DiCoRe: Enhancing Zero-shot Event Detection via Divergent-Convergent LLM Reasoning

Anonymous ACL submission

Abstract

Zero-shot Event Detection (ED), the task of identifying event mentions in natural language text without any training data, is critical for document understanding in specialized domains. Understanding the complex event ontology, extracting domain-specific triggers from the passage, and structuring them appropriately overloads and limits the utility of Large Language Models (LLMs) for zero-shot ED. To this end, we propose DiCoRe, a divergent-convergent reasoning framework that decouples the task of ED using Dreamer and Grounder. Dreamer encourages divergent reasoning through open-ended event discovery, which helps to boost event coverage. Conversely, Grounder introduces convergent reasoning to align the free-form predictions with the task-specific instructions using finite-state machine guided constrained decoding. Additionally, an LLM-Judge verifies the final outputs to ensure high precision. Through extensive experiments on six datasets across five domains and nine LLMs, we demonstrate how DiCoRe consistently outperforms prior zero-shot, transfer-learning, and reasoning baselines, achieving 4–7% average F1 gains over the best baseline – establishing DiCoRe as a strong zero-shot ED framework.

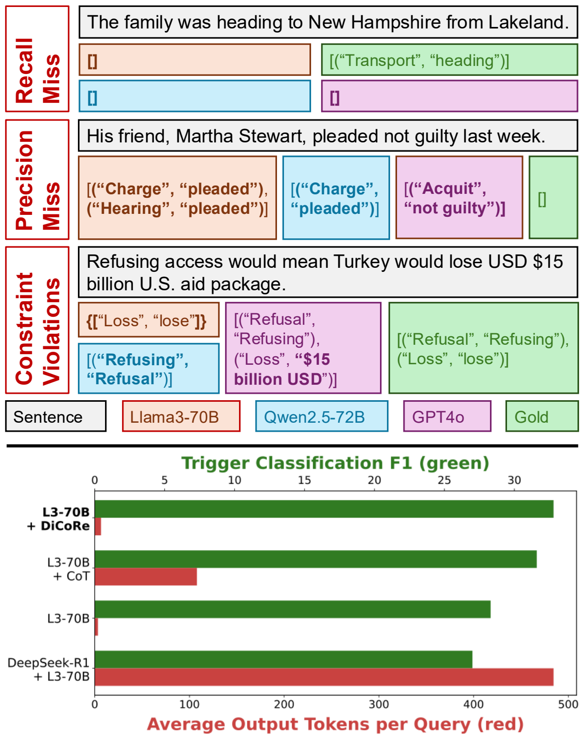

Figure 1: (top) Illustration of how prompting LLMs directly for Event Detection (ED) with all the task constraints can lead to precision, recall, and constraint violations (incorrect JSON, trigger not in sentence) across various LLMs. The errors are highlighted in **bold**. (bottom) Highlighting the superior model performance (**green bars**) of our proposed DiCoRe with minimal inference cost (**red bars**) relative to reasoning baselines.

1 Introduction

Event Detection (ED) is the task of identifying events by extracting and labeling event triggers (Sundheim, 1992; Doddington et al., 2004). ED aids in various downstream applications, including news monitoring (Tanev et al., 2008), biomedical literature mining (Pyysalo et al., 2012), epidemic forecasting (Parekh et al., 2024a,b), and legal understanding (Francesconi et al., 2010). Training effective ED models requires large amounts of expert-annotated domain-specific data, which is highly costly and labor-intensive. This underlines the need to develop zero-shot systems that can perform ED robustly without using any training data.

Recently, large language models (LLMs) have shown strong zero-shot performance across various tasks (Ouyang et al., 2022a; Li et al., 2023b). However, their effectiveness on ED remains limited (Gao et al., 2023; Huang et al., 2024), due to the requirement of extensive domain knowledge and the complex structural nature of ED. ED requires deep reasoning and imposes several intertwined constraints: study of the large, closed event ontology and ensuring the event types must be chosen from it; semantic understanding of the input passage and precisely identifying domain-specific triggers within it; and conforming the output to a strict, machine-parsable structured format. Encoding these constraints as natural language instructions in the prompt overloads the LLM cognitively, making it harder to effectively apply its reasoning skills (Tam et al., 2024). This increased difficulty in reasoning causes failures, such as missing relevant events, predicting irrelevant ones, and struggling to follow the expected format, as shown in Figure 1.

To this end, we propose DiCoRe, a novel pipeline introducing **Di**vergent-**Co**nvergent **Re**asoning, that facilitates better ED performance by reducing the cognitive burden of constraint adherence on the LLM. DiCoRe comprises two major components in a pipeline: Dreamer and Grounder. (1) Dreamer fosters divergent reasoning by prompting in an unconstrained, open-ended manner. This encourages broad semantic exploration of potential event mentions by removing rigid task constraints and, in turn, boosts the recall. (2) Grounder introduces convergent reasoning by mapping Dreamer's free-form predictions to the task-specific closed event ontology. To alleviate the constraint adherence burden on the LLM, we employ a finite-state machine (FSM) to encode structural and task-specific constraints. This FSM guides the generation process through constrained decoding, ensuring that the output adheres to the task requirements. Finally, we add an LLM-Judge to verify the grounded predictions against the original task instructions, ensuring high precision by filtering irrelevant predictions.

We conduct extensive experiments on six datasets from five domains across nine LLMs. Compared with various existing LLM inference works (Gao et al., 2023; Wang et al., 2023; Parekh et al., 2025), we show how DiCoRe performs the best with average improvements of 4-5% F1 Trigger Classification and 5.5-6.5% F1 Event Identification over the best baselines. DiCoRe, without any training, also consistently improves over transfer-learning baselines (Hsu et al., 2022; Sainz et al., 2023) fine-tuned on 15-30k datapoints by at least 5-12% F1. Furthermore, we demonstrate that DiCoRe provides 1-2% F1 gains while using 15-55x fewer inference tokens relative to strong thinking-based models and chain-of-thought (CoT), highlighting the significance of our proposed divergent-convergent reasoning.

In summary, we make the following contributions: (1) We propose Dreamer, introducing divergent reasoning to improve event coverage. (2) We develop Grounder, performing convergent reasoning to align free-form predictions to the event ontology. (3) We design FSM-guided decoding to enforce task-specific structure during inference. Through extensive evaluations across six datasets, five domains, and nine LLMs, we demonstrate the generalizability and efficacy of DiCoRe, establishing it as a robust zero-shot ED framework.[1]

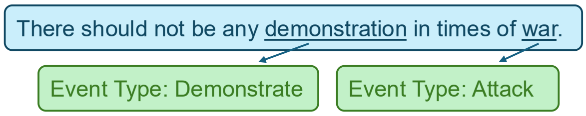

Figure 2: Illustration example for the task of Event Detection. Here, the blue box is the input sentence, and the green boxes are the event mentions. The underlined words indicate the event triggers.

2 Background and Related Works

Event Detection (ED) is the task of identifying event mentions from input text/document X based on a pre-defined ontology (Sundheim, 1992; Grishman and Sundheim, 1996; Doddington et al., 2004). We follow previous works (Doddington et al., 2004) to define an *event* as something that happens or describes a change of state. Each event is labeled by an *event type* e and the list of pre-defined event types constitutes an *event ontology* $\mathcal{E}$. An *event trigger* t is defined as the most distinctive word/phrase that indicates the event's presence in the text X. The trigger-event type pair (t, e) is jointly referred to as the *event mention*. The extraction of trigger words from the text and classifying them into one or more event types from the event ontology is the task of Event Detection, described by f below.

$$[(e_1, t_1), ...(e_n, t_n)] = f(X; \mathcal{E})$$

We provide an illustration of the task in Figure 2, wherein *demonstration* and *war* indicate the mentions of *Demonstrate* and *Attack* events, respectively, in the sentence.

Event Detection: Traditionally, ACE05 (Doddington et al., 2004) and ERE (Song et al., 2015) have been traditionally utilized for developing various sequence-tagging (Wadden et al., 2019; Hsu et al., 2023a) and generative (Li et al., 2021; Hsu et al., 2023b) models. However, procuring

[1] We will release our code upon acceptance.

expert-annotated data in specialized domains like biomedicine, law, cybersecurity, etc. is an expensive and labor-intensive task, leading to explorations in zero-shot and low-resource ED.

Zero-shot Event Detection: Recently, various diverse datasets such as MAVEN (Wang et al., 2020), FewEvent (Deng et al., 2019), GENEVA (Parekh et al., 2023) in general domain, GENIA2011 (Kim et al., 2011), GENIA2013 (Kim et al., 2013) in biomedical, CASIE (Satyapanich et al., 2020) in cybersecurity, PHEE (Sun et al., 2022) in pharmacovigilance, SPEED (Parekh et al., 2024b), SPEED++ (Parekh et al., 2024a) in epidemiology, etc. have been developed. To explore generalizability across these domains/datasets, initial works posed ED as a question-answering (Du and Cardie, 2020) or machine-reading comprehension problem (Liu et al., 2020). Various works explored transfer and joint learning across various IE tasks to build more universal IE models (Lu et al., 2022; Fei et al., 2023; Li et al., 2024). Some works have explored posing ED as a generative text-to-text approach with event-based templates (Lu et al., 2021; Li et al., 2021; Hsu et al., 2022), even for zero-shot cross-lingual transfer (Huang et al., 2022; Parekh et al., 2024b). However, these works require source data training for zero-shot transfer, limiting their utility. Recent works have also explored the utility of zero-shot prompting with LLMs - concluding their sub-par performance (Gao et al., 2023; Li et al., 2023a). Other works have explored utilizing LLMs for data generation (Ma et al., 2024; Zhang et al., 2024b; Parekh et al., 2025) to aid better generalizability. In our work, we focus on improving LLMs' zero-shot task generalizability to ED without any model fine-tuning.

Unconstraining LLMs for Better Reasoning: LLMs show immense language understanding and generation capabilities, but they need instructions and constraints to aid in meaningful human tasks (Ouyang et al., 2022b). However, imposing constraints also reduces LLM reasoning capabilities (Tam et al., 2024; Banerjee et al., 2025). To this end, works have explored constrained decoding by altering the output probability distribution (Willard and Louf, 2023; Netz et al., 2024; Zhang et al., 2024a). Some works explore grammar-aligned decoding strategies (Geng et al., 2023; Park et al., 2024). However, such strict enforcement has been shown to hurt LLMs' generations. Recently, Tam et al. (2024) explored better prompt design on math reasoning to unburden the constraints on the LLM. With similar inspiration, we explore decoupling LLMs from constraints to improve reasoning for IE tasks, specifically Event Detection, in our work.

3 Methodology

In our work, we frame ED through a generative outlook f_{gen} (Paolini et al., 2021; Huang et al., 2022) as they provide stronger zero-shot performance (Hsu et al., 2022) and are better suited for LLMs. We consider a structured list of tuples as the output generation as they provide stronger performance (§ C.1) and are easy to parse (Wang et al., 2023). However, these considerations introduce constraints (encoded as task instructions in LLM prompt) like the predicted event is from the provided list, the predicted trigger phrase is in the input text, and the output generation follows the JSON format, as technically described below.

$$Y = f_{gen}(X; \mathcal{E}) \qquad \text{where}$$
$$Y = \text{“}[(e_1, t_1), ...(e_n, t_n)]\text{”} \tag{1}$$
$$t \in X \qquad \forall t \in \{t_1, ...t_n\} \tag{2}$$
$$e \in \mathcal{E} \qquad \forall e \in \{e_1, ...e_n\} \tag{3}$$

We argue that these structured constraints inherent to ED (Eq. 1-3) increase the cognitive load on LLMs, making reasoning more difficult (Tam et al., 2024). This is one of the contributing factors to LLMs' subpar performance for ED (Huang et al., 2024). To address this, we propose DiCoRe, a novel pipeline that decouples and reduces constraint adherence through divergent open-ended discovery, convergent alignment, and constrained decoding. DiCoRe is lightweight, does not require additional training, and can be seamlessly applied to any LLM. Specifically, DiCoRe comprises a three-stage pipeline of a Dreamer-Grounder-Judge, as illustrated in Figure 3, and described below.

3.1 Dreamer

Our first component, Dreamer *aka Divergent open-ended thinker*, is designed to promote open-ended divergent discovery, encouraging the LLM to achieve high recall by freely identifying potential events without being constrained by the predefined event ontology. Specifically, the Dreamer component f_d removes the task-specific event constraint (Eq. 3), relaxes the trigger constraint (Eq. 2), and prompts the LLM to extract event mentions directly from the input sentence X as

$$Y_d = \text{“}[(e'_1, t_1), ...(e'_n, t_n)]\text{”} = f_d(X)$$

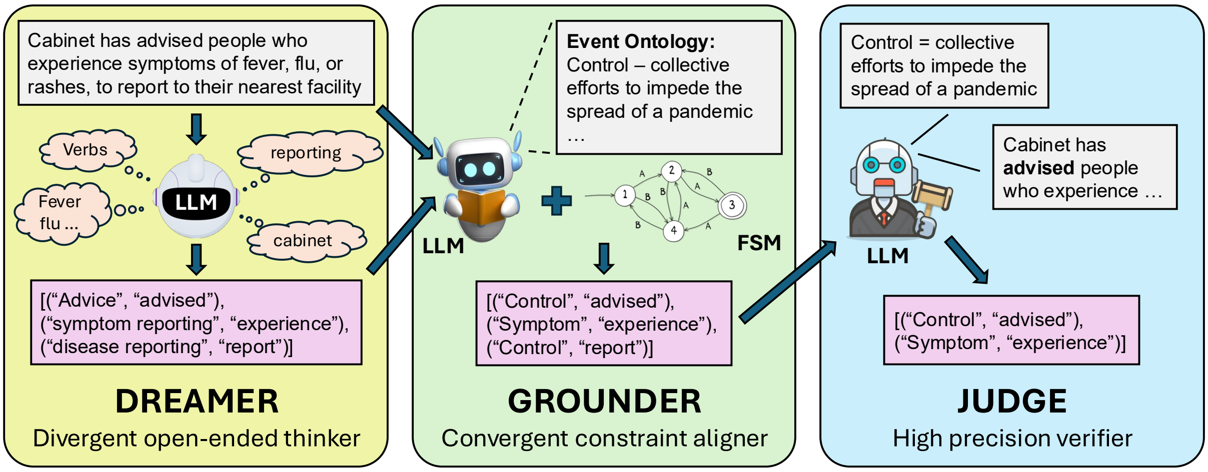

Figure 3: Illustration of our DiCoRe pipeline. First, the Dreamer reasons divergently in an open-ended unconstrained manner about all potential events in the text and generates free-form event names. Next, the Grounder reads the event ontology and convergently grounds the free-form predictions to one of the event types. It uses finite-state machine (FSM) guided constrained decoding to enforce task-specific constraints. Finally, the Judge evaluates each prediction and verifies its validity at a holistic scale.

where each e_i' is a free-form LLM-generated natural language event name. Notably, e_i' need not adhere to the event ontology $\mathcal{E}$. We provide an illustration of the LLM prompt in Figure 5.

By removing explicit references to the event ontology, the instructions become less restrictive and more semantically intuitive for the LLM. This simplification enables the model to divergently reason on the underlying semantics of the text, rather than rigidly aligning with predefined categories. This open-ended setup encourages broader event discovery, improving recall by allowing the model to identify diverse or implicit event types. Though it may lower precision, it produces a rich candidate set for downstream refinement.

3.2 Grounder

The second component, Grounder *aka Convergent constraint aligner*, convergently aligns the Dreamer's open-ended predictions Y_d with the closed, task-specific event ontology $\mathcal{E}$, while filtering the events that are not mappable. Technically, the Grounder component f_g infuses the original task-specific constraints into the prompt to generate the grounded event mentions Y_g as

$$Y_g = \text{“}[(e_1, t_1), ...(e_m, t_m)]\text{”} = f_g(X; \mathcal{E}, Y_d)$$

An illustration of the Grounder prompt and expected output is shown in Figure 6.

FSM-guided decoding for constraint enforcement: To reduce the burden of constraint-following on the LLM and ensure strict adherence

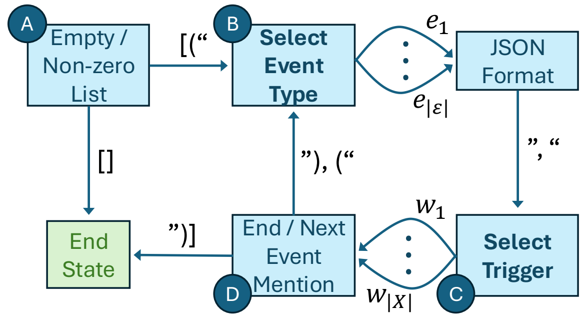

Figure 4: Finite state machine (FSM) illustration for guiding decoding to enforce constraints. Here $e_1, \ldots, e_{|\mathcal{E}|} \in \mathcal{E}$ represent all the possible event types and $w_1, \ldots, w_{|X|} \in X$ represent the atomized phrases in the sentence X.

to the task format, we incorporate a constrained decoding mechanism guided by a finite-state machine (FSM). Inspired by recent work (Willard and Louf, 2023; Zhang et al., 2024a), the FSM is designed to encode key constraints (Eq. 1–3) within the decoding process. We construct and demonstrate an FSM to encode constraints for our ED task in Figure 4.

The states of the FSM represent possible decision points (e.g., choice of event type, choice of trigger word, etc.). The state transitions denote possible LLM generations/choices (e.g., list of event types in the ontology $\mathcal{E}$) and are guided by their corresponding LLM generation probabilities. In our FSM in Figure 4, we first decide if there is any event or if it is an event-free sentence (state A). Next, we decide our first event type $e \in \mathcal{E}$ (state B)

LLM	Prompt Style	**MAVEN** (168)			**FewEvent** (100)			**ACE** (33)			**GENIA** (9)			**SPEED** (7)			**CASIE** (5)			**Average**		
		TI	TC	EI	TI	TC	EI	TI	TC	EI	TI	TC	EI	TI	TC	EI	TI	TC	EI	TI	TC	EI
Llama3-8B	ChatIE	33.7	7.3	13.8	20.8	10.2	**27.6**	30.6	24.9	46.8	8.6	3.2	11.3	28.4	15.5	**43.3**	10.8	3.6	20.4	22.2	10.8	27.2
	GEE	19.1	1.9	6.8	11.7	5.9	14.0	30.0	21.3	27.4	25.4	15.8	26.7	**35.9**	27.7	38.7	11.5	9.2	45.8	22.3	13.6	26.6
	DEE	33.7	6.0	9.2	21.1	10.6	17.8	26.9	19.8	36.1	25.3	16.9	**32.5**	29.1	20.3	39.2	8.7	7.6	48.3	24.1	13.5	30.5
	BD	**54.5**	10.7	12.3	22.3	9.9	15.0	34.2	19.5	31.4	**28.1**	11.2	30.2	35.3	24.7	37.2	16.8	7.4	44.5	31.9	13.9	28.4
	MD	45.9	2.8	4.0	25.2	9.5	15.2	35.6	22.4	30.1	22.8	15.3	25.4	34.9	**27.8**	42.4	10.3	8.8	47.9	29.1	14.4	27.5
	MS	46.2	10.3	11.2	20.2	10.2	17.0	26.7	17.6	23.1	27.6	**19.7**	30.5	34.1	27.3	40.6	11.9	10.3	48.3	27.8	15.9	28.4
	DiCoRe	53.5	**14.4**	**17.4**	**26.1**	**15.7**	25.0	**40.3**	**36.3**	**47.9**	25.8	15.4	30.0	35.5	23.6	42.4	**18.5**	**16.8**	**58.8**	**33.3**	**20.4**	**36.9**
Llama3-70B	ChatIE	47.9	19.8	24.8	33.3	20.8	**40.6**	45.5	37.9	47.0	14.6	6.4	17.3	41.8	31.0	50.9	12.9	10.2	48.9	32.7	21.0	38.3
	GEE	28.3	15.7	17.5	26.2	16.3	31.1	47.0	42.3	52.2	32.5	24.2	38.5	43.7	34.7	46.0	11.1	10.7	43.2	31.5	24.0	38.1
	DEE	60.8	14.8	16.4	34.0	21.3	33.6	47.4	38.3	45.4	39.2	30.5	46.0	41.7	32.2	44.7	16.6	16.4	63.1	40.0	25.6	41.5
	BD	63.0	13.9	15.2	34.0	14.5	22.6	49.1	36.6	41.7	**39.4**	26.5	45.4	**49.2**	33.6	45.7	16.5	11.7	48.8	41.9	22.8	36.6
	MD	**63.5**	14.2	14.7	34.0	20.9	32.6	51.2	40.2	46.8	36.8	28.9	43.0	45.4	36.8	49.0	13.9	13.7	64.4	40.8	25.8	41.8
	MS	33.9	21.6	22.3	35.3	24.9	39.9	49.9	42.8	46.9	37.4	**31.0**	45.0	43.8	35.5	49.6	14.0	14.0	59.5	35.7	28.3	43.9
	DiCoRe	62.5	**27.8**	**30.6**	**40.4**	**25.1**	36.1	**57.2**	**49.5**	**55.1**	38.6	**31.0**	**48.5**	45.0	**36.5**	**51.8**	**17.3**	**16.6**	**66.6**	**43.5**	**32.8**	**48.1**

Table 1: Main results comparing the zero-shot ED performance of our proposed DiCoRe with all other baselines for the Llama3-8B-Instruct and Llama3-70B-Instruct LLMs. TI: Trigger Identification, TC: Trigger Classification, EI: Event Identification. **bold** = best performance. (XX) = number of distinct event types.

and then the corresponding trigger $w \in X$ (state C). Finally, we decide if we want to list another event mention or end the predictions (state D). To ensure that the generations are natural, the FSM states are partitioned in alignment with the LLM tokenizer, i.e., the states are chosen such that the sequence of transition tokens is the most probable tokenization of the output text Y_g.

At each FSM state, we generate using the LLM while constraining its output space by zeroing out the probability mass of all tokens not corresponding to valid state transitions. This enforces that the LLM can only generate tokens permitted by the FSM at each step, effectively guiding the generation process according to the task-specific grammar. As a result, all structural constraints are directly enforced during decoding, ensuring well-formed and ontology-compliant outputs.

3.3 Judge

The final component of our pipeline, Judge *aka High precision verifier*, serves to ensure each predicted event mention adheres to the original task instructions. Specifically, for each candidate pair (e_i, t_i), the Judge f_j evaluates the hypothesis that the trigger t_i expresses the event type e_i in the context of the input sentence X as

$$y_j^i = \text{“}Yes/No\text{”} = f_j(e_i, t_i, X; \mathcal{E})$$

All predictions with $y_j^i = \text{“}Yes\text{”}$ are accepted into the final output, while the others are rejected. We provide an illustration of the prompt in Figure 7.

This verification step plays a crucial role in ensuring the semantic validity and task alignment of predictions at a holistic level. By filtering out irrelevant or uncertain outputs, the Judge substantially improves the precision of the overall system without requiring additional supervision or training.

4 Experimental Setup

In this section, we describe our experimental setup comprising the datasets, baselines, evaluation metrics, and implementation details. Additional setup and implementation details are provided in § B.

Datasets: We benchmark our model across six ED datasets spanning five diverse domains, listed as: (1) MAVEN (Wang et al., 2020) and (2) FewEvent (Deng et al., 2019) from the general domain, (3) ACE (Doddington et al., 2004) from the news domain, (4) GENIA (Kim et al., 2011), from the biomedical domain, (5) SPEED (Parekh et al., 2024b), from the epidemiological/social media domain, (6) CASIE (Satyapanich et al., 2020), from the cybersecurity domain. To avoid any distributional biases, following TextEE (Huang et al., 2024), we uniformly sample 250 datapoints from the complete dataset for evaluation.[2]

Baselines: We consider two major baselines, described below: (1) Multi-event Direct (MD) (Gao et al., 2023) directly prompts the LLM to provide the final output in a single pass, and (2) Multi-event Staged (MS) (Parekh et al., 2025) decomposes the task into two stages, where the first stage identifies the event and the second stage extracts the corresponding triggers. We also compare with other works like: (3) Binary-event Direct (BD) (Lyu et al.,

[2]We will release the test splits for reproduction.

LLM	Prompt Style	MAVEN (168) TI	TC	EI	FewEvent (100) TI	TC	EI	ACE (33) TI	TC	EI	GENIA (9) TI	TC	EI	SPEED (7) TI	TC	EI	CASIE (5) TI	TC	EI	Average TI	TC	EI
Qwen2.5-14B	MD	53.0	17.6	20.9	28.8	**21.1**	**34.2**	28.3	24.5	42.1	24.8	18.8	26.7	37.7	33.0	**51.2**	15.8	15.8	**61.5**	31.4	21.8	39.5
	MS	46.5	20.8	24.6	24.8	18.9	32.1	33.6	26.3	32.5	25.4	19.2	27.7	38.9	34.3	46.1	16.3	16.1	54.5	30.9	22.6	36.2
	DiCoRe	**53.1**	**23.3**	**27.6**	**29.7**	19.3	30.4	**38.4**	**37.7**	**48.8**	**29.9**	**22.6**	**38.6**	**42.9**	**35.3**	46.5	**19.7**	**19.5**	58.8	**35.8**	**26.1**	**41.8**
Qwen2.5-72B	MD	49.4	21.6	24.1	17.0	12.3	21.0	28.8	25.8	30.3	30.5	27.0	36.3	**41.4**	**37.4**	**45.4**	11.0	10.4	57.9	29.7	22.4	35.8
	MS	39.9	23.6	25.4	25.0	21.0	**34.2**	42.5	40.4	42.5	26.7	23.6	34.1	40.6	35.5	45.2	10.5	10.5	49.1	30.9	25.8	38.4
	DiCoRe	**54.1**	**27.5**	**30.2**	**30.8**	**22.3**	32.9	**46.8**	**44.8**	**47.8**	**33.6**	**29.8**	**43.9**	40.6	34.7	41.4	**15.9**	**15.8**	**59.3**	**37.0**	**29.2**	**42.6**
GPT3.5-turbo	MD	**50.9**	17.4	20.4	23.2	14.6	27.0	40.9	36.2	42.5	**27.0**	**19.9**	31.4	**36.5**	**30.6**	41.8	10.0	9.9	51.1	**31.4**	21.4	35.7
	MS	48.2	15.5	17.2	23.7	**15.9**	29.8	40.7	37.4	42.3	23.2	19.0	26.3	33.0	23.7	35.5	7.7	7.1	44.4	29.4	19.8	32.6
	DiCoRe	48.1	**21.6**	**26.1**	**25.3**	15.6	**31.1**	**41.7**	**41.7**	**48.9**	26.2	19.5	**36.3**	32.4	27.2	**49.0**	**11.4**	**10.6**	**55.7**	30.9	**22.7**	**41.2**
GPT4o	MD	**61.8**	28.9	31.9	30.6	23.9	35.4	52.3	52.3	52.3	**41.0**	**36.5**	49.5	44.1	40.2	48.0	10.1	10.1	55.7	40.0	32.0	45.5
	MS	49.4	30.8	33.3	25.6	20.6	32.2	36.2	36.2	38.3	36.6	33.2	45.0	**45.7**	**40.4**	**50.1**	13.4	13.4	46.9	34.5	29.1	41.0
	DiCoRe	58.5	**32.2**	**35.6**	**36.1**	**28.4**	**38.5**	**54.9**	**54.9**	**56.6**	40.7	35.4	**51.2**	43.3	37.3	46.1	**16.7**	**16.7**	**58.8**	**41.7**	**34.2**	**47.8**

Table 2: Generalization results for zero-shot ED performance comparing DiCoRe with the best baselines for four other LLMs of Qwen2.5-14B-Instruct, Qwen2.5-72B-Instruct, GPT3.5-turbo, and GPT4o. **bold** = best performance. (XX) = number of distinct event types.

2021; Li et al., 2023c) prompts the LLM to do binary classification for each event, (4) Decompose-Enrich-Extract (DEE) (Shiri et al., 2024) utilizes instruction enrichment with schema information for ED, (5) GuidelineEE (GEE) (Srivastava et al., 2025), similar to Code4Struct (Wang et al., 2023), converts ED into a code-generation problem using Python classes and instantiations, and (6) ChatIE (Wei et al., 2023) decomposes ED via multi-turn conversations. We ensure consistent, structured outputs for each baseline to maintain fair comparisons (analysis in § C.1). Furthermore, we add the Judge component to each baseline, if not already present, to ensure robust benchmarking of DiCoRe.

Base LLMs: We use the following LLMs for our base experiments: Llama3-8B-Instruct and Llama3-70b-Instruct from the Llama3 family (Dubey et al., 2024) and Qwen2.5-14B-Instruct; Qwen2.5-72B-Instruct from the Qwen2.5 (Yang et al., 2024) LLM family; and GPT3.5-turbo and GPT-4o (Brown et al., 2020; OpenAI, 2023) from OpenAI.

Evaluation Metrics: Following Ahn (2006); Parekh et al. (2025) we report the F1 scores for the following three metrics: (1) Trigger Identification (TI) - correct identification of triggers, and (2) Event Identification (EI) - correct classification of event types, and (3) Trigger Classification (TC) - correct identification of the trigger-event pair (event mention). To maintain consistency with traditional span-based evaluations, we used string matching to map the generated outputs to input spans.

Implementation Details: We use TextEE (Huang et al., 2024) for our benchmarking, datasets, and evaluation setup. To restrict LLM's generation choices for the FSM-guided constrained decoding, we utilize Outlines (Willard and Louf, 2023) over vLLM inference (Kwon et al., 2023). We use Curator (Marten et al., 2025) for querying the GPT family LLMs. We deploy a temperature of 0.4 and top-p of 0.9 for decoding. We report the averaged results over three runs for robust benchmarking.

5 Results and Analysis

In this section, we provide our main results and findings, and later provide supporting evidence through our analyses.

5.1 Main Results

We present the main zero-shot results for all baselines on the six datasets for Llama3 LLMs in Table 1. As seen from the average results (last three columns), DiCoRe performs the best, surpassing the best baseline of multi-event staged (MS) by a significant margin of 5.5-8% TI, 4-8.5% EI, and 4-5% TC. The performance disparity across different task decomposition methods of ChatIE, MS, and DiCoRe highlights how our divergent-convergent decomposition of Dreamer-Grounder provides a stronger inductive bias. Other baselines perform relatively better on the fewer-event and simpler datasets like GENIA/SPEED, but DiCoRe shows strong dominance on the larger event datasets like MAVEN/FewEvent/ACE.

Generalization across LLMs: To demonstrate the generalizability of DiCoRe, we benchmark it with the top-performing baselines on four additional LLMs from the Qwen and GPT families and show our results in Table 2. We note how DiCoRe

Model Setting	Average F1 TI	TC	EI
Test on GENIA, SPEED, CASIE			
GOLLIE-7B	6.0	5.3	15.3
GOLLIE-34B	15.6	11.7	29.4
Llama3-8B DiCoRe	<u>26.6</u>	<u>18.6</u>	<u>43.7</u>
Llama3-70B DiCoRe	<u>33.6</u>	<u>28.0</u>	<u>55.6</u>
Test on all but ACE dataset			
ACE-trained DEGREE	20.9	11.0	21.3
Llama3-8B DiCoRe	<u>31.9</u>	<u>17.2</u>	<u>34.7</u>
Llama3-70B DiCoRe	<u>40.8</u>	<u>27.4</u>	<u>46.7</u>
Test on all but MAVEN dataset			
MAVEN-trained DEGREE	31.8	25.0	38.6
Llama3-8B DiCoRe	29.2	21.6	<u>40.8</u>
Llama3-70B DiCoRe	<u>39.7</u>	<u>31.7</u>	<u>51.6</u>

Table 3: Comparison of pure zero-shot DiCoRe with fine-tuned transfer-learning baselines. <u>Underline</u> indicates scenarios of DiCoRe improvements.

performs the best across all LLMs with an overall average improvement of 5.5% TI, 6.5% EI, 4% TC over the multievent-staged baseline and 3.3% TI, 5.4%, 4.6% TC over the multievent-direct baseline. Across different LLMs, we note the strongest performance on GPT4o, followed by Llama3-70B-Instruct and Qwen2.5-72B, indicating how more parameters help better reasoning with DiCoRe.

5.2 Comparison with Fine-tuned Transfer-learning Methods

Various works utilize transfer-learning and universal Information Extraction (IE) training for zero-shot cross-dataset ED (Cai et al., 2024; Li et al., 2024). These works train on selected IE datasets and show performance on unseen IE datasets. We provide a comparison of DiCoRe with two such transfer-learning approaches: (1) DEGREE (Hsu et al., 2022), a generative framework utilizing text-based event templates to generalize, (2) GOLLIE (Sainz et al., 2023), a universal IE framework, fine-tuning LLMs on various IE instruction datasets. For DEGREE, we consider two versions where the source data is ACE and MAVEN, respectively. For GOLLIE, we consider the fine-tuned GOLLIE-7B and GOLLIE-34B models. We provide the averaged results across target datasets (not included in the source data) in Table 3, with detailed results in § C.2. Through these results, we demonstrate how, despite no fine-tuning, DiCoRe consistently outperforms the fine-tuned transfer-learning baselines across all settings. On average, DiCoRe improves by 3-10% F1 using Llama3-8B-Instruct and 10-22% F1 using Llama3-70B-Instruct and GPT4o.

Base LLM	Prompt Style	Average F1 TI	TC	EI
Chain-of-thought Baselines				
Llama3-8B	MD + CoT	25.0	13.5	27.1
Llama3-8B	MS + CoT	28.4	17.6	31.9
Llama3-70B	MD + CoT	41.0	30.9	48.0
Llama3-70B	MS + CoT	40.5	31.6	47.1
Qwen2.5-72B	MD + CoT	34.9	27.1	43.6
Qwen2.5-72B	MS + CoT	36.2	28.8	40.8
Thinking-based model Baselines				
DS-Qwen-32B	MD	39.2	30.0	46.3
DS-Qwen-32B	MS	39.5	30.4	45.2
DS-Llama3-70B	MD	29.0	23.3	36.1
DS-Llama3-70B	MS	33.3	27.0	37.8
O1-mini	MD	40.2	32.5	44.7
DiCoRe base model results				
Llama3-8B	DiCoRe	<u>33.3</u>	<u>20.4</u>	<u>36.9</u>
Llama3-70B	DiCoRe	<u>43.5</u>	<u>32.8</u>	<u>48.1</u>
Qwen2.5-72B	DiCoRe	<u>37.0</u>	<u>29.2</u>	42.6
GPT4o	DiCoRe	<u>41.7</u>	<u>34.2</u>	<u>47.8</u>
DiCoRe improvements with reasoning				
Llama3-8B	DiCoRe+ CoT	33.1	21.1	36.2
Llama3-70B	DiCoRe+ CoT	43.0	33.1	**49.8**
Qwen2.5-72B	DiCoRe+ CoT	37.0	29.1	43.5
DS-Qwen-32B	DiCoRe	**43.1**	**33.3**	49.5
DS-Llama3-70B	DiCoRe	41.4	33.0	48.3

Table 4: Comparison of DiCoRe with reasoning-based baselines like Chain-of-thought (CoT) and thinking-based models. <u>Underline</u> indicates DiCoRe improvements over reasoning baselines.

5.3 Comparison with Reasoning baselines

Reasoning by verbalizing thoughts using additional tokens has commonly helped improve performance across a wide range of tasks (Kojima et al., 2022; Latif et al., 2024). We evaluate the utility of reasoning, specifically Chain-of-thought (CoT) (Wei et al., 2022), along with thinking-based models like Deepseek-R1-Distilled-Qwen-32B (DS-Qwen-32B), Deepseek-R1-Distilled-Llama3-70B (DS-Llama3-70B) (DeepSeek-AI et al., 2025) and O1-mini (Jaech et al., 2024) on our task of zero-shot ED in Table 4 (complete results in § C.3). We demonstrate how the baselines improve with additional reasoning; however, DiCoRe with the base models (Llama3-70B) consistently outperforms all these reasoning baselines (even O1-mini) while using 15-55x fewer tokens on average (§ C.3). We also show how our method is complementary to reasoning by demonstrating further improvements up to 1-2% F1 using reasoning with DiCoRe.

Sentence	Best Baseline Prediction	Dreamer Prediction	Grounder Prediction	Judge Prediction
cass apd ra gave birth to her first daughter.	[("Life:Be-Born", "gave")]	[("Birth", "gave"), ("Birth", "birth")]	[("Life:Be-Born", "birth")]	[("Life:Be-Born", "birth")]
After passing the island, the hurricane turned to the northeast, and became extratropical on September 8, before dissipating two days later.	[("Change", "turned"), ("Change", "became"), ("Dissipating", "dissipating")]	[("Movement", "turned"), ("Transition", "became"), ("Dissipation", "dissipating")]	[("Change_event_time", "turned"), ("Becoming_a_member", "became"), ("Dispersal", "dissipating")]	[("Dispersal", "dissipating")]
Covid-19 has led to social distancing, but we can still be together through the quarantine with online gaming!	[]	[("Social_Distancing", "distancing"), ("Quarantine", "quarantine"), ("Gaming", "gaming")]	[("prevent", "distancing"), ("control", "quarantine")]	[("prevent", "distancing"), ("control", "quarantine")]

Table 5: Qualitative examples comparing DiCoRe's predictions (per component) with the best baseline. We highlight the correct predictions in **green** and incorrect ones in **red**.

Component	TI P	TI R	TI F	TC P	TC R	TC F
Llama3-8B-Instruct						
Dreamer	8.5	64.3	15.0	0.0	0.0	0.0
+ Grounder	20.4	47.9	28.6	15.5	37.1	21.9
+ FSM Decoding	22.3	56.8	32.1	16.2	42.3	23.4
+ Judge	41.8	**39.0**	**40.3**	**37.5**	**35.2**	**36.3**
MD Baseline	**48.4**	28.2	35.6	30.2	17.8	22.4
MS Baseline	22.0	33.8	26.7	14.4	22.5	17.6
Llama3-70B-Instruct						
Dreamer	15.5	77.5	25.8	0.0	0.0	0.0
+ Grounder	28.6	65.7	40.4	22.5	53.4	31.8
+ FSM Decoding	32.3	66.7	43.5	26.2	54.0	35.3
+ Judge	52.8	**62.5**	**57.2**	45.7	**54.0**	**49.5**
MD Baseline	57.2	46.5	51.2	44.0	37.1	40.2
MS Baseline	**66.4**	39.9	49.9	**57.0**	34.3	42.8

Table 6: Ablation Study on the ACE dataset highlighting the significance and contribution of each component of DiCoRe. P: Precision, R: Recall, F: F1 score.

5.4 Ablation Study

To demonstrate the role of each component of our pipeline, we ablate and show the model performance as we add each component in DiCoRe for the ACE dataset for Llama3-8B and Llama3-70B LLMs in Table 6. For reference, we also show the precision/recall splits of the baselines. Dreamer achieves a high recall for TI (albeit a low precision) - demonstrating the utility of divergent unconstrained reasoning. Grounder helps align the predictions, causing a slight drop in recall while improving the precision. FSM Decoding helps largely improve the recall for Llama3-8B-Instruct by improving the mapping, and precision for Llama3-70B-Instruct by fixing any constraint violations. Finally, Judge largely boosts the precision of the model. Analysis of the baselines reveals that they are conservative, making a low number of high-precision predictions. In comparison, DiCoRe makes many more predictions, largely improving recall while maintaining reasonably high precision.

Qualitative Study: We provide some qualitative examples for each component of DiCoRe, while comparing the best baseline across the datasets in Table 5 (more examples in § D). We see how the best baseline often reasons incorrectly, leading to precision loss, or remains conservative, predicting nothing, leading to recall errors. The split across the three components shows how Dreamer generates many plausible event mentions, Grounder aligns and removes some, while Judge verifies and filters irrelevant ones. These examples provide the internal workings of DiCoRe, highlighting the significance of divergent-convergent reasoning.

6 Conclusion and Future Work

In our work, we introduce DiCoRe, a novel divergent-convergent reasoning pipeline of Dreamer-Grounder-Judge, aimed at decoupling the LLM from task-specific constraints, and indirectly better exploiting LLMs' reasoning. Through experimentation on six ED datasets from five domains across nine LLMs, we confirm how DiCoRe provides a stronger inductive bias, improving over other zero-shot baselines, fine-tuned transfer learning methods, and reasoning-focused approaches. Future works can explore this paradigm on broader tasks and study to better elicit divergent-convergent reasoning.

Limitations

In our work, we focus on improving zero-shot LLM inference for Event Detection. This work is easily extendable to other low-resource settings as well as other Information Extraction (IE) tasks - but we leave these for future explorations. To keep experimentation consistent with prior works, we utilized/sampled 250 datapoints from each dataset as our test set. If working with a different data split, one might get different absolute model performance, but we believe the general trends should remain the same. Finally, there are various lines of work on improving the use of retrieval to select good in-context examples, or teaching the LLM to learn the schema. We believe these works are orthogonal and complementary to our work, and we do not compare/include them in our study.

Ethical Considerations

Our work focuses on using LLMs through the inductive bias of our method DiCoRe. Since we do not train the LLM, there could be inherent biases in the LLM that can crop up when using our pipeline. We do not study or provide methods to mitigate such biases, as it's not in the scope of our work.

We would like to acknowledge that we used AI assistants and chatbots for writing some parts of the paper, helping with coding up plots, and searching for related works. For each application, a human expert verified to ensure we do not add any spurious/harmful content.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, Sydney, Australia. Association for Computational Linguistics.

Debangshu Banerjee, Tarun Suresh, Shubham Ugare, Sasa Misailovic, and Gagandeep Singh. 2025. CRANE: reasoning with constrained LLM generation. *CoRR*, abs/2502.09061.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Zefan Cai, Po-Nien Kung, Ashima Suvarna, Mingyu Ma, Hritik Bansal, Baobao Chang, P. Jeffrey Brantingham, Wei Wang, and Nanyun Peng. 2024. Improving event definition following for zero-shot event detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2842–2863, Bangkok, Thailand. Association for Computational Linguistics.

Ruirui Chen, Chengwei Qin, Weifeng Jiang, and Dongkyu Choi. 2024. Is a large language model a good annotator for event extraction? In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17772–17780. AAAI Press.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 81 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Shumin Deng, Ningyu Zhang, Jiaojian Kang, Yichi Zhang, Wei Zhang, and Huajun Chen. 2019. Meta-learning with dynamic-memory-based prototypical network for few-shot event detection. *CoRR*, abs/1910.11621.

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

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Hao Fei, Shengqiong Wu, Jingye Li, Bobo Li, Fei Li, Libo Qin, Meishan Zhang, Min Zhang, and Tat-Seng Chua. 2023. Lasuie: Unifying information extraction with latent adaptive structure-aware generative language model. *CoRR*, abs/2304.06248.

Enrico Francesconi, Simonetta Montemagni, Wim Peters, and Daniela Tiscornia, editors. 2010. *Semantic*

Processing of Legal Texts: Where the Language of Law Meets the Law of Language, volume 6036 of *Lecture Notes in Computer Science*. Springer.

Jun Gao, Huan Zhao, Changlong Yu, and Ruifeng Xu. 2023. Exploring the feasibility of chatgpt for event extraction. *CoRR*, abs/2303.03836.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, Singapore. Association for Computational Linguistics.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference- 6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

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.

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

I-Hung Hsu, Zhiyu Xie, Kuan-Hao Huang, Prem Natarajan, and Nanyun Peng. 2023b. AMPERE: AMR-aware prefix for generation-based event argument extraction model. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10976–10993, Toronto, Canada. Association for Computational Linguistics.

Kuan-Hao Huang, I-Hung Hsu, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. Multilingual generative language models for zero-shot cross-lingual event argument extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4633–4646, Dublin, Ireland. Association for Computational Linguistics.

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.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, and 80 others. 2024. Openai o1 system card. *CoRR*, abs/2412.16720.

Jin-Dong Kim, Yue Wang, Toshihisa Takagi, and Akinori Yonezawa. 2011. Overview of Genia event task in BioNLP shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 7–15, Portland, Oregon, USA. Association for Computational Linguistics.

Jin-Dong Kim, Yue Wang, and Yamamoto Yasunori. 2013. The Genia event extraction shared task, 2013 edition - overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 8–15, Sofia, Bulgaria. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *CoRR*, abs/2205.11916.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Ehsan Latif, Yifan Zhou, Shuchen Guo, Yizhu Gao, Lehong Shi, Matthew Nayaaba, Gyeong-Geon Lee, Liang Zhang, Arne Bewersdorff, Luyang Fang, Xiantong Yang, Huaqin Zhao, Hanqi Jiang, Haoran Lu, Jiaxi Li, Jichao Yu, Weihang You, Zhengliang Liu, Vincent Shung Liu, and 8 others. 2024. A systematic assessment of openai o1-preview for higher order thinking in education. *CoRR*, abs/2410.21287.

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. *CoRR*, abs/2304.11633.

Dongfang Li, Zetian Sun, Xinshuo Hu, Zhenyu Liu, Ziyang Chen, Baotian Hu, Aiguo Wu, and Min Zhang. 2023b. A survey of large language models attribution. *CoRR*, abs/2311.03731.

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.

Sha Li, Qiusi Zhan, Kathryn Conger, Martha Palmer, Heng Ji, and Jiawei Han. 2023c. GLEN: General-purpose event detection for thousands of types. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2823–2838, Singapore. Association for Computational Linguistics.

Xiaomin Li, Zhou Yu, Zhiwei Zhang, Xupeng Chen, Ziji Zhang, Yingying Zhuang, Narayanan Sadagopan, and Anurag Beniwal. 2025. When thinking fails: The pitfalls of reasoning for instruction-following in llms. *CoRR*, abs/2505.11423.

Zixuan Li, Yutao Zeng, Yuxin Zuo, Weicheng Ren, Wenxuan Liu, Miao Su, Yucan Guo, Yantao Liu, Lixiang Lixiang, Zhilei Hu, Long Bai, Wei Li, Yidan Liu, Pan Yang, Xiaolong Jin, Jiafeng Guo, and Xueqi Cheng. 2024. KnowCoder: Coding structured knowledge into LLMs for universal information extraction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8758–8779, Bangkok, Thailand. 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.

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.

Qing Lyu, Hongming Zhang, Elior Sulem, and Dan Roth. 2021. Zero-shot event extraction via transfer learning: Challenges and insights. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 322–332, Online. Association for Computational Linguistics.

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. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 18751–18759. AAAI Press.

Ryan* Marten, Trung* Vu, Charlie Cheng-Jie Ji, Kartik Sharma, Shreyas Pimpalgaonkar, Alex Dimakis, and Maheswaran Sathiamoorthy. 2025. Curator: A Tool for Synthetic Data Creation. `https://github.com/bespokelabsai/curator`.

Lukas Netz, Jan Reimer, and Bernhard Rumpe. 2024. Using grammar masking to ensure syntactic validity in llm-based modeling tasks. *CoRR*, abs/2407.06146.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022a. Training language models to follow instructions with human feedback. *CoRR*, abs/2203.02155.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022b. Training language models to follow instructions with human feedback. *CoRR*, abs/2203.02155.

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. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Tanmay Parekh, Yuxuan Dong, Lucas Bandarkar, Artin Kim, I-Hung Hsu, Kai-Wei Chang, and Nanyun Peng. 2025. FIG: forward-inverse generation for low-resource domain-specific event detection. *CoRR*, abs/2502.17394.

Tanmay Parekh, I-Hung Hsu, Kuan-Hao Huang, Kai-Wei Chang, and Nanyun Peng. 2023. GENEVA: Benchmarking generalizability for event argument extraction with hundreds of event types and argument roles. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3664–3686, Toronto, Canada. Association for Computational Linguistics.

Tanmay Parekh, Jeffrey Kwan, Jiarui Yu, Sparsh Johri, Hyosang Ahn, Sreya Muppalla, Kai-Wei Chang, Wei

Wang, and Nanyun Peng. 2024a. SPEED++: A multilingual event extraction framework for epidemic prediction and preparedness. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12936–12965, Miami, Florida, USA. Association for Computational Linguistics.

Tanmay Parekh, Anh Mac, Jiarui Yu, Yuxuan Dong, Syed Shahriar, Bonnie Liu, Eric Yang, Kuan-Hao Huang, Wei Wang, Nanyun Peng, and Kai-Wei Chang. 2024b. Event detection from social media for epidemic prediction. 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 5758–5783, Mexico City, Mexico. Association for Computational Linguistics.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. 2024. Grammar-aligned decoding. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Sampo Pyysalo, Tomoko Ohta, Makoto Miwa, Han-Cheol Cho, Junichi Tsujii, and Sophia Ananiadou. 2012. Event extraction across multiple levels of biological organization. *Bioinform.*, 28(18):575–581.

Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. 2023. Gollie: Annotation guidelines improve zero-shot information-extraction. *CoRR*, abs/2310.03668.

Taneeya Satyapanich, Francis Ferraro, and Tim Finin. 2020. CASIE: extracting cybersecurity event information from text. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8749–8757. AAAI Press.

Fatemeh Shiri, Farhad Moghimifar, Reza Haffari, Yuan-Fang Li, Van Nguyen, and John Yoo. 2024. Decompose, enrich, and extract! schema-aware event extraction using llms. In *27th International Conference on Information Fusion, FUSION 2024, Venice, Italy, July 8-11, 2024*, pages 1–8. IEEE.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ERE: Annotation of entities, relations, and events. In *Proceedings of the 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 89–98, Denver, Colorado. Association for Computational Linguistics.

Saurabh Srivastava, Sweta Pati, and Ziyu Yao. 2025. Instruction-tuning llms for event extraction with annotation guidelines. *CoRR*, abs/2502.16377.

Zhaoyue Sun, Jiazheng Li, Gabriele Pergola, Byron Wallace, Bino John, Nigel Greene, Joseph Kim, and Yulan He. 2022. PHEE: A dataset for pharmacovigilance event extraction from text. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5571–5587, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Beth M. Sundheim. 1992. Overview of the fourth Message Understanding Evaluation and Conference. In *Fourth Message Understanding Conference (MUC-4): Proceedings of a Conference Held in McLean, Virginia, June 16-18, 1992*.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on large language model performance. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1218–1236, Miami, Florida, US. Association for Computational Linguistics.

Hristo Tanev, Jakub Piskorski, and Martin Atkinson. 2008. Real-time news event extraction for global crisis monitoring. In *Natural Language and Information Systems, 13th International Conference on Applications of Natural Language to Information Systems, NLDB 2008, London, UK, June 24-27, 2008, Proceedings*, volume 5039 of *Lecture Notes in Computer Science*, pages 207–218. Springer.

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.

Xiaozhi Wang, Ziqi Wang, Xu Han, Wangyi Jiang, Rong Han, Zhiyuan Liu, Juanzi Li, Peng Li, Yankai Lin, and Jie Zhou. 2020. MAVEN: A Massive General Domain Event Detection Dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1652–1671, Online. Association for Computational Linguistics.

Xingyao Wang, Sha Li, and Heng Ji. 2023. Code4Struct: Code generation for few-shot event structure prediction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3640–3663, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022.

Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. 2023. Zero-shot information extraction via chatting with chatgpt. *CoRR*, abs/2302.10205.

Brandon T. Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *CoRR*, abs/2307.09702.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Honghua Zhang, Po-Nien Kung, Masahiro Yoshida, Guy Van den Broeck, and Nanyun Peng. 2024a. Adaptable logical control for large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Weiyan Zhang, Wanpeng Lu, Jiacheng Wang, Yating Wang, Lihan Chen, Haiyun Jiang, Jingping Liu, and Tong Ruan. 2024b. Unexpected phenomenon: LLMs' spurious associations in information extraction. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9176–9190, Bangkok, Thailand. Association for Computational Linguistics.

A DiCoRe Prompts

We described our modeling paradigm of divergent-convergent reasoning through the Dreamer-Grounder-Judge paradigm in § 3. Here we provide some additional details and also share the prompts that we used for each component.

Dreamer: The Dreamer component induces divergent thinking, encouraging the model to think more widely. We induce this behavior by removing the event-based constraints from the task instructions and adding additional inductive bias to provide this encouragement inthe form of additional task instructions asking the model to be super liberal. We provide an illustration of this prompt in Figure 5. Specifically, sentences like "Try to be liberal and increase the coverage as much as possible. I will filter and improve the precision in the next step." and "Be very open and output all possible events that are potentially mentioned." provide this stronger divergent reasoning inductive bias.

Grounder: The Grounder component aligns the open-ended predictions of the Dreamer with the closed event ontology using convergent reasoning. To this end, we add the various task-specific constraints in the form of natural language instructions as well as use a finite-state machine (FSM) guided generation to aid with this convergent reasoning. Here, we describe the prompt and the inductive biases in it, as illustrated in Figure 6. Specifically, we first add all the verbalized constraints, including the ontology details in the form of event names and information. To provide more inductive bias, we also add a sentence like "Be conservative in your outputs - If a trigger word cannot be mapped, skip the trigger word. If the mapped event does not happen in the sentence, skip the trigger word.".

Judge: The Judge is tasked with the evaluation of the prediction to ensure that the trigger word triggers the specific event in the given sentence. We run the Judge for each prediction separately. To make this lightweight, we ensure that the output space is simple "Yes" or "No" without any explanation, which makes the parsing easier as well. We provide an illustration of this prompt in Figure 7. This component is very generic and can be easily applied to other methods/LLMs as well.

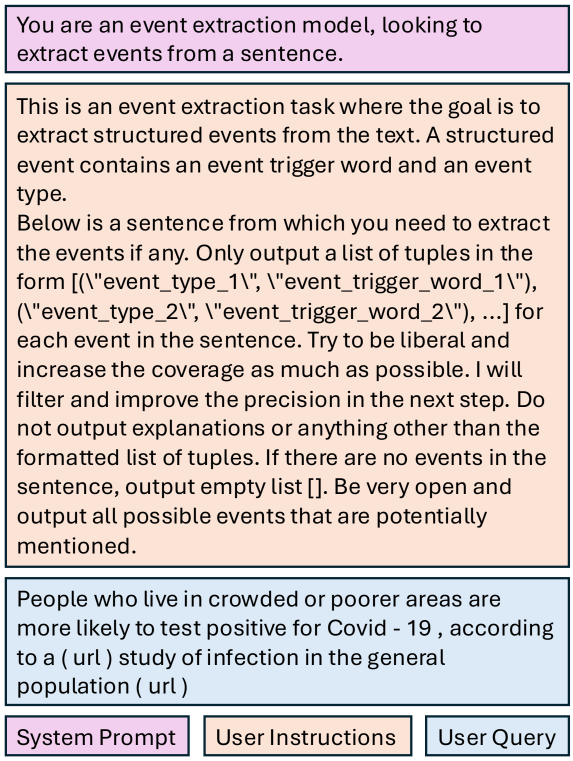

Figure 5: Illustration of the prompt utilized for Dreamer. To encourage divergent thinking, we remove event-based constraints from the model instructions. Furthermore, we add sentences that encourage the model to be liberal and open in its predictions.

B Additional Experimental Details

In § 4, we provided brief details about our experimental and implementation details. Here, we provide more intricate details about our different experimental setups and baseline models.

B.1 Dataset Statistics

Our experimental setup is a pure zero-shot setup where we do not use any training data. We provide statistics about the evaluation splits of the different datasets in Table 7. We follow TextEE (Huang et al., 2024) for the evaluation setup and consider a uniform random split of 250 test samples from each dataset to avoid any train-test split bias. Since CASIE is a smaller dataset, we only use 50 test samples for this dataset. The table highlights the domain diversity of the datasets covering common domains like news and general, while also focusing on technical domains like biomedical and epidemiology. The datasets also show variation in the density, with ACE, FewEvent, and SPEED being sparse with 1 event mention/sentence. On the other hand, MAVEN, CASIE, and GENIA are

You are an event extraction model, looking to map provided trigger words to potential event types.

This is an event extraction task where the goal is to extract structured events from the text. Given the sentence and possible event triggers, map these triggers to corresponding events from the provided event list. Omit triggers which are not mappable or if the mapped event is not mentioned in the sentence. The event list comprises 7 events. These events are:
Infect ... Spread ...
...
Below is the sentence and the list of trigger words. Map each trigger word from this list to a single event from above and output a list of tuples in the form [(\"event_type_1\", \"event_trigger_word_1\"), (\"event_type_2\", \"event_trigger_word_2\"), ...]. Be conservative in your outputs - If a trigger word cannot be mapped, skip that trigger word. If the mapped event does not happen in the sentence, skip that trigger word. Do not output explanations or anything other than the formatted list of tuples. If no triggers can be mapped, output empty list [].

Sentence: I hope this pandemic ends soon ...
Trigger List: ['ends']"

System Prompt | User Instructions | User Query

Figure 6: Illustration of the prompt utilized for Grounder. To encourage convergent thinking and alignment, we add event-based constraints in the model instructions. Furthermore, we add sentences that encourage the model to be more conservative in its predictions.

You are an event extraction verification model, looking to verify the provided trigger word triggers the event type in the given sentence.

This is an event extraction verification task where the goal is to verify if the extracted structured event is mentioned in the text. Given the sentence, a possible event mention with its trigger, verify if the event mention is correct or not.
Event Definition: The event of interest is infect. The event is related to the process of a disease/pathogen invading host(s).
Event Trigger: infection
Below is the sentence. Verify if the above trigger word triggers the above mentioned event in this given sentence. If yes, then output 'Yes' else output 'No'. Do not output explanations or anything other than 'Yes/No'.

People who live in crowded or poorer areas are more likely to test positive for Covid - 19 , according to a (url) study of infection in the general population (url)

System Prompt | User Instructions | User Query

Figure 7: Illustration of the prompt utilized for Judge. To encourage convergent thinking and alignment, we add event-based constraints in the model instructions. Furthermore, we add sentences that encourage the model to be more conservative in its predictions.

Dataset	Domain	# Doc	# Event Mentions	Avg. Doc Length
MAVEN	General	250	623	24.5
FewEvent	General	250	250	30.5
ACE	News	250	71	13.2
GENIA	Biomedical	250	2472	251.3
SPEED	Epidemiology	250	258	32.4
CASIE	Cybersecurity	50	291	283.1

Table 7: Data Statistics of the various ED datasets used in our experimental setup.

denser with 2.5-10 event mentions/passage. Finally, we also show the variation in token length, with ACE being the lowest with 13 average tokens, while GENIA and CASIE are longer with 250-280 average tokens per document.

B.2 Additional Implementation Details

In this section, we provide additional implementation details for DiCoRe and the various baselines. For open-source models, we ran them locally on NVIDIA RTX A6000/A100 machines with support for 8 GPUs.

B.2.1 DiCoRe

Trigger Atomization Adaptation for FSM-guided Decoding: Different datasets have varied annotation instructions and definitions for the trigger spans. Some datasets are strictly adhering to only single-word triggers (e.g., SPEED), while others are largely loose and support multi-word triggers (e.g., CASIE). We provide a small study of measuring multi-word triggers in Table 8, highlighting this disparity across datasets. To account for these varied definitions, we infuse a customizable atomization unit in our FSM-guided decoding. Specifically, state C from Figure 4 is customizable wherein for stricter datasets (SPEED, ACE, FewEvent), we impose an additional constraint of single-word trigger, while for other datasets (CASIE, GENIA, MAVEN), we apply a looser constraint of substring match with the query sentence.

B.2.2 Multi-event Direct (MD)

Multi-event direct (MD) (Gao et al., 2023; Huang et al., 2024; Chen et al., 2024) is the most common and simplest prompting technique used for ED. It prompts the model directly to reason across all the events and provide the relevant triggers based on

Dataset	% Multi-word Triggers
MAVEN	8%
FewEvent	3%
ACE	2.8%
GENIA	8.5%
SPEED	0%
CASIE	54.6%

Table 8: Measuring the percentage of multi-word triggers across the different ED datasets.

You are an event extraction model, looking to extract events from a sentence.

This is an event extraction task where the goal is to extract structured events from the text. A structured event contains an event trigger word and an event type.
Here are 7 events that we are interested in:
Infect ... Spread ...
...
Below is a sentence from which you need to extract the events if any. Only output a list of tuples in the form [(\"event_type_1\", \"event_trigger_word_1\"), (\"event_type_2\", \"event_trigger_word_2\"), ...] for each event in the sentence. Do not output explainations or anything other than the formatted list of tuples. If there are no events in the sentence, output empty list [].

People who live in crowded or poorer areas are more likely to test positive for Covid - 19 , according to a (url) study of infection in the general population (url)

System Prompt | User Instructions | User Query

Figure 8: Illustration of the prompt utilized for multi-event direct baseline.

the query text. We try various prompt versions and illustrate the best engineered prompt based on a small study in Figure 8. Majorly, we include all task-specific instructions and constraints in a single verbalized prompt, which can overload the LLM's reasoning capability.

B.2.3 Multi-event Staged (MS)

Multi-event staged (MS) (Parekh et al., 2025) was introduced as a way of forward generation to ensure higher trigger quality. We extend that in our work to build a strong task decomposition baseline. Simply, this model first extracts the event types from the texts in Stage 1 and then extracts triggers specific to these event types in Stage 2. We provide an illustration of the two stages of MS in Figures 9 and 10. In this case, the first stage majorly only focuses on the event-specific constraints, while the

You are an event extraction model, looking to extract events from a sentence.

This is an event extraction task where the goal is to extract structured events from the text. Given the sentence, figure if you find any event mention of the possible events.
Here are 7 events that we are interested in:
Infect ... Spread ...
...
Below is a sentence from which you need to extract the events if any. Only output a list of events in the form [\"event_type_1\", \"event_type_2\", ...] that you find in the sentence. Do not output explanations or anything other than the formatted list. If there are no events in the sentence, output empty list [].

People who live in crowded or poorer areas are more likely to test positive for Covid - 19 , according to a (url) study of infection in the general population (url)

System Prompt | User Instructions | User Query

Figure 9: Illustration of the Stage-1 prompt utilized for multi-event staged baseline.

You are an event extraction model, looking to extract event triggers for given events from a sentence.

This is an event extraction task where the goal is to extract structured events from the text. Given the sentence and possible events, find corresponding event triggers for the event. Event triggers are usually one word, many times verbs, and most indicative of the event presence.
Here are 1 events that are possibly present in the sentence: ['infect']
The event of interest is infect. The event is related to the process of a disease/pathogen invading host(s).
Below is a sentence. Identify the trigger word for the above listed events and output a list of tuples in the form [(\"event_type_1\", \"event_trigger_word_1\"), (\"event_type_2\", \"event_trigger_word_2\"), ...] for all the events mentions. Omit events which are not present or do not have a prominent trigger. Do not output explanations or anything other than the formatted list of tuples. If there are no events in the sentence, output empty list [].

Children can catch COVID - 19 .

System Prompt | User Instructions | User Query

Figure 10: Illustration of the Stage-2 prompt utilized for multi-event staged baseline.

You are an event extraction model, looking to extract event triggers for the given event from a sentence.

This is an event extraction task where the goal is to extract structured events from the text. Given the sentence and the event definition, find corresponding event triggers for the event. Event triggers are usually one word, many times verbs, and most indicative of the event presence.
The event of interest is infect. The event is related to the process of a disease/pathogen invading host(s).
Below is a sentence. Identify the trigger word for the above event of interest. Output a list in the form [\"trigger1\", \"trigger2\" ...] for all the events mentions. If the event is not present, output a empty list []. Do not output explanations or anything other than the output list.

Children can catch COVID - 19 .

System Prompt | User Instructions | User Query

Figure 11: Illustration of the prompt utilized for binary-event direct baseline.

second stage is focused on the trigger-specific ones.

B.2.4 Binary-event Direct (BD)

Binary-event direct (BD) (Lyu et al., 2021; Li et al., 2023c) has been a popular paradigm pre-dating LLMs when smaller generative text-to-text models were used. It drastically reduces the LLM's constraints by making the LLM focus on a single event type at a time, i.e., it prompts the LLM in a multi-event direct manner, but for each event type separately. Finally, the predictions are aggregated and output as the final prediction. We provide an illustration of the prompt in Figure 11. Overall, this is a highly expensive method, especially for larger event datasets.

B.2.5 Decompose-Enrich-Extract (DEE)

Decompose-Enrich-Extract (DEE) (Shiri et al., 2024) is a variation of the multi-event direct (MD) model, wherein it prompts the model to make predictions while enhancing the input schema. It also puts down additional rules to make the extraction more accurate, but we posit this also adds more constraints, restricting the model's reasoning. We provide an illustration of the prompt for this baseline in Figure 12.

B.2.6 GuidelineEE (GEE)

GuidelineEE (GEE) (Srivastava et al., 2025) is the method focused on providing extensive guidelines

You are an event extraction model, looking to extract events from a sentence.

Task Description: You are an assistant that helps extract the list of event types and their trigger words from input text.
Extraction Rules:
- The instance can contain any number of events.
- Limit responses to event types and their triggers only.
- Refrain from providing additional explanations.
- Do not enumerate the list.

Event Type Definitions: The possible event types and their definitions are as follows:
Infect ... Spread ...
...
Output Format: Output a list of events [{'event_type': <event_type_1>, 'trigger': <event_trigger_1>}, {'event_type': <event_type_2>, 'trigger': <event_trigger_2>}, ...]. Each event contains an event type and its trigger.

People who live in crowded or poorer areas are more likely to test positive for Covid - 19 , according to a (url) study of infection in the general population (url)

System Prompt | User Instructions | User Query

Figure 12: Illustration of the prompt utilized for Decompose-Enrich-Extract baseline.

You are an event extraction model, looking to extract events from a sentence.

This is an event extraction task where the goal is to extract structured events from the text. A structured event contains an event trigger word and an event type. For each different event type, please output the instances of the corresponding classes with the appropriate trigger i.e.
<Event_Name>(trigger='<Trigger_name>')
```
# The following lines describe the events as python classes:
@dataclass
class infect():
""" The event of interest is infect. The event is related to the process of a disease/pathogen invading host(s)."""
def __init__(self, trigger: str):
    self.trigger = trigger
...
```

```
# This is the text to analyze
text = "Children can catch COVID - 19 ."
```

```
# The list called result should contain the instances for the events in the above text according to the guidelines above (i.e. [ event_name1(trigger='trigger1'), event_name2(trigger='trigger2'), ...]):
result =
```

System Prompt | User Instructions | User Query

Figure 13: Illustration of the prompt utilized for GuidelineEE baseline.

to the LLM to improve its task understanding capability. This work is similar to Code4Struct (Wang et al., 2023), wherein the input and output are more code-oriented using Python class-like structures. The definition is provided as a docstring, and the trigger is extracted as an attribute of the class. The output is mainly instantiations of the right set of classes. We provide an illustration of the prompt for this baseline in Figure 13.

B.2.7 ChatIE

ChatIE (Wei et al., 2023) is a simple variation of multi-event staged (MS), but uses multi-turn conversation with the LLM. Specifically, stage-1 (Figure 9) is used as the initial prompt, and based on the output, stage-2 (Figure 10) is used as the second turn of the prompt.

B.2.8 GPT Runs

For the GPT models (i.e., GPT3.5-turbo, GPT4o, O1-mini), we utilized Curator (Marten et al., 2025) for the API calls. We noticed how the GPT models are already super conservative in their predictions, even when explicitly asked not to be. The Judge component was indeed hurting model performance by making the pipeline more conservative. Thus, we removed the Judge from all runs of the GPT LLMs.

C Additional Experimental Results

Here we provide additional and complementary results to the ones discussed in the main paper.

C.1 Structured v/s Unstructured Output

In our work, we largely maintain the output to be structured to ensure easy parsing and get stronger model performance as noted in Wang et al. (2023). To provide more evidence, we conducted a small experiment with different output formats: (1) Structured JSON output (the base version that we have currently) using a JSON list of tuples as the output, (2) Structured text wherein we ask the LLM to produce natural language text but in a structured way, and (3) Free-form text and re-structuring (Tam et al., 2024), wherein the LLM generates free-form text in the first generation and later restructures into JSON format using an additional LLM generation. We provide an illustration of these output formats in Figure 14.

We ablate these three output formats using the Multi-event Direct (MD) prompt setting for the ACE dataset using Llama3-8B-Instruct. We provide the results of the average of 3 runs in Table 9. As clearly evidenced, any kind of text-based output format is quite poor for TI and TC metrics, highlighting the significance of JSON-based output.

Below is a sentence from which you need to extract the events if any. Only output a list of tuples in the form [(\"event_type_1\", \"event_trigger_word_1\"), (\"event_type_2\", \"event_trigger_word_2\"), ...] for each event in the sentence. Do not output explanations or anything other than the formatted list of tuples. If there are no events in the sentence, output empty list [].

Below is a sentence from which you need to extract the events if any. Output the event and trigger information as natural sentences like "The event <event_name> is triggered by the trigger <trigger>." for each event type on a new line. Do not output explanations. If there are no events in the sentence, output "No events found.".

Below is a sentence from which you need to extract the events if any. Output the event and trigger information in natural language as you wish. Do not output any explanations.

Below is the text from which you need to extract the structured event-related information. Only output a list of tuples in the form [(\"event_type_1\", \"event_trigger_word_1\"), (\"event_type_2\", \"event_trigger_word_2\"), ...] for each event in the sentence. Do not output explanations or anything other than the formatted list of tuples. If there are no events in the sentence, output empty list [].

Structured JSON | Structured Text | Free-form & Restructuring

Figure 14: Illustration of the prompts utilized for the different output formats for ablating why the structured output format is better.

Output Format	TI	TC	EI
Structured JSON	**35.6**	**22.4**	30.1
Stuctured Text	14.9	11.0	**31.8**
Free-form & Restructuring	16.7	12.7	20.8

Table 9: Ablation Study on the ACE dataset using Llama3-8B-Instruct, highlighting the significance of utilizing structured JSON output compared to text outputs.

C.2 Complete Results for Transfer Learning Baselines

We discussed and compared DiCoRe with existing zero-shot cross-dataset transfer-learning approaches in § 5.2. We provide complete results for each dataset in Table 12 for a deeper analysis. We exclude results for MAVEN and FewEvent for GOLLIE as the generations were degenerate and led to 0 F1 performance. Across the three settings of various source-target datasets, we see how our pure zero-shot DiCoRe consistently outperforms all the fine-tuned transfer learning baselines by a considerable margin. In fact, DiCoRe, based on the smaller Llama3-8B-Instruct LLM is stronger than most of these transfer-learning baselines. This highlights the superior zero-shot generalization of

LLM	Prompt Style	Avg. Words
Llama3-8B	MD + CoT	36.8
	MS + CoT	82.4
Llama3-70B	MD + CoT	87.4
	MS + CoT	107.9
Qwen2.5-72B	MD + CoT	96.3
	MS + CoT	184.4
DS-Qwen-32B	MD	247.8
	MS	525.5
DS-L3-70B	MD	258.9
	MS	484.4
Llama3-8B	DiCoRe	11.6
Llama3-70B	DiCoRe	6.6
Qwen2.5-72B	DiCoRe	5.1

Table 10: Efficiency analysis in terms of average number of words per query (Avg. Words) of DiCoRe with other reasoning-based baselines on the ACE dataset.

Component/LLM	EI		
	P	R	F
Llama3-8B-Instruct			
Dreamer	0.0	0.0	0.0
+ Grounder	19.1	45.6	26.9
+ FSM Decoding	21.1	54.9	32.3
+ Judge	**49.5**	**46.5**	**47.9**
MD Baseline	40.5	23.9	30.1
MS Baseline	18.9	29.6	23.1
Llama3-70B-Instruct			
Dreamer	0.0	0.0	0.0
+ Grounder	25.4	61.5	36.0
+ FSM Decoding	29.1	60.1	39.2
+ Judge	50.8	**60.1**	**55.1**
MD Baseline	51.2	43.2	46.8
MS Baseline	**62.5**	37.5	46.9

Table 11: Ablation Study using Trigger Identification (TI) on the ACE dataset highlighting the significance and contribution of each component of DiCoRe. P: Precision, R: Recall, F: F1 score.

our proposed method.

C.3 Complete Results for Reasoning Baselines

In § 5.3, we discuss and compare DiCoRe with reasoning-based approaches and models. Here, we provide complete results of that comparison across datasets in Table 14. In comparison to the non-CoT numbers, we note how CoT provides gains for the baseline models, and larger gains for the larger LLMs. This indicates how reasoning improves model performance, but also requires more parameters and longer context handling. Thinking-based models somehow show poorer performance compared to CoT, and our observations align with Li et al. (2025). Next, we show how the base non-CoT performance of DiCoRe is better than the CoT-based baselines. This can also be seen when comparing thinking-based model baselines. This strongly indicates how the strong inductive bias of DiCoRe beats the reasoning-based improvements.

Additionally, we also infuse reasoning with DiCoRe, specifically only in the Grounder stage. Reasoning in the Dreamer stage makes the model more conservative and harms the divergent reasoning we want to encourage. We note how this additional reasoning provides further improvements of up to 1-2% F1 over the base DiCoRe performance.

Efficiency analysis: Apart from performance, we also analyze the effectiveness in terms of efficiency of the various methods. We measure efficiency by the average number of output words generated per query (which should be equivalent to the average number of output tokens). We provide this comparison for the different methods and LLMs for the ACE dataset in Table 10. As evident, CoT and thinking-based models expend a large amount of tokens on token-based reasoning, which is zero in the case of DiCoRe. On average, DiCoRe reduces the output words by 15x compared to CoT and by up to 55x compared to the thinking-based models. This highlights the practical utility of DiCoRe where it can provide higher performance at vastly reduced token generation cost.

C.4 Additional results for Ablation Study

We provided an ablation study for DiCoRe's components in § 5.4. Here we provide additional results for the same study, specifically for the Event Identification (EI) evaluation metric in Table 11. We conclude observations similar to those noted in the main paper, highlighting how DiCoRe helps increase the recall without much decreasing the precision of the model. Dreamer has a 0% score since the event names are free-form text generations in this stage.

D Broader Qualitative Study

We provided a brief qualitative study eliciting some common errors of previous baselines and how DiCoRefixes them in § 5.4. Here, we provide some more examples to highlight the various errors made by previous baselines in Table 13. Next, we also show some more examples to elicit the internal component-wise predictions of DiCoRe in Table 15. Overall, these examples demonstrate the utility of the divergent-convergent reasoning

LM/LLM	Prompt Style	MAVEN (168) TI	TC	EI	FewEvent (100) TI	TC	EI	ACE (33) TI	TC	EI	GENIA (9) TI	TC	EI	SPEED (7) TI	TC	EI	CASIE (5) TI	TC	EI	Average TI	TC	EI
Trained on ACE data* → Tested on other datasets																						
BART-large	DEGREE	29.4	11.0	13.8	42.6	22.5	27.2	-	-	-	5.1	3.5	11.6	23.4	16.2	26.7	3.8	2.0	27.0	20.9	11.0	21.3
Llama3-8B	DICORE	53.5	14.4	17.4	26.1	15.7	25.0	-	-	-	25.8	15.4	30.0	35.5	23.6	42.4	18.5	16.8	58.8	31.9	17.2	34.7
Llama3-70B	DICORE	62.5	27.8	30.6	40.4	25.1	36.1	-	-	-	38.6	31.0	48.5	45.0	36.5	51.8	17.3	16.6	66.6	40.8	27.4	46.7
GPT4o	DICORE	58.5	32.2	35.6	36.1	28.4	38.5	-	-	-	40.7	35.4	51.2	43.3	37.3	46.1	16.7	16.7	58.8	39.1	30.0	46.0
Trained on MAVEN data* → Tested on other datasets																						
BART-large	DEGREE	-	-	-	31.1	18.7	25.0	43.3	36.6	38.2	33.9	27.6	46.2	44.8	37.1	44.8	6.1	5.2	38.6	31.8	25.0	38.6
Llama3-8B	DICORE	-	-	-	26.1	15.7	25.0	40.3	36.3	47.9	25.8	15.4	30.0	35.5	23.6	42.4	18.5	16.8	58.8	29.2	21.6	40.8
Llama3-70B	DICORE	-	-	-	40.4	25.1	36.1	57.2	49.5	55.1	38.6	31.0	48.5	45.0	36.5	51.8	17.3	16.6	66.6	39.7	31.7	51.6
GPT4o	DICORE	-	-	-	36.1	28.4	38.5	54.9	54.9	56.6	40.7	35.4	51.2	43.3	37.3	46.1	16.7	16.7	58.8	38.3	34.5	50.2
Trained on ACE data* → Tested on GENIA, SPEED, CASIE																						
GOLLIE-7B	GOLLIE	-	-	-	-	-	-	-	-	-	3.2	2.2	7.1	12.6	11.6	24.3	2.1	2.1	14.4	6.0	5.3	15.3
GOLLIE-34B	GOLLIE	-	-	-	-	-	-	-	-	-	26.5	22.8	40.4	15.9	10.9	19.1	4.5	1.5	28.6	15.6	11.7	29.4
Llama3-8B	DICORE	-	-	-	-	-	-	-	-	-	25.8	15.4	30.0	35.5	23.6	42.4	18.5	16.8	58.8	26.6	18.6	43.7
Llama3-70B	DICORE	-	-	-	-	-	-	-	-	-	38.6	31.0	48.5	45.0	36.5	51.8	17.3	16.6	66.6	33.6	28.0	55.6
GPT4o	DICORE	-	-	-	-	-	-	-	-	-	40.7	35.4	51.2	43.3	37.3	46.1	16.7	16.7	58.8	33.6	29.8	52.0

Table 12: Complete results for comparison of DICORE with other fine-tuned transfer-learning approaches for zero-shot ED. *Training done for models other than DICORE. DICORE results are pure zero-shot, i.e., without any training. "-" indicates training data or where results were degenerate. (XX) = number of distinct event types.

paradigm for ED.

Sentence	Baseline Prediction
Precision Errors	
In the near future we will be expanding this to include all the other organizations that we can contact, but we are just keeping things safe for now.	[(**"Phone-Write"**, **"contact"**)]
The Holocaust of the Jews and Zigeuner was motivated by racial prejudices.	[(**"Attack"**, **"Holocaust"**)]
My friend, an ER physician has said over 70% of people who test positive for covid NEVER have a fever.	[(**"symptom"**, **"fever"**)]
On 4 April 2013, a building collapsed on tribal land in Mumbra.	[(**"Destroying"**, **"collapsed"**)]
Recall Errors	
Pasko was released in January for good behavior after serving more than two-thirds of the sentence.	[(**"Release-Parole"**, **"released"**)] **Missed: ("Sentence", "sentence")**
People who live in crowded or poorer areas are more likely to test positive for Covid - 19	[] **Missed: ("infect", "positive")**
WOW debuted on January 18 as part of AXS's Friday Night Fights schedule	[] **Missed: ("Process_start", "debuted")**
He is got it pretty easy Id say even with the international travel	[] **Missed: ("Transport-person", "travel")**

Table 13: Qualitative examples highlighting the various errors by zero-shot LLM baselines. We highlight the correct predictions in **green** and incorrect ones in **red**.

LLM	Prompt Style	**MAVEN** (168)			**FewEvent** (100)			**ACE** (33)			**GENIA** (9)			**SPEED** (7)			**CASIE** (5)			**Average**		
		TI	TC	EI	TI	TC	EI	TI	TC	EI	TI	TC	EI	TI	TC	EI	TI	TC	EI	TI	TC	EI
Chain-of-thought																						
Llama3-8B	MD	45.9	2.8	4.0	25.2	9.5	15.2	35.6	22.4	30.1	22.8	15.3	25.4	34.9	27.8	42.4	10.3	8.8	47.9	29.1	14.4	27.5
	+ CoT	35.4	3.2	4.8	15.4	6.8	13.8	30.6	18.7	27.6	24.3	15.9	26.9	34.6	27.8	42.1	9.8	8.7	47.1	25.0	13.5	27.1
	MS	46.2	10.3	11.2	20.2	10.2	17.0	26.7	17.6	23.1	**27.6**	19.7	30.5	34.1	27.3	40.6	11.9	10.3	48.3	27.8	15.9	28.4
	+ CoT	35.9	7.2	8.2	20.5	11.1	19.3	34.3	23.4	32.9	27.2	**20.1**	29.6	**39.4**	**31.9**	**46.6**	13.1	12.2	54.8	28.4	17.6	31.9
	DiCoRe	53.5	14.4	17.4	26.1	**15.7**	**25.0**	**40.3**	36.3	**47.9**	25.8	15.4	30.0	35.5	23.6	42.4	**18.5**	**16.8**	**58.8**	**33.3**	20.4	**36.9**
	+ CoT	**53.6**	**15.5**	**17.9**	**27.5**	15.4	24.7	39.8	**36.6**	45.0	25.8	16.4	**31.9**	35.1	26.6	41.5	16.7	15.9	56.0	33.1	**21.1**	36.2
Llama3-70B	MD	**63.5**	14.2	14.7	34.0	20.9	32.6	51.2	40.2	46.8	36.8	28.9	43.0	45.4	36.8	49.0	13.9	13.7	64.4	40.8	25.8	41.8
	+ CoT	56.0	29.4	32.5	37.1	25.3	37.2	54.9	48.5	57.1	35.4	28.2	45.5	47.1	39.5	50.3	15.7	14.8	65.4	41.0	30.9	48.0
	MS	33.9	21.6	22.3	35.3	24.9	**39.9**	49.9	42.8	46.9	37.4	31.0	45.0	43.8	35.5	49.6	14.0	14.0	59.5	35.7	28.3	43.9
	+ CoT	55.7	29.5	32.6	34.9	25.4	38.6	56.1	51.3	56.5	31.8	26.4	37.7	**49.7**	**42.5**	**56.6**	14.8	14.6	60.6	40.5	31.6	47.1
	DiCoRe	62.5	27.8	30.6	40.4	25.1	36.1	**57.2**	49.5	55.1	**38.6**	31.0	**48.5**	45.0	36.5	51.8	17.3	16.6	66.6	**43.5**	32.8	48.1
	+ CoT	61.2	**34.1**	**36.4**	**40.9**	**27.3**	37.5	55.4	**51.7**	**58.5**	37.9	**31.7**	48.1	44.3	36.5	50.8	**18.0**	**17.4**	**67.1**	43.0	**33.1**	**49.8**
Qwen2.5-72B	MD	49.4	21.6	24.1	17.0	12.3	21.0	28.8	25.8	30.3	30.5	27.0	36.3	41.4	37.4	45.4	11.0	10.4	57.9	29.7	22.4	35.8
	+ CoT	54.0	27.9	**33.8**	26.7	20.5	33.3	46.1	41.6	47.3	29.5	26.1	38.9	**42.6**	36.8	**48.1**	10.3	9.9	**60.0**	34.9	27.1	43.6
	MS	39.9	23.6	25.4	25.0	21.0	34.2	42.5	40.4	42.5	26.7	23.6	34.1	40.6	35.5	45.2	10.5	10.5	49.1	30.9	25.8	38.4
	+ CoT	**54.2**	28.0	31.1	28.3	21.5	33.6	**48.5**	**46.3**	**48.9**	30.7	26.5	38.7	44.9	**39.7**	47.9	10.6	10.6	44.5	36.2	28.8	40.8
	DiCoRe	54.1	27.5	30.2	30.8	22.3	32.9	46.8	44.8	47.8	33.6	**29.8**	**43.9**	40.6	34.7	41.4	15.9	15.8	59.3	37.0	29.2	42.6
	+ CoT	**54.2**	**29.7**	**33.8**	**31.7**	**23.5**	**35.5**	45.4	42.2	45.4	**34.2**	29.2	43.6	40.5	34.6	44.8	**16.8**	**16.7**	**60.0**	**37.1**	**29.3**	**43.8**
Thinking-based models																						
DS-Qwen-32B	MD	55.3	26.7	30.1	34.0	23.7	36.8	**56.3**	51.8	60.2	33.2	27.5	41.2	45.5	39.0	54.5	11.1	11.1	54.9	39.2	30.0	46.3
	MS	55.0	25.8	29.6	33.8	23.3	**38.5**	50.6	48.9	59.6	30.5	25.0	36.6	**52.7**	**44.7**	54.7	14.6	14.6	51.9	39.5	30.4	45.2
	DiCoRe	**60.1**	**30.2**	**32.6**	**38.5**	**26.1**	36.8	**56.3**	**53.9**	**60.5**	**36.3**	**30.4**	**47.6**	48.6	41.1	**55.2**	**18.5**	**17.8**	**64.4**	**43.1**	**33.3**	**49.5**
DS-L3-70B	MD	48.3	31.2	32.5	13.7	9.6	17.3	31.5	27.8	34.5	24.5	21.6	31.9	45.3	38.9	50.6	10.5	10.5	50.0	29.0	23.3	36.1
	MS	50.3	28.3	31.3	23.9	18.5	28.3	36.8	33.7	38.0	27.8	24.6	35.3	48.2	**44.2**	49.2	12.6	12.6	44.7	33.3	27.0	37.8
	DiCoRe	**59.5**	**34.7**	**37.2**	**36.2**	**25.9**	**35.0**	**53.0**	**51.3**	**55.8**	**32.3**	**28.6**	**42.7**	**49.3**	39.8	**53.4**	**18.0**	**17.9**	**65.9**	**41.4**	**33.0**	**48.3**
O1-mini	MD	59.1	32.8	35.7	36.8	28.0	40.3	53.9	48.5	53.0	35.8	33.7	43.8	44.2	40.2	48.1	11.5	11.5	47.5	40.2	32.5	44.7

Table 14: Complete results for comparison of DiCoRe with reasoning approaches like Chain-of-thought (CoT) and thinking-based models for zero-shot ED. **bold** = best performance. (XX) = number of distinct event types.

Sentence	**Dreamer Prediction**	**Grounder Prediction**	**Judge Prediction**
Police also arrested two Moroccan men suspected of trafficking in human beings and navigating the Zodiac boat across from Africa, Efe said.	[("arrest", "arrested"), ("trafficking", "trafficking"), ("navigating", "navigating"), ("said", "said")]	[("Arrest-Jail", "arrested"), ("Charge-Indict", "trafficking")]	[(**"Arrest-Jail", "arrested"**)]
Only 4 men have competed without eliminating a single opponent Fire, Mini Maximo, Sombrita and Stukita.	[("compete", "competed"), ("eliminate", "eliminating")]	[("Competition", "competed")]	[(**"Competition", "competed"**)]
Weird as hell: the Covid-19 patients who have symptoms for months | Coronavirus outbreak | The Guardian (url)	[("Disease_Spread", "outbreak"), ("Infection", "patients"), ("Symptom_Show", "symptoms")]	[("symptom", "symptoms"), ("spread", "outbreak")]	[(**"symptom", "symptoms"**), (**"spread", "outbreak"**)]
The time he has spent inside roughly equates to 2 years per woman he killed	[("Kill", "killed"), ("Spend", "spent"), ("Equate", "equates")]	[("Life.Die", "killed")]	[(**"Life.Die", "killed"**)]

Table 15: Qualitative examples eliciting DiCoRe's predictions per component for various input sentences. We highlight the correct predictions in **green** and incorrect ones in **red**.