Towards Event Extraction with Massive Types: LLM-based Collaborative Annotation and Partitioning Extraction

Wenxuan Liu, Zixuan Li*, Long Bai, Yuxin Zuo,
Daozhu Xu, Xiaolong Jin*, Jiafeng Guo, Xueqi Cheng
Key Laboratory of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences
State Key Laboratory of AI Safety
School of Computer Science and Technology, University of Chinese Academy of Sciences
{liuwenxuan2024z, lizixuan, jinxiaolong}@ict.ac.cn

Abstract

Developing a general-purpose system that can extract events with massive types is a long-standing target in Event Extraction (EE). In doing so, the basic challenge comes from the absence of an efficient and effective annotation framework to construct the corresponding datasets. In this paper, we propose an LLM-based collaborative annotation framework. Through collaboration among multiple LLMs and a subsequent voting process, it refines annotations of triggers from distant supervision and then carries out argument annotation. Finally, we create EEMT, the largest EE dataset to date, featuring over **200,000** samples, **3,465** event types, and **6,297** role types. Evaluation on the human-annotated test set demonstrates that the proposed framework achieves the F1 scores of **90.1%** and **85.3%** for event detection and argument extraction, strongly validating its effectiveness. Besides, to alleviate the excessively long prompts caused by massive types, we propose an LLM-based Partitioning method for EE called LLM-PEE. It first recalls candidate event types and then splits them into multiple partitions for LLMs to extract. After fine-tuning on the EEMT training set, the distilled LLM-PEE with 7B parameters outperforms state-of-the-art methods by **5.4%** and **6.1%** in event detection and argument extraction. Besides, it also surpasses mainstream LLMs by **12.9%** on the unseen datasets, which strongly demonstrates the event diversity of the EEMT dataset and the generalization capabilities of the LLM-PEE method.

1 Introduction

Event Extraction (EE) aims to identify structural event information from text, which contains two subtasks (Zhan et al., 2023), i.e., Event Detection (ED) and Event Argument Extraction (EAE). The former detects the trigger words of events (event triggers) and their corresponding types, while the latter extracts arguments and their roles based on the given trigger and its event type. EE has demonstrated its value across a variety of domains, including finance (Lee et al., 2021), biomedical (Sun et al., 2022) and Cyber-security (Satyapanich et al., 2020). Each of these domains has its own specific event types, which jointly form a large event schema containing massive types. It leads researchers (Spaulding et al., 2023) to pursue a general-purpose event system with massive types in different domains.

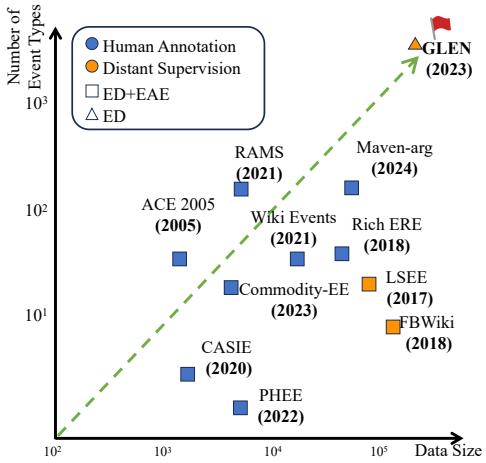

Figure 1: Statistics on the existing EE datasets.

In doing so, the basic challenge is the lack of an effective and efficient annotation framework to construct datasets. As illustrated in Figure 1, existing datasets can be divided into two types: human-annotation and distant supervision. The former is generally effective but inefficient. Consequently, the obtained datasets are often limited in both event type and scale. The latter (Chen et al., 2017; Zhan et al., 2023) automatically annotate triggers and arguments if they have been annotated in the knowledge bases (e.g., Propbank (Kingsbury and Palmer, 2002)), offering a more efficient alternative. However, taking the largest dataset, GLEN (Zhan et al.,

* Corresponding authors.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 34377–34399
November 4-9, 2025

2023), for example, it often suffers from noise in three aspects: 1) **Unreasonable Trigger Annotation:** Some predicates in knowledge bases are not considered as events in the view of EE, leading to the incorrect annotation of irrelevant words as triggers. For example, the adverb "voluntarily" is wrongly annotated as a trigger in GLEN; 2) **Coarse-Grained Type Annotation:** Due to the hierarchical structure of event types, a predicate often has types with multiple granularities. Previous works (e.g, GLEN) struggle to assign precise fine-grained types to these predicates; 3) **Missing Argument Annotation:** Unlike triggers, where candidate triggers are relatively limited, the number of potential arguments is much larger and cannot be fully enumerated by existing knowledge bases. Thus, distant supervision-based annotation often misses argument annotations when the arguments are not included in the knowledge bases.

In this paper, we propose an LLM-based collaborative annotation framework. To improve efficiency and avoid excessively long prompts when directly handling massive event types, we leverage LLMs to perform annotations based on distant supervision, rather than starting from scratch. Based on the distant supervision results, it first performs event trigger filtering by removing irrelevant triggers. This is followed by event type refinement, which assigns more fine-grained event types for triggers based on context. Finally, it identifies arguments and their roles in each event and refines the original annotation with human annotation rules. After each step, multiple LLMs collaborate to generate annotations, and then a voting phase is used to unify the annotation preferences across annotating LLMs. Finally, we obtain a new dataset EEMT, with over **200,000** annotated samples, covering **3,465** event types and **6,297** argument role types, which is the largest EE dataset regarding event type and scale to date. Besides, we create a **human-annotated** test set for unbiased evaluation. The annotation framework achieves F1 scores of **90.1%** and **85.3%** for event detection and argument extraction on the human-annotated test set, strongly validating the effectiveness of the annotation framework and the quality of the EEMT dataset.

Additionally, in some situations, there is a need to train custom open-source smaller LLMs and directly apply them for EE with massive event types from scratch. To alleviate the prompt length limitation, we further propose a partitioned event extraction method called LLM-PEE. The method begins by recalling the top-k most possible event types, then divides them into several partitions, which are assembled into prompts for the LLMs to conduct event extraction. After fine-tuning on the EEMT training set, we obtain a distilled LLM-PEE with 7B parameters. It outperforms state-of-the-art methods by **5.4%** and **6.1%** in event detection and argument extraction on the human-annotated test set. Furthermore, it surpasses mainstream LLMs (e.g., DeepSeek-V3-671B) by **12.9%** on unseen datasets in the zero-shot setting, strongly demonstrating the event diversity of the EEMT and the generalization capabilities of the LLM-PEE method.

Our contributions can be summarized as follows:

- We propose an LLM-based collaborative annotation framework for EE with massive types. Based on the framework, we construct the EEMT dataset, which is the largest EE dataset to date regarding coverage and scale. Besides, we create a human-annotated test set for unbiased evaluation.
- We propose an LLM-based Partitioning Extraction method, called LLM-PEE, which alleviates the prompt length problem when directly apply LLMs to handling massive schemas from scratch.
- Distilled on the constructed EEMT dataset, LLM-PEE (7B) achieves significant improvements in both supervised and zero-shot settings compared to the state-of-the-art methods.

2 Related Work

Event Extraction Dataset. In human annotated datasets, ACE 2005 (Walker, 2005) is the most used dataset, including 33 event types and 22 roles. Followed by this, Rich ERE (Song et al., 2015) is proposed to further enhance the scale of the EE dataset. MAVEN-arg(Wang et al., 2023b) is constructed based on MAVEN(Wang et al., 2020), the current largest EAE dataset annotated by human experts with 162 event types and 612 roles. However, due to the high cost of annotating events, human annotators can not further extend the schema and data scale. For datasets from distant supervision-based methods, LSEE (Chen et al., 2017) is constructed from FrameNet (Fillmore and Baker, 2009) and WikiPedia (Milne, 2008) to automatically annotate

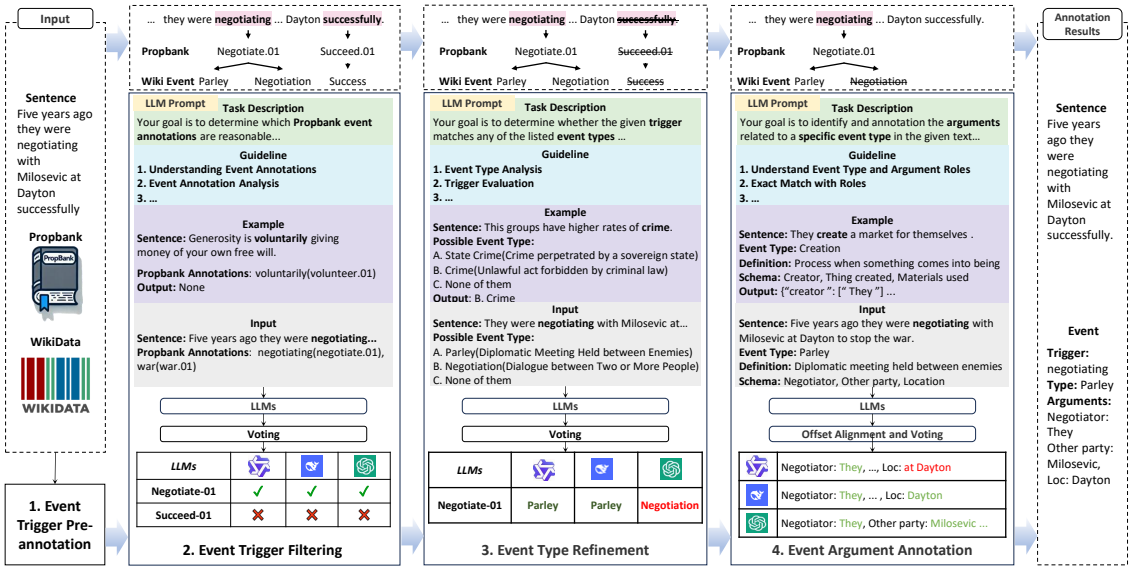

Figure 2: Overview of the proposed LLM-based collaborative annotation framework for EE with massive types.

events. GLEN (Zhan et al., 2023) is the largest ED dataset, including 3465 types and 200,000 samples, using Propbank and Wikidata event nodes to conduct distant supervision. Besides, some datasets (Tong et al., 2022; Han et al., 2022) focus on document-level or single-instance event extraction, which is different from our setting. And our work focuses on trigger-based EE with multiple instances at the sentence level.

Event Extraction Method. EE methods can be divided into two kinds: classification-based and generation-based ones. Classification-based methods (Zhang and Ji, 2021; Zhan et al., 2023) tend to formulate EE as a token classification or a sequential labeling task. Generation-based methods (Du and Cardie, 2020; Liu et al., 2020; Hsu et al., 2022) aim to generate the text containing a structured event, while these methods require manual schema-specific templates, which are difficult to adapt to massive types. Nowadays, due to the strong generation abilities of LLMs, they have been widely used in EE, like InstructUIE (Wang et al., 2023a), KnowCoder (Li et al., 2024), AlignXIE (Zuo et al., 2024). Besides, some ICL (In-context-Learning) IE works (Wang et al., 2022; Li et al., 2023b) focus on few-shot or zero-shot settings.

3 LLM-based Collaborative Annotation

3.1 Annotation Framework

Given that LLMs demonstrate strong capabilities in understanding natural language, we incorporate them into the EE annotation process. To improve efficiency, we use LLMs to perform annotation based on trigger annotations from distant supervision, rather than starting from scratch. By collaborating across multiple LLMs using offset alignment and voting, we achieve consistent and precise annotations. Specifically, as shown in Figure 2, the proposed annotation framework consists of four steps: (1) Event trigger pre-annotation annotates the triggers and their potential types based on distant supervised methods; (2) Event trigger filtering filters the unreasonable event triggers via LLMs; (3) Event type refinement maps the trigger to more fine-grained event types via LLMs; (4) Event argument annotation further annotates the arguments via LLMs. The detailed prompts and voting strategies are listed in Appendix D.2 and D.3.

Event Trigger Pre-annotation. Due to the massive types in the schema, directly using LLMs to annotate events is inefficient, as it requires including extensive guidelines in the prompt of LLMs. To improve the efficiency of the annotation process and mitigate the missing of events, we perform event trigger pre-annotation using distant supervision to annotate events as much as possible. We follow the event definitions used GLEN and DWD Overlay (Spaulding et al., 2023), both of which include certain generic concept types. Specifically, following GLEN, we take the distant supervision-based methods based on Propbank (Fillmore and Baker, 2009) and Wikidata (Vrandečić and Krötzsch, 2014), which serve as a general-purpose semantic labeling system comprising over 15,000 event

types. A sentence is first processed by annotating words as triggers if they are included in the Propbank predicate annotations. Then, the event type (referred to as roleset in Propbank) of each trigger is mapped to Wikidata QNode types via DWD Overlay (Spaulding et al., 2023). After this step, we obtain the initial annotations of event triggers and their corresponding candidate event types.

Event Trigger Filtering. The event annotation rules in semantic knowledge bases, such as Propbank, differs from the event definitions in mainstream EE datasets (Walker, 2005). For example, some adverbs or adjectives are treated as event triggers in Propbank, which leads to the issue of unreasonable trigger annotation. Thus, this step leverages LLMs to filter out invalid triggers. By observing these differences in event definition and describing them as guidelines, we instruct LLMs to evaluate the validity of event annotations. To help the LLMs better comprehend the task, carefully written examples are also included in the prompt. Due to potential biases in the filtering process provided by different LLMs, a voting strategy is applied to obtain the final results. Specifically, we aggregate the valid events identified by each LLM. An event is considered valid if the majority of LLMs support it. If there is a tie during the voting, we will instruct each LLM to re-annotate the case until the majority of LLMs support the result.

Event Type Refinement. In the original GLEN annotation, 60% of triggers have more than one event type. Distantly supervised methods cannot assign more fine-grained types to these triggers, as this requires more accurate event definitions and a deeper semantic understanding of the sentence. Therefore, this step uses LLMs to refine the event types to a more fine-grained level automatically.

We formalize the event type refinement task as a multiple-choice question for the LLMs. Specifically, this step takes the candidate event types from event trigger and their corresponding descriptions as input. It instructs the LLMs to select the event types most accurately aligned with the event triggers and the sentence. Additionally, we include a "None of them" option for cases where none of the candidate types align with the event trigger. Each LLM performs this task independently, and the final fine-grained event type is determined by selecting the candidate with the highest number of votes from different LLMs. If there is a tie during the voting, each LLM is instructed to re-annotate the case. However, since the most probable fine-grained types typically converge within one or two candidates, such ties are generally resolved within two rounds of iterative voting.

Event Argument Annotation. After the fine-grained event type is selected, we need to annotate the event arguments based on the event type and its corresponding schemas. As mentioned above, distant supervised methods often fail to conduct argument annotations when the relevant arguments are absent from the knowledge bases. Thus, we adopt LLMs to annotate arguments and their roles. However, annotating event arguments introduces two additional challenges for LLMs: 1) **Roles Understanding:** Event argument annotation requires LLMs to comprehensively understand the complex event schema, including event types and role definitions. Additionally, LLMs must be capable of analyzing syntactic structures within sentence spans and mapping each span to a specific role. 2) **Bias in Span Offset:** LLMs exhibit inherent bias in span offset when dealing with different styles of text. Different LLMs are more likely to give out different offsets in argument annotation.

Considering the additional challenges, we develop a set of guidelines for analyzing logical relationships in sentences and assigning roles accordingly. Then, LLMs are adopted to identify logical relationships within sentences, and map text spans to their corresponding roles. Additionally, we include examples of manual annotations in the prompts to help the LLM better understand the roles of events. To eliminate bias in span offset across different LLMs, we instruct the LLMs to refine and update the original annotation results, especially in span offset, using the rules derived from human observations. We refer to this process as **Offset Alignment**, which ensures greater consistency in the annotations generated by different LLMs. Following this alignment, we employ a voting strategy to determine the final argument annotations. For each argument in the annotated event, if a specific argument-role pair appears in more than half of LLMs' annotation results, it is deemed valid. If the LLMs generate completely different annotations for certain roles, we employ GPT-4o to annotate the case given the original annotation from each LLM. The details of voting strategies are listed in Appendix D.3.

Data Source	Event Typess	Argument Types	Cases	Event Mentions	Argument Mentions	Domain
ACE 2005	33	22	3,869[1]	4,090	9,683	General
CASIE	5	26	5,166	3,027	6,135	Cybersecurity
PHEE	2	16	4,827	5,019	7,129	Biochem
Commodity EE	18	19	3,949	3,949	8,123	Commodity
GLEN	3,465	-	208,454	185,047	-	General
Maven-arg	162	612	4,480	98,591	290,613	General
EEMT	3,465	6,297	208,454	170,849	465,856	General

Table 1: A comparison of the EEMT dataset statistics with other datasets.

3.2 The Constructed EEMT Dataset

Dataset Construction. As some datasets already pre-annotate event triggers, we reuse the GLEN to construct a more comprehensive dataset. Considering that GLEN does not contain schema with role types, we use the original event schema from DWD Overlay, which contains 6,297 different role types. Specifically, we employ three mainstream LLMs for collaborative annotation: Deepseek-V3 (DeepSeek-AI et al., 2024), Qwen-Plus (Team, 2024b), and GPT-4o-mini (OpenAI et al., 2024).

We follow the splits of GLEN for the training, valid, and test sets. Additionally, we select 1,500 samples from the test set and create a **human-annotated** test set, with the help of three graduate students specializing in NLP. The LLM-annotated test set will reflect the relative performance of models on large-scale data, both currently and over a recent period. The human-annotated test set will serve as a no-bias test set independent of the annotation LLM and be used to evaluate both the annotation quality and extraction methods. More discussion and details are provided in Appendix A and D.

The statistics of the EEMT dataset and existing datasets are shown in Table 1. Compared to commonly used datasets, EEMT surpasses them in the size of event types, role types, and data scale. Particularly, EEMT contains 10 times more event and role types than the largest human-annotated EE dataset, MAVEN-arg. Compared to GLEN, we filter out nearly 7.64% of the unreasonable event triggers, refine the coarse-grained type annotation (which accounts for 61.3% in the original GLEN) into fine-grained ones, and annotate the arguments along with their roles. More statistics are in Appendix E.

Quality Assessment. To evaluate the effectiveness of the proposed annotation framework, we assess the annotation quality of each of the three LLM-based steps on the human-annotated test set. The results are in the last column of Table 2. The F1 scores for all steps surpass 85%, with the event type refinement achieving an impressive F1 score of 96.2%. These results strongly validate the high consistency between our collaborative annotation framework and human annotations. Additionally, we evaluate the results annotated by a single LLM. By leveraging collaborative annotation, the F1 score improves at each step, further mitigating the biases inherent in the single LLM and enhancing the quality of the annotations.

	Deepseek-V3†	Qwen-Plus†	GPT-4o-mini†	CA
ETF	92.1	91.4	91.6	93.2
ETR	95.8	94.9	94.2	96.2
EAA	84.7	83.5	84.2	85.3

Table 2: Results between different single-LLMs and collaborative annotation framework(denoted as CA). ETF, ETR, EAA indicate Event Trigger Filtering, Event Type Refinement, and Event Argument Annotation, respectively. † indicates that we apply the offset alignment in EAA. We calculate the F1 score for each step.

4 LLM-based Partitioning Extraction

Considering many applications involve training open-source smaller LLMs, there is a need to equip the small-parameter LLMs with the capabilities to handle massive event types from scratch. And the LLM-based EE methods require placing all event schemas in single prompt, which leads to an excessive prompt. To alleviate the prompt length limitation, we propose an LLM-based partitioning extraction method for EE, called LLM-PEE. As shown in Figure 3, LLM-PEE consists of three key components: similarity-based type recall, type-partitioning prompting, and LLM-based event extraction. In the similarity-based type recall step, we reduce the number of candidate event types that may appear in a sentence to a small subset using a similarity-based model. Next, the type-partitioning prompt divides the subset into several partitions us-

[1]The statistics are from IEPILE.

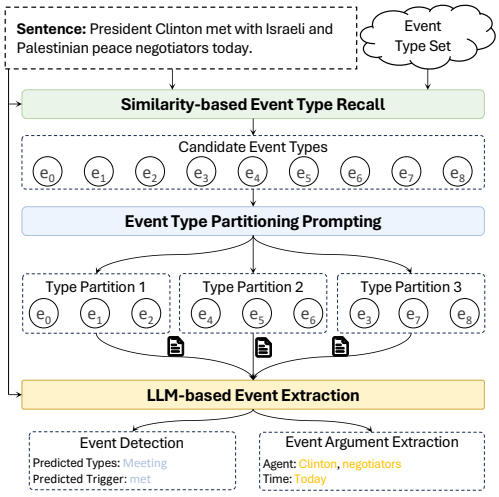

Figure 3: Overview of the proposed LLM-based Partitioning Extraction Method.

ing three different strategies, thus further reducing the prompt length of each partition. Finally, based on the partitioning prompts, LLMs extract event triggers and their corresponding arguments.

Similarity-based Event Type Recalling. Given a sentence $s = s_1, ..., s_i, ..., s_n$ and a set of candidate event types $\{e_1, ..., e_i, ..., e_m\}$, the similarity-based event type recalling step identifies the potential event types described by the sentence based on the similarity between the sentence and the event types. Following CDEAR (Zhan et al., 2023), we employ ColBERT (Khattab and Zaharia, 2020) as the encoder of sentence and event types. Based on the embeddings from encoders, we calculate the similarities and get k candidate event types for each sentence. More details are in Appendix B.1.

Event Type Partitioning Prompting. Existing LLM-based EE methods (Guo et al., 2023; Zhu et al., 2024) typically adopt prompting learning (Liu et al., 2023b). Their prompts generally contain three parts, i.e., event schema, task description, and the input sentence. Although the number of event types is reduced to k after the event type recalling, the prompt describing the event schema information remains long, particularly for some open-source LLMs with relatively short context lengths. Moreover, some studies (Liu et al., 2023a) have shown that as the prompt length increases, the difficulty of understanding the prompt also increases. Motivated by this, the event type partitioning prompting step splits the event types into smaller partitions to mitigate this issue. To determine which event types within a partition enhance LLM performance, we design three partitioning strategies based on the confidence scores obtained in the similarity-based event type recall step: 1) **Random**: The k event Types are evenly divided into N parts randomly; 2) **Average**: The k event types are evenly divided into N parts, ensuring that the sum of the confidences for event types in each part is as equal as possible, which aims to ensure that the extraction difficulty is balanced across different partitions. 3) **Level**: Sort the k event types based on their confidences, and then evenly divide them into N parts. This strategy ensures different partitions have varying difficulty levels. We analyze different partition strategies in Appendix C.2.

LLM-based EE. With the prompt as input, we use LLMs to conduct extraction. Specifically, we conduct the two-stage extraction process following KnowCoder (Li et al., 2024). For the ED task, with the partitioning prompts as input, we generate the triggers and corresponding event types. For the EAE task, with the golden event types and triggers in the single sentence as input, LLMs predict potential arguments and their corresponding roles. The details of the extraction prompt, including schema, instruction and completion, are in Appendix F.

5 Experiment

5.1 Experiment Setting

Metrics. Following GLEN (Zhan et al., 2023) and KnowCoder (Li et al., 2024), we use Trigger Identification (TI) and Classification (TC) F1 to evaluate ED. For the EAE, we use Argument Identification (AI) and Classification (AC) F1.

Baselines. For the ED task, following GLEN, we use four classification-based methods(denoted as C.B. Method), including DMBERT (Wang et al., 2019), token-level classification and span-level classification, and CDEAR (Zhan et al., 2023). For the EAE task, we use two classification methods, CRF-Tagging and Tag-Prime (Hsu et al., 2023), as well as a generative method, Bart-Gen (Li et al., 2021) (denoted as G.B. Method). For LLM-based methods, we compare with three 7B-level ones, including InstructUIE (Wang et al., 2023a), IEPILE (Gui et al., 2024), and KnowCoder (Li et al., 2024). All of them are fine-tuned on EEMT. We also evaluate the LLMs used to annotating events, Qwen-Plus, GPT-4o-mini and DeepSeek-V3. For fairness, we evaluate the three LLMs with same similarity-

Method	TI (LLM-Annotated)			TC (LLM-Annotated)			TI (Human-Annotated)			TC (Human-Annotated)		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
CA	100.0	100.0	100.0	100.0	100.0	100.0	92.1	94.3	93.2	89.8	90.5	90.1
C.B. Method												
DMBERT	50.7	74.2	60.2	32.2	47.3	38.3	49.8	74.0	59.5	31.7	47.0	37.9
Token-Level	55.5	73.3	63.1	35.5	49.2	41.2	56.0	73.1	63.4	35.9	48.6	41.3
Span-Level	53.2	74.7	62.1	33.5	50.2	40.5	53.6	73.5	62.0	62.0	49.2	39.5
CDEAR	60.0	78.2	67.9	45.1	55.2	49.7	61.2	77.7	68.5	46.3	54.5	50.1
G.B. Method												
Qwen-Plus	52.1	76.1	61.8	42.4	69.7	52.7	51.5	75.5	61.2	42.0	68.6	52.1
GPT-4o-mini	50.3	78.7	61.4	43.6	68.2	53.2	49.8	78.7	61.0	42.6	67.2	52.1
Deepseek-V3	49.7	**81.2**	61.7	43.9	**72.1**	54.6	49.6	**80.5**	61.4	43.0	**71.7**	53.8
InstructUIE	72.9	72.3	72.6	53.5	55.5	54.5	73.5	74.4	73.9	53.9	54.9	54.4
IEPILE	74.6	71.9	73.2	56.0	54.3	55.1	74.8	72.3	73.5	55.2	53.5	54.3
KnowCoder	74.5	72.3	73.4	57.2	57.4	57.2	74.1	73.7	73.9	56.0	56.4	56.2
LLM-PEE	**76.1**	72.9	**74.5**	**61.9**	59.9	**60.9**	**76.9**	72.4	**74.6**	**60.7**	57.9	**59.3**
LLM-PEE w/o. e.t.r	74.4	47.6	58.1	56.3	43.5	49.1	73.7	47.1	57.4	55.4	42.9	48.4
LLM-PEE w/o. t.p.p	74.5	72.7	73.6	57.3	57.2	57.2	73.6	73.7	73.6	55.9	55.7	55.8

Table 3: Performance (in percentage) for ED both on the LLM-annotated and human-annotated test sets of EEMT. We apply the similarity-based event type recalling (denoted as e.t.r) for all generation based models. t.p.p indicates the event type partitioning prompting.

Method	AI (LLM-Annotated)			AC (LLM-Annotated)			AI (Human-Annotated)			AC (Human-Annotated)		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
CA	100.0	100.0	100.0	100.0	100.0	100.0	90.1	89.5	89.8	86.4	84.2	85.3
C.B. Method												
CRF-Tagging	25.8	24.9	25.3	24.4	23.6	24.0	24.8	24.7	24.7	23.8	23.2	23.5
Tag-Prime	29.1	25.6	27.3	27.8	23.4	25.4	28.8	25.1	26.8	27.4	22.8	24.9
G.B. Method												
Qwen-Plus	69.4	68.1	68.8	65.8	64.5	65.1	67.4	66.9	67.1	63.0	61.7	62.3
GPT-4o-mini	70.4	67.9	69.1	66.0	65.0	65.5	67.9	66.9	67.4	63.7	61.2	62.4
Deepseek-V3	71.9	68.5	70.2	66.4	65.2	65.8	68.0	68.7	68.3	64.1	62.2	63.1
Bart-Gen	38.0	37.2	37.6	35.0	35.4	35.2	36.8	36.5	36.6	33.6	34.9	34.2
InstructUIE	65.8	64.7	65.2	60.1	59.1	59.6	64.0	63.6	63.8	59.1	57.6	58.3
IEPILE	66.8	66.5	66.7	62.1	59.7	60.9	66.8	64.2	65.5	62.1	59.7	60.9
KnowCoder	72.1	68.2	70.1	65.1	62.4	63.7	68.7	67.6	68.2	63.8	60.2	61.9
LLM-PEE	**74.7**	**73.3**	**73.9**	**69.9**	**67.6**	**68.7**	**72.9**	**70.1**	**71.5**	**66.9**	**64.8**	**65.8**

Table 4: Performance (in percentage) for EAE both on the LLM-annotated and human-annotated test sets of EEMT.

based event type recalling in ED and remove the offset alignment in EAE. Besides, we evaluate other mainstream LLMs in Appendix C.4.

Benchmark. For supervised evaluation, we evaluate the methods both on the LLM-based testset and human-annotated testset. We evaluate methods on ACE 2005 (Walker, 2005) under the zero-shot setting to evaluate the generalization capabilities.

Implement Details. We utilize LLaMA2-7B-Base (Touvron et al., 2023) as the backbone. Specifically, we use LoRA (Hu et al., 2021) and set the LoRA rank to 8. The maximum sequence length is set to 2048, and the batch size is set to 256. Training is conducted for 4 epochs. We employ greedy search and a maximum output length of 500, using vllm (Kwon et al., 2023). We recall the top 15 similar event types in the similarity-based event type recalling stage and divide them into two partitions with the level strategy.

5.2 Experiment Results

We conduct the supervised and zero-shot evaluation and provide more detailed analysis in Appendix C.

5.2.1 Supervised Evaluation

The results of the supervised evaluation are listed in Tables 3 and 4. We conduct a comprehensive analysis from the next four perspectives.

Analyses on the Annotation Framework. The collaborative annotation framework (denoted as CA) is presented in the first row of Tables 3 and 4. First, the collaborative annotation framework achieves an F1 score of 85% even on the human-annotated test set, demonstrating the high consistency between our collaborative annotation framework and human annotations. Furthermore, our collaborative annotation framework outperforms the direct extraction performance of any single annotation model, thereby validating the rationality

and efficacy of the design underlying our method. Specifically, in the ED task, there is a significant performance gap (90.1 v.s.56.2) between the collaborative annotation framework and individual annotation models. This disparity arises because **distant supervision** is incorporated as auxiliary information during the annotation process, whereas directly performing event detection using individual LLMs remains highly challenging. In the EAE task, the performance of a single LLM significantly declines due to the absence of a collaborative mechanism and offset alignment. This observation aligns with the results reported in Table 2, further reinforcing the advantages of our method that mitigates bias and enhances the consistency of dataset styles.

Analyses on ED. Table 3 presents the results of ED on the proposed EEMT dataset. Firstly, the relative performance of different models on the LLM-annotated test sets remains consistent with that on the human-annotated test sets, validating the reasonableness of LLM-annotated test set. Compared with the KnowCoder (the generation method based on LLM), LLM-PEE outperforms 1.7% on TI and 5.4% on TC, respectively. These results demonstrate the effectiveness of our partitioned extraction framework in addressing event detection with massive types. In comparison with CDEAR (designed for ED with massive types), LLM-PEE integrates prompting learning into the framework, effectively unleashing the potential of LLMs in event detection. Besides, without fine-tuning on the dataset, mainstream LLMs tend to over-generate events, which leads to high recall but low precision. LLM-PEE demonstrates superior event prediction accuracy, leading to a higher F1 score compared to the mainstream LLMs (e.g., DeepSeek-V3-671B). Additionally, the relatively low performance on TC highlights the inherent difficulty of accurately identifying event types from a large-scale schema, which remains a challenge for our dataset.

Analyses on EAE. Table 4 presents the results of EAE on the proposed EEMT dataset. Compared to KnowCoder, LLM-PEE achieves improvements of 3.9% in AI and 6.2% in AC, respectively. In LLM-PEE, the schema representations of ED and EAE are consistent. We guess that this consistency enables the ED task to facilitate the EAE task, thereby deepening the model's understanding of event types and roles and contributing to higher performance on both tasks. In comparison with the mainstream LLMs, after fine-tuning on the dataset, the argument extraction performance of LLM-PEE (trained on LLaMA2-7B) surpasses that of the original annotation LLMs. This improvement can be attributed to the high-quality dataset after the offset alignment and collaborative annotation. Additionally, classification-based methods often face challenges in accurately identifying the boundaries of arguments, particularly for continuous spans (e.g., multi-token arguments) in roles such as "purpose", which result in lower performance in both AI and AC. That suggests generation-based methods might be more suitable for the complex EAE task.

Ablation Analysis. To verify the effectiveness of LLM-PEE, we conduct ablation experiments by removing its two key modules: similarity-based event type recalling (denoted as LLM-PEE w/o e.t.r) and event type partitioning prompting (denoted as LLM-PEE w/o t.p.p). The results are presented in the bottom two rows of Table 3. Since the EAE task needs to specify event types and triggers in advance, we focus our ablation analysis solely on the ED task. LLM-PEE w/o e.t.r exhibits a significant drop of 18.3% in TC. It suggests that, without recalling the most similar event types, the model struggles to accurately distinguish the correct event type from the event set, particularly for unseen event types. Furthermore, LLM-PEE w/o t.p.p shows a performance decline of 5.1% in TC, further validating the effectiveness of the partitioning strategies in our framework. More detailed results of the partitioning strategies are provided in Appendix C.2. Additionally, to verify the effectiveness of LLM-PEE in other EE datasets, we conduct an additional supervised experiment on ACE 2005. The results are listed in Appendix C.6.

5.2.2 Zero-Shot Evaluation

Method	TI	TC	AI	AC
Qwen-Plus	20.7	14.7	35.1	25.4
GPT-4o-mini	20.0	13.7	35.1	25.6
Deepseek-V3	20.5	**15.6**	34.9	27.0
LLM-PEE	**22.3**	13.2	**38.8**	**30.5**

Table 5: Performance (in percentage) for ED and EAE tasks on ACE 2005.

To ensure that the constructed EEMT data does not rely solely on LLM-induced patterns and can exhibit generalization, we evaluate the distilled LLM-PEE on ACE 2005, an unseen dataset that contains event types derived from EEMT. The re-

sults are listed in Table 5. LLM-PEE outperforms other LLMs in TI, AI, and AC. In TI and AI, LLM-PEE demonstrates superior consistency in identifying the spans of trigger and argument. This improvement can be attributed to the high quality of the EEMT dataset, which significantly mitigates biases in span offsets. And the 12.9% improvements in AC can be attributed to the capabilities in understanding new event schemas rather than simply capturing the patterns present in the annotating LLMs. Though LLM-PEE exhibits slightly lower performance in TC compared to other methods, this is because it predicts more fine-grained event types than those in the original ACE 2005 schema. The results convincingly demonstrate the event diversity of the EEMT dataset and the generalization capabilities of the LLM-PEE method.

Model	Supervised		Zero-shot	
	TC	AC	TC	AC
LLM-PEE w.GLEN	60.3	52.0	21.1	12.5
LLM-PEE w.EEMT	**74.7**	**58.3**	**23.1**	**13.7**

Table 6: Performance comparison of LLM-PEE under supervised and zero-shot settings.

5.2.3 Comparison with original GLEN

Since we use the GLEN as the original annotation corpus, we further validate whether the quality of the dataset has been improved by training LLM-PEE with two different corpus. Considering that the GLEN only includes event annotations for the ED task, we report the ED performance includes both supervised and zero-shot settings, and the results are summarized in Table 6. These results indicate that LLM-PEE trained on EEMT outperform when trained on GLEN. We attribute this to the role of event trigger filtering and event type refinement, which lead to predicting more reasonable triggers and classifying them into the correct event types. The same issue also affects zero-shot evaluation on unseen event types. Besides, we conduct a case study in Appendix C.5 to further illustrate that training with EEMT can predict more fine-grained event types than GLEN.

6 Conclusion

In this paper, we proposed a novel LLM-based collaborative annotation framework. It refines trigger annotations from distant supervision and then performs argument annotation through collaboration among multiple LLMs. We created the new EEMT dataset based on the annotation framework, which is the largest EE dataset in terms of event types and data scale. To further adapt smaller LLMs for EE with massive types, we introduced a Partitioning EE method for LLMs called LLM-PEE. The results in both supervised and zero-shot settings demonstrated that distilled LLM-PEE with 7B parameters outperformed other baselines and even surpasses mainstream LLMs, which shows its effectiveness.

Limitations

We summarize the limitations of this work and look at them as areas for future improvement.

- **Hierarchical level of event extraction**. We believe that an important factor restricting our model is that the event hierarchy is not sufficiently distinguished. We will explore how to improve the understanding of the hierarchical level of events by better event definition or positive and negative sample strategy.
- **End-to-End event extraction**. The proposed LLM-PEE method still divides the event extraction into two sub-tasks ED and EAE, we try to proceed directly with end-to-end event extraction based on LLMs.
- **Document level event extraction**. The proposed EEMT dataset is only annotated on the sentence-level instance. The difficulties of document-level annotation lies in the to enable the LLMs to fully associate the document with massive event types. The common annotation framework splits the document into several sentences, and we are trying to annotate large-scale events in documents from document-self.

7 Acknowledgments

This work is partially funded by the Strategic Priority Research Program of the CAS under Grants No. XDB0680102, National Key Research and Development Program of China under Grants No. 2024YFC3308200, National Natural Science Foundation of China under grants 62306299 and 62441229, the Lenovo-CAS Joint Lab Youth Scientist Project and the project under Grants No. JCKY2022130C039. We thank anonymous reviewers for their insightful comments and suggestions.

References

AI@Meta. 2024. Llama 3 model card.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–419, Vancouver, Canada. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, and Bochao Wu et. al. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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*. Association for Computational Linguistics.

Charles J Fillmore and Collin Baker. 2009. A frames approach to semantic analysis.

Google. 2024. Gemini 2.0 flash. `https://gemini.google.com`. Accessed: 12/2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and Abhinav Pandey et.al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yucan Guo, Zixuan Li, Xiaolong Jin, Yantao Liu, Yutao Zeng, Wenxuan Liu, Xiang Li, Pan Yang, Long Bai, Jiafeng Guo, et al. 2023. Retrieval-augmented code generation for universal information extraction. *arXiv preprint arXiv:2311.02962*.

Cuiyun Han, Jinchuan Zhang, Xinyu Li, Guojin Xu, Weihua Peng, and Zengfeng Zeng. 2022. Duee-fin: A large-scale dataset for document-level event extraction. In *Natural Language Processing and Chinese Computing: 11th CCF International Conference, NLPCC 2022, Guilin, China, September 24–25, 2022, Proceedings, Part I*, page 172–183, Berlin, Heidelberg. Springer-Verlag.

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. *Preprint*, arXiv:2108.12724.

I-Hung Hsu, Kuan-Hao Huang, Shuning Zhang, Wenxin Cheng, Premkumar Natarajan, Kai-Wei Chang, and Nanyun Peng. 2023. Tagprime: A unified framework for relational structure extraction. *Preprint*, arXiv:2205.12585.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Paul R Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*, pages 1989–1993.

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*, pages 611–626.

Meisin Lee, Lay-Ki Soon, and Eu-Gene Siew. 2021. Effective use of graph convolution network and contextual sub-tree forcommodity news event extraction. *arXiv preprint arXiv:2109.12781*.

Minzhi Li, Taiwei Shi, Caleb Ziems, Min-Yen Kan, Nancy Chen, Zhengyuan Liu, and Diyi Yang. 2023a. Coannotating: Uncertainty-guided work allocation between human and large language models for data annotation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Peng Li, Tianxiang Sun, Qiong Tang, Hang Yan, Yuanbin Wu, Xuanjing Huang, and Xipeng Qiu. 2023b. Codeie: Large code generation models are better few-shot information extractors. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 15339–15353. Association for Computational Linguistics.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. *Preprint*, arXiv:2104.05919.

Zixuan Li, Yutao Zeng, Yuxin Zuo, Weicheng Ren, Wenxuan Liu, Miao Su, Yucan Guo, Yantao Liu, Xiang Li, Zhilei Hu, et al. 2024. Knowcoder: Coding structured knowledge into llms for universal information extraction. *arXiv preprint arXiv:2403.07969*.

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.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023a. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

David et al Milne. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, and Adam Perelman et. al. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Taneeya Satyapanich, Francis Ferraro, and Tim Finin. 2020. Casie: Extracting cybersecurity event information from text. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8749–8757.

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.

Elizabeth Spaulding, Kathryn Conger, Anatole Gershman, Rosario Uceda-Sosa, Susan Windisch Brown, James Pustejovsky, Peter Anick, and Martha Palmer. 2023. The DARPA Wikidata overlay: Wikidata as an ontology for natural language processing. In *Proceedings of the 19th Joint ACL-ISO Workshop on Interoperable Semantics (ISA-19)*, pages 1–10, Nancy, France. Association for Computational Linguistics.

Zhaoyue Sun, Jiazheng Li, Gabriele Pergola, Byron C. Wallace, Bino John, Nigel Greene, Joseph Kim, and Yulan He. 2022. Phee: A dataset for pharmacovigilance event extraction from text. *Preprint*, arXiv:2210.12560.

Qwen Team. 2024a. Qwen2.5: A party of foundation models.

Qwen Team. 2024b. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Meihan Tong, Bin Xu, Shuai Wang, Meihuan Han, Yixin Cao, Jiangqi Zhu, Siyu Chen, Lei Hou, and Juanzi Li. 2022. Docee: A large-scale and fine-grained benchmark for document-level event extraction. *2022 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

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.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

C. et al Walker. 2005. *ACE 2005 Multilingual Training Corpus*. LDC corpora. Linguistic Data Consortium.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, et al. 2023a. Instructuie: Multi-task instruction tuning for unified information extraction. *arXiv preprint arXiv:2304.08085*.

Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. 2019. Adversarial training for weakly supervised event detection. In *Proceedings of the 2019 Conference of the North*.

Xiaozhi Wang, Hao Peng, Yong Guan, Kaisheng Zeng, Jianhui Chen, Lei Hou, Xu Han, Yankai Lin, Zhiyuan Liu, Ruobing Xie, Jie Zhou, and Juanzi Li. 2023b. Maven-arg: Completing the puzzle of all-in-one event understanding dataset with event argument annotation. *Preprint*, arXiv:2311.09105.

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. *Preprint*, arXiv:2004.13590.

Xingyao Wang, Sha Li, and Heng Ji. 2022. Code4struct: Code generation for few-shot event structure prediction.

Ying Zeng, Yansong Feng, Rong Ma, Zheng Wang, Rui Yan, Chongde Shi, and Dongyan Zhao. 2017. Scale up event extraction learning via automatic training data generation. *CoRR*, abs/1712.03665.

Qiusi Zhan, Sha Li, Kathryn Conger, Martha Palmer, Heng Ji, and Jiawei Han. 2023. Glen: General-purpose event detection for thousands of types.

Kechi Zhang, Huangzhao Zhang, Ge Li, Jia Li, Zhuo Li, and Zhi Jin. 2023. Toolcoder: Teach code generation models to use api search tools. *Preprint*, arXiv:2305.04032.

Zixuan Zhang and Heng Ji. 2021. Abstract meaning representation guided graph encoding and decoding for joint information extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

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

Yuxin Zuo, Wenxuan Jiang, Wenxuan Liu, Zixuan Li, Long Bai, Hanbin Wang, Yutao Zeng, Xiaolong Jin, Jiafeng Guo, and Xueqi Cheng. 2024. Alignxie: Improving multilingual information extraction by cross-lingual alignment. *arXiv preprint arXiv:2411.04794*.

A Discussion

A.1 The reasonableness of Annotation Framework

Similar to other LLM-annotating methods and their datasets (Li et al., 2023a; Zhang et al., 2023), the annotation framework is independent of certain annotating LLMs, more powerful LLMs can replace the original annotating LLMs. Considering the effectiveness and cost, we select the Qwen-Plus, DeepSeek-V3 and Gpt-4o-mini as our annotating LLMs here. As the improvements of LLMs, we believe that it can lead to a dataset with higher quality under the proposed annotation frameworks. With this dataset, we can produce more powerful models.

A.2 The reasonableness of EEMT

The reasonableness of EEMT is listed here through three major parts of the datasets.

LLM-annotated Training Set Considering many practical applications involve training smaller models, domain-specific systems, or fine-tuned architectures. Thus, they rely on the annotated datasets. Moreover, existing models can be further improved by finetuning on the annotated datasets. As shown in Tables 3 and 4, the 7B-parameters model LLM-PEE finetuned on the training set outperforms larger, closed-source LLMs (e.g., GPT-4) by 5.5 points in ED and 2.7 points in EAE. In Table 5, the LLM-PEE achieves up to 12.9% improvement compared to larger LLMs (DeepSeek) in the zero-shot setting. Regarding there may be noisy annotations in the training dataset, this is a common phenomenon of the model-generated datasets in IE and other domains. For example, some studies in RE and EE rely on Distant Supervision (Chen et al., 2017; Zeng et al., 2017), which develops their models based on noisy datasets. Similarly, recent studies in LLM-based data synthesis also utilize LLM-generated data to train open-source models (Li et al., 2023a; Zhang et al., 2023). These studies demonstrate that LLM-annotated training data are both valuable and practically useful, although they are noisy. Therefore, the LLM-annotated training set provides large-scale, low-cost event annotations that enable the development of practical EE systems.

LLM-annotated Valid and Test Set The meaningfulness of LLM-annotated valid and test sets lie in its ability to reflect the relative performance of models, both currently and over a recent period, on large-scale data. From the experimental results in Tables 3 and 4, the results show that the relative performance of different models on the LLM-annotated test sets remains consistent with that on the human-annotated test sets, despite the presence of noise in the former one. Besides, the statistic in Appendix E shows that the LLM-annotated test set expands the coverage compared to the human-annotated one.

If more powerful LLMs emerge, we can observe inconsistent performance between the human-annotated test set and LLM-annotated test set (i.e., higher performance on the former but lower on the latter). In such a case, the LLM-annotated test set fails to reflect the relative performance of these emerging LLMs in real-world scenarios. Therefore, it would be necessary to update the LLM-annotated test set to ensure its results are consistent with the human-annotated test sets.

Human-annotated Test Set Human-annotated test set serves as the ground truth for evaluating the performance of all the existing or future EE methods, and assessing the quality of LLM-annotated test sets. We conduct all experiments on both the human-annotated and LLM-annotated test sets.

When more powerful LLMs emerge, improvements in LLM capabilities can be detected through performance on the human-annotated test set. It can also serve as an indicator to measure whether the emerging LLMs are more powerful in the EE tasks. However, as illustrated earlier, the human-annotated test sets are limited in the data scale,

which requires more specialized annotators to conduct the annotation.

B LLM-based Partitioning Extraction

B.1 Similarity-based Event Type Recalling

ColBERT consists of a BERT (Devlin, 2018) layer, a convolution layer and an L2 normalization layer. In this paper, we use $\mathrm{ColBERT}(\cdot)$ to denote the encoder. Specifically, the embedding list of all tokens in a sentence s, $h_s = [\mathbf{h}_1^s, \mathbf{h}_2^s, ...]$, is calculated as follows:

$$h_s = \mathrm{ColBERT}(\text{[CLS][SENT]}s_1, s_2, ...\text{[SEP]}), \quad (1)$$

where [SENT] is a special token indicating the object being encoded is a sentence.

For an event type e, the corresponding embedding list of all tokens in the event type name, $h_e = [\mathbf{h}_1^e, \mathbf{h}_2^e, ...]$, is calculated as follows:

$$h_e = \mathrm{ColBERT}(\text{[CLS][EVENT]}\tau_i\text{[SEP]}), \quad (2)$$

where [EVENT] is a special token indicating the object being encoded is an event type.

Then, the similarity score between sentence s and event type e is computed as the sum of the maximum similarity between the token embeddings in sentence and event type: $\rho_{(s,e)} = \sum_{h_s} \max_{h_e}(\mathbf{h}_i^e, \mathbf{h}_j^s)$.

A similar margin loss to CDEAR is adopted for training, which ensures that the best candidate is scored higher than all negative samples.

$$\mathcal{L} = \frac{1}{N} \sum_{s} \sum_{e^-} \max\{0, (\tau - \max_{e \in C_y} \rho_{(e,s)} + \rho_{(e^-,s)})\}. \quad (3)$$

Based on the similarity score, we can get k candidate event types for each sentence.

C Experiment

C.1 Bias Analysis

Due to the dual use of both annotating and extraction stages, we analyze the potential biases from each stage.

C.1.1 Annotation Stage

To assess whether our annotation framework introduces significant pattern biases, we evaluated the annotation performance on a human-annotated test set. The F1 scores of the proposed LLM-based collaborative annotation on this test set are presented in Tables 3 and 4 (denoted as CA in the first line), reaching 90.1 in ED and 85.3 in EAE. These results validate the high consistency between our collaborative annotation framework and human annotations, confirming that our annotation approach does not introduce substantial annotation biases.

Furthermore, the results in Table 2 demonstrate that our collaborative annotation framework maintains high consistency with human annotations across all annotation steps. Specifically, we evaluated the F1 scores for each step—Event Type Filtering (ETF), Event Type Refinement (ETR), and Event Argument Annotation (EAE), which reached 93.2, 96.2, and 85.3, respectively, all exceeding 85%. This further substantiates the robustness of our method in mitigating annotation biases at each stage of the process.

C.1.2 Extraction Stage

To examine whether our annotation framework affects the generalization ability of the trained model, we trained our model using the automatically annotated training set and evaluated its performance on both LLM-annotated and human-annotated test sets. The results, presented in Tables 3 and 4 , indicate that the performance gap between the LLM-annotated and human-annotated test sets is minimal (e.g., for TC, 60.9 in LLM-annotation vs. 59.3 in human-annotation; for AC, 68.7 in LLM-annotation vs. 65.8 in human-annotation). These findings suggest that our method does not introduce any significant bias.

To further evaluate the generalization capabilities of our model and ensure it is not overly reliant on biases present in the training data, we conducted evaluations on an unseen dataset with no biases from the training data. The results in Table 5 indicate that our model significantly outperforms other annotation LLMs, such as Deepseek-V3, by an average of 12.9% in TI, AI, and AC. This demonstrates that our model has developed a deeper understanding of unseen event schemas beyond merely capturing the patterns present in the annotation LLMs, underscoring its strong generalization ability in handling complex event extraction tasks.

C.2 Influence of partitioning strategy

We conduct the experiment on ED to figure out the influence of the partitioning strategy in Table 7. Our strategy outperforms IEPLIE[2], because we in-

[2]IEPILE method builds a hard negative dictionary, however, this method leads to train-test inconsistency in our

corporate the information of the sentence when recalling the similar event types and ensures consistency between the training and testing phases. Besides, it demonstrates that the Level strategy significantly enhances the precision of event extraction. We hypothesize that sorting samples in descending order of confidence enables the grouping of the most challenging-to-distinguish event types into the same partition. This approach facilitates the model's ability to learn fine-grained distinctions between different event types, including their corresponding triggers and classifications. Notably, the inclusion of naturally occurring challenging negative samples within these partitions allows the model to better grasp the nuanced boundaries between similar or ambiguous event categories.

Partitioning strategy	**TI**	**TC**
IEPILE	73.7	58.2
Random	74.3	58.5
Average	74.7	58.7
Level	**75.2**	**59.3**

Table 7: Results with different partitioning strategy.

C.3 Generalization on various backbones

We further evaluate the generalization ability of LLM-PEE across a range of backbones, including different sizes of LLaMA (AI@Meta, 2024) and Qwen (Team, 2024a). As shown in Table 8, under the supervised setting, smaller LLMs (0.5B/1.5B/3B) achieve performance relatively close to that of the larger 7B model. This suggests that with sufficient labeled data, compact models are able to capture task-specific patterns effectively, narrowing the gap with larger backbones. In contrast, under the zero-shot setting, the performance disparity between small and large models becomes much more evident. For instance, the AC score drops from 30.9 with Qwen2.5-7B to 21.1 with Qwen2.5-0.5B, while TC decreases from 14.9 to 12.6. This indicates that smaller models, despite being competitive in supervised scenarios, exhibit substantially weaker generalization ability when explicit task-specific supervision is absent.

C.4 Evaluation on Mainstream LLMs

To further evaluate how the mainstream LLMs perform on our dataset, we conduct an extra evaluation on our human annotation benchmark. We use the LLAMA3.1-405B-instruct (Grattafiori et al., 2024), Gemini-2.0-flash (Google, 2024), GPT-4o, Qwen-Max to conduct further evaluation. The results are listed in 9.

dataset. We built the hard negative dictionary and employed hard negative sampling during the training phase, and during testing, the ranked samples were randomly allocated to simulate the effect of the IEPLIE method as closely as possible.

Model	Supervised		Zero-shot	
	TC	AC	TC	AC
LLM-PEE w.Qwen2.5-0.5B	56.1	61.8	12.6	21.1
LLM-PEE w.LLaMA3.1-1.5B	56.9	63.6	12.5	26.5
LLM-PEE w.Qwen2.5-3B	57.1	64.1	12.4	27.2
LLM-PEE w.Qwen2.5-7B	59.9	67.5	14.9	30.9

Table 8: Performance of LLM-PEE with various backbones.

Model	TI	TC	AI	AC
LLama-3.1-405B	59.6	51.9	65.1	60.2
Gemini-2.0-flash	60.5	53.1	67.4	62.9
Qwen-Turbo	58.3	49.7	64.9	59.3
Qwen-Plus	61.3	52.8	67.0	62.3
Qwen-Max	61.3	53.4	67.5	62.4
GPT-4o-mini	60.1	52.1	67.4	62.5
GPT-4o	60.6	52.7	67.7	62.7
Deepseek-V3	61.4	53.8	68.3	63.1

Table 9: Results on different mainstream LLMs.

The results suggests that without training, the mainstream LLMs exhibit almost the same annotation and extraction capabilities, which is still a challenging task for mainstream LLMs. Considering the effectiveness and efficiency, we employ the Qwen-Plus, GPT-4o-mini and DeepSeek-V3 as the final annotation LLMs.

C.5 Case Study

We further analyze whether more fine-grained event type can be predicted based on the EEMT, we randomly selected a subset of examples from the test set for verification. As shown in Table 10, we observed that compared to the original GLEN dataset, when trained on the EEMT, the model predictions shifted from "Assessment" to "Risk Assessment" and from "Occupation" to "Military occupation", which better aligns the sentence. Upon analyzing the training data, we find that the key is the increase in the number of corresponding events in the train set, the number of "Risk assesement" increases from 0 to 5, and the number of "Military occupation" increases from 1 to 4, which results in more accurate fine-grained event predictions.

Context	Predictions
In the most recent Third **Assessment** Report (2001), IPCC wrote there is new and stronger evidence that most of the warming observed over the last 50 years is attributable to human activities.	GLEN: Assessment EEMT: Risk Assessment
The **occupying** armies existing in german territory will end soon.	GLEN: Occupation EEMT: Military Occupation

Table 10: Comparison of the prediction results across different datasets.

Model	TI	TC	AI	AC
IEPILE	72.9	72.4	69.9	68.2
KnowCoder	73.1	72.3	70.9	68.6
LLM-PEE	73.3	72.6	71.3	69.2

Table 11: Results under the supervised evaluation in ACE 2005.

C.6 Supervised Evaluation on other dataset

Considering the generality of LLM-PEE on other datasets, we apply LLM-PEE on the supervised setting in the dataset ACE 2005 in Table 11. Compared to other fine-tuned methods, LLM-PEE achieves state-of-the-art performance. However, the margin of advantage is narrower than when applied to the EEMT dataset, which contains a significantly larger number of event types. This reduction in relative performance can be attributed to the core design philosophy of our model, which primarily focuses on addressing the challenges posed by long prompts resulting from massive event types. When applied to datasets with fewer event types, such as ACE 2005, the inherent advantages of LLM-PEE are somewhat diminished, as the problem it was specifically designed to tackle becomes less pronounced.

D Annotation Details

D.1 Annotation Setting

We use the DeepSeek-V3 (DeepSeek-AI et al., 2024), Qwen-Plus (Team, 2024b) and GPT-4o-mini as the annotation LLMs on web-based API services, costing approximately $500. In terms of licensing for using LLMs to conduct annotation, GPT-4o-mini is governed by the OpenAI Proprietary License [3], DeepSeek-V3 [4] follows the MIT License, and Qwen-Plus is governed by the Qwen License [5]. All three licenses permit the use of these LLMs as annotation tools for the development of non-commercial datasets. Considering the accuracy of the annotation and the divergence of the answers, we set the temperature to 0.5. Besides, we set an answer detection mechanism, if the obtained result can not be correctly parsed, the model is required to annotate the events again. Following GLEN (Zhan et al., 2023), the annotation corpus are AMR Annotation Release 3.0 [6] and OntoNotes Release 5.0 [7] from LDC (Linguistic Data Consortium), each of which has specific licenses for access and usage.

D.2 Prompts Examples

We introduce the prompts we used in the Collaborative annotation framework including Event Triggers Filtering, Event Type Refinement and Event Argument Annotation (two-stage) in Table 13, 14, 15 and 16.

D.3 Voting Strategies Details

After different stages, we employ different voting strategies. Considering that we employ majority voting, an odd number of annotating LLMs is needed. In this paper, we take three LLMs as an example, and it can be easily adapted to more annnotating LLMs.

For voting after Event Trigger Filtering, each LLM independently judges the validity of each trigger. For example, in a three-LLM voting scenario, if there are more than two LLMs that consider the event to be reasonable, it will consider the event to be reasonable.

For voting after Event Type Refinement, we tally and vote on the most potential event type identified by each LLM, selecting the event with the highest number of votes as the corresponding fine-level event. In cases where a tie occurs (e.g., LLM1: A, LLM2: B, LLM3: C), we apply a repeated voting procedure until one type secures a majority (e.g., LLM1: A, LLM2: B, LLM3: B). We set the maximum number of voting rounds at five to prevent excessive iterations. However, in practice, ambiguous potential events were typically limited

[3] https://openai.com/policies/row-terms-of-use

[4] https://deepseeklicense.github.io

[5] https://github.com/QwenLM/Qwen/blob/main/Tongyi%20Qianwen%20LICENSE%20AGREEMENT

[6] https://catalog.ldc.upenn.edu/LDC2020T02

[7] https://catalog.ldc.upenn.edu/LDC2013T19

Data Source	Cases	Event Mentions	Argument Mentions
Train	187,468	158,005	430,629
Dev	10,359	6,465	18,459
Test	10,627	6,379	16,768
Human Annotation	1,500	973	2,546

Table 12: The detailed statistics of EEMT.

to two or three, with most reaching a definitive outcome within two voting rounds.

For voting after Event Argument Annotation, since each role corresponds to a list of arguments, we first count the length of the argument list from different annotations, (e.g., LLM1: [A, B], LLM2: [A, C], LLM3: [C, D]) and set 2 as the true length of the argument list for this role because most of the annotations support that there are 2 arguments corresponding to this role. Then, we check each argument in the annotations (e.g., A, B, C, D). Since A and C receive more than half of the votes, we consider A and C as valid arguments and discard B and D. In extreme cases, if the length of argument list after voting does not match the true length of the argument list (e.g., LLM1: [A, B], LLM2: [A, C], and LLM3: [A, D]), the voting result is [A], and the length of argument list is 1, which does not match the true length of the argument list 2. We assume this case as a hard case, and we provide three LLMs' annotations to GPT-4o for re-annotation using the prompt in Table 14. However, this situation occurs infrequently (less than 5% of instances), and most roles are not annotated by 3 different arguments in a single instance. Therefore, we do not overly rely on GPT-4o. The offset alignment prompt with multiple input are listed in Table 17.

E Dataset Statistics

E.1 The split of dataset

We introduce the spilt of our dataset in this section. For the train/dev/test set, we follow GLEN's 90/5/5 setting, besides, to further improve the quality of our dataset. We manually annotate 1,500 cases as the human annotation set.

E.2 Arguments Distribution

Furthermore, we analyze the distribution of argument in Figure 4 and categorize the role types into five broad classes: "agent", "entity", "location", "purpose", and "others". Agent, location, and entity are the three most frequently occurring argument roles, aligning with real-world distributions. This alignment facilitates knowledge sharing of similar roles across different event schemas, thereby reinforcing the validity of our dataset construction.

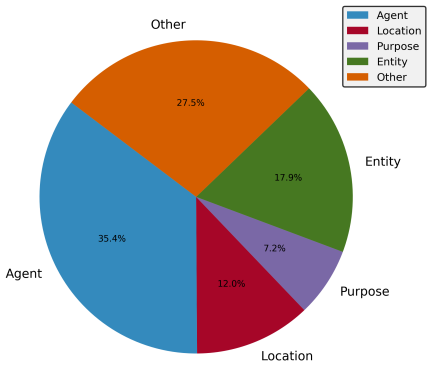

Figure 4: The argument distribution in our dataset.

F Prompt for Training

We present examples of training prompts for both EAE and ED task, including instruction and completion in figure 5 and 6.

More specifically, we employ the same Python class style prompt with KnowCoder (Li et al., 2024), which includes the schema and instruction. Besides, we use the same output representation as the object of a certain class of event type. Following the KnowCoder, we adopt class comments to provide clear definitions of concepts, which includes the event definition and several samples from the training set. However, while the KnowCoder uses different schema representation for ED and EAE tasks, we use the same representation, which could better facilitate the model understanding about a certain event type.

PROMPT FOR EVENT TRIGGER FILTERING.

Identify Reasonable Propbank Event Annotations in a Sentence

Task Overview: Your goal is to determine which **Propbank event annotations** are reasonable for the given sentence. If an annotation aligns with the context and semantics of the sentence, it is considered reasonable. Output the reasonable annotations in the specified format. If no reasonable annotations are found, output "None of them."

Guidelines:
1. **Understanding Event Annotations**: Each Propbank event annotation consists of two parts: the **event trigger ** (outside the parentheses) and the **Propbank event type** (inside the parentheses). The trigger word is the specific word in the sentence that activates the event, while the Propbank event type defines the semantic category of the event. For example, in 'robbery(rob.01)', "robbery" is the trigger, and 'rob.01' is the Propbank event type. Ensure you understand both components when evaluating the reasonableness of the annotation.
2. **Event Annotation Analysis**: Review the definitions and usage patterns of the Propbank event annotations provided. Understand their semantic boundaries and structural characteristics.
3. **Sentence Context Evaluation**: Assess whether the sentence provides sufficient context to support each Propbank event annotation. Consider whether the verb or action described in the sentence matches the annotation's definition.
4. **Reasonableness Assessment**: Check if the Propbank event annotation is used appropriately in the given context and whether it matches the event definition in a clear and complete manner.
5. **Adverb/Adjective Trigger Exclusion**: Be aware that some adverbs, adjectives, or partial verbs may be annotated as triggers by Propbank, leading to the generation of events. However, these annotations may not align with how humans typically define events. Such cases should be considered **unreasonable** and excluded from the final output.

Output Format:
Adhere to the specified format for clarity and consistency. Follow the **Example Output Format** to provide the **Reasonable Annotations** and **Reasoning**.

Example
Input: **Sentence**: Generosity is voluntarily giving money on your own free will.
Propbank Annotations: 'voluntarily(volunteer.01)'
Output:
Reasonable Annotations: None
Reasoning:
The event 'voluntarily(volunteer.01)' is not reasonable because "voluntarily" is an adverb describing the manner in which the action of giving occurs. It does not represent an event or action itself but rather modifies the act of generosity.

Input
Sentence: Five years ago they were negotiating with Milosevic at Dayton to stop the war.
Propbank Annotations: 'negotiating(negotiate-01)', 'war(war-01)'
Output

Table 13: Prompt for Event Trigger Filtering.

PROMPT FOR EVENT TYPE REFINEMENT.

Identify the Event Type Based on the Trigger Word

Task Overview: Your goal is to determine whether the given trigger matches any of the listed event types. If a match is found, output the serial number of the corresponding event type. If no match is found, output the last option.

Guidelines:
1. **Event Type Analysis:** Thoroughly review the definitions of all listed event types to grasp their semantic boundaries and structural characteristics.
2. **Trigger Evaluation:** Assess whether the trigger word aligns with the semantic and contextual requirements of any event type. If the trigger does not align with any event type, output the last option.
3. **Contextual Validation:** Analyze the sentence in which the trigger word appears to ensure the match is precise and meaningful.
4. **Context Reasonableness Analysis:** Check if the trigger word is used appropriately in the given sentence and whether it matches the event definition in a clear and complete manner.
5. **Annotation Accuracy Evaluation:** Assess whether the annotation of the trigger word is reasonable based on the event definition and sentence.
6. **Output Format:** Adhere to the specified format for clarity and consistency. Follow the Example Output Format to provide the Event Type and Reasoning.

Example
Input: **Event Types**: A. state_crime: crime perpetrated by a sovereign state
B. war_crime: serious violation of the laws of war
C. crime: unlawful act forbidden and punishable by criminal law
D. corporate_crime: crimes committed either by a corporation or its representatives
E. environmental_crime: illegal act which directly harms the environment
F. ethnic_cleansing: systematic removal of a certain ethnic or religious group
G. alcohol-related_crime: criminal activities that involve alcohol use
H. cybercrime: any crime that involves a computer and a network
I. criminal_case: investigation case under criminal law
J. None of them.
Trigger: crime
Sentence: this is not because their race or culture is wrong or bad in any way, it 's simply a fact that certain groups have higher rates of crime.

Output: **Event Type**: C
Reasoning: The trigger word "crime" aligns with the core semantics of the crime event type (Event Type C). The context discusses crime in general terms without specifying a sovereign state (state_crime), war-related violations (war_crime), or any specific subtype such as corporate, environmental, or cybercrime. Thus, it best fits the generic "crime" definition.

Input **Event Types**:
A. negotiation: dialogue between two or more people or parties intended to reach a beneficial outcome
B. parley: type of diplomatic meeting held between enemies
C. None of them.
Trigger: crime
Sentence: Five years ago they were negotiating with Milosevic at Dayton to stop the war.
Output

Table 14: Prompt for Event Type Refinement.

PROMPT FOR EVENT ARGUMENT ANNOTATION.

Annotating the Event Arguments based on the Schema.

Task Overview: Your goal is to identify and annotation the arguments related to a specific event type in the given text. Each event type has specific roles (arguments), and you are required to annotation the arguments according to the provided event type and its corresponding argument roles. Do not introduce new roles or categories that are not part of the given event type and argument roles.
Guidelines:

1. **Understand Event Type and Argument Roles:** Before starting the annotation process, make sure you understand the provided event type and the roles (arguments) associated with it. Do not assume new roles or make connections outside the given structure. Only use the roles provided in the event type definition.
2. **Exact Match with Roles:** Each argument should align directly with one of the provided roles. For example: - If the event type is **Attack Event**, the roles are **Agent**, **Target**, **Location**, **Time**, **Cause**, and **Method**. You must only annotate arguments that fall under these categories. Do not annotate anything as **Purpose** or any role not explicitly listed.
3. **Contextual Accuracy:** Ensure that the arguments labeled in the text match the defined roles and are appropriate for the context. The roles should be matched based on both the meaning and the context of the text.
4. **Multiple Entities per Role:** Some roles may correspond to multiple entities. For example, **Agent** could involve more than one person or entity (e.g., ‘[’A’, ’B’]‘). In such cases, include all relevant entities under the role.
5. **Output Format:** Provide the annotated arguments in the following JSON format. Ensure that the roles and arguments strictly correspond to those defined for the given event type.

Example
Example Input:
Event Type: creation
Event Description: process during which something comes into being and gains its characteristics
Trigger: create
Argument Roles: agent_creator, result_thing_created, material_materials_used, location
Sentence: they create a market for themselves .
Example Output:
“‘json {
"agent_creator": ["drug pushers"],
"result_thing_created": ["market"],
"material_materials_used": [],
"location": [],
}
“‘json

Input
Event Type: parley
Event Description: type of diplomatic meeting held between enemies
Trigger: negotiating
Argument Roles: Negotiator, Other party, Location
Sentence: Five years ago they were negotiating with Milosevic at Dayton to stop the war.

Table 15: Prompt for Event Argument Annotation.

PROMPT FOR EVENT OFFSET ALIGNMENT.

Refining the Annotation with the Rules

Task Overview: You will be provided with an initial event argument extraction result. Your task is to analyze this result according to the given annotation rules. Based on the rules, you will need to ensure the arguments are correctly selected or re-generate the correct extraction if necessary. The final output should be a **single JSON** string containing the event arguments, formatted according to the rules.

Rules for Evaluation and Modification:

1. **Logical Consistency:** - Ensure that each argument logically participates in the event. For example, the **Agent** typically refers to the entity executing the trigger, while the **Entity** refers to the recipient or affected party of the trigger.

2. **Accuracy of Extraction:** - Arguments must be extracted concisely and clearly. If a role is extracted as "A and B," it should be represented as ‘["A", "B"]‘ rather than a combined form. - Avoid redundancy or unnecessary details. - The arguments must be extracted from the context.

3. **Conciseness:** - Arguments should be represented as single words or phrases. If the extraction includes a sentence, reduce it to the smallest meaningful segment (e.g., remove unnecessary words like "that").

4. **Avoid Redundant or Overlapping Arguments:** - Check for any overlapping or redundant arguments. If an overlap exists, determine whether it is logical or whether one of the arguments should be removed or merged.

5. **Argument Selection:** - Use the provided **Input** to determine which argument is the most accurate or appropriate for each role. - In cases where the input is not fully accurate, modify or re-generate the argument extraction to match the correct role.

6. **Final Output Format:** - Provide the final output in **JSON format**, with each event role and its arguments clearly defined. Do not include the event type or trigger word.

Example:

Example Input:

Event Type:

- **Event Type:** Attack
- **Event Definition:** action to injure another organism

Event Trigger: attacked

Event Roles Definition:

- **Roles:** Agent, Location, Time, Target

Sentence: John and Sarah attacked the enemy base at night.

Input:

{ "Agent": ["John and Sarah"], "Target": ["the enemy base"], "Time": ["at night"], "Location": ["the enemy base"] }

Example Output:

{ "Agent": ["John", "Sarah"], "Target": ["enemy base"], "Time": ["night"], "Location": [] }

Input

Event Type:

- **Event Type:** parley
- **Event Definition:** type of diplomatic meeting held between enemies

Event Trigger: negotiating

Event Roles Definition:

- **Roles:** Negotiator, Other party, Location

Sentence: Five years ago they were negotiating with Milosevic at Dayton to stop the war.

Input: {

{ "Negotiator": ["They"], "Other party": ["Milosevic"], "Location": ["at Dayton"], }

Output

Table 16: Prompt for Event Offset Alignment.

PROMPT FOR EVENT OFFSET ALIGNMENT WITH MULTIPLE INPUT.

Task Overview: You will be provided with three initial event argument extraction results, named **Input1**, **Input2**, and **Input3**. Your task is to analyze and combine these results according to the given annotation rules. Based on the rules, you will need to choose the most appropriate argument for each role or re-generate the correct extraction if necessary. The final output should be a **single JSON** string containing the event arguments, formatted according to the rules.

Rules for Evaluation and Modification:

1. **Logical Consistency:** - Ensure that each argument logically participates in the event. For example, the **Agent** typically refers to the entity executing the trigger, while the **Entity** refers to the recipient or affected party of the trigger.

2. **Accuracy of Extraction:** - Arguments must be extracted concisely and clearly. If a role is extracted as "A and B," it should be represented as ‘["A", "B"]‘ rather than a combined form. - Avoid redundancy or unnecessary details.

3. **Conciseness:** - Arguments should be represented as single words or phrases. If the extraction includes a sentence, reduce it to the smallest meaningful segment (e.g., remove unnecessary words like "that").

4. **Avoid Redundant or Overlapping Arguments:** - Check for any overlapping or redundant arguments. If an overlap exists, determine whether it is logical or whether one of the arguments should be removed or merged.

5. **Argument Selection:** - Compare **Input1**, **Input2**, and **Input3** to determine which argument is the most accurate or appropriate for each role. If there is disagreement, use the rules to guide which argument should be selected. - In cases where none of the inputs is fully accurate, combine or re-generate the most accurate argument extraction.

6. **Final Output Format:** - Provide the final output in **JSON format**, with each event role and its arguments clearly defined. Do not include the event type or trigger word.

—

Input Format: You will be given: - **Input1, Input2, Input3:** Three JSON objects containing the initial event argument extractions. - **Event Roles Definition:** A list of roles defined for the event (e.g., Agent, Target, Location, etc.). - **Context:** The original text from which the arguments are extracted.

Output Format: A **single JSON object** with the final, modified extraction, including the most accurate and logically consistent arguments for each role.

Example:

Input:

Event Roles Definition: - **Roles:** Agent, Location, Time, Target

Context: "John and Sarah attacked the enemy base at night."

Input1: “‘json "Agent": ["John"], "Location": ["enemy base"], "Time": ["at night"], "Target": [] ”’

Input2: “‘json "Agent": ["John", "Sarah"], "Location": ["enemy base"], "Time": ["night"], ”’

Input3: “‘json "Agent": ["John and Sarah"], "Location": ["enemy base"], "Time": ["night"], ”’ **Output:** “‘json "Agent": ["John", "Sarah"], "Location": ["enemy base"], "Time": ["night"], "Target": [] ”’

Explanation:

-The Agent role was chosen as ["John", "Sarah"] since both individuals participated in the action.

-The Location and Time were correctly extracted as "enemy base" and "night," respectively. -The Target role remains empty as no direct target is mentioned in the context.

Table 17: Prompt for Event Offset Alignment with multiple input.

```
-----------------------------------------instruction-----------------------------------------
class Event:
    """
    The base class for all events.
    """
    def __init__(self, trigger: str, arg_names, *args):
        self.trigger = trigger
        self.arguments = {}
        for arg_name, arg_values in zip(arg_names, args):
            self.arguments[arg_name] = arg_values

class Writing(Event):
    """
    Description: activity of composing a text, narrating through writing
    Examples: written, writer, Write, writes, Writers, writ, writings
    """
    def __init__(self, trigger: str, arg_names, *args):
        arg_names = ["writer", "thing_written", "benefactive", "location"]
        super().__init__(trigger = trigger, arg_names = arg_names, *args)
class Web_writing(Event):
    """
    Description: online writing
    Examples: written, writer, Write
    """
    def __init__(self, trigger: str, arg_names, *args):
        arg_names = ["writer", "thing_written", "benefactive", "location"]
        super().__init__(trigger = trigger, arg_names = arg_names, *args)

class Script(Event):
    """
    Description: style of handwritten language
    Examples: script, scripting, scripted, scripts
    """
    def __init__(self, trigger: str, arg_names, *args):
        arg_names = ["writer", "thing_written", "benefactive", "attribute_of_arg_1", "location"]
        super().__init__(trigger = trigger, arg_names = arg_names, *args)

class Write_off(Event):
    """
    Description: reduction in recognized value of an entity
    Examples:write down, write
    """
    def __init__(self, trigger: str, arg_names, *args):
        arg_names = ["changer_of_monetary_value", "thing_with_value", "amount_of_change",
"old_high_value", "new_lower_value", "location"]
        super().__init__(trigger = trigger, arg_names = arg_names, *args)

"""
This is an object-oriented programming task from Dataset others: Some Event Classes are defined
above. Please instantiate all the corresponding Event Objects in the following sentence.
"""

sentence = "Who wrote the hymn `` Amazing Grace '' ?"
-----------------------------------------completion-----------------------------------------

results = [
    Writing(trigger = "wrote")
]
```

Figure 5: An example of training data in ED task.

```
--------------------------------------------instruction--------------------------------------------
class Entity:
    """
    The base class for all entities.
    """
    def __init__(self, name: str):
        self.name = name
class Event:
    """
    The base class for all events.
    """
    def __init__(self, trigger: str):
        self.trigger = trigger

class Departure(Event):
    """
    Description: act of departing or something that has departed
    Examples: Examples:Departed, departure, departs, departed, depart, departing, Departing
    """
    def __init__(self, trigger: str, arg_names, *args):
        arg_names = ["entity_leaving", "departed_from", "departed_for", "location"]
        super().__init__(trigger = trigger, arg_names = arg_names, *args)
"""
This is an object-oriented programming task: Some Event Classes are defined above. Please
instantiate all the corresponding Event Objects in the following sentence. It is important to note
that the triggers of the events are confirmed as follows: "departing" is the trigger of event type
"departure".
"""

sentence = "Former President Clinton drew cheers from admirers at a US Air Force base outside
Washington today before departing for his home in New York , and although Mr. Clinton is out of
office , he says he 'll still be around ."

--------------------------------------------completion--------------------------------------------
results = [
    Departure(trigger = "departing",
              entity_in_motion = [Entity("Former President Clinton")],
              starting_point_location_vacated = [Entity("US Air Force base outside Washington")],
              destination_must_also_be_a_location = [Entity("New York")])
]

```

Figure 6: An example of training data in EAE task.