Retrieval, Reasoning, and Generation: A Universal Format Generation Method for Few-Shot Event Argument Extraction

Pengfei Yin[1,2], Hao Li[1,2], Boxiang Hu[1,2], Xixun Lin[1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences;
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{yinpengfei, lihao1998, huboxiang, linxixun}@iie.ac.cn

***Abstract*—Event Argument Extraction (EAE) is essential for identifying event participants and their roles. However, in few-shot settings, existing generative approaches typically heavily rely on manually crafted, ontology-specific prompts and formats, which are labor-intensive, error-prone, and exhibit poor transferability across domains and event schemas. To address these limitations, we propose FGEE, a universal JSON-based generation framework for EAE. FGEE tackles the data scarcity through three key innovations: (1) Unified I/O formats, which represent complex event structures with a standardized JSON schema, thereby eliminating per-ontology prompt engineering; (2) Pseudo-output retrieval, which retrieves structurally similar pseudo-outputs to improve task understanding and mine salient event clues; and (3) Key sentence-based reasoning, which performs trigger-guided analysis to focus on informative evidence and deduce arguments from redundant contexts. We further introduce an N-Event-K-Shot evaluation framework to assess EAE under strict few-shot conditions. Experiments on ACE05, RAMS and WikiEvents datasets show that FGEE consistently outperforms the latest few-shot supervised models, yielding about 2.73 improvement in F1 across all benchmarks. These results show that FGEE substantially improves the robustness and generalization of generative EAE in low-resource environments.**

***Index Terms*—Event Argument Extraction, Information Extraction, Natural Language Processing, Large Language Models.**

I. Introduction

Event argument extraction (EAE) aims to extract event arguments from a given event and identify the specific roles they play, which is an important sub-task of event extraction [1], [2]. By providing structured representations of event information, EAE can support a wide range of downstream applications, such as machine reading comprehension and dialog system [3]. As illustrated in Figure 1, given the trigger word *evacuated* for the evacuationrescue event, an event argument extractor is expected to identify *90 people*, *hospital*, and *remote area* as the event arguments, and predict their roles as passenger, destination, origin.

From a task formulation perspective, existing EAE methods can be broadly categorized into extractive and generative methods [4]. Extractive methods typically employ dedicated classification modules to identify and classify event arguments directly from the input context [1]. In contrast, generative methods formulate EAE as a sequence-to-sequence problem and generate all event arguments jointly [5]. Owing to the strong contextual reasoning capabilities of pre-trained language models, generative methods have witnessed rapid development in recent years [6]. Early generative methods rely on natural-language templates to guide models in generating arguments for predefined role slots [7]. Subsequent work use specific delimiters to distinguish different arguments and roles [8]. Considering the similarity between the event structure and the code, Wang et al. [9] completes the sentence-level EAE tasks through code generation, using the code to represent the event argument structure as a class object and generate the corresponding arguments.

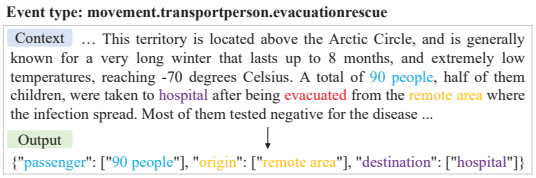


Fig. 1: An example of event argument extraction from RAMS dataset. Red fonts represent trigger words, other color fonts represent arguments.

To facilitate language models' understanding of event structures and task formulations, and to prevent the generation of unnecessary content, these methods usually carefully design specific generation formats and prompts for each event type, as well as the use of complex decoding strategies to derive arguments from the output. However, such ontology-specific, handcrafted designs are labor-intensive and brittle, and they generalize poorly across event schemas and domains, making the overall pipeline hard to maintain and extend. This greatly hinders the application of the model in real-world scenarios. The emergence of large language models (LLM) represents a significant advancement in natural language processing [10], [11], designing a universal generation format to apply LLM to EAE tasks conveniently is an urgent and worth exploring research trend.

In this paper, we explore multiple generation formats for EAE tasks, including TEXT, CODE, and JSON. Based on this, we propose a novel **F**ormat **G**eneration method for

Corresponding author: huboxiang@iie.ac.cn

EAE tasks (FGEE), which can be easily applied to different event ontologies. Specifically, we design the universal input and output format based on JSON, which can efficiently represent complex event structures. We propose the pseudo-output retrieval strategy that innovatively uses pseudo-output as the criteria to retrieve similar events, helping the model understand task forms and mine event clues. A reasoning module based on the key sentence is proposed to guide the model to start from the trigger word and enhance the ability to reason the arguments in redundant context. Furthermore, taking events as the basic unit, we construct N-Event-K-Shot low-resource scenarios based on hierarchical event types to facilitate Few-Shot EAE evaluation. We conduct experiments on multiple LLMs and datasets, and FGEE achieves excellent performance in sentence-level and document-level EAE tasks. Specifically, our method achieves new state-of-the-art (SOTA) performance, achieving up to 2.81, 1.43, and 3.96 Arg-C F1 improvements on ACE05, RAMS, and WikiEvents.

We summarize our contributions as follows:

- We propose a universal EAE method FGEE based on JSON generation that can be easily migrated to different event ontologies.
- Pseudo-output retrieval strategy and reasoning module based on the key sentence are proposed to provide the model with relevant event clues and help it understand the task form, enhancing its argument extraction ability.
- We construct the N-Event-K-Shot scenario based on hierarchical event types to facilitate the evaluation of Few-Shot EAE tasks.
- Extensive evaluations on three benchmark sentence-level/document-level EAE datasets demonstrate our method achieves significant performance gains across diverse scenarios.

II. Related Works

A. Extractive Event Argument Extraction

Extractive EAE usually extracts corresponding arguments as spans in the input text based on a specific task paradigm, which can be roughly divided into:

Classification-based methods: Researchers [12] adopt a classic classification pipeline: enumerate candidate arguments (mentions/spans) and score $(\text{trigger}, \text{role}, \text{span})$ tuples to select role-consistent spans. In the LLM era, classification-style DocEAE mainly enhances long-range trigger–argument dependency modeling and suppresses distracting context. APSR [13] leverages AMR semantic graphs with sparse, role-aware attention (inter-/intra-sentential encoders) to focus on long-range dependencies and filter noise for better role classification. CsEAE [14] couples a small extractor with a large model to refine boundaries and reduce cross-sentence redundant extractions.

Question-answer methods: Researchers [15] treat EAE as the QA task, extracting arguments in the context through questions constructed in natural language. A representative formulation is to cast event extraction as machine reading comprehension, where questions encode role semantics and the model extracts answer spans from context [16].

Prompt-based methods: Researchers [17] use event-specific prompt templates to fill role slots while keeping outputs as extractive spans. Recent extractive DocEAE advances include: (i) graph-augmented prompting to connect mentions across the document and prompt representations (e.g., GAM [18]); (ii) prefix/prompt tuning for stronger long-context slot querying (e.g., DEGAP [19]); (iii) multi-event prompting to jointly process multiple events and exploit inter-event correlations (e.g., DEEIA [20]); and (iv) prompt enrichment with additional same-event or cross-event clues to improve robustness [21].

B. Generative Event Argument Extraction

Benefiting from the strong reasoning and generation abilities of pre-trained language models, generative EAE extracts arguments by generating structured outputs instead of selecting spans [22]. It typically specifies an output format and then parses argument–role pairs from the generation. Researchers [7] use a TEXT-style output format to generate all arguments in context. Recent generation-based EAE further emphasizes prompt diversification and structure/ontology constraints to better model inter-argument dependencies and reduce order sensitivity. GEMS [23] adopts multi-perspective prompts with voting aggregation and ontology-driven steering, showing strong performance in low-resource settings. Inspired by the similarity between event structures and code, [24] formulates outputs as CODE to leverage programming-language constructs (e.g., inheritance and type annotations). This line is extended by schema-aware fine-tuning: KnowCoder [25] converts schemas into Python-class-style code and trains LLMs with a two-phase framework (schema understanding and schema following), achieving strong results on EAE.

III. Methodology

In this section, we first formulate the Few-Shot EAE task, then introduce the overall framework of our method, as illustrated in Figure 2, which contains three core modules: Universal Prompt Construction (§III-A), Pseudo-output Retrieval (§III-B), and Key Sentence Reasoning (§III-C).

Task Definition We first describe the task formalization of EAE. Formally, given a context $\mathbf{c} = \{c_1, ..., c_n\}$ of n words, the event trigger word $c_e \in \mathbf{c}$, the event type $e \in \mathcal{E}$ and the set of event-specific role types $\mathcal{R}^e$. The EAE task aims to extract all the arguments $\{a_1, ..., a_u\}$ related to c_e in $\mathbf{c}$ and assign a role $r \in \mathcal{R}^e$ to each extracted argument a_i, where a_i is a text span in context $\mathbf{c}$.

In the Few-Shot EAE scenario, researchers [17] perform ratio sampling on the train data $\mathcal{D}_{\text{train}}$, low resource setting ($|\mathcal{D}'_{\text{train}}| \ll |\mathcal{D}_{\text{test}}|$) can evaluate the generalization ability of the model. However, ratio sampling further worsens the long-tailed distribution of event types [26]; Researcher [27] constructs the N-Way-D-Doc scenario and proposes a complex sampling strategy to ensure that the D documents contain N roles. However, events usually contain part of the roles in the

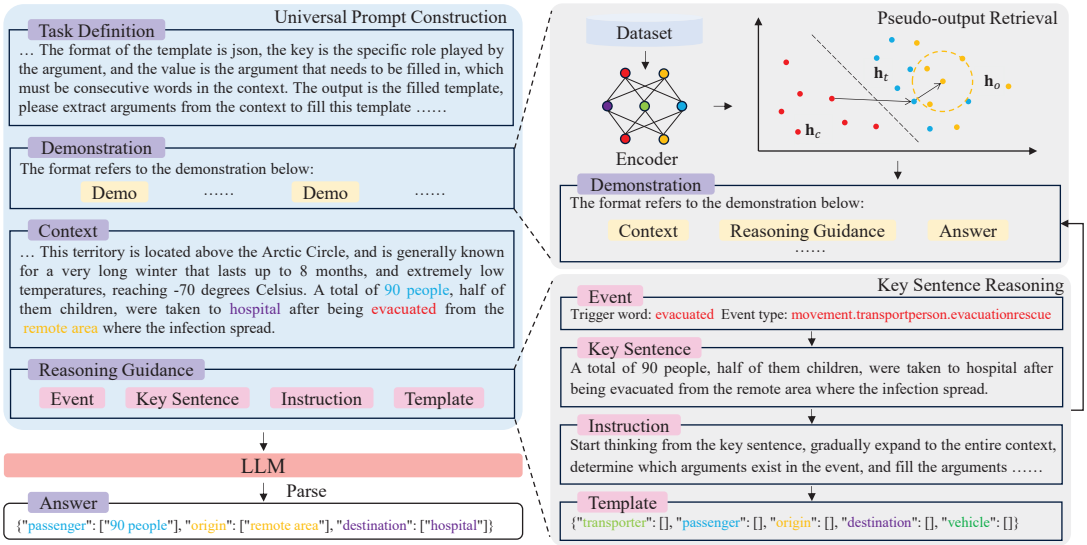


Fig. 2: The overall architecture of FGEE, where the left side is the overall process, and the right side is the Pseudo-output Retrieval module and the Key Sentence Reasoning module.

ontology, making it difficult to strictly satisfy N roles in real-world scenarios and difficult to use in practical applications.

Based on the pros and cons presented here, we utilize events as the basic unit and propose the N-Event-K-Shot low resource setting. Specifically, given the train set $\mathcal{D}_{\text{train}} = \{(\mathbf{c}_1, e_1), ..., (\mathbf{c}_K, e_1), ..., (\mathbf{c}_{N\times K}, e_N)\}, N \leq |\mathcal{E}|$, which contains N event types, each associated with K event instances, and where each event instance includes its corresponding trigger word and arguments, the goal is to extract the arguments of events in the test set $\mathcal{D}_{\text{test}} = \{(\mathbf{c}_1, e_1), ..., (\mathbf{c}_M, e_i)\}, M \gg (N \times K), i \in \{1, ..., N\}$.

A. Universal Prompt Construction

Benefiting from the language model's excellent contextual reasoning capabilities, we generatively complete the EAE tasks, parsing arguments from the output. Specifically, given the previous tokens $\mathbf{y}_{<i}$ and input $\mathbf{x}$ during the generation process, the language model models the conditional probability of selecting a new token $\mathbf{y}_i$. Therefore, the total probability $p(\mathbf{y}|\mathbf{x})$ for generating output $\mathbf{y}$ is calculated as follows:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i}^{|\mathbf{y}|} p(\mathbf{y}_i|\mathbf{y}_{<i}, \mathbf{x}). \tag{1}$$

a) Universal Input Format: By providing task information to the language model, it guides the model to understand the task form and generate arguments that exist in the context. Specifically, we construct a universal prompt format that can be easily applied to different event ontologies, which introduces event information and task instructions, including task definition $\mathbf{z}$, demonstration $\mathbf{m}_c$, context $\mathbf{c}$, reasoning guidance $\mathbf{g}_c$:

$$\mathbf{x} = [\mathbf{z}; \mathbf{m}_c; \mathbf{c}; \mathbf{g}_c], \tag{2}$$

where $[;]$ denotes the concatenation operation; $\mathbf{z}$ provides a universal explanation of the definitions of trigger word, argument, and the generation format; $\mathbf{m}_c$ includes some demonstrations to help the model understand the task form and requirements; $\mathbf{g}_c$ guides the steps of model reasoning, helps the model extract key clues in redundant context, and improves the ability of reasoning arguments.

b) JSON-based Generation Format: Considering that there are multiple argument roles in the event, and a role may have more than one argument, we define the generation format as JSON, where Key represents the argument role and Value is a list representing all arguments corresponding to the argument role. This format universally represents the event with clear structures, making it easy to migrate to other event ontologies.

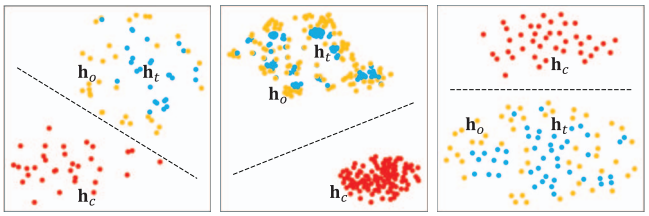
$$\mathbf{y} = \{r_1 : [a_1^1, ..., a_j^1], ..., r_i : [a_1^i, ..., a_j^i]\}. \tag{3}$$

Finally, we parse the arguments from the output $\mathbf{y}$. The correct JSON format only requires a simple format conversion to parse arguments. For outputs that include redundant text, we use heuristic rules to get the position of the JSON result in the output based on the JSON start and end symbols. If an argument appears multiple times in the context, select the argument closest to the trigger word as the final extraction result.

Benefiting from the Universal Input Format and JSON-based Generation Format, we can apply FGEE to any event ontology scenario by simply replacing the content of the specific event (context $\mathbf{c}$, trigger word c_e, event type e, and argument roles $\mathcal{R}^e$). This operation is convenient and does not require any manual participation.

B. Pseudo-output Retrieval

In in-context learning (ICL), embedding-space proximity between demonstrations and test samples enhances perfor-



(a) ACE05 (b) RAMS (c) WikiEvents

Fig. 3: Visualization of context, output template, and output representations via t-SNE [28] projection on three mainstream datasets. Red indicates the context representation $\mathbf{h}_c$, blue indicates the output template representation $\mathbf{h}_t$, and golden indicates the output representation $\mathbf{h}_o$.

mance consistency [29]. For EAE tasks, selecting demonstrations via context embedding similarity is intuitive but suboptimal due to context redundancy and distribution mismatch between long texts and compact event structures (triggers + arguments). While argument-based retrieval would be ideal, arguments remain unknown during testing. Leveraging JSON output templates ($\mathbf{t}_e$) with empty argument lists partially resolves this, as templates and outputs share embedding similarities (Figure 3). However, templates' limited representation of argument roles (without actual content) inadequately captures event nuances, particularly when event types overlap or share multiple roles in the ontology.

To this end, we propose the Pseudo-output Retrieval strategy that uses the variation between the template and output of the similar event to obtain an alternative output as the retrieval criteria. This strategy aims to retrieve events with similar semantics and structure, helping the model understand task forms and mine event clues. Specifically, we first encode the context $\mathbf{c}$ and output template $\mathbf{t}_e$:

$$\mathbf{h}_c = \text{Encoder}(\mathbf{c}), \tag{4}$$

$$\mathbf{h}_t = \text{Encoder}(\mathbf{t}_e), \tag{5}$$

where Encoder represents the encoding language model, $\mathbf{h}_c$ and $\mathbf{h}_t$ represent the embedding of context and output template. Then we concatenate these embeddings as representations of the current event $\mathbf{h}_e = [\mathbf{h}_c; \mathbf{h}_t]$, and use cosine similarity to retrieve the most similar events from the train set:

$$\text{Score}(\mathbf{h}_e^i|\mathbf{h}_e) = \frac{f(\mathbf{h}_e^i) \cdot f(\mathbf{h}_e)}{||f(\mathbf{h}_e^i)|| \times ||f(\mathbf{h}_e)||}, \tag{6}$$

$$d_s = \underset{d_i \in \mathcal{D}_{\text{train}}}{argmax}(\text{Score}(\mathbf{h}_e^i|\mathbf{h}_e)), \tag{7}$$

where f denotes the mean-pooling operation, d_s denotes the event most similar to the current event d. $d_i \in \mathcal{D}_{\text{train}}$ denotes the event in the train set, and $\mathbf{h}_e^i$ denotes the corresponding event representation.

We construct a pseudo-output representation $\mathbf{h}'_o$ for the current event d based on the variation between the template and output of similar event d_s, which replaces the ground truth output $\mathbf{h}_o$ and serves as the criteria for retrieval:

$$\mathbf{h}'_o = \mathbf{h}_t + \Delta_s, \tag{8}$$

where $\Delta_s = \mathbf{h}_o^s - \mathbf{h}_t^s$ represents the variation between the template and output of event d_s. Finally, we retrieve k similar events based on pseudo-output representation $\mathbf{h}'_o$ as demonstrations:

$$\mathbf{m}_c = \underset{d_i \in \mathcal{D}_{\text{train}}}{\text{KNN}}(\text{Score}(\mathbf{h}_o^i|\mathbf{h}'_o)), \tag{9}$$

where KNN represents the k-nearest neighbor search, $\mathbf{m}_c = [m_1, ..., m_k]$ includes k demonstrations, each demonstration $m_i = [\mathbf{c}_i; \mathbf{g}_c^i; \text{ans}_i]$ consists of context $\mathbf{c}_i$, reasoning guidance $\mathbf{g}_c^i$ and ground truth answer ans_i.

C. Key Sentence Reasoning

Document-level EAE faces heightened extraction challenges due to argument scattering. We propose a Key Sentence Reasoning (KSR) module that leverages the trigger's local context to construct reasoning chains from trigger to output, enhancing argument extraction and mitigating context redundancy. Specifically, we first introduce the reasoning path ρ:

$$\rho : (c_e, e) \rightarrow \mathbf{s}_c \rightarrow \mathbf{c}, \tag{10}$$

which guides the model to start from the trigger word c_e, think event information contained in the key sentence $\mathbf{s}_c \in \mathbf{c}$, and finally expand to the entire context $\mathbf{c}$. Then construct the reasoning guidance $\mathbf{g}_c$ based on the reasoning path ρ:

$$\mathbf{g}_c = [c_e; e; \mathbf{s}_c; \mathcal{I}; \mathbf{t}_e], \tag{11}$$

where $\mathcal{I}$ represents the instruction constructed in natural language, guiding the model to reason in the order of ρ. In the sentence-level EAE tasks, we construct the clause where the trigger word is located as the key sentence according to heuristic rules; In the document-level EAE tasks, we use the sentence where the trigger word is located as the key sentence. As shown in Figure 4, the reasoning guidance $\mathbf{g}_c$ is obtained by concating the corresponding content according to Equation 11.

We construct reasoning guidance $\mathbf{g}_c$ and $\mathbf{g}_m$ for the current event d and demonstrations $\mathbf{m}_c$ respectively, where $\mathbf{g}_c$ and $\mathbf{g}_m$ keep the same format. It will guide the model to learn the reasoning process from the reasoning guidance $\mathbf{g}_m$ of demonstrations, and then extract arguments from the redundant context based on the current reasoning guidance $\mathbf{g}_c$.

Trigger word: evacuated
Event type: contact.discussion.meet
Key sentence: A total of 90 people, half of them children, were taken to hospital after being evacuated from the remote area where the infection spread.
Instruction: Start thinking from the key sentence, gradually expand to the entire context, determine which arguments exist in the event, and fill the arguments with high confidence into the value of the corresponding argument role in the template.
Template: {"transporter": [], "passenger": [], "origin": [], "destination": [], "vehicle": []}

Fig. 4: An example of key sentence reasoning.

TABLE I: N-Event-1-Shot scenario EAE results on three mainstream datasets, where N represents the number of event types in the event ontology. The supervised learning method uses N data for training. The best and suboptimal results are marked in bold and underline fonts respectively.

Model	Method	ACE05		RAMS		WikiEvents		
		Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Head-C
Llama2-7b	TEXT	31.67	23.03	23.48	17.46	15.04	12.43	15.11
	CODE	32.29	22.89	22.74	17.16	10.86	9.16	12.12
	JSON	37.60	26.91	25.10	19.17	13.52	11.11	15.91
	FGEE (Ours)	**43.35**	**34.43**	**28.13**	**23.35**	**23.96**	**19.66**	**24.65**
Mistral-7b	TEXT	31.96	23.55	24.36	19.79	12.32	10.99	15.75
	CODE	29.01	22.18	24.28	19.10	11.75	10.48	17.16
	JSON	32.45	26.60	29.38	23.83	13.35	11.79	20.84
	FGEE (Ours)	**37.78**	**30.55**	**33.25**	**27.13**	**20.35**	**17.53**	**29.96**
Vicuna-7b	TEXT	20.93	13.45	16.63	12.48	12.76	10.96	13.62
	CODE	32.48	21.76	23.68	17.40	10.77	8.53	11.73
	JSON	34.12	23.58	25.66	19.16	12.17	10.47	16.27
	FGEE (Ours)	**41.47**	**31.50**	**28.44**	**22.10**	**22.22**	**18.88**	**24.46**
GPT-3.5	TEXT	34.93	24.17	35.16	29.28	18.51	15.72	24.11
	CODE	34.40	25.65	32.82	27.08	16.86	14.73	21.98
	JSON	38.02	30.37	38.03	31.61	18.67	16.39	25.88
	FGEE (Ours)	**43.78**	**36.68**	**41.67**	**33.71**	**29.74**	**26.89**	**37.32**
GPT-4	TEXT	36.90	29.95	34.20	29.22	17.26	15.60	26.91
	CODE	35.82	31.00	32.11	27.57	14.71	13.12	21.86
	JSON	36.42	30.43	36.96	31.41	18.09	16.51	26.42
	FGEE (Ours)	**42.09**	**37.28**	**42.21**	**36.92**	**24.61**	**22.70**	**34.76**
Supervised Learning (Few-Shot)	BART-Gen	12.02	7.90	25.32	16.80	5.76	4.65	5.02
	NNShot	29.71	22.81	16.10	12.84	13.11	10.56	12.62
	ProtoNet	34.67	24.52	17.41	12.66	16.54	12.15	13.58
	PAIE	24.81	23.07	38.73	31.69	18.73	15.47	18.42
	TabEAE	36.58	27.67	**43.01**	34.03	20.81	18.11	23.80
	FSGBA	39.13	**34.47**	40.42	**35.49**	24.19	21.81	28.73
	HERO	**41.90**	33.81	41.9	34.82	**24.75**	**22.93**	**32.59**

IV. Experiments

A. Experimental Setup

a) Datasets: We conduct experiments on three commonly used EAE datasets: ACE05 [30], RAMS [26] and WikiEvents [7]. We construct the N-Event-K-Shot low resource setting based on hierarchical event types on ACE05, RAMS, and WikiEvents datasets. We first divide the event types according to the main type and then construct a set of N event types based on the number of overlapping argument roles, giving priority to event types with the same main type. Finally, sample K data in the train set for each event type.

b) Evaluation Metrics: We adopt three evaluation metrics: **Argument Identification** (Arg-I), **Argument Classification** (Arg-C), and **Argument Head Classification** (Head-C). The F-measure (F1) score is used to evaluate the performance of the model.

Arg-I: An event argument is correctly identified if its offsets and event type match those of any of the argument mentions.

Arg-C: An event argument is correctly classified if its role type is also correct.

Head-C: For WikiEvents dataset, we follow previous work [7], additionally evaluate argument head score, which only concerns the matching of the headword of an argument.

c) Baselines: According to the generation format, three baselines are included: TEXT, CODE, and JSON. The supervised learning baselines include ProtoNet [31], NNShot [32], BART-Gen [7], PAIE [17], TabEAE [33],Fusion Selection-Generation-Based [34],HERO [35].

BART-Gen [7]: BART-Gen formulates the EAE as conditional generation following predefined manual event templates, using BART-large as the pre-trained model. **ProtoNet** [31]: Following work [27], Prototypical Networks (ProtoNet) is applied to the Few-Shot EAE task, which learns a prototype representation for each argument role and labels each query according to its distance from the prototype representation. ProtoNet follows the settings [27] and uses BERT-base as the pre-trained model. **NNShot** [32]: Following [27], Nearest Neighbor Tagger (NNShot) is applied to the Few-Shot EAE task, which is based on token-level nearest neighbor classification for Few-Shot sequence tagger tasks. NNShot follows the settings [27] and uses BERT-base as the pre-trained model. **PAIE** [17]: PAIE is a method based on prompt learning that extracts the spans corresponding to the arguments by constructing the role template.PAIE uses its manual prompts setting and uses BART-large as the pretrained model. **TabEAE** [33]: TabEAE reformulates EAE as a problem of table generation, which generates arguments in a non-

TABLE II: Ablation study results.

Method	ACE05		RAMS		WikiEvents	
	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C
FGEE	43.35	**34.43**	28.13	**23.35**	**23.96**	**19.66**
w/o reasoning	**44.20**	33.82	24.76	19.13	18.54	15.85
w/o retrieval	38.03	26.61	**29.09**	22.78	16.92	14.46

autoregressive way by constructing the slotted table prompts. TabEAE uses its manual prompts setting and uses RoBERTa-large as the pre-trained model. **FSGBA** [34]: It unifies extractive selection and generative prediction within a single encoder–decoder framework for EAE, jointly training both by aligning inputs, procedures, and outputs. It adopts manual prompts and uses BART-large as the backbone. **HERO** [35]: It frames DocEAE as rule-guided in-context learning, using fine-grained hierarchical extraction rules to boost few-shot performance without parameter tuning. It adopts manual prompts and uses Llama3-8B-Instruct [36] as the backbone.

d) Implementation Details: Our approach is based on five language models: Llama-2-7b, Mistral-7B-Instruct-v0.2, Vicuna-7b-v1.5, GPT-3.5 (`gpt-3.5-turbo-0125`) and GPT-4 (`gpt-4-0125-preview`). The encoding language model is Jina embeddings 2 (`jina-embeddings-v2-base-en`). Llama2, Mistral, Vicuna and Jina embeddings 2 available in HuggingFace Transformers library.[1] GPT-3.5 and GPT-4 are accessed through the OpenAI API,[2] and the temperature is fixed as 0. The number of demonstrations is set to 3, the maximum length is set to 4096, and top_k is set to 10. We use the Llama2 model for ablation study and experimental analysis.

B. Overall Performance

Table I compares the performance of FGEE and all baselines in the N-Event-1-Shot scenario on ACE05, RAMS, and WikiEvents datasets. There are the following observations and discussions:

(1) FGEE achieves SOTA performance, surpassing all format baselines with **6.39**, **3.60**, and **9.06** Arg-C F1 improvements on ACE05, RAMS, and WikiEvents, respectively. This demonstrates FGEE's superiority in both sentence-level and document-level EAE tasks.

(2) FGEE surpasses supervised baselines with manual prompts, achieving **2.81**, **1.43**, and **3.96** Arg-C F1 improvements on three datasets. The Pseudo-output Retrieval and Key Sentence Reasoning modules enhance event clue mining and argument deduction in redundant contexts.

(3) JSON Format Dominates: Outperforms other formats in nearly all scenarios, proving its efficiency in structured event representation. JSON's standardized syntax also ensures easier parsing and broader applicability in real-world systems.

C. Ablation Study

Table II reveals the necessity of FGEE's components: Removing the Key Sentence Reasoning module causes significant performance drops (**0.61**, **4.22**, and **3.81** Arg-C F1), particularly in document-level tasks, validating its role in guided argument deduction. Eliminating the Pseudo-output Retrieval module degrades results by **7.82**, **0.57**, and **5.20** Arg-C F1, underscoring its ability to mine critical event clues. Both modules are indispensable—reasoning drives context-aware argument generation, while retrieval enhances event understanding and extraction efficiency.

V. Analysis

A. Few-Shot Scenario

As shown in Table III, FGEE outperforms the JSON baseline (BASE) across all scenarios, achieving 5.16, 4.00, and 4.23 Arg-C F1 improvements on three datasets. Demonstrations enhance task understanding, with performance gains in multi-event (N-Event-1-Shot) scenarios (avg. **8.14**, **4.43**, and **3.21** Arg-C F1). Results confirm FGEE's ability to retrieve event clues and provide reasoning guidance, excelling in both 1-Event-K-Shot (strong intra-type relevance) and N-Event setups.

B. Impact of Demonstrations

In Figure 5, we explore the impact of different retrieval strategies and the number of demonstrations on EAE task performance. "Random" means random sampling demonstrations; "Context" means retrieval using context embedding; "Template" means retrieval using the embedding of context and output template; "Gold" means retrieval using the ground truth output embedding.

a) Impact of Retrieval Strategies: As shown in Figure 5, FGEE's pseudo-output retrieval boosts Arg-C F1 by **7.42**, **3.60**, and **7.62** on three datasets, outperforming random/context-based methods. Output-aware strategies (pseudo-output and gold output retrieval) prove event-guided retrieval enhances relevance.

b) Impact of Demonstrations Number: Multi-Demo Analysis: Multi-demo strategies outperform single-demo approaches across retrieval methods, as demonstrations enhance task/event comprehension. Performance plateaus for sentence-level EAE at ≥ 4 demos, while document-level tasks show slight degradation due to input length inflation, which weakens the model's focus on current event reasoning.

C. Efficiency Analysis

FGEE is a universal JSON-based generation framework for EAE, compatible with generative language models while maintaining practical efficiency. Although the pseudo-output retrieval module adds computational overhead, it is acceptable since retrieval operates on precomputed embedding representations of the training set, avoiding redundant computations.

D. Error Analysis

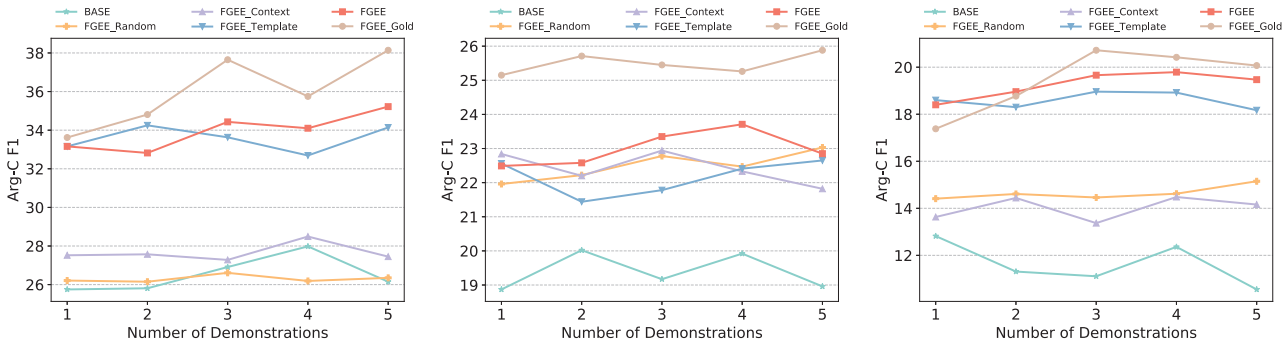
We analyze errors in language model–based EAE via JSON generation, including Key, Value, and JSON errors. As shown in Fig. 6, models achieve high format accuracy, especially GPT models, with rare JSON ($< 1\%$) and Key ($< 3\%$) errors. This

[1]https://github.com/huggingface/transformers

[2]https://openai.com/api/

TABLE III: Few-Shot event argument extraction experimental results on three mainstream datasets.

Dataset	Method	1-Event-1-Shot		1-Event-5-Shot		1-Event-10-Shot		5-Event-1-Shot		10-Event-1-Shot	
		Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C	Arg-I	Arg-C
ACE05	BASE	**41.58**	30.08	**41.97**	30.76	38.58	29.34	39.35	29.10	38.10	24.31
	FGEE	41.34	**32.69**	39.16	**31.54**	**44.26**	**35.48**	**41.77**	**34.16**	**44.22**	**35.54**
RAMS	BASE	23.50	17.24	25.16	19.62	25.82	20.72	24.29	17.91	24.57	17.98
	FGEE	**28.83**	**22.12**	**28.73**	**22.96**	**29.61**	**23.67**	**27.61**	**22.04**	**28.79**	**22.71**
WikiEvents	BASE	18.62	16.01	18.93	15.64	18.69	15.68	19.20	16.07	17.39	14.50
	FGEE	**22.60**	**18.98**	**24.45**	**21.44**	**25.21**	**21.67**	**23.99**	**19.57**	**21.14**	**17.43**



(a) ACE05 (b) RAMS (c) WikiEvents

Fig. 5: The impact of different retrieval strategies and the number of demonstrations on EAE performance.

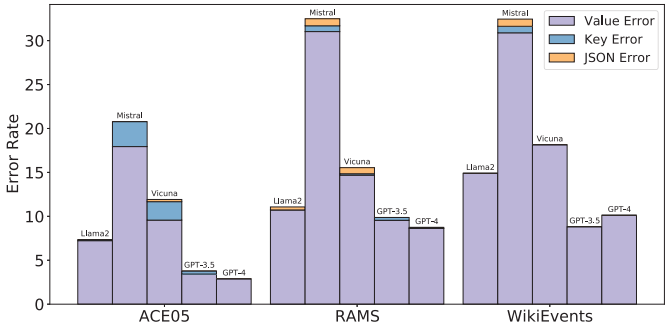


Fig. 6: Error analysis.

indicates our method effectively generates event arguments in JSON format, enabling simple parsing without manual event template design and allowing easy transfer to other event ontologies. Most errors are Value errors, where generated arguments are out of context (e.g., spelling mistakes), which can be filtered using heuristic rules.

VI. Conclusion

In this paper, we revisit document-level EAE by casting it as universal JSON-format generation. Our approach integrates pseudo-output retrieval and key-sentence reasoning to enhance evidence grounding and structured argument prediction, whose effectiveness is validated by extensive experiments. In the future, we will extend FGEE to broader event ontologies and other information extraction tasks, and further investigate its effectiveness in few-shot and low-resource scenarios.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No.U2336202, No.62402491) and the China Postdoctoral Science Foundation (No.2025M771524).

References

[1] N. Ding, C. Hu, K. Sun, S. Mensah, and R. Zhang, "Explicit role interaction network for event argument extraction," in *Findings of the Association for Computational Linguistics: EMNLP 2022*. Association for Computational Linguistics, 2022. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.254/

[2] Y. Ren, Y. Cao, F. Fang, P. Guo, Z. Lin, W. Ma, and Y. Liu, "Clio: Role-interactive multi-event head attention network for document-level event extraction," in *Proceedings of the 29th International Conference on Computational Linguistics*, 2022, pp. 2504–2514.

[3] T. Zhang, M. Chen, and A. A. Bui, "Diagnostic prediction with sequence-of-sets representation learning for clinical events," in *Artificial Intelligence in Medicine: 18th International Conference on Artificial Intelligence in Medicine, AIME 2020, Minneapolis, MN, USA, August 25–28, 2020, Proceedings 18*. Springer, 2020, pp. 348–358.

[4] É. Simon, H. Olsen, H. You, S. Touileb, L. Øvrelid, and E. Velldal, "Generative approaches to event extraction: Survey and outlook," in *Proceedings of the Workshop on the Future of Event Detection (FuturED)*, J. Tetreault, T. H. Nguyen, H. Lamba, and A. Hughes, Eds. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 73–86. [Online]. Available: https://aclanthology.org/2024.futured-1.7/

[5] Y. Lu, H. Lin, J. Xu, X. Han, J. Tang, A. Li, L. Sun, M. Liao, and S. Chen, "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)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 2795–2806. [Online]. Available: https://aclanthology.org/2021.acl-long.217/

[6] Z. Zhang, W. You, T. Wu, X. Wang, J. Li, and M. Zhang, "A survey of generative information extraction," in *Proceedings of the 31st International Conference on Computational Linguistics*, O. Rambow, L. Wanner, M. Apidianaki, H. Al-Khalifa, B. D. Eugenio, and S. Schockaert, Eds. Abu Dhabi, UAE: Association for Computational Linguistics, Jan. 2025, pp. 4840–4870. [Online]. Available: https://aclanthology.org/2025.coling-main.324/

[7] S. Li, H. Ji, and J. Han, "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*. Online: Association for Computational Linguistics, Jun. 2021, pp. 894–908. [Online]. Available: https://aclanthology.org/2021.naacl-main.69

[8] Y. Ren, Y. Cao, P. Guo, F. Fang, W. Ma, and Z. Lin, "Retrieve-and-sample: Document-level event argument extraction via hybrid retrieval augmentation," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 293–306. [Online]. Available: https://aclanthology.org/2023.acl-long.17

[9] X. Wang, S. Li, and H. Ji, "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)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 3640–3663. [Online]. Available: https://aclanthology.org/2023.acl-long.202

[10] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[11] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[12] W. Liu, S. Cheng, D. Zeng, and Q. Hong, "Enhancing document-level event argument extraction with contextual clues and role relevance," in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 12 908–12 922. [Online]. Available: https://aclanthology.org/2023.findings-acl.817

[13] M. Zhang and H. Chen, "Document-level event argument extraction with sparse representation attention," *Mathematics*, vol. 12, no. 17, p. 2636, 2024.

[14] J. Peng, H. Sun, W. Yang, F. Wei, L. He, and L. Wang, "One small and one large for document-level event argument extraction," *arXiv preprint arXiv:2411.05895*, 2024.

[15] X. Du and C. Cardie, "Event extraction by answering (almost) natural questions," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 671–683. [Online]. Available: https://aclanthology.org/2020.emnlp-main.49

[16] J. Liu, Y. Chen, K. Liu, W. Bi, and X. Liu, "Event extraction as machine reading comprehension," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1641–1651. [Online]. Available: https://aclanthology.org/2020.emnlp-main.128

[17] Y. Ma, Z. Wang, Y. Cao, M. Li, M. Chen, K. Wang, and J. Shao, "Prompt for extraction? paie: Prompting argument interaction for event argument extraction," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 6759–6774.

[18] J. Zhang, C. Yang, H. Zhu, Q. Lin, F. Xu, and J. Liu, "A semantic mention graph augmented model for document-level event argument extraction," *arXiv preprint arXiv:2403.09721*, 2024.

[19] G. Wang, D. Liu, J.-Y. Nie, Q. Wan, R. Hu, X. Liu, W. Liu, and J. Liu, "Degap: Dual event-guided adaptive prefixes for templated-based event argument extraction with slot querying," in *Proceedings of the 31st International Conference on Computational Linguistics*, 2025, pp. 7598–7613.

[20] W. Liu, L. Zhou, D. Zeng, Y. Xiao, S. Cheng, C. Zhang, G. Lee, M. Zhang, and W. Chen, "Beyond single-event extraction: Towards efficient document-level multi-event argument extraction," *arXiv preprint arXiv:2405.01884*, 2024.

[21] C. Liang, "Event argument extraction with enriched prompts," *arXiv preprint arXiv:2501.06825*, 2025.

[22] H. Li, Y. Ren, Y. Cao, Y. Li, F. Fang, Z. Lin, and S. Wang, "Bridging the gap: Aligning language model generation with structured information extraction via controllable state transition," in *Proceedings of the ACM on Web Conference 2025*, ser. WWW '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 1811–1821. [Online]. Available: https://doi.org/10.1145/3696410.3714571

[23] R. Lin, Y. Liu, Y. Gan, Y. Cai, T. Lan, and Q. Liu, "Gems: Generation-based event argument extraction via multi-perspective prompts and ontology steering," in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025, pp. 26 392–26 409.

[24] X. Wang, S. Li, and H. Ji, "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)*, 2023, pp. 3640–3663.

[25] Z. Li, Y. Zeng, Y. Zuo, W. Ren, W. Liu, M. Su, Y. Guo, Y. Liu, L. Lixiang, Z. Hu *et al.*, "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)*, 2024, pp. 8758–8779.

[26] S. Ebner, P. Xia, R. Culkin, K. Rawlins, and B. Van Durme, "Multi-sentence argument linking," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 8057–8077. [Online]. Available: https://aclanthology.org/2020.acl-main.718

[27] X. Yang, Y. Lu, and L. Petzold, "Few-shot document-level event argument extraction," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 8029–8046. [Online]. Available: https://aclanthology.org/2023.acl-long.446

[28] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[29] Z. Wan, F. Cheng, Z. Mao, Q. Liu, H. Song, J. Li, and S. Kurohashi, "GPT-RE: In-context learning for relation extraction using large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 3534–3547. [Online]. Available: https://aclanthology.org/2023.emnlp-main.214

[30] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel, "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), May 2004. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2004/pdf/5.pdf

[31] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[32] Y. Yang and A. Katiyar, "Simple and effective few-shot named entity recognition with structured nearest neighbor learning," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6365–6375. [Online]. Available: https://aclanthology.org/2020.emnlp-main.516

[33] Y. He, J. Hu, and B. Tang, "Revisiting event argument extraction: Can EAE models learn better when being aware of event co-occurrences?" in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 12 542–12 556. [Online]. Available: https://aclanthology.org/2023.acl-long.701

[34] G. Ding, X. Guo, X. Wang, L. Wang, T. Fu, N. Mu, and D. Zha, "Fusion meets function: The adaptive selection-generation approach in event argument extraction," in *Proceedings of the 31st International Conference on Computational Linguistics*, 2025, pp. 4359–4369.

[35] Y. Zuo, Y. Fei, W. Ning, J. Huang, Y. Feng, and L. Li, "Rule-guided extraction: A hierarchical rule optimization framework for document-level event argument extraction," in *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025, pp. 21 155–21 171.

[36] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.