Event Schema Miner with locally contrastive optmization

Anonymous ACL submission

Abstract

Event Schema Induction is an important task in natural language processing (NLP) that aims to summarize event types and their associated argument roles from a corpus. However, the task remains challenging due to several issues: limited coverage of event element extraction, ambiguous semantics of event reprensentation, and insufficient semantic distinctiveness in the event embedding space. In this paper, we propose *Event Schema Miner* (ESM), a novel framework with locally contrastive optmization for mining event schemas. The framework effectively addresses these challenges through three key components, each promoting the next: scenario-aware event extraction to improve the coverage, instruction-driven event respresentaion to resolve semantic ambiguity, and target-centric model optimization to refine embedding space. Experimental results show that ESM surpasses state-of-the-art methods on standard evalution metrics, excelling in discovering high-quality, high-coverage event schemas from rather complicated contexts with severe semantic ambiguity.

1 Introduction

Event schemas represent abstract structure of events by identifying their core event elements and semantic relationships, playing a crucial role in enabling machines to comprehend and organize about events. This enhances downstream tasks such as information extraction (Lin et al., 2020; Chambers and Jurafsky, 2011; Lu et al., 2022), event extraction (Liu et al., 2019; Ji and Grishman, 2008; Ahn, 2006), event prediction(Du et al., 2022; Zhao, 2021), and knowledge base construction (Zhang et al., 2020; He et al., 2024) and so on.

Traditional event schemas are manually designed by experts, such as in TAC-KBP (Ji and Grishman, 2011), ACE (Doddington et al., 2004), and MUC (Chinchor et al., 1993). However, these predefined schemas are time-consuming, labor-intensive to create and lack flexibility and scalability.

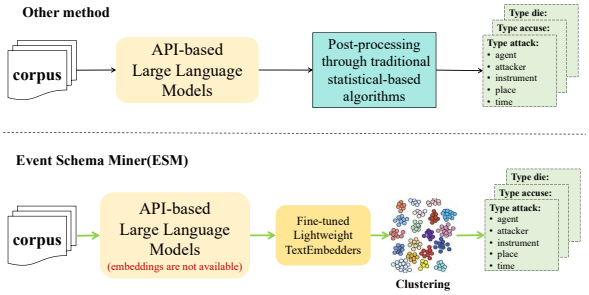

Figure 1: LLMs like ChatGPT do not perform well in directly generating schemas from the corpus, and thus require statistical-based algorithms for post-processing. Additionally, since LLM embeddings are not readily available, ESM introduces fine-tuned lightweight TEs to work with LLMs for clustering-based schema induction.

The limitations have driven the development of automated schema induction, such as resource-based and clustering-based induction. Resource-based induction methods use knowledge bases (e.g., WordNet (Miller, 1995), VerbNet (Schuler, 2005), FrameNet (Baker et al., 1998), and PropBank (Palmer et al., 2005)) as external semantic resources to provide reference standard for schema induction while using word sense disambiguation tools(Zhong and Ng, 2010; Huang et al., 2016). However, their effectiveness is constrained by quality and coverage of these resources, which limits their applicability in open-domain scenarios.

In contrast, clustering-based methods (Yuan et al., 2018; Nguyen et al., 2015; Chambers, 2013; Ahn, 2017) provide greater flexibility through three steps: event extraction (extracting event elements from raw texts, including triggers and arguments), event representation (constructing and vectorizing event features based on extracted event elements), and clustering induction (clustering for event types and argument roles based on event representation).

However, such methods still face obvious limitations. For example, many event extraction methods rely on dependency parser, which result in low coverage due to their susceptibility to noise introduced by tool inaccuracies (Shen et al., 2021). In addition, many event representation methods (Edwards and Ji, 2022; Huang et al., 2016; Shen et al., 2021; Tang et al., 2023; Qin et al., 2024) solely focus on triggers alone or simple combinations of triggers and arguments, which lead to ambiguous semantics of event expression, as shown in Table 1. Meanwhile, some embedding methods employ the distributed word embedding models (e.g. Word2Vec (Church, 2017)) or pre-trained models(e.g. Bert (Koroteev, 2021)) to vectorize event features, which struggle to generate task-specific distinctive embeddings (Huang et al., 2016; Tang et al., 2023).

Table 1: Ambiguous semantics of words. The first four sentence S1, S2, S3 and S4 contain the same trigger 'charge' but each corresponds to a different event type, similar to sentences S7 and S10. Sentence S2, S5, S6 and S7 have different triggers but correspond to the same event type, similar for S1 and S9. Sentence S5, S6 and S8 have almost identical contexts except for the trigger, yet they may still belong to different event types.

ID	Sentence
S1	The police **charged** the suspect with premeditated assault.
S2	Protesters **charged** at the security barricades during the demonstration.
S3	The waves **charged** against the rocky shore during the storm.
S4	The company **charged** customers an extra fee for expedited shipping.
S5	Protesters **assaulted** the security barricades during the demonstration.
S6	Protesters **attacked** the security barricades during the demonstration.
S7	The boxer **struke** his opponent with a powerful punch just now.
S8	Volunteers **defended** the security barricades during the demonstration.
S9	The prosecutor **accused** the suspect of murder.
S10	Today 19000 flight attendants of Lufthansa Airlines are **striking** for higher pay.

Until recently, some researchers (Tang et al., 2023) attempt to use large language models (LLMs, such as BLOOM (Scao et al., 2022).) to generate event schemas directly from the input corpus, bypassing intermediate steps. However, LLMs may generate noisy schemas that are inconsistent with input event descriptions, resulting in only partially relevant and relatively common schemas are obtained after post-processing.

In this paper, we propose Event Schema Miner (ESM), a framework with locally contrastive optimization for automatically mining event schemas from complex contexts in open domain, as shown in Figure 2. ESM effectively addresses challenges in clustering-based induction through three key components. Firstly, scenario-aware event extraction introduces a refined prompt, equipped with candidate triggers tailored to the given scenario, to query the LLM for high-coverage event extraction. Secondly, instruction-driven event respresentaion introduces a feature augmentation mechanism that integrates both event context and event element, which augments event expression and resolves semantic ambiguity. Thirdly, target-centric model optimization introduces a locally contrastive adjustment strategy that fine-tunes the model centered on target feature, which generates a semantically distinctive, task-specific embedding space.

We evaluate ESM on multiple datasets across diverse domains and languages, including DuEE (Li et al., 2020), DuEE-finDuEE-fin[1], FewEvent (Deng et al., 2020), ACE05 (Doddington et al., 2004), and ERE-EN (Song et al., 2015). Experimental results demonstrate that our approach achieves state-of-the-art performance on standard evaluation metrics, which can induce high-quality event schemas from rather complicated contexts, easy to deal with semantic ambiguity.

Contributions. The main contributions are summaried as follows: 1) a novel event schema induction framework ESM is proposed to effectively mine event schemas from rather complicated contexts in open-domain corpus. 2) a refined prompt is designed to query LLM for high-coverage event element extraction. 3) a feature augmentation mechanism is proposed to resolve semantic ambiguity of event representation. 4) a locally contrastive adjustment strategy is developed to optimize the embedding space more semantically distinctive for the task. 5) Extensive experiments on many datasets verify the effectiveness of ESM to mine event schemas from complicated contexts.

2 Preliminary

In this section, we describe some basic concepts and the task definitions.

Key Concepts. An event is represented as $e = \{tr, a_{tr}\}$, where *tr* denotes event trigger, and $a_{tr} =$

[1]https://aistudio.baidu.com/competition/detail/65/0/task-definition

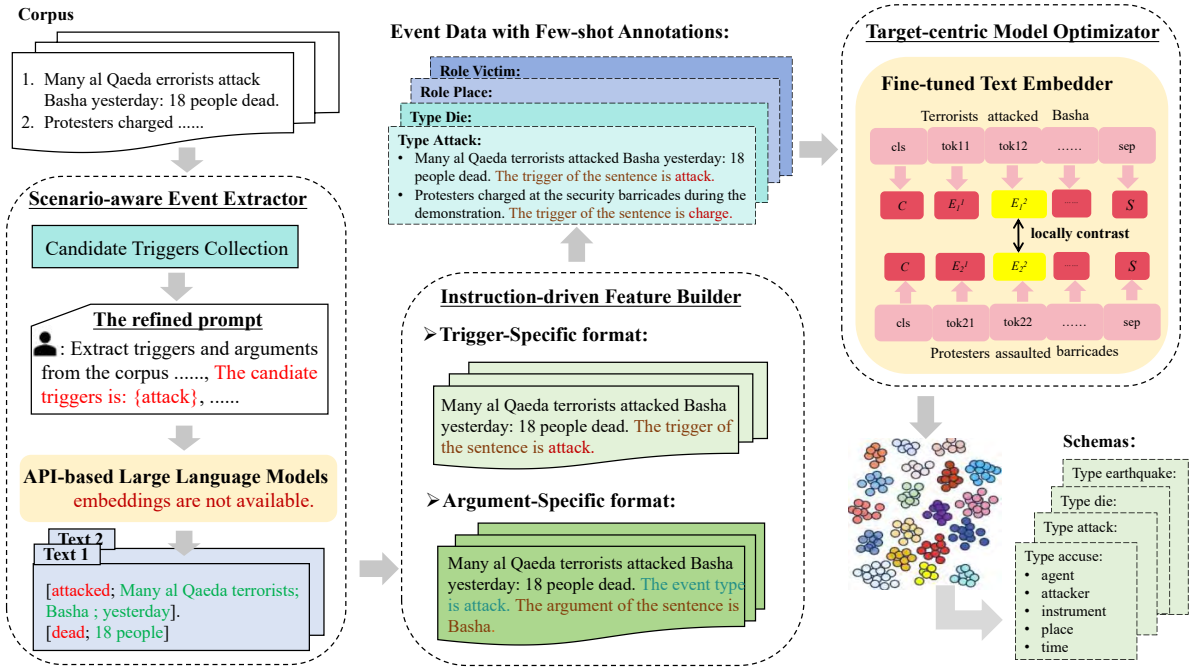

Figure 2: An overview of ESM. It is a novel framework with locally contrastive optmization to mine event schemas from complicated contexts.

$\{a_i\}$ $(i \in [1, n])$ are associated arguments describing participants or circumstances of the event.

An event schema is defined as $s = \{ty, r_{ty}\}$, where *ty* is event type, representing semantic category of event, abstracted from multiple instances of similar events. And $r_{ty} = \{r_i\}$ $(i \in [1, m])$ is a set of semantic roles, specifying relationships between event type and its associated arguments.

Task Definition. Given an unlabeled corpus $D = \{x_1, x_2, \ldots, x_n\}$ where each sentence x_i describes one or more events, the goal of event schema induction task is to induce a set of k schemas $S = \{s_1, s_2, s_3, \ldots, s_k\}$ from D. Each schema s_i $(i \in [1, k])$ is defined as described above.

Specifically, the task involves identifying and clustering triggers and arguments to unify event types ty and generalize associated semantic roles r_{ty}, which is typically divided into two subtasks: *event type induction* and *argument role induction*. To address these subtasks, we design two different **event representation models** (RM[2]) based on the feature augmentation mechanism:

- $\mathrm{RM_{ty}} = \{T, \mathrm{Instruct(tr)}\}$.
- $\mathrm{RM}_{r_{\mathrm{ty}}} = \{T, \mathrm{Instruct(ty)}, \mathrm{Instruct(a)}\}$.

[2]RM denotes the Representation Model, which is used to construct augmented event features tailored for specific subtasks.

Here, T indicates the context of event sentence and $\mathrm{Instruct}(x)$ is a natural language description of x.

3 Method

In this section, we describe how ESM mines event schemas from open-domain corpus. The framework significantly improves task performance by addressing challenges mentioned above.

The improvements stem from both data-driven and model-optimization perspectives. From a data-driven perspective, ESM constructs augmented event features based on event elements extracted by LLM, ensuring a more comprehensive and accurate data preparation. For a model-optimization perspective, ESM generates event embeddings for the augmented event features through TE models, which are then fine-tuned to better capture task-specific discriminative semantics. Finally, the optimized embeddings are subsequently clustered to induce event schemas.

To achieve these improvements, as shown in Figure 2, ESM incorporates the following key components: 1) Scenario-aware Event Extraction, 2) Instruction-driven Event Representation, 3) Target-centric Locally Contrastive Optimization. we will describe these components in detail next.

3.1 Scenario-aware Event Extraction

The Scenario-aware Event Extraction leveraging LLMs (such as ChatGPT) with a refined prompt to extract event elements (including event triggers and arguments) from open-domain corpus.[3]

By leveraging powerful contextual understanding capability and abundant knowledge reserve, LLMs effectively addresses diverse descriptions in open-domain contexts. To further improve the extraction coverage, we design a refined prompt that incorporates candidate triggers tailored to the given scenario. The refinement expands scenario-awareness of model's search scope and broades its cognitive vision for different event types.

Specifically, an unsupervised text-to-event structure based on LLMs is proposed. Given an input sentence x_i and a refined prompt P, we query the LLMs to extract an event list $E = [e_1, e_2, e_3, \dots]$, where each event e_i is defined as described above. Formally, the extraction process is as follows:

$$E = \textit{LLM}\left(P(I, D, C), x_i\right) \tag{1}$$

where E is extraction results given by LLM, representing event elements extracted from input x_i. The refined prompt P includes the following parts:

- **Task Commands (I):** Explicitly define the extraction task objective and specify desired output format as a nested list.
- **Demonstration Commands (D):** Randomly provide multiple demonstrations, each consisting of $\langle \text{text}, \text{events} \rangle$. The demonstrations show the mapping from input text to output list, helping the model learn task commands through imitation.
- **Candidate Trigger Commands (C):** Provide a list of candidate triggers, either user-supplied keywords or salient words filtered from the text. Since LLMs may struggle with specialized domains, candidate triggers guide them to extract event elements related to candidate triggers, reducing omissions of key event information to improve extraction coverage.

The generation of Candidate Triggers. The candidate triggers can be generated in various ways. Intuitively, we hypothesize that the most salient words in an event sentence are the most likely candidate triggers. That is, those appearing frequently in a specific event but rarely elsewhere. Following the TF-IDF principle, we define the saliency of a word w as follows:

$$s(w) = \left(1 + (\log \text{fr}(w))^2\right) \cdot \log\left(\frac{|N|}{\sum_k |N| \cdot \text{fr}(w)_k}\right) \tag{2}$$

where $\text{fr}(w)$ denotes the frequency of word w in the input text, and $|N|$ represents the total number of event sentences in the corpus. We ultimately select the top 10% of terms ranked by their salience scores as candidate triggers.

3.2 Instruction-driven Event Representation

Once event elements are extracted, next challenge lies in representing events to comprehensively capture their complex semantics within the context.

We observe that this issue is further complicated by semantic ambiguity of words in event descriptions. Take triggers for example, triggers serve as lexical cues that signal the occurrence of an event, and play a critical role in determining its specific type. However, as shown in table 1, a single trigger can carry multiple meanings when considered in isolation, and may indicate different event types depending on its surrounding context.

To address this problem, we draw inspiration from instruction learning, which incorporates omnifarious task-specific instructions into input text to guide the model's output, similar to the approach used in Instructor (Su et al., 2022). Following this principle, we propose a feature augmentation mechanism that incorporates both event context and contextually relevant instructions. The additional instruction, which serves as a domain-specific knowledge, are explicitly inserted into event sentence to augment event description, helping the model better generalize across different event types.

Specifically, building on the event representation models introduced earlier, their specific implementation and application in ESM are as follows:

Trigger-Specific Format. This format is designed for event type induction by integrating entire event context with a trigger-specific contextual instruction. The trigger-specific instruction $I_{trig}(tr)$ mark the trigger *tr* explicitly and instruct the model to focus on it, as defined below:

$I_{trig}(tr)$ = *"The trigger word of the sentence is* tr.

[3] Unless otherwise specified, for descriptive convenience, event elements refer to triggers and/or arguments, with triggers being the event's trigger words.

The trigger-specific augmented event feature x^+_{trig} is constructed by appending the trigger-specific instruction to raw event sentence x:

$$x^+_{trig} = x \oplus I_{trig}(tr) \tag{3}$$

where $\oplus$ denotes the concatenation operation.

For example, given the event sentence x: *"The prosecutor charged the suspect with murder."* and the trigger t extracted by the LLM, which is "*charged*", the trigger-specific augmented event feature x^+_{tr} becomes: *"The prosecutor charged the suspect with murder. The trigger word of the sentence is charge"*.

Argument-Specific Format. The construction principle of this format is similar to that of the trigger-specific format, with the key difference being the inclusion of an event type instruction. Due to space constraints, the detailed construction of this format is provided in Appendix C.

3.3 Target-centric Locally Contrastive Optimization

While the previous two compenents improve task performance from a data-driven prospective, the Target-centric Locally Contrastive Optimization achieves performance improvements from a model-optimization prospective.

By leveraging efficient embedding generation capabilities and flexible fine-tuning adaptability, TEs like GTE, built on the Sentence-Transformer architecture, map event features into a dense low-dimensional embedding space. Initally, define the embedder as $f_e(\theta_e)$, where θ_e represents the learnable parameters. Then the embeddings are formulated as follows:

$$E_i = f_e(B_i \mid \theta_e) \tag{4}$$

where B_i ($i \in \{1, 2\}$) represents pairwise batches.

To enable the model to learn both the shared features in event types and the distinctive features across different types, we apply contrastive learning to refine the embedding spaces. The parameter optimization process for θ^*_e can be formally expressed as follows:

$$\theta^*_e = \begin{cases} \arg\max\limits_{\theta} \left(\sum\limits_{B_u} \sum\limits_{k,l \in Y} (z^k_i z^l_j) \right), & \text{if } k = l, \\ \arg\min\limits_{\theta} \left(\sum\limits_{B_u} \sum\limits_{k,l \in Y} (z^k_i z^l_j) \right), & \text{others.} \end{cases} \tag{5}$$

where z_i, z_j denotes the normalized embedding of sample in batch B_u ($u \in \{1, 2\}$). k and l are the event type of each sample, and Y is the set of all event types. $(z_i z_j)$ represents the inner product.

Many contrastive learning methods for sentence-level tasks optimize embeddings based on full-sentence representations, which distribute attention across entire sentence. This limits the model's ability to capture fine-grained distinctions of individual event elements, especially in complex contexts where information dilution can occur.

In contrast, we propose a locally contrastive adjustment strategy that centers on target features (triggers or arguments), and optimizes by comparing their individual embeddings. This method helps the model foucs on distinguishing task-specific key features, avoiding distractions from the full sentence context and preventing semantic drift. To further refined embedding space, we adopt supervised contrastive loss (SCL) (Khosla et al., 2020) as the training objective, explicitly modeling relationships between events to refine the model's semantic discrimination. Let $\mathcal{L}_{\text{constra}}$ denote the contrastive loss, which is defined as:

$$\mathcal{L}_{\text{constra}} = -\sum_{i \in \text{batch}} \log \left(\frac{\exp(\text{sim}(z_i, z_{i'})/\tau)}{\exp(\text{sim}(z_i, z_{i'})/\tau) + \sum\limits_{j \neq i} \exp(\text{sim}(z_i, z_j)/\tau)} \right) \tag{6}$$

Here, z_i and $z_{i'}$ represent positive pairs, z_i and z_j denotes negative samples, $\text{sim}(\cdot, \cdot)$ is cosine similarity function, and τ is temperature parameter.

After fine-tuning, event embeddings for the test samples are generated using the optimized GTE model, with event features constructed similarly to those in the training set. We then apply a clustering algorithm (such as k-means and Lovain Graph) to group samples based on semantic similarity, with the resulting clusters represent different event types. Fine-tuned embeddings capture more context and finer distinctions, improving the clustering process and enhancing semantic grouping.

4 Experiments

This section presents extensive experiments to evaluate the effectiveness of the proposed method. The source code and instructions for reproducing are available at https://github.com/.

4.1 Experimental Settings

Datasets. We evaluate ESM on multiple datasets across diverse domains and languages, including Chinese datasets Duee (Du Baike Event Extraction) (Li et al., 2020) and DuEE-fin[4] (Du Baike Event Extraction-Finance), as well as English datasets ACE (Automatic Content Extraction) (Doddington et al., 2004) and ERE-EN (Entity Relation Event-English) (Song et al., 2015), which are widely used for event-related research tasks. All datasets were used for scientific research only, conditions permitting. The statisitics information of these datasets are given in Table 2.

Table 2: Statistics of different datasets. Size refers to the sum of event sentences in the training, validation and test sets. No.of Types refers to the number contained in the dataset of event types.

Dataset	Size	No.of Types	Domain	language
Duee	16956	65	General	CN
Duee-fin	11745	13	Financial	CN
ACE	18927	33	General	EN
ERE-EN	16510	9	General	EN

Table 3: Coverage of Event Extraction. We run each method 10 times and report its averaged result.

Dataset	Method	Triggers	Arguments
Duee	Syntax	0.525	0.482
	LLMs	0.662	0.605
	LLMs+RP	0.788	0.775
Duee-fin	Syntax	0.512	0.496
	LLMs	0.659	0.623
	LLMs+RP	0.786	0.763
ACE	Syntax	0.509	0.471
	LLMs	0.644	0.602
	LLMs+RP	0.765	0.736
ERE-EN	Syntax	0.518	0.477
	LLMs	0.656	0.625
	LLMs+RP	0.774	0.743

Implementation. For event extraction, we leverage GPT-4, accessed via standard APIs[5] as a paid service. For event schema induction, we employ GTE-base[6] model for Chinese datasets and GTE-base-zh[7] model for English datasets, to generate text embeddings. The embeddings are then fine-tuned using few-shot annotations for task-specific adaptations and optimizations. For the selection of label data, we randomly selected 6 Chinese event types from DuEE dataset, which include a total of 2,530 labeled event instances. For English datasets, we selected 4 event types from ACE dataset, comprising 500 labeled instances. Other details about runtime environment, hyperparameters, and configurations are provided in Appendix A.

4.2 Evaluation of Event Extraction

Since the quality of event elements directly influences event representation and model training, we first evaluate the effectiveness of event extraction.
Compared Method. (1) Syntax: Use the dependency parsing techniques to extract event elements, which represents a commonly used method before the application of LLMs. (2) LLMs: Use LLMs without any additional prompt refinement. (3) LLMs+RP: Use LLMs with a refined prompt to guide them toward higher-coverage extraction.
Evaluation Metircs. To access the alignment of the extracted event elements with the ground-truth annotations, we follow Named Entity Evaluation to evaluate at the token level. The Exact[8] matching mode that requires a complete and exact match between extracted results and annotated elements, is adopted for computing extraction coverage.
Results and Analysis. The results presented in Table 3 show that the event extraction method using LLMs with the refined prompt outperforms other methods. This improvement can be attributed to the powerful contextual understanding ability of LLM and the guiding effect of the refined prompt.

4.3 Evaluation of Event Schema Induction

4.3.1 Event Mention Clustering

In this section, we evaluate the effectiveness of ESM through event mention clustering task, which aims to generate event mention clusters when the number of clusters is specified. Unlike (Shen et al., 2021; Tang et al., 2023), which choose 15 types for evaluation, we select all types but follow their practice to access alignment with the ground-truth.
Compared Methods. We compare ESM with the following methods: Kmeans, JCSC, ETypeClus and ESHer. Due to space constraints, the detailed description of these methods are in Appendix D. We refer to the default settings of (Shen et al., 2021) for implementation details.
Evaluation Metrics. To evaluate the alignment of

[4]https://aistudio.baidu.com/competition/detail/65/0/task-definition
[5]https://api.openai.com/v1
[6]https://huggingface.co/thenlper/gte-base
[7]https://huggingface.co/thenlper/gte-base-zh
[8]Disregarding entity type, there is also a Partial mode that allows partial overlaps between two comparison, works better.

the clustering results with the ground-truth event schemas in the reference set, we choose commonly used cluster evaluation metircs (such as NMI, ARI, ACC, and BCubed-F) to measure the clustering performance. For these metrics, larger values indicate better performance. The math formulas of these metrics are in Appendix B.

Results and Analysis. Table 5 presents the overall experimental results. We can observe that ESM achieves state-of-the-art performance, which fully demonstrates the superiority of our approach. Its key advantages stem from the innovative event representation and optimization strategy, which effectively capture task-specific features and distinctive senmantics from both data and model perspectives. As shown in the table, ESM excels in handling complex contexts with semantically ambiguous triggers and arguments, aligning with our original design goals. However, for simpler datasets like Dueefin, which has simple event contexts with minimal semantic ambiguity (as most event types contain only one or two triggers), our approach still performs comparably to other traditional methods using event representations based on trigger itself. How to better deal with these simple contexts we leave to future work.

Table 4: Ablation Study results of the instruction and strategy. Each method is run 10 times, and the average result for each metric is provided.

Method	NMI	ARI	ACC	BC-F1
Only trigger	0.828	0.576	0.629	0.672
Only sentence	0.723	0.41	0.509	0.617
Instruction.EndTrig	0.9	0.751	0.766	0.963
Instruction.StrTrig	0.917	0.813	0.823	0.985
No locally	0.741	0.602	0.599	0.683
locally	0.9	0.751	0.766	0.963

4.3.2 Event Schema Induction

Unlike event mention clustering, which evaluate the framework from a task-oriented perspective, event schema discovery is designed to show the potential of ESM from a framework-oriented perspective, extending beyond the schemas that have already been discovered. Follow (Huang et al., 2016; Shen et al., 2021; Tang et al., 2023), we analyze the induced result by ESM. To facilitate visualization for comparison, as shown in Figure 3, we forced the number of clusters to be 64. It would be better not to set this value and allow the model to operate freely. The top image shows the results using our framework, the middle image shows the results using only the context and optimizing the entire sentence embedding, and the bottom image shows the clustering results using the trigger word embeddings generated by a non-finetuned text embedding model.

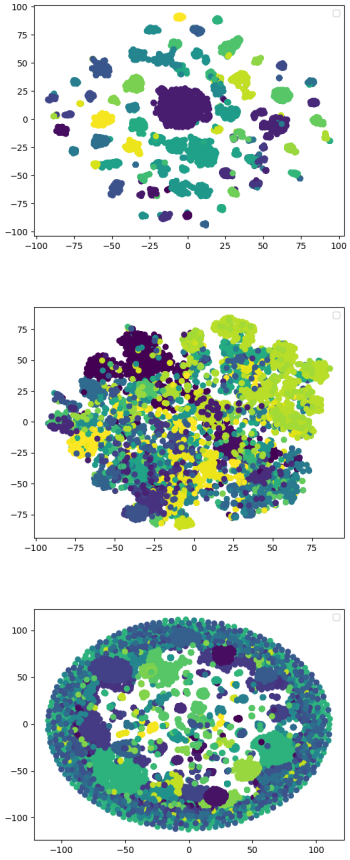

Figure 3: Visualization of Event Schema Induction results.

4.3.3 Ablation Study

To evaluate the contribution of each component in the framework, we conduct a series of ablation experiments. Specifically, we systematically remove or modify key components and assess their impact on the model's performance. Since the important of the refined prompt has been verified in the event extraction experiments, here we focus on verifing the feature augmention mechanism and the locally contrastive optimization strategy. Table 4 presents the ablation study results. The table above shows the results of the feature augmentation mechanism experiments, while the table below displays those related to the locally contrastive optimization strategy.

Table 5: Event Mention Clustering result. All values are reported as percentages. Each method is run 10 times, and the average result for each metric is provided. Note that ETYPECLUS is not applicable to argument role induction, as it is specifically designed for event type clustering.

Datasets	Methods	Event Type Induction				Argument Role Induction			
		NMI	ARI	ACC	BCubed-F1	NMI	ARI	ACC	BCubed-F1
Duee	Kmeans	54.71	38.61	54.71	38.61	43.72	11.21	23.06	63.21
	JCSC	44.43	34.31	47.44	51.45	54.65	31.57	44.65	77.64
	ETYPECLUS	77.03	28.56	45.45	55.56	-	-	-	-
	ESHer	85.33	73.01	70.93	90.32	54.68	23.47	21.37	34.36
	ESM	90.01	75.17	76.66	96.31	83.57	65.67	61.48	81.67
Duee-fin	Kmeans	90.32	83.34	89.28	88.45	57.34	31.48	41.39	65.91
	JCSC	82.33	81.00	87.56	83.43	68.31	41.34	65.44	69.87
	ETYPECLUS	67.36	65.73	73.07	67.67	-	-	-	-
	ESHer	90.07	89.32	91.67	91.25	59.31	15.91	37.14	35.34
	ESM	91.64	87.81	93.5	90.67	87.35	64.37	88.64	81.27
ACE05	Kmeans	48.02	26.27	41.57	41.33	23.41	11.83	27.61	28.40
	JCSC	49.50	36.10	46.17	43.83	44.70	28.71	51.31	47.60
	ETYPECLUS	57.57	40.78	48.35	51.58	-	-	-	-
	ESHer	65.30	50.27	55.34	61.27	39.34	15.50	24.48	34.60
	ESM	70.23	47.48	60.34	69.31	73.44	52.81	67.61	69.44
ERE-EN	Kmeans	37.65	12.51	31.01	31.01	39.16	13.11	29.40	31.24
	JCSC	39.50	17.07	33.76	37.64	43.41	22.20	39.41	41.94
	ETYPECLUS	49.40	24.09	41.10	40.09	-	-	-	-
	ESHer	62.72	51.59	57.43	62.43	44.95	11.10	35.21	37.40
	ESM	71.42	48.56	53.01	66.01	67.34	53.22	64.38	62.21

5 Related work

Event Schema Induction. Many traditional event schema induction heavily relies on predefined schemas crafted by domain experts(Ji and Grishman, 2011; Chinchor et al., 1993; Doddington et al., 2004). While these schemas provide a strong foundation for domain-specific tasks, their manual design process is resource-intensive and lacks the scalability to adapt to new scenarios. Afterwards, automated event schema induction subsequently emerged. Resource-based methods leverage NLP tools and external semantic resources (Miller, 1995; Schuler, 2005; Baker et al., 1998; Palmer et al., 2005) and using word sense disambiguation tools (Zhong and Ng, 2010) to extract event elements and align them with standard schemas in knowledge bases. However, their effectiveness are severely constrained by the quality and coverage of the predefined resources. Clustering-based methods (Yuan et al., 2018; Nguyen et al., 2015; Chambers, 2013; Ahn, 2017) induce event schemas by grouping events with similar embeddings, typically through three main steps. Early event element extraction rely on templates or dependency parsers(Shen et al., 2021), which are limited by scalability, flexibility and noise sensitivity. Many studies solely focus on triggers alone or simple combinations of triggers and arguments to express event(Edwards and Ji, 2022; Huang et al., 2016; Shen et al., 2021; Tang et al., 2023; Qin et al., 2024) and use distributed word embeddings (Church, 2017) or pre-trained models (Koroteev, 2021)) to represent events. However, these methods often ignore the contextual semantics of events and fail to generate fine-grained embedding. Moreover, while clustering algorithms can group embeddings effectively, their success heavily depends on the quality of event representations, which current methods struggle to achieve.

6 Conclusion

In this paper, we propose the Event Schema Miner (ESM) framework with locally contrastive optmization for mining event schemas, which offers a novel approach to event schema induction by combining scenario-aware event extraction, instruction-driven event representation, and locally contrastive optimization. These components jointly improve the accuracy and coverage of event schema induction by comprehensively capturing task-specific event features and distinctive senmantics from both data-driven and model-optimization perspectives. ESM excels in handling complex contexts with semantically ambiguous triggers and arguments, as demonstrated by our experimental results across multiple datasets. Our work contributes to the field by providing an effective and scalable solution for high-quality event schema induction.

Limitations

In this paper, we randomly selected some event types as training data for model optimization. However, different event type data can have a significant impact on the inductive results. Therefore, in future work, we will focus on how to select appropriate event types as training data from event datasets. Secondly, we only used GTE-base and GTE-base-zh as the base models in our experiments. In the future, we will explore the impact of different base models on the event schema induction task.

Impact Statement

Both event extraction and event type induction are standard tasks in NLP. We do not see any significant ethical concerns. The expected usage of our work is to induce event schemas from the input corpus such as a set of news articles or a collection of scientific papers.

References

Hervé Abdi and Lynne J Williams. 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

David Ahn. 2006. The stages of event extraction. pages 1–8.

Natalie Ahn. 2017. Inducing event types and roles in reverse: Using function to discover theme. pages 66–76.

David Arthur and Sergei Vassilvitskii. 2006. k-means++: The advantages of careful seeding.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. pages 722–735.

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. pages 1797–1807.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. pages 976–986.

Nancy Chinchor, Lynette Hirschman, and David D Lewis. 1993. Evaluating message understanding systems: An analysis of the third message understanding conference (muc-3). *Computational linguistics*, 19(3):409–450.

Kenneth Ward Church. 2017. Word2vec. *Natural Language Engineering*, 23(1):155–162.

Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. 2005. Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment*, 2005(09):P09008.

Shumin Deng, Ningyu Zhang, Jiaojian Kang, Yichi Zhang, Wei Zhang, and Huajun Chen. 2020. Meta-learning with dynamic-memory-based prototypical network for few-shot event detection. pages 151–159.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. 2(1):837–840.

Xinya Du, Zixuan Zhang, Sha Li, Pengfei Yu, Hongwei Wang, Tuan Lai, Xudong Lin, Ziqi Wang, Iris Liu, Ben Zhou, and 1 others. 2022. Resin-11: Schema-guided event prediction for 11 newsworthy scenarios. pages 54–63.

Carl Edwards and Heng Ji. 2022. Semi-supervised new event type induction and description via contrastive loss-enforced batch attention. *arXiv preprint arXiv:2202.05943*.

Mutian He, Tianqing Fang, Weiqi Wang, and Yangqiu Song. 2024. Acquiring and modeling abstract commonsense knowledge via conceptualization. *Artificial Intelligence*, 333:104149.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. pages 258–268.

L Hubert and P Arabie. 1985. Comparing partitions journal of classification 2 193–218. *Google Scholar*, pages 193–128.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. pages 254–262.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. pages 1148–1158.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673.

Mikhail V Koroteev. 2021. Bert: a review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*.

Xinyu Li, Fayuan Li, Lu Pan, Yuguang Chen, Weihua Peng, Quan Wang, Yajuan Lyu, and Yong Zhu. 2020. Duee: a large-scale dataset for chinese event extraction in real-world scenarios. pages 534–545.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. pages 7999–8009.

Xiao Liu, Heyan Huang, and Yue Zhang. 2019. Open domain event extraction using neural latent variable models. pages 2860–2871.

Stuart Lloyd. 1982. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. *arXiv preprint arXiv:2203.12277*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. 2015. Generative event schema induction with entity disambiguation. pages 188–197.

Lawrence Page. 1999. The pagerank citation ranking: Bringing order to the web.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and 1 others. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Wei Qin, Hao Wang, and Xiangfeng Luo. 2024. Open-domain event schema induction via weighted attentive hypergraph neural network. *Concurrency and Computation: Practice and Experience*, 36(12):e8029.

Teven Le Scao, Thomas Wang, Daniel Hesslow, Lucile Saulnier, Stas Bekman, M Saiful Bari, Stella Biderman, Hady Elsahar, Niklas Muennighoff, Jason Phang, and 1 others. 2022. What language model to train if you have one million gpu hours? *arXiv preprint arXiv:2210.15424*.

Karin Kipper Schuler. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon.

Jiaming Shen, Yunyi Zhang, Heng Ji, and Jiawei Han. 2021. Corpus-based open-domain event type induction. *arXiv preprint arXiv:2109.03322*.

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. pages 89–98.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. 2022. One embedder, any task: Instruction-finetuned text embeddings. *arXiv preprint arXiv:2212.09741*.

Jialong Tang, Hongyu Lin, Zhuoqun Li, Yaojie Lu, Xianpei Han, and Le Sun. 2023. Harvesting event schemas from large language models. pages 57–69.

Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, and 1 others. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*, 17.

Thomas Wolf. 2020. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Quan Yuan, Xiang Ren, Wenqi He, Chao Zhang, Xinhe Geng, Lifu Huang, Heng Ji, Chin-Yew Lin, and Jiawei Han. 2018. Open-schema event profiling for massive news corpora. pages 587–596.

Hongming Zhang, Daniel Khashabi, Yangqiu Song, and Dan Roth. 2020. Transomcs: From linguistic graphs to commonsense knowledge. *arXiv preprint arXiv:2005.00206*.

Liang Zhao. 2021. Event prediction in the big data era: A systematic survey. *ACM Computing Surveys (CSUR)*, 54(5):1–37.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. pages 78–83.

A Running Environment and Hyperparameters

Except for the ChatGPT model, all experiments were conducted on a server equipped with 12 CPU cores (12 vCPUs, Intel(R) Xeon(R) Platinum 8352V @ 2.10GHz) and a high-performance GPU with a Turing architecture (RTX 4090, 24GB). Our implementation relies on the Huggingface Library (Wolf, 2020) to manage pre-trained models and related tools. For data preprocessing, we followed the steps outlined in (Shen et al., 2021) and filtered out sentences that were too long or contained excessive numerical symbols. In the baseline experiments, we used the Wikipedia version from March 1, 2022, as background corpus and preprocessed it using WikiExtractor [9]to generate a clean and structured dataset for subsequent experiments.

[9]https://github.com/attardi/wikiextractor.

B Clustering metrics

We denote the ground truth clusters as C^*, the predicted clusters as C, and the total number of event mentions as N.

(1) NMI (Normalized Mutual Information) (Danon et al., 2005) measures the amount of information shared between the predicted and true cluster assignments, normalized by the total amount of information in both sets. Let $MI(\cdot;\cdot)$ be the Mutual Information between two cluster assignments, and $H(\cdot)$ denote the Entropy. Then, the NMI is formulated as follows:

$$\text{NMI} = \frac{2 \times MI(C^*; C)}{H(C^*) + H(C)}.$$

(2) ARI (Adjusted Rand Index) (Hubert and Arabie, 1985) evaluates the similarity between the predicted and true cluster assignments, adjusted for chance, providing a score between -1 and 1. Let $TP(TN)$ denote the number of element pairs in the same (different) cluster(s) in both C^* and C. Then, ARI is calculated as follows:

$$\text{ARI} = \frac{\text{RI} - E(\text{RI})}{\max(\text{RI}) - E(\text{RI})},$$

$$\text{RI} = \frac{TP + TN}{N},$$

where $E(\text{RI})$ is the expected RI of random assignments.

(3) B-Cubed F1 (Bagga and Baldwin, 1998) computes the precision and recall of each cluster by considering each data point's contribution to its cluster and comparing it to other clusters. B-Cubed precision, recall, and F1 are thus calculated as follows:

$$\text{BCubed-P} = \frac{1}{N} \sum_{i=0}^{N} \frac{|C(e_i) \cap C^*(e_i)|}{|C(e_i)|}$$

$$\text{BCubed-R} = \frac{1}{N} \sum_{i=0}^{N} \frac{|C(e_i) \cap C^*(e_i)|}{|C^*(e_i)|}$$

$$\text{BCubed-F1} = \frac{\text{BCubed-P}^{-1} + \text{BCubed-R}^{-1}}{2}$$

where $C^*(\cdot)$ and $C(\cdot)$ are the mapping functions from an element to its ground truth (predicted) cluster.

(4) Accuracy measures the proportion of correctly assigned points to clusters, comparing predicted labels to true labels. Let y_i (y_i^*) denote the i-th element's predicted (ground truth) cluster ID, the ACC is formulated as follows:

$$\text{ACC} = \max_{\sigma \in \text{Perm}(k)} \frac{1}{N} \sum_{i=1}^{N} I(y_i^* = \sigma(y_i))$$

where k is the number of clusters for both C^* and C, $\text{Perm}(k)$ is the set of all permutation functions on the set $\{1, 2, \ldots, k\}$, and $I(\cdot)$ is the indicator function.

Shared Hyper-parameter	ESM/Baselines
Max Number of Tokens	256
Ratio of Numerical Tokens	0.25
Min Frequency of Verbs	3
Salient Ratio of Verbs	0.25
Min Frequency of Arguments	3
Salient Ratio of Arguments	0.25
Random Seed	1234

Table 6: Shared hyperparameters for ESM

C Argument-Specific Format

Argument-Specific Format. This format is designed for argument role induction. In contrast to the trigger-specific format, it incorporates both an argument-specific instruction and an event type instruction. The purpose of this is to capture only the argument information related to the current event type from sentence containing multiple events.

The argument-specific instruction $I_{argu}(a)$ mark the argument *a* explicitly and instruct the model to focus on it, as defined below:

$$I_{argu}(a) = \textit{"The argument of the sentence is a."}$$

Meanwhile, the event type *ty*, obtained from the previous subtask, forms the event type instruction $I_{type}(ty)$ that provides event type information to help the model distinguish arguments associated with different event types, as defined below:

$$I_{type}(ty) = \textit{"The event type is ty."}$$

So under this event type, the argument-specific augmented event feature x_{ar}^{+} is constructed by appending the argument-specific instruction and the event type instruction to raw event sentence x:

$$x_{argu}^{+} = x \oplus I_{type}(ty) \oplus I_{argu}(a) \qquad (7)$$

where $\oplus$ denotes the concatenation operation.

For example, after obtaining the event type *'accuse'*, for one of the arguments *'prosecutor'* in sentence x (the example in trigger-specific format), the argument-specific augmented event feature x_a^+ becomes: *"The prosecutor charged the suspect with murder. The event type is accusation. The argument of this sentence is prosecutor"*.

Additionally, since an event sentence may contain multiple arguments, it is essential to construct argument-specific augmented event features for each argument. Moreover, as each event type determines the semantic scope of its associated arguments, argument role induction must follow event type induction to ensure coherence.

D Compared Methods

Compared Methods. (1) Kmeans (Lloyd, 1982). This method relies on the Scikit-learn codebase (Pedregosa et al., 2011)and uses L2 (Euclidean distance) as the similarity measure. The initial centroids are selected using k-means++ strategy (Arthur and Vassilvitskii, 2006). (2) JCSC (Huang et al., 2016). This method use spectral clustering algorithm (Von Luxburg, 2007) which are implemented based on the above Scikit-learn codebase. The label allocation policy is K-means, and 30 random initialization times are used for each time. (3) ETypeClus (Shen et al., 2021). This method uses the OntoNotes (Weischedel et al., 2011) and Wikipedia corpus (Auer et al., 2007) for support. The clustering process is based on a latent space model implemented with the Huggingface library (Wolf, 2020). The PCA (Abdi and Williams, 2010) is applied to reduce the dimension of original representations. (4) ESHer (Tang et al., 2023). This method pre-processes event expression, uses BLOOM (Scao et al., 2022) to generate event schemas, and then post-processes the model's outputs using traditional statistical-based algorithms (e.g. PageRank (Page, 1999)). (5) ESM: Our proposed method. All methods start with the same random seed and embeddings of unfine-tuned text embedders.