME": "January 1",
    "LOCATION": "Paris",
    "HUMAN_PART": "He",
    "NON_HUMAN_PART": ""
  }
},
{
  "event_id": "E2",
  "trigger": "...",
  "arguments": {
    "TIME": "...",
    "LOCATION": "...",
    "HUMAN_PART": "...",
    "NON_HUMAN_PART": "..."
  }
}
...
]
```

Instructions to the model:
Read each sentence, extract events in order, and output exactly the JSON array above.
Here is the document to process:
{}

Step 3: Cross-document event coreference resolution

Task Objective:
You are given a single document with:

- doc_id: a unique identifier.
- text: the full document text.
- events: a list of objects, each with:
  - event_id, trigger, arguments: keys TIME, LOCATION, HUMAN_PART, NON_HUMAN_PART

Clustering Steps:

1. Gather all events into a flat list.
2. Identify coreference groups by trigger lemma and argument overlap.
3. For each group:
   - Assign cluster_id (C1, C2, ...).
   - Select earliest trigger.
   - For each role, pool non-empty values, dedupe, pick earliest; omit if none.
   - Add original_events: merged event_ids.
4. Singleton events form their own cluster.

Output Format:
Return only JSON, no extra text:

```
{
  "clusters": [
  {
    "cluster_id":"C1",
    "triggers":["hold"],
    "arguments":{
      "HUMAN_PART":["He"],
      "NON_HUMAN_PART":["the screen"]
    },
    "original_events":["E7","E8"]
  },
  {
    "cluster_id":"C2",
    "triggers":["want"],
    "arguments":{
      "HUMAN_PART":["we"],
      "NON_HUMAN_PART":["the ceremony
          "]
    },
    "original_events":["E5"]
  }
  ...
  ]
}
```

Instructions to the model:
Read the input JSON, perform clustering, and output exactly the JSON under "clusters".
Here is the document:
{}

Step 4: Event consolidation

Input Format:
You will be given a list of document entries. Each entry has:

- "doc_id": the document identifier
- "clusters": an array of intra-document clusters, each with:
  - "cluster_id"
  - "trigger": [...]
  - "arguments": {"TIME": [...

], "LOCATION": [...],
"HUMAN_PART": [...],
"NON_HUMAN_PART": [...]}
– "original_events": [...]

TASK: Merge Across Documents
Merge these clusters into global event clusters. Return only the final JSON.
Steps:

1. Pool all clusters into one flat list.
2. Identify cross-document coreference sets by:
   - Matching or synonym-equivalent trigger lemmas.
   - Overlapping or identical argument values.
3. For each set:
   - Assign "EC_id" (EC1, EC2, ...).
   - Select earliest trigger.
   - For each role, collect non-empty values, dedupe, pick earliest; omit if none.
4. Non-coreference clusters become singleton global clusters.

Output Format:
Return a JSON array:

```
[
{
  "EC_id": "EC1",
  "trigger": ["hold"],
  "arguments": {
    "HUMAN_PART": ["He"],
    "NON_HUMAN_PART": ["the screen"]
  }
},
...
]
```

Here is the INPUT:
{}

C.3 LLM Sequence-to-Sequence Prompts

LLM seq2seq ICL prompt

Task Objective:
The goal is to first identify events within individual documents, where each event consists of a trigger and its associated arguments, and then consolidate these events from multiple documents into corpus-level event clusters. Some events from different documents may refer to the same real-world event, and these should be merged into a single cluster after consolidation.

An example is provided as follows:

Input documents: {}
Example: "Doc_1":". . . ",
"Events_of_this_document":". . . ",
"Doc_2":". . . ", . . .
Event cluster ids consisting of event ids from individual documents: {}
Example: "event_cluster_id":"EC1",
"related_event_ids":"D1_E2","D2_E3",
"D3_E1",. . .
Output event clusters: {}
Example: "event_cluster_id":"EC1",
"canonical_trigger": "Oscar",
"arguments": ". . . ",. . .

An exercise is provided as follows:

Input documents:
{}

LLM seq2seq SFT prompt

Task Definition:
To accomplish the multi-document event extraction task, you need to extract a structured representation of events from the given multiple documents. You are required to extract all parameter information of the events, including time, location, participants, etc., and mark the role information for all time parameters.
For event entities, if there are multiple forms of synonymous entities, please select the most representative form.
For the role information of entities, if the same entity has multiple roles, choose the most representative role as the final role.
Output in the following JSON format:

"trigger": "Event Trigger",
"arguments": [
"role": "Role1", "mention": "Entity1",
"role": "Role2", "mention": "Entity2"
...
]

Input documents:
{}
Output:

C.4 Our Collaborative Framework Prompts

Reasoning stage 1.1: Event-to-cluster assignment

Task Objective:
Find the most relevant event cluster for merging.

Document-Level Event Structure:
Each event in a document has the following structure:

- `event_id`: A unique identifier for the event.
- `trigger`: A keyword or phrase that triggers the event.
- `arguments`: A list of arguments, each with:
  - `argument_id`: A unique identifier for the argument.
  - `role`: One of
    - `time`: specific moment or duration when the event occurs
    - `location`: the place or area where the event takes place
    - `human participant`: the people who participate in or influence the event
    - `non_human participant`: entities or nested events
  - `mention`: The specific textual expression that refers to the argument.
  - `argument_cluster_id`: The ID of the argument cluster it belongs to.

Corpus-Level Event Cluster Structure:
Each cluster has:

- `EC_id`: A unique ID for the event cluster.
- `triggers`: A list of triggers concatenated from the document-level events.
- `arguments`: A list of merged arguments, each with:
  - `roles`: List of roles concatenated from the document-level arguments.
  - `mentions`: List of mentions concatenated from the document-level arguments.
  - `argument_cluster_ids`: List of argument cluster IDs concatenated from the document-level arguments.

Procedure:
Stage 1: Identify the most relevant cluster

- **Context:** Compare document contexts.
- **Trigger:** Check for synonymy.
- **Arguments:** Compare time, location, participants; infer missing details.

Stage 2: Select relevance level

- `a`: Different seminal news.
- `b`: Same news, not synonymous triggers.
- `c`: Same news, synonymous triggers, significant differences.
- `d`: Same news, synonymous triggers, basic match.
- `e`: Same news, synonymous triggers, all aspects match.
- **Merge decision:**
  - If level `d` or `e`, `merge`.
  - Otherwise, `create`.

Output Requirements:

- Detailed analysis for both stages.
- A JSON object with fields:
  - `decision`: `"merge"` or `"create"`.
  - `most_relevant_event_cluster_id`: cluster ID or `"none"`.
- No extraneous text in the JSON.

Input documents and corresponding events: {}
Existing event clusters: {}
Candidate event: {}

Reasoning stage 1.2: Argument alignment

Task Objective:
Merge the candidate event into the target event cluster.

Document-Level Event Structure:
Each event in a document has the following structure:

- `event_id`: A unique identifier for the event.
- `trigger`: A keyword or phrase that triggers the event.
- `arguments`: A list of arguments, each with:
  - `argument_id`: A unique identifier for the argument.
  - `role`: One of
    - `time`: specific moment or duration when the event occurs
    - `location`: the place or area where the event takes place
    - `human participant`: the people who participate in or influence the event

    - * non_human participant: entities or nested events
  - mention: The specific textual expression that refers to the argument.
  - argument_cluster_id: The ID of the argument cluster it belongs to.

Corpus-Level Event Cluster Structure:
Each corpus-level event cluster has:

- EC_id: A unique ID for the event cluster.
- triggers: List of triggers concatenated from document-level events.
- arguments: List of merged arguments, each with:
  - roles: List of roles concatenated from document-level arguments.
  - mentions: List of mentions concatenated from document-level arguments.
  - argument_cluster_ids: List of argument cluster IDs concatenated from document-level arguments.

Procedure:
For each component in the candidate event:

- **Trigger merging**: Append the candidate trigger to the cluster's triggers list.
- **Argument merging**: For each candidate argument:
  - Determine match by semantic similarity and argument_cluster_id:
    - * Similar & same cluster → match
    - * Similar & different cluster → match
    - * Not similar & same cluster → no match
    - * Not similar & different cluster → no match
  - If matched: append its argument_cluster_id, role, and mention to the corresponding lists.
  - If not matched: add as a new argument entry.

Output Requirements:

- Provide a detailed analysis.
- Then output a JSON object with key:
  - updated_event_cluster: The newly merged event cluster.
- No extraneous text in the JSON.

Input documents and corresponding events:
{}
Target event cluster: {}
Candidate event: {}

Reasoning stage 2: Event consolidation

Task Objective:
The event cluster constitutes the structured representation of events at the corpus-level by integrating information from the corresponding document-level events. We aim to fine-tune this structure by correcting errors in the integrated information.

Procedure:

- **Step 1.** Identify the corresponding document-level events using the listed event_ids.
- **Step 2.** From the triggers list, select one core trigger and remove all others.
- **Step 3.** For each entry in arguments:
  - Check relevance: an argument is relevant if it factually contributes to the event cluster.
  - If relevant:
    - * Retain the argument.
    - * In its mentions list, choose one core mention and remove the rest.
  - If irrelevant: remove the entire argument entry.

Document-Level Event Structure:
Each document-level event has:

- event_id: Unique identifier.
- trigger: Keyword or phrase.
- arguments: List of arguments, each with:
  - argument_id: Unique identifier.
  - mention: Textual expression of the argument.

Corpus-Level Event Cluster Structure:
Each cluster has:

- EC_id: Unique cluster ID.
- event_ids: List of document-level event IDs.
- triggers: Concatenated triggers from member events.
- arguments: List of merged arguments, each with:
  - mentions: Concatenated mentions from member events.

Input documents and corresponding events:

{}
Event cluster: {}

Output Requirements:

- Provide a detailed analysis of each step.
- Then output a JSON object with the updated event cluster under the key:
  - `updated_event_cluster`: The refined cluster structure.
- Do not merge arguments; keep the original cluster schema intact.
- The JSON must contain no additional text or notes.

C.5 A Detailed Running Example of Our Framework Components

To further clarify the components of our framework, we provide a detailed running example based on the example in Figure 1, demonstrating one iteration of Stage 1 and Stage 2 reasoning. For clarity, we have structured the reasoning process in the form of tables.

Reasoning Stage I: Event Composition

Input 1: Existing Event Clusters

Cluster ID: EC1
Trigger: injured
Human Participant: dozens (A777)
Non-Human Participant: quake (A768)
Time: Tuesday (A784)
Location: in a region (A783)

Cluster ID: EC2
Trigger: injured
Human Participant: others (A781)
Non-Human Participant: quakes (A768)
Time: Sunday (A764)
Location: in Papua in easternmost of Indonesia (A848)

Input 2: Candidate Event

Trigger: injuring
Human Participant: dozens (A777)
Non-Human Participant: quake (A768)
Location: across the affected region (A798)

Sub-step 1.1: Event-to-cluster Assignment

Feature: Context
EC1: Same event: 6.1-magnitude earthquake in Aceh
EC2: Different event: Quake in Papua
Decision: EC1 context aligned

Feature: Trigger
EC1: injuring $\approx$ injured
EC2: injuring $\approx$ injured
Decision: Both match

Feature: Human Participant
EC1: dozens (A777)
EC2: others (A781)
Decision: Exact match with EC1

Feature: Non-Human Participant
EC1: quake (A768)
EC2: quakes (A768)
Decision: Shared across both

Feature: Time
EC1: (missing) vs. Tuesday (A784)
EC2: (missing) vs. Sunday (A764)
Decision: No contradiction with EC1

Feature: Location
EC1: across the affected region (A798) vs. in a region (A783)
EC2: across the affected region (A798) vs. Papua (A848)
Decision: Consistent with EC1

Sub-step 1.2: Merge Candidate Event

Relevance Level: e (Same news, synonymous triggers, consistent arguments)
Action: Merge the candidate event into **Event Cluster 1 (EC1)**

Sub-step 2: Argument Alignment

Type: Trigger
Mentions: injured, injuring
LLM Reasoning: "injuring" is a variant form of "injured," indicating the same event trigger.

Type: Human Participant
Mentions: dozens
Argument Clusters: A777
LLM Reasoning: Exact match on mention and cluster ID, confirming identity of human participants.

Type: Non-Human Participant

Mentions: quake
Argument Clusters: A768
LLM Reasoning: Exact match on mention and cluster ID, confirming same non-human participant.

Type: Location
Mentions: in a region, across the affected region
Argument Clusters: A783, A798
LLM Reasoning: Mentions are semantically similar; LLM infers they refer to the same location despite different cluster IDs.

Type: Time
Mentions: Tuesday
Argument Clusters: A784
LLM Reasoning: Candidate lacks explicit time but does not contradict existing "Tuesday" argument; inferred as same.

Reasoning Stage II: Event Consolidation

Event Consolidation

Component: Trigger
Mentions: "injured" (D13_E2), "injuring" (D14_E4)
Selected: "injured"
Reasoning: More concise and commonly used; semantically similar to "injuring"

Component: Argument 1
Mentions: "dozens", "dozens"
Selected: "dozens"
Reasoning: Identical mentions; quantify injured people; keep one

Component: Argument 2
Mentions: "quake", "quake"
Selected: "quake"
Reasoning: Identical mentions; refers to cause of injury; keep one

Component: Argument 3
Mentions: "in a region", "across the affected region"
Selected: "across the affected region"
Reasoning: Both relevant, second is more precise; keep more specific mention

Component: Argument 4
Mentions: "Tuesday"
Selected: "Tuesday"
Reasoning: Time argument relevant for event temporal context; retain for completeness

D Case Study

D.1 Stage 1.1: Event-to-cluster assignment

An example of event-to-cluster assignment is presented in Table 8. The LLM takes the new candidate event D7_E1 ("ARRESTS" of "RCMP" and "MEN" in "THE RELIGIOUS COMMUNITY OF BOUNTIFUL IN THE BC INTERIOR") and, in the first event composition phase, recognizes that the uppercase trigger "ARRESTS" is semantically equivalent to the first event cluster's existing "arrested," aligns "RCMP" and "MEN" with the cluster's "leaders" according to the texts, thereby merging D7_E1 into cluster 1 (Relevance Level e).

D.2 Stage 1.2: Argument alignment

An example of argument alignment is given in Table 9. In the Model Input section, we show (1) the excerpt of the target event cluster—complete with its existing triggers and argument lists—and (2) the new candidate event D7_E1, formatted as nested JSON. In the Model Output section, we first detail the trigger merging step, where "ARRESTS" is appended to the cluster's triggers, and then the argument merging step, where each D7_E1 argument ("RCMP," "MEN," and the Bountiful location) is aligned and merged into its corresponding cluster entry. The table concludes with the full updated_event_cluster JSON, demonstrating how the LLM produces a single, coherent multi-document event representation.

D.3 Event Consolidation

An example of event Consolidation is presented in Table 10. The LLM receives as input the aggregated cluster—complete with its seven triggers (five instances of "arrested" plus "ARRESTS") and nine argument mention lists—and, in the Detailed Analysis phase, first selects "arrested" as the single canonical trigger (removing the variant "ARRESTS"). It then evaluates each argument mention group : retaining "leaders," "practicing," "in Canada," "Wednesday," "in Bountiful, B.C.," and "RCMP" as relevant core mentions, and discarding "on Tuesday," "residents," and "charges" as inconsistent or redundant. The table concludes with the final 'updated_event_cluster' JSON, showing a streamlined cluster with one trigger and six core arguments.

Model Input:

Documents and Events (excerpt):
Doc 1: . . . ; Events: . . .
Doc 2: . . . ; Events: . . .
. . .
Existing event clusters (excerpt):

```
[{ "EC_id": 1, "event_ids": ["D1_E1",. . . ],
"triggers": ["arrested","arrested",. . . ],
"arguments": [
{ "roles": ["human participant",. . . ], "mentions": ["leaders",. . . ], "argument_cluster_ids": [. . . ]
},
}, { "EC_id": 2, "event_ids": [. . . ],
"triggers": ["polygamy","polygamy",. . . ], "arguments": [. . . ] },
{ "EC_id": 3, "event_ids": [. . . ],
"triggers": ["arrests"], "arguments": [. . . ] },
{ "EC_id": 4, "event_ids": [. . . ],
"triggers": ["practicing"], "arguments": [. . . ] },
{. . . }
]
]
```

Candidate event (excerpt):

```
{ "event_id": "D7_E1", "trigger": "ARRESTS", "arguments": [
{ "argument_id": ". . . ", "role": "human participant", "mention": "MEN" },
{ "argument_id": ". . . ", "role": "human participant", "mention": "RCMP" },
{ "argument_id": ". . . ", "role": "location",
"mention": "THE RELIGIOUS COMMUNITY OF BOUNTIFUL IN THE BC INTERIOR" },
{. . . }
] }
```

Model Output:

Stage 1: Identify the Most Relevant Cluster

Context Comparison:

– The candidate event (D7_E1) reports the arrest of two men (Winston Blackmore and James Oler) from the polygamous community of Bountiful, British Columbia, on charges of polygamy. This aligns closely with the context of existing event cluster 1 (EC_id: 1), which also describes the arrests of the same individuals in the same location (Bountiful, British Columbia) for practicing polygamy. The context is consistent across all documents in cluster 1 and the candidate event.

Trigger Comparison:

– The candidate event's trigger is "ARRESTS" (uppercase, likely due to headline formatting), which is synonymous with "arrested" and "arrests" found in cluster 1 and cluster 3. The semantic meaning is identical, despite minor formatting differences.

Argument Comparison:

– **Human Participants:** The candidate event mentions "RCMP" (A973) and "MEN" (A894). Cluster 1 includes "RCMP" (A890) and "leaders" (A893, A953), which are semantically similar to "MEN" (the men arrested are the leaders). The roles and mentions align well.

– **Location:** The candidate event mentions "THE RELIGIOUS COMMUNITY OF BOUNTIFUL IN THE BC INTERIOR" (A916), which is semantically equivalent to "in Bountiful, B.C." (A926) and "in Canada" (A897, A936) in cluster 1. All refer to the same geographic area.

– **Non-Human Participants:** The candidate event does not explicitly mention a non-human participant, but the context implies the charges are related to polygamy, which is covered in cluster 1's arguments ("practicing", "charges").

– **Time:** The candidate event does not specify a time, but this is not a significant discrepancy given the strong alignment in other arguments.

Comparison with Other Clusters:

– Cluster 2 (EC_id: 2) focuses on a different event involving "another member" of the FLDS, which does not match the candidate event's context or arguments.

– Cluster 3 (EC_id: 3) uses the trigger "arrests" but involves "authorities" as the human participant and lacks specific details about Bountiful or the leaders. The context is less aligned.

Stage 2: Select Relevance Level

– Same news (arrests of Blackmore and Oler in Bountiful for polygamy).
– Synonymous triggers ("ARRESTS" vs. "arrested").
– Matching human participants (RCMP and leaders/men).
– Matching location (Bountiful, British Columbia).
– No significant differences in time, location, or participants.

This fits **relevance level e**: "Same news, synonymous triggers, and most aspects match."

Merge Decision:

– Merge the candidate event into cluster 1 (EC_id: 1).

JSON Output:

```
{ "decision": "merge", "most_relevant_event_cluster_id": 1 }
```

Table 8: LLM I/O in event-to-cluster assignment

Model Input:

Documents and Events (excerpt):
Doc 1: . . . ; Events: . . .
Doc 2: . . . ; Events: . . .
. . .
Target event cluster (excerpt):

```
['EC_id': 1, 'event_ids': ['D1_E1','D8_E4',. . . ], 'triggers': ['arrested','arrested',. . . ],
'arguments':      ['roles':      ['human_participant',. . . ],    'mentions':      ['leaders',. . . ],
'argument_cluster_ids': [. . . ],
'roles': ['location',. . . ], 'mentions': ['in Canada',. . . ], 'argument_cluster_ids': [. . . ],
'roles': ['time',. . . ], 'mentions': ['Wednesday','Wednesday morning'],
'argument_cluster_ids': [. . . ],
'roles': ['location'], 'mentions': ['in Bountiful , B.C.'], 'argument_cluster_ids': [. . . ],
'roles': ['human_participant'], 'mentions': ['RCMP'], . . . ]]
```

Candidate event (excerpt):

```
{ "event_id": "D7_E1", "trigger": "ARRESTS", "arguments": [
{ "argument_id": ". . . ", "role": "human participant", "mention": "MEN" },
{ "argument_id": ". . . ", "role": "location",
"mention": "THE RELIGIOUS COMMUNITY OF BOUNTIFUL IN THE BC INTERIOR" }, {. . . }] }
```

Model Output:

Trigger Merging:

– **Candidate Trigger**: "ARRESTS" is appended to the existing triggers {"arrested", . . . }, since they are semantically equivalent.

Argument Merging:

1. **Argument D7_E1_A1**: mention "RCMP" matches the cluster's "RCMP" despite differing cluster IDs, so we append its role, mention, and ID.
2. **Argument D7_E1_A2**: mention "MEN" is semantically similar to "leaders"/"residents", so we merge it into that argument entry.
3. **Argument D7_E1_A3**: the location mention "THE RELIGIOUS COMMUNITY OF BOUNTIFUL IN THE BC INTERIOR" aligns with "in Bountiful, B.C.", so we merge it accordingly.


```
{"updated_event_cluster": {
    "EC_id": 1,
    "event_ids": ["D1_E1","D8_E4","D9_E1","D3_E2","D5_E1","D6_E2","D7_E1"],
    "triggers": ["arrested","arrested","arrested","arrested","arrested","ARRESTS"],
    "arguments": [
      {"roles": ["human participant","human participant","human participant","human participant"],
        "mentions": ["leaders","leaders","leaders","MEN"],
        "argument_cluster_ids": ["A953","A893","A893","A894"]},
      {"roles": ["non_human participant"],
        "mentions": ["practicing"],
        "argument_cluster_ids": ["A895"]},
      {"roles": ["location","location","location"],
        "mentions": ["in Canada","in Canada","in Canada"],
        "argument_cluster_ids": ["A897","A897","A936"]},
      {"roles": ["time","time"],
        "mentions": ["Wednesday","Wednesday morning"],
        "argument_cluster_ids": ["A903","A903"]},
      {"roles": ["location","location"],
        "mentions": ["in Bountiful , B.C.","THE RELIGIOUS COMMUNITY OF BOUNTIFUL IN THE BC INTERIOR"],
        "argument_cluster_ids": ["A926","A916"]},
      {"roles": ["human participant","human participant"],
        "mentions": ["RCMP","RCMP"],
        "argument_cluster_ids": ["A890","A973"]},
      {"roles": ["time"],
        "mentions": ["on Tuesday"],
        "argument_cluster_ids": ["A903"]},
      {"roles": ["human participant"],
        "mentions": ["residents"],
        "argument_cluster_ids": ["A893"]},
      {"roles": ["non_human participant"],
        "mentions": ["charges"],
        "argument_cluster_ids": ["A913"]}]}}
```


Table 9: LLM I/O in argument alignment

Model Input:

Documents and Events (excerpt):
Doc 1: . . . ; Events: . . .
Doc 2: . . . ; Events: . . .
. . .
Event cluster (excerpt):

```
'EC_id': 1, 'event_ids': ['D1_E1','D8_E4',. . . ],
'triggers': ['arrested','arrested',. . . ],
'arguments': [
'mentions': ['leaders','leaders','leaders','MEN'],
'mentions': ['practicing'],
'mentions': ['in Canada','in Canada','in Canada'],
'mentions': ['Wednesday','Wednesday morning'],
'mentions': ['in Bountiful , B.C.','THE RELIGIOUS COMMUNITY OF BOUNTIFUL IN THE BC INTERIOR'],
'mentions': ['RCMP','RCMP'],
'mentions': ['on Tuesday'],
'mentions': ['residents'],
'mentions': ['charges']
]
```

Model Output:

Detailed Analysis:

1. **Trigger Selection**: "arrested" is chosen as the single core trigger (the most consistent) and all others are removed.
2. **Argument Relevance and Mention Selection**:

• Argument 1 ({'mentions': ['leaders','leaders','leaders','MEN']}): retain, choose "leaders".
• Argument 2 ({'mentions': ['practicing']}): retain, "practicing".
• Argument 3 ({'mentions': ['in Canada','in Canada','in Canada']}): retain, "in Canada".
• Argument 4 ({'mentions': ['Wednesday','Wednesday morning']}): retain, "Wednesday".
• Argument 5 ({'mentions': ['in Bountiful , B.C.','THE RELIGIOUS COMMUNITY OF BOUNTIFUL IN THE BC INTERIOR']}): retain, "in Bountiful , B.C.".
• Argument 6 ({'mentions': ['RCMP','RCMP']}): retain, "RCMP".
• Argument 7 ({'mentions': ['on Tuesday']}): remove (inconsistent).
• Argument 8 ({'mentions': ['residents']}): remove (redundant).
• Argument 9 ({'mentions': ['charges']}): remove (too vague).

```
{
  "EC_id": 1,
  "event_ids": ["D1_E1","D8_E4","D9_E1","D3_E2","D5_E1","D6_E2","D7_E1"],
  "triggers": ["arrested"],
  "arguments": [
    {"mentions": ["leaders"]},
    {"mentions": ["practicing"]},
    {"mentions": ["in Canada"]},
    {"mentions": ["Wednesday"]},
    {"mentions": ["in Bountiful , B.C."]},
    {"mentions": ["RCMP"]}
  ]
}
```

Table 10: LLM I/O in event consolidation