Dynamic Chain-of-thought for Low-Resource Event Extraction

Anonymous ACL submission

Abstract

Event extraction is a critical task that encompasses several interdependent sub-tasks. The complex interplay among these sub-tasks renders the overall task particularly challenging, particularly in low-resource scenarios where data availability is limited. However, the inherent logical coherence among these sub-tasks presents a promising avenue for addressing these challenges. This logical structure is particularly advantageous in low-resource settings, as it facilitates a deeper understanding of the tasks by the model and reduces dependence on available data. Building on this observation, we explore the logical structure of event extraction with a focus on low-resource scenarios. Specifically, we propose a three-step Chain-of-Thought pattern to guide the model through the logical reasoning process. Additionally, we design a step-wise navigator that dynamically provides the model with relevant knowledge. Empirical results demonstrate the robustness of our approach in low-resource event extraction.

1 Introduction

Event extraction (EE) is an important but challenging task in information extraction. Event extraction is typically decomposed into multiple interdependent sub-tasks, which involve trigger identification, event type classification, argument identification, and role classification.

Traditional methods (Lin et al., 2018; Wadden et al., 2019; Yang et al., 2019; Wang et al., 2019; Ma et al., 2020; Zhang et al., 2020; Liu et al., 2020; Li et al., 2020) rely heavily on extensively annotated datasets for training, which are often unavailable or costly to obtain in real-world applications. While sampling from large language models has been proposed as a solution (Wang et al., 2023a; Ma et al., 2024), this approach can be expensive and unstable. Thus developing effective low-resource event extraction techniques is imperative in this area.

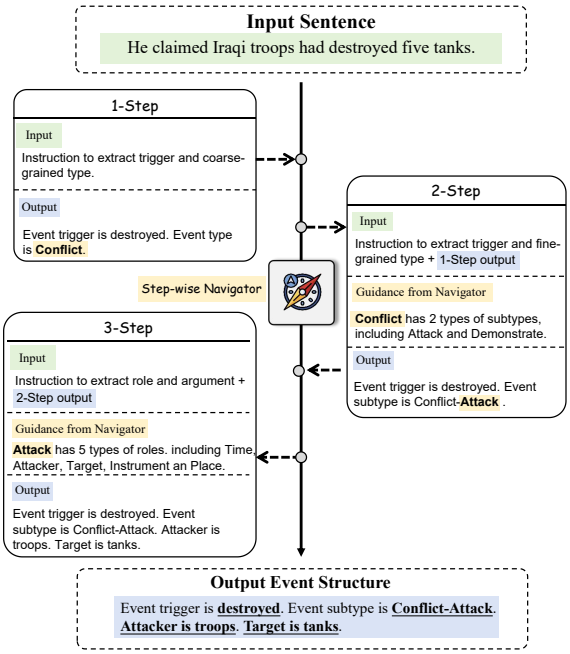

Figure 1: An illustration of our dynamic chain of thought method for event extraction.

Recent studies have explored sequence-to-sequence methods for low-resource event extraction. In contrast to pipeline methods (Lin et al., 2018; Wadden et al., 2019; Yang et al., 2019), these methods are typically end-to-end, some relying on structured language (Lu et al., 2022, 2021) and others using natural language prompts (Hsu et al., 2022; Ma et al., 2022; Zhao et al., 2023). However, they overlook the logical structure inherent in the sub-tasks of event extraction, which could potentially enhance the extraction process.

Intuitively, we propose that Chain-of-Thought (CoT) reasoning (Wei et al., 2022; Fei et al., 2023; Trivedi et al., 2023) can significantly enhance low-resource event extraction. From the task perspective, event extraction follows a clear logical structure: recognizing a coarse-grained event type naturally leads to the detection of fine-grained event

types. Once the event type is determined, identifying the corresponding roles becomes easier. This logical reasoning simplifies the extraction process by narrowing the search space, reducing irrelevant choices, and focusing on the most likely outcomes. From the data perspective, the class imbalance problem is particularly significant in low-resource tasks, with smaller event categories often being overlooked due to insufficient data. The hierarchical classification approach—first classifying into coarse-grained event types, then into fine-grained event types—can effectively mitigate the class imbalance problem. Without hierarchical classification, the model tends to be concentrated on a few dominant classes, neglecting the smaller ones.

Based on our analysis, we propose a dynamic Chain-of-Thought framework enhanced by a step-wise navigator to address these challenges. As shown in Figure 1, the framework follows the inherent logical structure of the sub-tasks. We first design a three-step reasoning prompt for inferring the coarse-grained event type, fine-grained event type, and roles in a step-by-step manner. Secondly, we propose a step-wise navigator to attach specialized guidance to each step for navigating the most possible reasoning path to improve the correctness of each reasoning step. Specifically, the step-wise navigator extracts key components from the output of the previous step and provides corresponding guidance to guide the next reasoning path. The main idea of this framework is to effectively utilize the limited data by leveraging the model's reasoning ability, thereby enhancing performance even in low-resource scenarios.

Experimental results show that our model outperforms competitive models in low-resource scenarios. Deep analysis indicates that the proposed framework is capable of eliciting the inherent logical structure of events, leveraging reasoning capabilities to improve performance even with limited data.

Our contributions can be summarized as follows:

- We investigate the inherent logic of event extraction tasks to enhance the model's reasoning capabilities in low-resource scenarios and design a dynamic chain of thought framework.
- We design a specific step-wise navigator to guide dynamic chain of thought prompting, providing the model accurate guidance in low-resource scenarios.
- Comprehensive empirical studies show the effectiveness of the proposed method in low-resource event extraction.

2 Related Works

Event Extraction has been tackled through several mainstream approaches. Traditionally, researchers have relied on classification-based sequence labeling models within pipeline frameworks (Lin et al., 2018; Wadden et al., 2019; Yang et al., 2019; Wang et al., 2019; Ma et al., 2020; Zhang et al., 2020). However, pipeline models often suffer from error propagation. To address this, several studies (Yang and Mitchell, 2016; Nguyen et al., 2016; Lin et al., 2020) have proposed integrating global features for joint learning of both event triggers and arguments.

In recent years, researchers have turned their attention to low-resource event extraction, proposing generation-based models within end-to-end frameworks (Paolini et al., 2021; Li et al., 2021; Lu et al., 2021; Liu et al., 2022a; Du et al., 2022; Wang et al., 2023b). For example, Text2Event(Lu et al., 2021) introduces a sequence-to-structure generation paradigm for event extraction, while UIE(Lu et al., 2022) employs a text-to-structure approach with a Structural Extraction Language (SEL) to unify various information extraction tasks. However, these approaches may face limitations due to the mismatch between the generated structural output and the training objectives of pre-trained models such as T5 and BART.

Therefore, another line of work focuses on natural language prompts (Liu et al., 2022b; Hsu et al., 2022; Zhao et al., 2023). For instance, DEGREE(Hsu et al., 2022) frames event extraction as a conditional generation task, employing label-guided prompts to convert events into template-based sentences. However, manually designing templates is time-consuming. DemoSG(Zhao et al., 2023) introduces a demonstration-based event extraction paradigm, using annotated data and label semantics to guide model generation.

Building on these advances, our work takes a more comprehensive approach by considering both the task structure and data characteristics. Based on the inherent logical structure of event extraction, we propose chain-of-thought prompting with multi-step reasoning. Additionally, we introduce a step-wise navigator to automatically adjust the input prompts, minimizing the need for manual intervention.

3 Method

3.1 Task Definition

The event extraction(EE) task consists of two subtasks: 1) event detection(ED), in which the model extracts trigger text and predicts the event type; and 2) event argument extraction(EAE), in which the model extracts arguments and predicts the role of each argument given event triggers.

Formally, given a set of sentences $X = \{x_i\}_{i=1}^{|X|}$, the event extraction task aims to extract a set of event records $S = \{S_j\}_{j=1}^{|X|}$. Herein, S_j is a natural language sentence, consisting of 1) a trigger word T_j; 2) a set of arguments $A_j = \{A_{jk}\}_{k=1}^{|A_j|}$ and 3) a set of roles $R_j = \{R_{jk}\}_{k=1}^{|R_j|}$. Each trigger T_j corresponds to an event type E_j. Each argument corresponds to an event role, thus $|A_j| = |R_j|$.

In this paper, we model event extraction as a language generation task, in which a natural language text x_i is given as input and outputs the event record S_j. As illustrated in Figure 2, the model takes the natural language text as the input and outputs an event record, where "destroyed", "Conflict-Attack", "troops", and "tanks" are denoted as the event trigger, event sub-type, attacker and target, respectively.

3.2 Dynamic CoT for Event Extraction

By investigating the inherent logical structure of the event extraction, we design a dynamic Chain-of-thought(CoT) framework supported by a step-wise navigator. In this framework, we employ three-step prompts for different sub-tasks to obtain the complete event record in a step-by-step manner.

3.2.1 Chain-of-Thought Design

The event extraction task consists of two subtasks: event detection and event argument extraction. Upon further analysis, we observe a clear logical structure in which coarse-grained event types categorize related fine-grained event types. Each event type has a defined set of participant roles. For instance, events such as ***birth***, ***death***, and ***marriage*** belong to the coarse-grained event type ***Life***, whereas events like ***attack*** and ***demonstrate*** fall under the coarse-grained event type ***Conflict***. The ***Conflict-Attack*** event type includes five roles: *"Attacker"*, *"Target"*, *"Instrument"*, *"Time"* and *"Place"*. This hierarchical structure is illustrated in Figure 2(c).

Consequently, the first step in our Chain-of-Thought(CoT) process aims to identify the trigger word and classify events into these coarse-grained event types. In the second step, we further divide these coarse-grained event types to identify the fine-grained event types. After determining the event type, the third step in our CoT process focuses on role classification, where we identify the roles present in the event and their corresponding arguments.

3.2.2 Step-wise Navigator Module

In the section 3.1, we introduce the logical structure of the event extraction task. We further design the step-wise navigator to guide the reasoning path at each step. Due to the hierarchical structure having different search paths, the role of the stepwise navigator is to provide specific guiding information to the model, thereby narrowing the search space and selecting the most probable path.

Specifically, we first construct a hierarchical knowledge base based on the definitions in the dataset.[1] In this knowledge base, we define coarse-grained and fine-grained event types and provide detailed descriptions for each event type. Additionally, since roles are inherent attributes of events, we also provide the roles and their definitions for each event. In the second step, the step-wise navigator module analyzes the previous step's results and adjusts the model's input prompt for the next step based on the predefined knowledge base. We show a case in Figure 2(c).

This mechanism ensures that the model consistently moves along the most probable reasoning path, minimizing the accumulation of errors that often arise in multi-step processes. This is beneficial for low-resource scenarios.

3.2.3 Dynamic CoT Framework

Based on the CoT design and the step-wise navigator, we develop the dynamic templates. Our model extracts event structure in three steps, where each step's input and output follow a specific template. The input template of each step consists of the sentence x_i, a dynamic prompt $p_i \in P_i$ and a question $q_i \in Q_i$. Herein, we divide the task into three steps, thus formulating $p_i =< p_i^1, p_i^2, p_i^3 >$, $q_i =< q_i^1, q_i^2, q_i^3 >$. Specifically, the input prompt p_i is dynamic because it not only concatenates the response from the previous step but also adds guidance derived from the navigator. The output

[1]Definition from ACE English Annotation Guidelines for Events V5.4.3:https://www.ldc.upenn.edu/collaborations/past-projects/ace/annotation-tasks-and-specifications

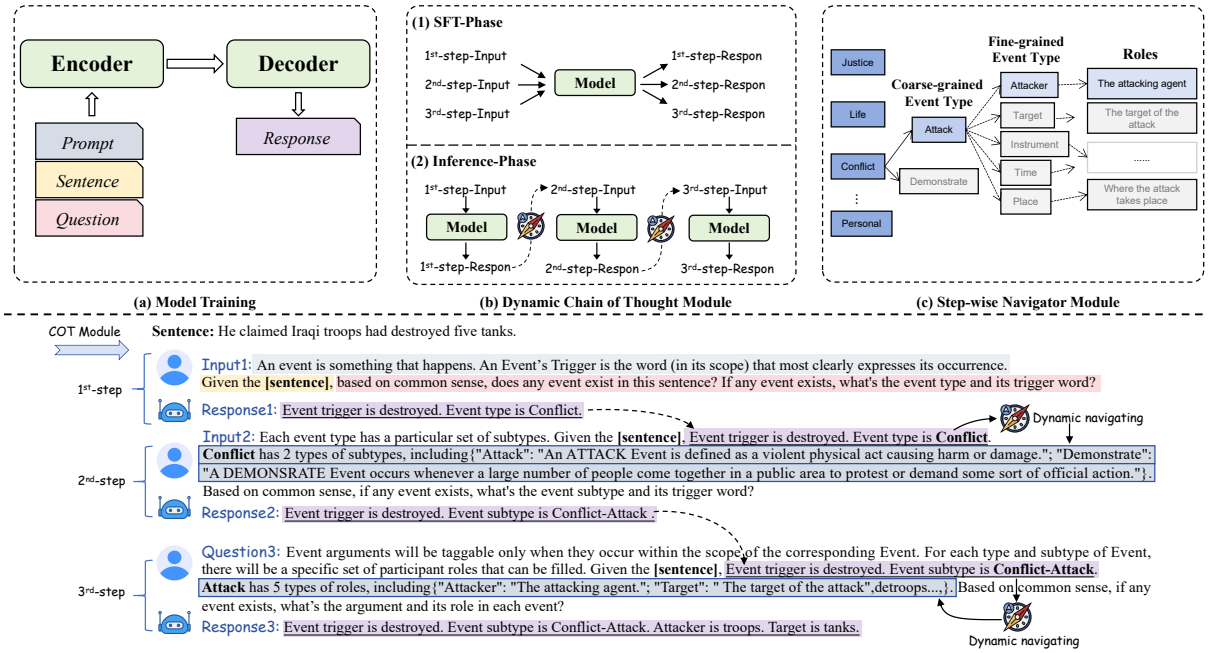

Figure 2: An illustration of the proposed method.

template of each step also follows a particular format, allowing us to use a deterministic algorithm to parse the event type from the previous step's output. Given the event type parsed from the previous step, the navigator will attach the corresponding guidance to the input template for the next reasoning step, thereby dynamically adjusting our input template based on the inference results.

The three-step templates we constructed are as follows:

Step 1 The first step involves extracting event triggers and classifying them into coarse-grained event types. The dynamic input template for this step includes a sentence x_i, a prompt about event extraction p_i^1 and question q_i^1. We construct the following input template: The output template for this step is: *"Event trigger is <tri>. Event type is <type>"*. If there is no event, special placeholders will be generated(*i.e. "Event trigger is [trigger]. Event type is [type]"*). If there are multiple events, different events are separated using the special placeholder "[SEP]". This step can be formulated as follows:

$$r_i^1 = LLM(x_i, p_i^1, q_i^1) \quad (1)$$

Where, p_i^1 and q_i^1 denote the prompt and question in step 1.

Step 2 The second step involves identifying the event trigger and classifying it into fine-grained event types. The dynamic input template for this step includes a sentence x_i, a prompt about event type p_i^2 and question q_i^2. The prompt for this step includes the description of the inferred coarse-grained event type and the definitions of the corresponding fine-grained event types. For example, as illustrated in Figure 2, the output from the previous step is parsed to identify the coarse-grained event type as ***Conflict***. The navigator indicates that ***Conflict*** includes two fine-grained event types: ***Attack*** and ***Demonstrate*** with definitions for each of them. We construct the following input template:

The output template for this step is: *"Event trigger is <tri>. Event subtype is <sub-type>"*. If there is no event, special placeholders will be generated(*i.e. "Event trigger is [trigger]. Event subtype is [event-type]"*). This step can be formulated as follows:

$$r_i^2 = LLM(x_i, r_i^1, p_i^2, q_i^2) \quad (2)$$

Step 3 The third step aims to extract the arguments and classify them into event roles. The dynamic input template for this step includes a sentence x_i, a prompt about event roles p_i^3 and question q_i^3. The prompt for this step includes the inherent roles of the event type and their definitions. We construct the following input template:

The output template for this step is: *"Event trigger is <tri>. Event subtype is <sub-type>. <Role> is <Arg>"*. When there is no valid role for the

queried event type, the special placeholder "[SEP]" is generated(*i.e. "Event trigger is [trigger]. Event subtype is [event-type]. [Role] is [Arg]"*). This step can be formulated as follows:

$$r_i^3 = LLM(x_i, r_i^2, p_i^3, q_i^3) \quad (3)$$

3.3 Training and Inference

3.3.1 Training

During the training phase, we follow the CoT three-step strategy. Given a passage, we construct three parallel data instances. For each step, the input includes the ground truth response from the previous step. As illustrated in Figure 2:

- For the first step, given the passage and prompt, the model is expected to replace *"<tri>"* with the gold trigger *"destroyed"* and replace *"<type>"* with the gold event type *"Conflict"*.
- For the second step, given the passage, prompt, and the ground truth response from the first step, the model is expected to replace *"<sub-type>"* with the gold subtype *"Conflict-Attack"*.
- For the third step, given the passage, prompt, and the ground truth response from the second step, the model is expected to replace *"<role>"* with the gold roles *"Attacker"* and replace *"<arg>"* with the gold arguments *"troops"*.

The goal is to maximize the objective text T probability given the input text X, prompts P and questions Q. Therefore, we optimize the negative log-likelihood loss function:

$$\mathcal{L} = -\frac{1}{|\tau|} \sum_{(X,T)\in\tau} \log p(T|X;P;Q;\theta) \quad (4)$$

where θ is the model parameters, P is the dynamic CoT prompts at each step, Q is the question at each step, and (X, T) is a (*input,output*) pair in training set τ.

3.3.2 Inference

In the inference phase, we also follow the CoT three-step strategy, but in a sequential manner. Unlike the training phase, the input for each step is the model's prediction from the previous step, rather than the ground truth.

Dataset	Split	# Sentence
ACE05-EN	Train	17,172
	Validation	923
	Test	832
ACE05-EN$^+$	Train	19,216
	Validation	901
	Test	676

Table 1: Distribution of data across different datasets.

Given a passage, the model conducts event extraction through a three-step process. In the first step, it predicts the event trigger and event type. In the second step, the model's response from the first step is utilized by a step-wise navigator to dynamically adjust the input template, guiding the prediction of the event subtype. In the third step, the response from the second step is incorporated into the input template, and the model predicts the roles and arguments to generate the final event records.

4 Experiment

This section introduces the dataset and experimental setup employed to validate our method. Subsequently, we present the experimental results and analyze the findings.

Dataset As shown in Table 1, we assess our method using two benchmarks, ACE05-EN and ACE05-EN$^+$, derived from the ACE2005 dataset, which features 33 event types and 22 entity roles. Our data splitting and preprocessing procedures align with those used in prior studies by Wadden et al. (2019) and Lin et al. (2020). Additionally, for low-resource settings, we adopt a data sampling strategy following Hsu et al. (2022), conducting experiments with various proportions of training data.[2].

Baselines (1) **OneIE** (Lin et al., 2020): This state-of-the-art classification model excels in high-resource settings, utilizing global feature extraction to optimize event extraction outcomes. (2) **BERT_QA** (Du and Cardie, 2021): This classification model approaches event extraction as a Question Answering (QA) task, enabling end-to-end argument extraction. (3) **TANL** (Paolini et al., 2021): This generative method leverages translation between augmented natural languages to

[2]Detailed statistics presented in Table 5

Method	Event Detection F1-Score (%) ↑											
	ACE05-EN						ACE05-EN$^+$					
	1%	3%	5%	10%	20%	30%	1%	3%	5%	10%	20%	30%
BERT-QA	20.5	44.2	50.7	53.1	60.9	61.8	–	–	–	–	–	–
OneIE	40.9	50.1	58.5	60.4	65.3	65.5	39.0	52.5	60.6	58.1	66.5	66.4
Text2Event	14.2	35.2	46.6	47.0	55.6	60.7	15.7	38.4	43.9	46.3	56.5	62.0
TANL	34.1	48.1	53.4	54.8	61.8	61.6	30.3	50.9	53.1	55.7	60.8	61.7
DEGREE	**55.1**	**62.8**	**63.8**	**66.1**	64.4	64.4	**56.4**	**62.5**	**61.1**	62.3	62.5	67.1
DICE	22.9	44.5	53.1	53.4	61.5	64.1	18.9	41.7	48.7	54.0	63.8	66.0
FlanT5-Large	25.9	44.0	50.3	54.8	60.1	63.9	29.1	40.3	48.8	51.3	58.2	61.7
Llama2-7b	16.5	20.1	28.6	43.0	48.1	51.0	24.1	35.8	25.1	23.1	35.0	31.4
Ours	36.5	48.5	54.8	61.6	62.6	65.6	37.0	50.8	56.5	63.0	64.5	66.5
Ours(Type)	45.5	58.7	59.8	64.4	**66.3**	**67.2**	48.8	57.2	60.9	**65.7**	**66.8**	**67.4**

Method	Event Argument Classification F1-Score (%) ↑											
	ACE05-EN						ACE05-EN$^+$					
	1%	3%	5%	10%	20%	30%	1%	3%	5%	10%	20%	30%
BERT-QA	4.7	14.7	21.6	25.5	30.3	35.3	–	–	–	–	–	–
OneIE	9.4	19.1	24.4	26.5	41.7	43.3	10.4	20.6	29.7	35.5	46.7	48.0
Text2Event	3.9	12.2	19.1	24.9	32.3	39.2	5.7	16.5	21.3	26.4	35.2	42.1
TANL	8.5	17.2	24.7	29.0	34.0	39.2	8.6	22.3	30.4	29.2	34.6	39.0
DEGREE	13.1	26.1	27.6	**42.1**	40.7	44.0	16.0	26.4	29.9	39.5	41.3	48.5
DICE	5.4	16.5	21.1	26.2	34.3	36.5	5.2	19.5	21.7	27.4	36.5	46.3
FlanT5-Large	8.6	16.7	23.3	24.3	32.5	38.2	8.7	17.5	24.4	25.9	35.0	41.1
Llama2-7b	2.6	6.7	5.0	10.3	21.3	25.5	7.7	14.1	10.5	13.8	15.6	27.4
Ours	15.9	24.6	28.8	34.1	40.9	44.8	16.8	28.2	32.6	39.3	43.4	48.2
Ours(Type)	**20.6**	**30.5**	**33.3**	38.6	**42.3**	**47.1**	**23.9**	**34.1**	**36.8**	**43.4**	**48.2**	**51.0**

Table 2: Trigger classification F1-scores and argument classification F1-scores (%) across datasets with different training data."Ours(type)" refer to the method enhanced by the label semantics of coarse-grained event types following Hsu et al. (2022).

.

solve structured prediction tasks, including event extraction. (4) **Text2Event** (Lu et al., 2021): This generative approach converts event records into a structured tree, proposing a sequence-to-structure model. (5) **DEGREE** (Hsu et al., 2022): This generative method manually designs templates for various event types, using label semantics to guide the model to capture entity relationships for event extraction. (6) **DICE** (Ma et al., 2023): This generative method designs a Mention Identification module to improve trigger word recognition and argument extraction. (7) **FlanT5** (Chung et al., 2022): This generative approach is the base model of our method, solving event extraction without employing CoT prompting. (8)**LLAMA** (Touvron et al., 2023): This generative method employs Lora Fine-tuning to specifically tailor large language models for the event extraction task.

Evaluation Metrics Our criteria align with those used in previous studies Wadden et al. (2019); Lin et al. (2020); Hsu et al. (2022). We employ two main metrics: (1) Trigger F1-score, where a trigger is correctly identified (Tri-I) if its offset corresponds to the label and correctly classified (Tri-C) if its event type also matches. (2) Argument F1-score, which considers an argument correctly identified (Arg-I) if both its offset and event type align with the label, and correctly classified (Arg-C) if its role is also accurate.

Implementations We use Flan-T5-large as our base model. AdamW (Loshchilov and Hutter, 2018) is used as the optimizer, with a learning rate of $2e-5$. The gradient accumulation steps are set to 4, and the number of epochs is set to 100 with an early stopping mechanism. All experiments are conducted on a single 3090 GPU. The experimental results are obtained by averaging three runs with random initialization. "Ours" relies on dynamic prompts and simple templates to facilitate logical reasoning, while DEGREE's templates depend on label semantics. To ensure a fairer comparison with DEGREE, we also evaluate our model enhanced with the label semantics of coarse-grained event types, referred to as "Ours(type)".

4.1 Main Results

We compare our method with several strong baselines. These baselines can be roughly separated into two categories classification methods(i.e., BERT-QA, OneIE) and generative methods (Text2Event, TANL, DEGREE, DICE, Flan-T5, LLAMA). Table 2 shows the results on two different datasets. Following the consensus of previous work, we prioritize the Argument classification F1-score as the key metric for comparing model performance.

It can be observed that classification-based methods, such as OneIE, demonstrate competitive performance in the event detection(ED) task when sufficient training data is available (e.g., 30%). However, their performance degrades significantly under low-resource conditions (e.g., 1%). Furthermore, these methods exhibit limitations in event argument extraction(EAE). In contrast, generative approaches have shown strong capabilities in both ED and EAE, particularly in low-resource scenarios. Specifically, the state-of-the-art method DEGREE enhanced with design prompt outperforms other baselines, which shows the effectiveness of the prompt strategy. Notably, Llama2-7b, a large language model, does not perform well in either task. This may be due to the complex task definition involved. In the ED task, our method achieves competitive results using only 25% of the data compared to DEGREE, significantly reducing compute resource requirements. Furthermore, in the EAE task, our method consistently outperforms other strong baselines, highlighting the effectiveness of the dynamic chain-of-thought framework in low-resource scenarios. The results demonstrate that leveraging the inherent logic of event extraction sub-tasks allows for more efficient use of limited data, mitigating the performance degradation typically caused by data scarcity.

4.2 Ablation Study

We conduct an ablation study to analyze the contributions of different components in our model to the low-resource event extraction task. Results are shown in Table 3. Firstly, removing all the Chain-of-Thought design leads to the largest performance drop, highlighting its key role in reasoning through complex event structures. Furthermore, dividing the ED task into two steps using the CoT strategy is beneficial. Removing the classification of coarse-grained event types leads to a moderate decline in performance. Finally, the Step-wise navigator improves event extraction by providing adaptive guidance, with a notable impact. This evidence highlights that all components are highly beneficial for low-resource event extraction by guiding the model through step-by-step reasoning to investigate its inherent logic and effectively mitigate the lack of insufficient labeled data.

Method	Tri./Arg. F1-Score (%) ↑	
	ACE05-EN	ACE05-EN+
Ours	**61.6/34.1**	**63.0/39.3**
w/o Chain-of-thought	54.8/24.3	51.3/25.9
w/o Event type	56.3/30.9	59.2/35.6
w/o Step-wise navigator	55.6/30.1	57.1/32.8

Table 3: Ablation study for the key components of the dynamic chain of thought framework with 10% training data on ACE05-EN and ACE05-EN$^+$.

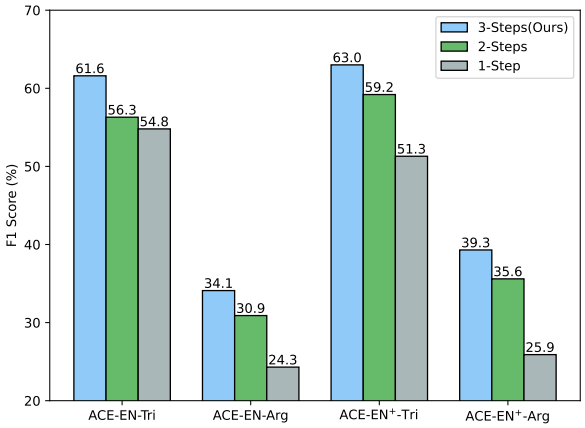

Figure 3: Influence of each step in the dynamic chain of thought design.

5 Analysis and Discussion

In this section, we provide a detailed analysis and discussion to highlight the importance of various factors within the proposed model when applied to a low-resource scenario, where only 10% of the training data is available.

5.1 Influence of Dynamic Chain of thought

As shown in Figure 3, the results reveal that the dynamic chain of thought framework contributes significantly to performance improvements in low-resource event extraction(EE). Its effectiveness by dynamically adjusting the reasoning steps based on the step-wise navigator module, allowing for more accurate and adaptive decision-making. The 3-step Chain-of-thought(CoT) outperforms both 2-step and 1-step methods across all tasks and datasets.

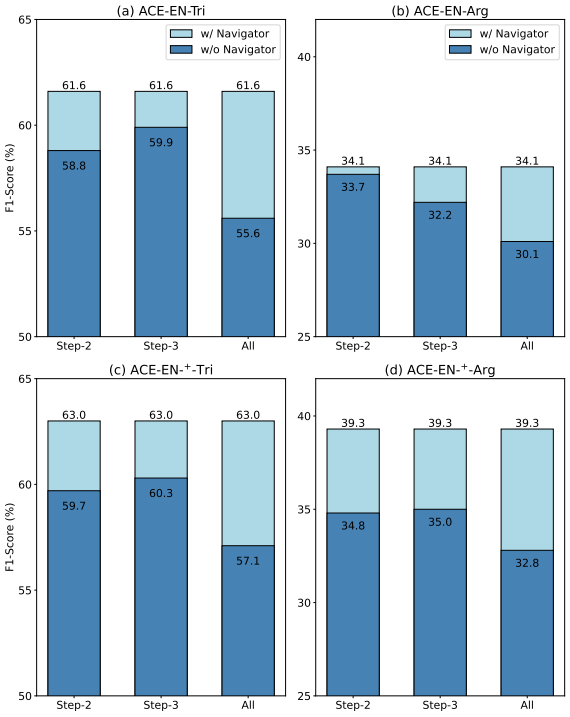

Figure 4: Impact of guidance in each step provided by step-wise navigator.

This indicates that investigating the inherent logical consistency of EE sub-tasks provides the model with a deeper and more comprehensive understanding of event triggers and arguments, which leads to improved performance in low-resource scenarios.

5.2 Effects of Step-wise Navigator

The step-wise navigator is crucial in the dynamic CoT strategy, as it supplies relevant information to the model and guides it along the next reasoning path. To evaluate its impact, we analyze its effect at various stages of the reasoning process.

As shown in Figure 4, the results across datasets demonstrate that the step-wise navigator effectively improves the F1 score in both the second and third steps. The most significant gains occurred in the "All-step", suggesting that the navigator plays an essential role in maintaining coherence and enhancing decision-making throughout the chain of reasoning steps. The navigator's role becomes even more crucial as the complexity of the reasoning task increases, where intermediate decisions (Step 2 and Step 3) require more careful alignment and logical consistency. By guiding the model through each reasoning step, the navigator helps reduce error propagation and enhance the model's capacity to make accurate predictions in later steps.

Method	**Base Model**	F1-Score (%) ↑	
		Trigger	Argument
In-context learning			
Few shot(2-shot)	GPT4o-mini	16.5	5.4
Vanilla CoT	GPT4o-mini	14.5	6.2
Dynamic CoT	GPT4o-mini	16.6	8.7
Supervised fine-tuning with 1% data			
Flan-T5	FlanT5-Large	25.9	8.6
Ours	FlanT5-Large	**36.5**	**15.9**

Table 4: Results of different methods using GPT4o-mini and Flan-T5-Large.

5.3 Impact of Training Paradigms

Although Chain-of-Thought (CoT) prompting has demonstrated significant capabilities in reasoning tasks with a large language model, it also exhibits substantial limitations in low-resource event extraction(EE)(Gao et al., 2023; Han et al., 2023). We analyze our method on a large language model(LLM) using few-shot (2-shot), vanilla CoT ('think step by step'), and our dynamic CoT strategy. Notably, vanilla CoT does not show significant improvement compared to the few-shot method, suggesting that random reasoning paths in the large model cannot help with limited resources. Besides, our method achieves higher performance than both the few-shot and vanilla CoT methods, indicating the effectiveness of our CoT design. Compared to direct inference from the LLM, the fine-tuning method using a small model achieves higher performance. This suggests that LLMs still have limitations in this task, possibly due to the complex definitions involved in EE sub-tasks. Instead, our method with only 1% data outperforms all baselines by leveraging the dynamic CoT template that effectively guides the model through event structures and enhances reasoning despite limited data.

6 Conclusion

In this study, we present a novel approach for low-resource event extraction by leveraging the inherent logic in the event extraction task through a dynamic Chain-of-Thought reasoning pattern. Additionally, we propose a step-wise navigator to dynamically provide relevant knowledge based on the previous step and minimize distractions from irrelevant options for focusing on the most likely reasoning paths. The empirical results validate the effectiveness of our approach, demonstrating harnessing the inherent logic within the sub-tasks can significantly improve performance in low-resource settings.

Limitations

Our method introduces a three-step chain-of-thought(CoT) template based on the inherent logic of event extraction sub-tasks, effectively leveraging limited data to elicit the model's reasoning ability to address the challenge of generalizing to complex event records. However, there are some limitations in our approach. Firstly, since we employ a pipeline model, there is a risk of error propagation, where mistakes made in earlier steps can affect the final results. Additionally, our method relies on a manually designed CoT template, which may limit generalization. Although we apply a step-wise navigator to alleviate this, it still remains a rule-based design, limiting its flexibility in more diverse contexts. Future work could focus on exploring automated approaches for generating diverse CoT templates to further enhance generalization in a low-resource scenario. Additionally, Our method achieves strong performance on the English dataset ACE05-EN; however, its extension to other English datasets and multilingual datasets remains limited. Expanding the applicability of our approach to a broader range of datasets will be a key focus of our future research.

References

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, pages 837–840. Lisbon.

Xinya Du and Claire Cardie. 2021. Event extraction by answering (almost) natural questions. *Preprint*, arXiv:2004.13625.

Xinya Du, Sha Li, and Heng Ji. 2022. Dynamic global memory for document-level argument extraction. *Preprint*, arXiv:2209.08679.

Hao Fei, Bobo Li, Qian Liu, Lidong Bing, Fei Li, and Tat-Seng Chua. 2023. Reasoning implicit sentiment with chain-of-thought prompting. *Preprint*, arXiv:2305.11255.

Jun Gao, Huan Zhao, Changlong Yu, and Ruifeng Xu. 2023. Exploring the feasibility of chatgpt for event extraction. *arXiv preprint arXiv:2303.03836*.

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*.

I Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, Nanyun Peng, et al. 2021. Degree: A data-efficient generation-based event extraction model. *arXiv preprint arXiv:2108.12724*.

I-Hung Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. DEGREE: A data-efficient generation-based event extraction model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1890–1908, Seattle, United States. Association for Computational Linguistics.

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.

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.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2018. Nugget proposal networks for Chinese event detection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1565–1574, Melbourne, Australia. Association for Computational Linguistics.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651, Online. Association for Computational Linguistics.

Xiao Liu, Heyan Huang, Ge Shi, and Bo Wang. 2022a. Dynamic prefix-tuning for generative template-based event extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Xiao Liu, Heyan Huang, Ge Shi, and Bo Wang. 2022b. Dynamic prefix-tuning for generative template-based event extraction. In *Annual Meeting of the Association for Computational Linguistics*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. 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 2795–2806, Online. Association for Computational Linguistics.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland. Association for Computational Linguistics.

Jie Ma, Shuai Wang, Rishita Anubhai, Miguel Ballesteros, and Yaser Al-Onaizan. 2020. Resource-enhanced neural model for event argument extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3554–3559, Online. Association for Computational Linguistics.

Mingyu Derek Ma, Alexander K. Taylor, Wei Wang, and Nanyun Peng. 2023. Dice: Data-efficient clinical event extraction with generative models. *Preprint*, arXiv:2208.07989.

Mingyu Derek Ma, Xiaoxuan Wang, Po-Nien Kung, P. Jeffrey Brantingham, Nanyun Peng, and Wei Wang. 2024. Star: Boosting low-resource information extraction by structure-to-text data generation with large language models. *Preprint*, arXiv:2305.15090.

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. *Preprint*, arXiv:2202.12109.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cícero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. *ArXiv*, abs/2101.05779.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *Preprint*, arXiv:2212.10509.

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.

Bo Wang, Heyan Huang, Xiaochi Wei, Ge Shi, Xiao Liu, Chong Feng, Tong Zhou, Shuaiqiang Wang, and Dawei Yin. 2023a. Boosting event extraction with denoised structure-to-text augmentation. *Preprint*, arXiv:2305.09598.

Xiaozhi Wang, Ziqi Wang, Xu Han, Zhiyuan Liu, Juanzi Li, Peng Li, Maosong Sun, Jie Zhou, and Xiang Ren. 2019. HMEAE: Hierarchical modular event argument extraction. 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 5777–5783, Hong Kong, China. Association for Computational Linguistics.

Xingyao Wang, Sha Li, and Heng Ji. 2023b. Code4struct: Code generation for few-shot event structure prediction. *Preprint*, arXiv:2210.12810.

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

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299, San Diego, California. 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.

Zhisong Zhang, Xiang Kong, Zhengzhong Liu, Xuezhe Ma, and Eduard Hovy. 2020. A two-step approach for implicit event argument detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7479–7485, Online. Association for Computational Linguistics.

Gang Zhao, Xiaocheng Gong, Xinjie Yang, Guanting Dong, Shudong Lu, and Si Li. 2023. DemoSG: Demonstration-enhanced schema-guided generation for low-resource event extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1805–1816, Singapore. Association for Computational Linguistics.

A Dateset Detail

In this study, we use the ACE2005[3]English dataset (Doddington et al., 2004). For low-resource scenarios, we adopt the data partitioning strategy from the DEGREE (Hsu et al., 2021) experimental setup. The training data is split by documents, offering a more realistic approach compared to instance-based splitting. Table 5 provides the statistics for both ACE05-E and ACE05-E+.

B Prompt Detail

We analyze our prompting strategy on a large language model using three approaches: few-shot (2-shot), vanilla CoT (i.e., 'think step by step'), and our dynamic CoT strategy. Tables 7, 8, and 9 separately present the prompts used for each approach. Since the few-shot and vanilla CoT methods lack sufficient knowledge of event extraction, we incorporate additional knowledge and examples to provide a more comprehensive understanding of the EE task. For dynamic CoT, we use the LangChain framework to conduct multi-round conversations and apply the GPT model to implement the CoT prompts we designed.

C Implementation Detail

This section describes the implementation details of our method. In the main experiment table 2, we present two results for our method: 'Ours' and 'Ours (type).' The 'Ours' approach uses only the sentence, the prompts, and the questions as input, while 'Ours (type)' refers to a type-enhanced version that also incorporates coarse-grained event types.

To ensure a fairer comparison, we follow a similar experimental setup to DEGREE (Hsu et al., 2022), but our type-enhanced method is more efficient and resource-saving. Specifically, given a passage, annotated coarse-grained event types are treated as positive examples. During training, we sample m unrelated event types as negative examples, with m typically set to 3 in our experiments. During inference, each coarse-grained event type is extracted separately. Our type-enhanced approach tests on 8 coarse-grained event types, whereas DEGREE tests on 33 fine-grained types, which provides DEGREE with more semantic information and increases the pressure for our model. This explains why our approach initially performs worse than DEGREE with less than 5% of the data. However, as more training examples become available, models can learn more sophisticated features from the data, allowing our approach to gradually surpass DEGREE once more than 10% of the data is used. Table 6 shows the results of high-resource event extraction. We can observe that by using only 25% of the data, our method achieves competitive results with DEGREE.

[3]https://catalog.ldc.upenn.edu/LDC2006T06

Dataset	Split	Number of Sentences
ACE05-EN	Train	17,172
	Train-30%	5429
	Train-20%	3467
	Train-10%	1688
	Train-5%	649
	Train-3%	451
	Train-1%	103
	Val.	923
	Test	832
ACE05-EN$^+$	Train	19,216
	Train-30%	6159
	Train-20%	3834
	Train-10%	1915
	Train-5%	628
	Train-3%	434
	Train-1%	92
	Val.	901
	Test	676

Table 5: Data Statics. Our data splitting for low resources aligns with those used in prior studies by (Hsu et al., 2021).

Model	ACE05-E		ACE05-E+	
	Tri-C	Arg-C	Tri-C	Arg-C
DyGIE++	70.0	50.0	-	-
Joint3EE	69.8	52.1	-	-
EE_QA	72.4	53.3	-	-
MQAEE	71.7	53.4	-	-
TANL	68.4	47.6	-	-
BART-Gen*	71.1	53.7	-	-
OneIE	**74.7**	**56.8**	**72.5**	55.9
Text2Event	71.9	53.8	70.3	53.4
DEGREE	72.2	55.8	71.7	**56.8**
UIE	73.4	54.8	70.7	52.6
Ours	72.5	55.2	71.8	53.8

Table 6: Performance comparison with 100% data on ACE05-E and ACE05-E+ datasets.

2-shot example for GPT4o-mini

Knowledge of Event Extraction
An event is something that happens. An Event's Trigger is the word (in its scope) that most clearly expresses its occurrence. Each event type has a particular set of roles and arguments.
Set of event types: {Life.Be-Born, Life.Marry, Life.Divorce, Life.Injure, Life.Die, Movement.Transport, Transaction.Transfer-Ownership, Transaction.Transfer-Money, Business.Start-Org, Business.Merge-Org, Business.Declare-Bankruptcy, Business.End-Org, Conflict.Attack, Conflict.Demonstrate, Contact.Meet, Contact.Phone-Write, Personnel.Start-Position, Personnel.End-Position, Personnel.Nominate, Personnel.Elect, Justice.Arrest-Jail, Justice.Release-Parole, Justice.Trial-Hearing, Justice.Charge-Indict, Justice.Sue, Justice.Convict, Justice.Sentence, Justice.Fine, Justice.Execute, Justice.Extradite, Justice.Acquit, Justice.Pardon, Justice.Appeal}.
Set of Roles: {'Origin', 'Entity', 'Seller', 'Money', 'Defendant', 'Destination', 'Price', 'Person', 'Time', 'Sentence', 'Recipient', 'Plaintiff', 'Target', 'Position', 'Artifact', 'Beneficiary', 'Adjudicator', 'Agent', 'Prosecutor', 'Giver', 'Instrument', 'Vehicle', 'Place', 'Buyer', 'Org', 'Crime', 'Attacker', 'Victim'}
Example
Example of no event:
Sentence: Martha Stewart is one of those stories that is in the news big time again.
Result: Event trigger is [trigger]. Event type is [event_type].[event_subtype]. [role] is [arg]. [SEP]
Example of one event:
Sentence: Senator Christopher Dodd of Connecticut made the announcement today that he would not be the 10th candidate for the nomination.
Result: Event trigger is nomination. Event subtype is Personnel.Nominate. Person is candidate. [SEP]
Input
Given the sentence: "He claimed Iraqi troops had destroyed five tanks.", does any event exist in this sentence? If any event exists, what is the event type and its trigger word? What's the role and argument?
Please only return the event extraction result. Return format: Event trigger is [trigger]. Event type is [event_type].[event_subtype]. [role] is [arg]. If two or more events exist, use [SEP] to concatenate.

Table 7: 2-shot example for GPT4o-mini.

Vanilla chain of thought example for GPT4o-mini

Knowledge of Event Extraction
An event is something that happens. An Event's Trigger is the word (in its scope) that most clearly expresses its occurrence. Each event type has a particular set of roles and arguments.
Set of event types: {Life.Be-Born, Life.Marry, Life.Divorce, Life.Injure, Life.Die, Movement.Transport, Transaction.Transfer-Ownership, Transaction.Transfer-Money, Business.Start-Org, Business.Merge-Org, Business.Declare-Bankruptcy, Business.End-Org, Conflict.Attack, Conflict.Demonstrate, Contact.Meet, Contact.Phone-Write, Personnel.Start-Position, Personnel.End-Position, Personnel.Nominate, Personnel.Elect, Justice.Arrest-Jail, Justice.Release-Parole, Justice.Trial-Hearing, Justice.Charge-Indict, Justice.Sue, Justice.Convict, Justice.Sentence, Justice.Fine, Justice.Execute, Justice.Extradite, Justice.Acquit, Justice.Pardon, Justice.Appeal}.
Set of Roles: {'Origin', 'Entity', 'Seller', 'Money', 'Defendant', 'Destination', 'Price', 'Person', 'Time', 'Sentence', 'Recipient', 'Plaintiff', 'Target', 'Position', 'Artifact', 'Beneficiary', 'Adjudicator', 'Agent', 'Prosecutor', 'Giver', 'Instrument', 'Vehicle', 'Place', 'Buyer', 'Org', 'Crime', 'Attacker', 'Victim'}
Example
Example of no event
Sentence: Martha Stewart is one of those stories that is in the news big time again.
Result: Event trigger is [trigger]. Event type is [event_type].[event_subtype]. [role] is [arg]. [SEP]
Reason: ...
Example of one event
Sentence: Senator Christopher Dodd of Connecticut made the announcement today that he would not be the 10th candidate for the nomination.
Result: Event trigger is nomination. Event subtype is Personnel.Nominate. Person is candidate. [SEP]
Reason: ...
Input
Given the sentence: "He claimed Iraqi troops had destroyed five tanks.", does any event exist in this sentence? If any event exists, what is the event type and its trigger word? What's the role and argument?
Let's think step by step. Please analyze the reason and return the event extraction result. Return format: Event trigger is [trigger]. Event type is [event_type].[event_subtype]. [role] is [arg]. If two or more events exist, use [SEP] to concatenate.

Table 8: Vanilla chain of thought example for GPT4o-mini

Dynamic chain of thought example for GPT4o-mini
Input-1: Firstly let us extract the trigger and event type. An event is something that happens. An Event's Trigger is the word (in its scope) that most clearly expresses its occurrence. Let us first consider the 9 event types, namely: Life, Movement, Transaction, Business, Conflict, Contact, Personnel, Justice. There is also [event_type], which means the event does not exist. Given the "He claimed Iraqi troops had destroyed five tanks.", does any event exist in this sentence? If any event exists, what is the event type and its trigger word? eturn format like: Event trigger is [trigger]. Event type is [event_type]. If two and more events exist, use [SEP] concat. If no event exists, return the format prompt. return format example: Event trigger is [trigger]. Event type is [event_type]. Event trigger is attend. Event type is Contact. [SEP] Event trigger is attend. Event type is Contact. [SEP] Event trigger is war. Event type is Conflict. [SEP] **Output-1:** Event trigger is destroyed. Event type is Conflict.
Input-2: Secondly, let us extract the event subtype. Each event type has a particular set of subtypes. Given the "He claimed Iraqi troops had destroyed five tanks.", does any event exist in this sentence? If any event exists, what is the event type and its trigger word? Event trigger is claimed. Event type is Conflict. Based on common sense, if any event exists, what's the event subtype and its trigger word? Please return format like: Event trigger is [trigger]. Event type is [event_type]. If two and more events exist, use [SEP] concat. If no event exists, return the format prompt. return format example: Event trigger is [trigger]. Event type is [event_type]. Event trigger is attend. Event type is Contact. [SEP] Event trigger is attend. Event type is Contact. [SEP] Event trigger is war. Event type is Conflict. [SEP] **Output-2:** Event trigger is destroyed. Event subtype is Conflict.Attack.
Input-3: Thirdly, let us extract the role and argument. Each event subtype has a particular set of roles and arguments. Given the "He claimed Iraqi troops had destroyed five tanks.", does any event exist in this sentence? If any event exists, what is the event type and its trigger word? Event trigger is claimed. Event subtype is Conflict.Attack. Based on common sense, if any event exists, what's the role and argument? Please return format like: Event trigger is [trigger]. Event subtype is [event_type]. [role] is [arg]. If two and more events exist, use [SEP] concat. If no event exists, return the format prompt. return format example: Event trigger is [trigger]. Event subtype is [event_type]. [role] is [arg]. [SEP] Event trigger is war. Event subtype is Conflict.Attack. Target is building. [SEP] Event trigger is war. Event subtype is Conflict.Attack. Target is building. Instrument is tanks. [SEP] Event trigger is attend. Event subtype is Movement.Transport. Artifact is Mike. [SEP] **Output-3:** Event trigger is destroyed. Event subtype is Conflict.Attack. Attacker is troops. Target is tanks.

Table 9: Dynamic chain of thought example for GPT4o-mini