Complex Event Schema Induction with Knowledge-Enriched Diffusion Model

Yupu Hao[1,2], **Pengfei Cao**[1,2], **Yubo Chen**[1,2], **Kang Liu**[1,2], **Jiexin Xu**[3],
Huaijun Li[3], **Xiaojian Jiang**[3], **Jun Zhao**[1,2]

[1] The Laboratory of Cognition and Decision Intelligence for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3] China Merchants Bank
haoyupu2023@ia.ac.cn, {pengfei.cao, yubo.chen, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

The concept of a complex event schema pertains to the graph structure that represents real-world knowledge of events and their multi-dimensional relationships. However, previous studies on event schema induction have been hindered by challenges such as error propagation and data quality issues. To tackle these challenges, we propose a knowledge-enriched discrete diffusion model. Specifically, we distill the abundant event scenario knowledge of Large Language Models (LLMs) through an object-oriented Python style prompt. We incorporate this knowledge into the training data, enhancing its quality. Subsequently, we employ a discrete diffusion process to generate all nodes and links simultaneously in a non-auto-regressive manner to tackle the problem of error propagation. Additionally, we devise an entity relationship prediction module to complete entity relationships between event arguments. Experimental results demonstrate that our approach achieves outstanding performance across a range of evaluation metrics.[1]

1 Introduction

Event schema induction aims to summarize common patterns and structures from historical events. Current studies mainly induce the atomic schema for each independent event type and their arguments separately (e.g. "*Attack*" event with the arguments: "*Attacker*", "*Target*", "*Instrument*" and "*Place*"), without considering the correlation between events (Chambers and Jurafsky, 2008; Chambers, 2013; Nguyen et al., 2015). However, some real-world events are usually very complex, consisting of multiple events and their relations. For example in Figure 1, *Bombing* is a complex event, which involves some fine-grained events, such as *Assemble*, *Detonate* and *Injure*. Therefore, some
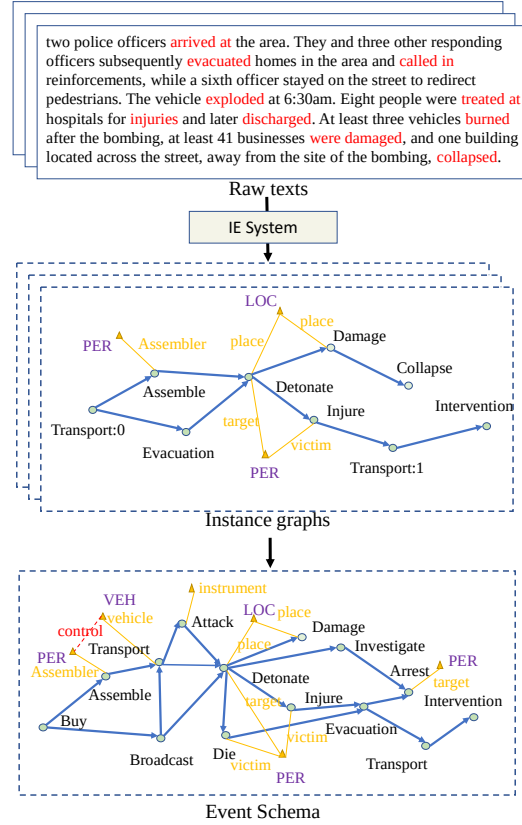

Figure 1: An example of schema induction process for complex event "*Bombing*".

researchers attempt to study the *complex event schema induction* task, which abstracts typical structures for complex events from event data. Figure 1 illustrates an example of the complex event schema induction process for the scenario of *Bombing*. Initially, an information extraction (IE) tool (Du et al., 2022) is utilized to extract instance graphs from raw texts. Subsequently, we induce the event schema based on these extracted instance graphs. The resulting event schema is represented as a graph, where events are interconnected through temporal links (e.g., *Damage* occurs after *Detonate*) and their argument relations (e.g., the *target*

---

[1]Code is available at https://github.com/hypasd-art/KDM/

of the *Detonate* event assumes the *victim* role in the subsequent *Injure* event).

However, inducing complex event schema is non-trivial. As shown in Figure 1, it necessitates the model's ability to summarize the events within instance graphs and possess a profound understanding of the multi-dimensional relationships between these events. Recently, graph-based methods are proposed for this task by utilizing graph generation techniques (Li et al., 2021; Jin et al., 2022). For example, Li et al. (2021) proposes an auto-regressive generation method that generates the schema following event temporal order. Similarly, Jin et al. (2022) leverages an auto-encoder to encode the global skeleton information and decode the schema graph event by event. Despite successful efforts, these methods still face two critical challenges:

Knowledge Coverage of Instance Graphs: The event schema induction task summarizes the instance graphs to obtain the event schema. Thus, the quality of the instance graphs is crucial for the event schema induction. However, the instance graphs are extracted via Information Extraction (IE) tools (Rui et al., 2022), whose knowledge coverage is very limited. For example, as the representative IE tool, RESIN (Wen et al., 2021) is trained on fixed datasets and can only extract predefined types of entities and events. Besides, the extraction performance of RESIN is also unsatisfactory, which only achieves approximately 64% of F1-score for event detection on the ACE dataset. It indicates that the IE tool is difficult to extract complete instance information, even for predefined event types. Therefore, how to improve the knowledge coverage of instance graphs is an important problem.

Error Propagation of Auto-regressive Decoding: Previous graph-based approaches are based on the auto-regressive generation manner (Li et al., 2021; Jin et al., 2022), generating the entire event schema graph node by node, which may lead to error accumulation over time and therefore degrade the generation performance. For example, in Figure 1, the model may mistakenly generate "Injure" instead of "Detonate" leading to the omission of subsequent events such as "Damage" and "Investigate" in the generated schema or resulting in incorrect nodes being generated in the next. The final generated event schema graph will consist of dozens of nodes and edges at the minimum, as each instance graph used for training contains an average of 117 event nodes and 246 temporary links according to our statistics on the Suicide-IED dataset (Li et al., 2021). The need to generate so many nodes and edges will inevitably exacerbate the problem of error accumulation. Thus, it is essential to address the error propagation problem during schema graph generation.

In this paper, we propose a novel method termed as **K**nowledge-Enriched **D**iffusion **M**odel (**KDM**) to address aforementioned problems. Firstly, to improve the knowledge coverage of instance graphs, we devise a *Instance Graph Expansion* module. As Large Language Models (LLMs) are trained on vast corpora of texts (Touvron et al., 2023; Zhao et al., 2023; Wang et al., 2023) and therefore possess extensive event and entity knowledge of the real world, we leverage the LLMs (Chowdhery et al., 2022; Ouyang et al., 2022) as the knowledge databases to inject knowledge into instance graphs. The module utilizes a Python style object-oriented prompt to extract event knowledge from LLMs, and adds the knowledge into the instance graphs. Secondly, to tackle error propagation of auto-regressive decoding, we propose an *Event Skeleton Generation* module, which utilizes discrete diffusion model to predict all nodes and links simultaneously in non-auto-regressive manner but not generates individually based on time series, which alleviate the error propagation problem (Austin et al., 2021; Yang et al., 2023; Vignac et al., 2022). Finally, we devise an *Entity Relation Prediction* module, which expands the event skeleton with corresponding arguments and predicts their relations to get a complete schema.

The contributions of our work include: (1) We propose a **K**nowledge-Enriched discrete **D**iffusion **M**odel (**KDM**) for complex event schema induction task. To the best of our knowledge, we are the first to simultaneously utilize LLMs and diffusion models to accomplish the task. (2) To improve knowledge coverage of instance graphs, we propose an Instance Graph Expansion module, which distillates the event knowledge in LLMs with python code-style prompt. To solve the error propagation problem, we design an event skeleton generation module, which predicts all nodes and links simultaneously. (3) We conduct extensive experiments on three widely used datasets. Experimental result indicates that our proposed method outperforms state-of-the-art baselines.
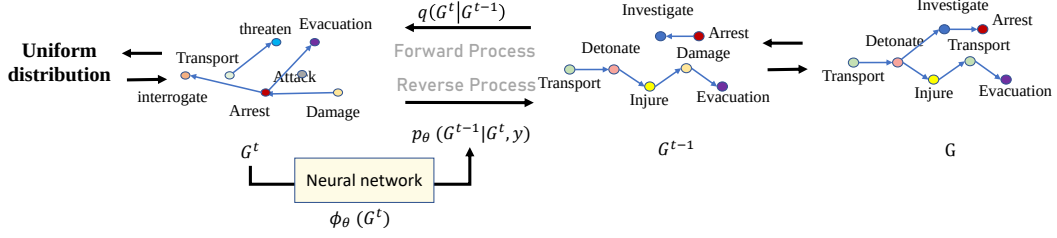
Figure 2: The discrete diffusion process. In forward process, the noise changes the types of nodes and edges.

2 Preliminaries and Problem Formulation

2.1 Preliminaries

Discrete diffusion model preserves the discrete characteristics of each element in the training data x_0, it introduces noise to each element $x_0^r \in x_0$ to into the uniform distribution and reverses them by removing the noise (Austin et al., 2021). Figure 2 shows the process of graph-based discrete diffusion.

The forward process. This process progressively adds noise to x_0 by transition probability matrix Q_t at t step.

$$q(x_t|x_{t-1}) = x_{t-1}Q_t \qquad (1)$$

where $|Q_t|_{ij} = q(x_t = j|x_{t-1} = i)$ indicates the probability of transition from $x_{t-1} = i$ to $x_t = j$. The forward process gradually converts each $x_0^r \in x_0$ to a uniform distribution when T is large enough.

The reverse process. Reverse process p_θ with learnable parameters θ aims to convert the noise distribution x_T back to the original x_0:

$$p_\theta(x_{0:T}) = p_\theta(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \qquad (2)$$

$$p_\theta(x_{t-1}|x_t) = \int q(x_{t-1}|x_t, x_0)dp_\theta(x_0|x_t) \qquad (3)$$

and according to Bayes formula as follows:

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)} \qquad (4)$$

Therefore, the task becomes predicting $p_\theta(x_0|x_t)$ using a neural network.

2.2 Problem Formulation

In the instance graphs about a specific topic y (e.g., *Bombing*), nodes represent events and entities, while edges have three types: the temporal link, the argument link and the entity relation link. The instance graph is denoted as $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, We define the distribution of graph as $G = (N, E)$, where node set $\mathcal{N}$ is sampled from node feature distribution $N \in \mathcal{R}^{n \times a}$ and edge set $\mathcal{E}$ is sampled from edge feature distribution $E \in \mathcal{R}^{n \times n \times b}$. Here, $c(p)$ represents the one-hot vector (category scalar) sampled based on probability distribution p. At time t, the instance graph is defined as $\mathcal{G}^t = (\mathcal{N}^t, \mathcal{E}^t) = (c(N^t), c(E^t))$.

The objective of this task is to learn an event schema $\mathcal{S}_y$ from a set of instance graphs $\mathcal{D}_y = \{\mathcal{G}(1), \mathcal{G}(2), \ldots, \mathcal{G}(m)\}$ that belong to the same specific topic.

3 Our Approach

To solve the complex event schema induction task, we propose a Knowledge-Enriched Discrete Diffusion Model, as shown in Figure 3. Our method mainly consists of three modules: (1) Instance Graph Expansion, which expands the instance graphs using the complex event knowledge obtained from LLMs atomically. (2) Event Skeleton Induction, which summarizes the event evolution skeleton using a discrete diffusion model. (3) Entity Relation Prediction, which decorates the arguments to the event skeleton and then use a simple graph transformer to predict the entity relations. We will illustrate each component in detail.

3.1 Instance Graph Expansion

In this section, we will illustrate how to obtain knowledge about event schemas from LLMs and inject them into instance graphs. Complex event schemas involve intricate graph structures, while LLMs are good at processing unstructured language tasks. To retain structured information of instance graphs, we need LLMs to be able to handle structured inputs and outputs. Considering the powerful coding capabilities of LLMs, we treat events as Python objects. In detail, events, entities, and their intricate relations can correspond to classes, attributes, and instances in the object-
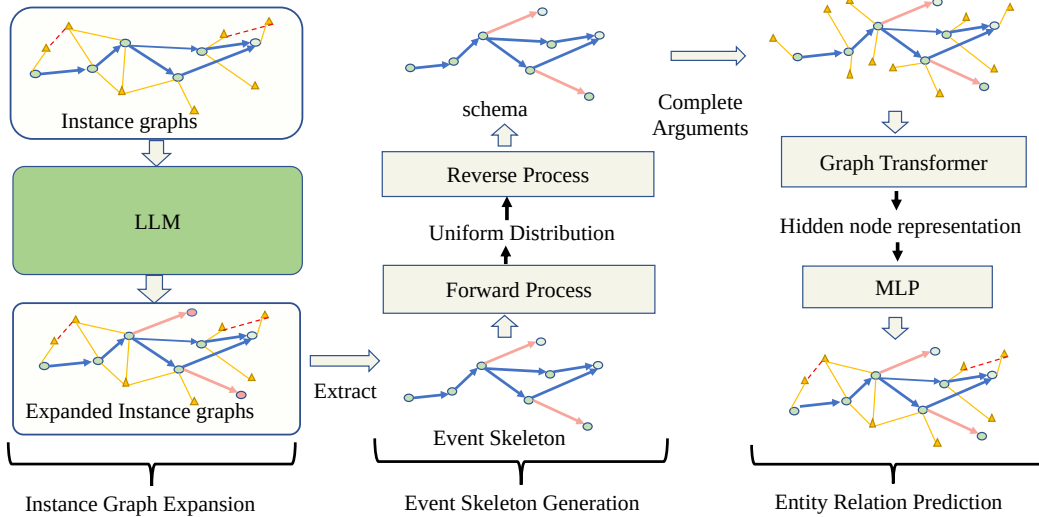
Figure 3: The model structure. The data passes through Instance Graph Expansion module, Event Skeleton Generation module, and Entity Relation Prediction module in sequence to obtain the final schema.

oriented paradigm, respectively. This module includes three aspects: event knowledge expansion, temporal relation expansion, and entity relation expansion.

In event knowledge expansion, we select frequently occurring event sequences from the training instance graphs and write them into Python classes. Then we ask the LLM to enrich Python code. In this way, we will obtain new classes which represent new events that have a high correlation with the scenario. We filter out new events that occur less frequently than a hyperparameter K and are not in the predefined event categories. In temporal relation expansion, we write the obtained events as multiple-choice questions to establish their temporal relation with existing event sequences. In entity relation expansion, we obtain the argument connection relation between the new and existing events by encoding new events into Python code and instantiating the class. By effectively leveraging the complex event knowledge contained in LLMs, our approach enhances the event schema generation process. The details and examples refer to Appendix E.

3.2 Event Skeleton Generation

In this section, we will introduce the forward and reverse processes of discrete diffusion based on the instance graphs through Instance Graph Expansion module. We have adopted the diffusion framework of Vignac et al.(2022) and made improvements based on it. Here, we denote the distribution $\boldsymbol{G}$ at

time t as $\boldsymbol{G}_t$.

The forward diffusion process. In this process, we apply noise separately on each node and edge. This is achieved by multiplying the node and edge distributions with the transition probability matrix $\boldsymbol{Q}$. By doing so, we can obtain the graph $\mathcal{G}_t$ from the previous graph $\mathcal{G}_{t-1}$. Mathematically, this can be expressed as:

$$\begin{aligned} q(\boldsymbol{G}_t|\boldsymbol{G}_{t-1}) &= (\boldsymbol{N}_{t-1}\boldsymbol{Q}_t^N, \boldsymbol{E}_{t-1}\boldsymbol{Q}_t^E) \\ &= (\boldsymbol{N}\bar{\boldsymbol{Q}}_t^N, \boldsymbol{E}\bar{\boldsymbol{Q}}_t^E) \end{aligned} \quad (5)$$

Where $\bar{\boldsymbol{Q}}_t = \boldsymbol{Q}_0\boldsymbol{Q}_1\cdots\boldsymbol{Q}_t$. The transition probability matrix $\boldsymbol{Q}_t$ is defined as $\alpha^t\boldsymbol{I} + (1 - \alpha^t)\mathbf{1}(\mathbf{1})^T/K$, where $\mathbf{1}$ is a column vector of all ones, and α^t varies from 1 to 0 (Austin et al., 2021). This formulation ensures that the distribution $q(\boldsymbol{G}_t|\boldsymbol{G}_0)$ consistent with uniform distribution when time t becomes sufficiently large.

Next, we sample the node and edge types from these probability distributions to obtain a discrete graph:

$$\mathcal{G}_t = \boldsymbol{c}(q(\boldsymbol{G}_t|\boldsymbol{G}_{t-1})) \quad (6)$$

The reverse diffusion process. We aim to remove the noise from the graphs using a parameterized reverse process p_θ. Following the formulation presented by Austin et al.(2021), we can express the posterior $p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t)$ as:

$$p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t) = \int q(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t, \boldsymbol{G}_0)dp_\theta(\boldsymbol{G}_0|\boldsymbol{G}_t) \quad (7)$$

To predict the clean graph distribution $\boldsymbol{G}_p^t = p_\theta(\boldsymbol{G}_0|\boldsymbol{G}_t)$ at time t given the noisy input $\mathcal{G}_t$, we

train a graph transformer ϕ_θ that outputs the clean graph representation:

$$\boldsymbol{G}_p^t = (\boldsymbol{N}_p^t, \boldsymbol{E}_p^t) = \phi_\theta(\mathcal{G}_t) \tag{8}$$

Our model ϕ_θ adopt transformer structure (Vaswani et al., 2017). Previous graph transformer model (Ying et al., 2021) is not appropriate for encoding directed graphs based on time series, because the relative position information between nodes is lost during the noise adding process. For instance, the self-attention mechanism module in the transformer cannot differentiate two "transport" events that occur in different time periods. To address this issue, we encode the depth information of event nodes as a fixed-feature embedding $\boldsymbol{n}_{dep}$ into the model. Before inputting the graph into the transformer, we add the depth feature to the corresponding node feature.

The depth fixed-feature embedding is encoded as follows:

$$\boldsymbol{n}_{dep} = \begin{cases} \sin(w_k \cdot n_d), & i = 2k \\ \cos(w_k \cdot n_d), & i = 2k+1 \end{cases} \tag{9}$$

where $w_k = 1/10000^{2k/n_d}$ and n_d is the average depth of node n, i is the index of the depth embedding.

Inspired by Vignac et al.(2022), our transformer model comprises several layers, each of which consists of a self-attention module and a feed-forward network. For layer l, the self-attention module takes as input time features $\boldsymbol{t}$, node features $\boldsymbol{N}_l^t$, edge features $\boldsymbol{E}_l^t$, and updates their representation as follows:

$$\boldsymbol{Att}_l = \boldsymbol{E}_l^t \boldsymbol{W}_a + \boldsymbol{E}_l^t \boldsymbol{W}_m \odot \boldsymbol{N}_l^t \boldsymbol{W}_Q (\boldsymbol{N}_l^t \boldsymbol{W}_K)^T \tag{10}$$

$$\boldsymbol{N}_{l+1}^t = MLP(\boldsymbol{t}\boldsymbol{W}_{tn} + \boldsymbol{Att}_l \boldsymbol{N}_l^t \boldsymbol{W}_V) \tag{11}$$

$$\boldsymbol{E}_{l+1}^t = MLP(\boldsymbol{t}\boldsymbol{W}_{te} + \boldsymbol{Att}_l) \tag{12}$$

where $\odot$ denotes the pairwise multiplication. $\boldsymbol{W}_a$, $\boldsymbol{W}_m$, $\boldsymbol{W}_{tn}$, $\boldsymbol{W}_{te}$, $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, $\boldsymbol{W}_V$ are trainable parameters.

To optimize our model, we use the cross-entropy loss $\mathcal{L}_{CE}$ weighted by λ:

$$\mathcal{L}_{CE}(\boldsymbol{G}_p^t, \boldsymbol{G}) = \sum_{\mathcal{G} \in \mathcal{D}} CE(\boldsymbol{N}_p^t, \boldsymbol{N}) + \lambda CE(\boldsymbol{E}_p^t, \boldsymbol{E}) \tag{13}$$

Once we obtain the clean graph distribution $\boldsymbol{G}_p^t$, we can infer the node distribution $p_\theta(\boldsymbol{n}_{t-1}|\boldsymbol{n}_t)$ and edge distribution $p_\theta(\boldsymbol{e}_{t-1}|\boldsymbol{e}_t)$ using the equations:

$$p_\theta(\boldsymbol{n}_{t-1}|\boldsymbol{n}_t) = \sum_{n_0=1}^{K_n} q(\boldsymbol{n}_{t-1}|\boldsymbol{n}_t, n_0) p_\theta(n_0|\boldsymbol{n}_t)$$

$$p_\theta(\boldsymbol{e}_{t-1}|\boldsymbol{e}_t) = \sum_{e_0=1}^{K_e} q(\boldsymbol{e}_{t-1}|\boldsymbol{e}_t, e_0) p_\theta(e_0|\boldsymbol{e}_t) \tag{14}$$

where K_n is the node type number, and K_e is edge type number. Before the next reverse process, we will get the discrete graph $\mathcal{G}_{t-1}$ from its distribution by probability sampling.

Our model obtains the final event schema $\mathcal{G}$ through T-step reversing process in non-AR manner. For further algorithm and derivation details, please refer to Appendix C.

Conditional Generation. Previous approaches (Li et al., 2020, 2021; Jin et al., 2022) need to train separate models for each scenario to ensure accurate generation. However, in order to generate event schemas for various scenarios using a single model and improve the model's generalization capabilities, we also propose the conditional diffusion model named as **KDMall** as a supplement.

We incorporate the category information $\boldsymbol{y}$ of the instance graphs as an additional attribute to control the training process of the model (Ho and Salimans, 2022; Dhariwal and Nichol, 2021). This allows us to influence the category of the generated schema. The formulation is as follows:

$$p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t, \boldsymbol{y}) = \int q(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t, \boldsymbol{G}_0) p_\theta(\boldsymbol{G}_0|\boldsymbol{G}_t, \boldsymbol{y}) \tag{15}$$

Therefore, we only need to encode the category information $\boldsymbol{y}$ into the neural network. We simply concatenate it into temporal features, enabling the conditional diffusion model to generate event schemas of different categories:

$$\boldsymbol{G}_p^t = (\boldsymbol{N}_p^t, \boldsymbol{E}_p^t) = \phi_\theta(\mathcal{G}_t, \boldsymbol{y}) \tag{16}$$

3.3 Entity Relation Prediction

In this module, we have developed a simple architecture that combines a graph transformer for obtaining node representations with an MLP layer for relation prediction. This module takes the event skeleton, expanded with event argument roles, as input and generates the complete event schema by predicting the relations between entities.

While previous models have primarily focused on entity types in the classification process, neglecting the significance of events and event roles, we address this limitation by artificially aggregating them together. We initialize the node features using BERT model (Devlin et al., 2018). Specifically, for each event or entity node n_i, $BERT(n_i)$ represents its type embedding encoded by BERT. For entity nodes, n_i^e indicates the event node that entity node n_i belongs to, and $\boldsymbol{n}_i^r$ is a fixed embedding representing the role played by entity node n_i in

event n_i^e, The encoding formula of embeddings $\boldsymbol{n}_i^r$ is the same as that of equation 9.

$$\boldsymbol{n}_i = \text{concat}(BERT(n_i), BERT(n_i^e)) \qquad (17)$$

Our transformer encoder is the same as the model used for Event Skeleton Generation, except that it lacks the time feature. The graph transformer outputs $\hat{\boldsymbol{n}}_i$ corresponding to the input $\boldsymbol{n}_i$, which is then passed to the MLP predictor.

The predicted relation type $\boldsymbol{r}_{ij}$ of entity node n_i and n_j is then computed as:

$$\boldsymbol{r}_{ij} = MLP([\hat{\boldsymbol{n}}_i + \boldsymbol{n}_i^\tau, \hat{\boldsymbol{n}}_j + \boldsymbol{n}_j^\tau]) \qquad (18)$$

It is worth noting that the classification problem is highly unbalanced. To address this issue, we set different weights for different categories of the loss function, defined as:

$$\mathcal{L} = \sum_{entity\ i,j} H(\hat{\boldsymbol{r}}_{ij})CE(\hat{\boldsymbol{r}}_{ij}, \boldsymbol{r}_{ij}) \qquad (19)$$

where $\hat{\boldsymbol{r}}_{ij}$ denotes the true relation between entity i and j and $H(\cdot)$ is a scalar function that assigns balanced weights to different relationships, with each relationship corresponding to a specific value.

4 Experiments

4.1 Datasets

We conduct experiments using the IED Schema Learning Corpus released by Li et al.(2021). The dataset utilizes the DARPA KAIROS ontology. The corpus specifically focuses on three sub-types of complex events related to Improvised Explosive Devices (IEDs): General-IED, Mass-Car-Bombing-IED, and Suicide-IED.

However, the test data in the corpus has data quality issues since it is also extracted through IE tools. To address this, we manually modify the test data, generating golden test event schemas based on the modified data. Additionally, to ensure the objective evaluation of our model's effectiveness, we record the test results using the original, unmodified data, which are provided in Appendix 7.

4.2 Baselines

In this work, we compare the proposed event schema induction model with two baselines:

Frequency-Based Sampling (FBS) model which constructs the event schema according to frequency distributions of temporal links in the training data. At each timestamp, FBS samples a pair of event types according to their frequency and adds the sampled edge into the schema graph. The procedure is repeated until FBS detects a cycle in the schema graph after adding a new edge.

Double Graph Auto-encoders Model (Double-GAE) (Jin et al., 2022), the state-of-the-art schema induction model which designs a variational directed acyclic graph auto-encoder to extract the event skeleton. Then it uses another GCN based auto-encoder to reconstruct entity-entity relations.

Large Language Model (LLM) have strong understanding and generation abilities, We ask the large language model (ChatGPT) to directly generate the event schema and use it as the baseline.

4.3 Evaluation Metrics

To evaluate the quality of the generated schema, we compare the schema with test instance graphs in terms of the following metrics to see how well the schema match real world instance graphs, the following evaluation metrics are employed:

(1) Event type match. we calculate the F1 score between the event types present in the schema graph and test instance graphs.

(2) Event sequence match. A good schema is able to track events through a timeline. we calculate the F1 score between the event sequences of length 2 or 3 present in the schema graph and the test instance graphs.

(3) Node/edge type distribution. we compare the Kullback-Leibler (KL) divergence of the node and edge type distributions between the schema graph and each test instance graph.

(4) Event Argument Connection Match (CM). Complex event graph schema includes entities and their relations, representing how events are connected through arguments. Because there is a serious long tail issue with the data, we calculate the macro F1-score for every pair of relationships between entities.

4.4 Overall Results

As shown in Table 1, the result demonstrates the effectiveness of **KDM** in capturing important events and their relationships. Specifically, our approach outperforms the baseline methods in terms of event sequence matching, particularly for longer path lengths (l=3).

These improvements can be attributed to the discrete diffusion process employed in our model. This process allows our model to simultaneously predict the categories of all nodes and edges, making it well-suited for graph generation. Addition-

Dataset	Model	Event type Match(F1)	Event sequence match		KL divergence		CM
			l=2	l=3	Node-type	Edge-type	
General-IED	FBS	0.614	0.199	0.064	2.98	6.13	-
	DoubleGAE	0.627	0.266	0.093	2.43	5.57	0.046
	LLM	0.520	0.176	0.041	2.72	5.84	-
	KDM (ours)	**0.704**	**0.380**	**0.181**	**2.32**	**4.53**	**0.185**
Car-IED	FBS	0.650	0.198	0.065	1.86	5.85	-
	DoubleGAE	0.654	0.285	0.107	2.12	5.71	0.044
	LLM	0.515	0.150	0.031	2.70	6.34	-
	KDM (ours)	**0.701**	**0.395**	**0.207**	**1.91**	**4.16**	**0.176**
Suicide-IED	FBS	0.626	0.210	0.061	2.11	5.59	-
	DoubleGAE	0.624	0.272	0.096	2.19	5.33	0.046
	LLM	0.493	0.174	0.053	2.75	5.65	-
	KDM (ours)	**0.713**	**0.462**	**0.268**	**1.91**	**3.45**	**0.176**

Table 1: Schema matching score (%) is calculated by checking the intersection of the induced schemas and the manually checked test schemas.

Dataset	Model	Event type Match(F1)	Event sequence match	
			l=2	l=3
General-IED	**KDM**	0.704	0.380	0.181
	KDMall	**0.723**	**0.413**	**0.202**
Car-IED	**KDM**	0.701	**0.395**	**0.207**
	KDMall	**0.703**	0.389	0.198
Suicide-IED	**KDM**	**0.714**	**0.462**	**0.268**
	KDMall	0.714	0.450	0.251

Table 2: Schema matching score (%) by checking the intersection of induced schemas and manual checked test schemas. **KDMall** is the conditional diffusion model trained on three IED datasets.

ally, the Transformer architecture leveraged in our model effectively utilizes global features through the self-attention mechanism, resulting in improved prediction accuracy.

Furthermore, our model shows remarkable improvements in the Connection Match evaluation, indicating the effectiveness of our graph transformer model than GCN graph auto-encoder in Double-GAE.

4.5 Conditional Generation Results

Building upon the aforementioned diffusion model, in order to improve the possibility of the model's generalization ability, we present an extension in the form of a conditional diffusion model as a supplement. This model enables the generation of event schemas for various scenarios using a single model.

As shown in Table 2, when comparing **KD-Mall** with our model trained on a specific-dataset,

we find that **KDMall** shows improved generalization capabilities and better understanding of event relationships, particularly in the "General-IED" scenario. Additionally, in other datasets, **KDMall** demonstrates comparable results to the model trained on a single dataset, indicating the potential of our conditional generation process. The incorporation of diverse training data enables the model to learn common patterns and associations across different scenarios, leading to improved performance and broader applicability (Sastry et al., 2023; Kim et al., 2022).

4.6 Ablation Experiment

To demonstrate the effectiveness of our approach, we conduct ablation studies on the "Suicide-IED" dataset. (1) **IGE Module Ablation Experiment:** To prove the effectiveness of our approach, we conduct experiments as shown in Table 3. JSON is a prevalent format for representing structured data. We encode the data in JSON format and instruct the LLMs to perform expansion, while maintaining the rest of the process consistent with the Python prompt approach. As shown in the Table, the results obtained through the use of Python prompts are noticeably better than those achieved with JSON prompts. And after filtering, the event types generated by the Python prompt are significantly more numerous than those generated by the JSON prompt. This observation underscores the effectiveness of the Python prompt approach. (2) **Diffusion Model Ablation Experiment:** In Table 4, comparing our **KDM** model with a variant

Model	Event type Match(F1)	Event sequence match(F1) l=2	l=3	KL divergence Node-type	Edge-type	EN
without IGE	0.672	0.411	0.216	1.96	3.67	-
IGE with JSON prompt	0.685	0.418	0.221	1.94	3.89	2
IGE with Python prompt	**0.713**	**0.462**	**0.268**	**1.91**	**3.45**	**9**

Table 3: Results of different prompts for the IGE Module on Suicide-IED dataset. **EN** is the number of effective events generated by the LLM after filtering, which are used for Instance Graph Expansion.

Model	Event type Match(F1)	Event sequence match(F1) l=2	l=3	KL divergence Node-type	Edge-type
KDM	**0.713**	**0.462**	**0.268**	**1.91**	**3.45**
w/o depth	0.709	0.429	0.209	1.92	3.83
w/o IGE	0.672	0.411	0.216	1.96	3.67

Table 4: The diffusion model ablation experiment on Suicide-IED dataset. "w/o depth" denoted as trained the model without depth features in graph transformer; "w/o IGE" denoted as trained the model on the dataset without Instance Graph Expansion module.

that removes the Instance Graph Expansion module, Our model achieves a 4.1% increase in node matching accuracy, proving the effectiveness of Instance Graph Expansion module. Additionally, by incorporating depth information, we observe a notable 5.9% improvement in sequence matching. These results demonstrate that the inclusion of depth information enhances our model's ability to capture the structural characteristics of graph, proving the effectiveness of adding depth features. (3) **Entity Predictor Ablation Experiment:** In Figure 4, Compared to not setting weight hyperparameters, our model achieves a significant 5.57% improvement in the macro F1 index, demonstrating that our weight scalar function significantly addresses the long-tail data problem. Moreover, when comparing the results of "w/o RE" and "w/o IGE", the improvements highlight the effectiveness of adding role and event features and Instance Graph Expansion module.

4.7 Case Study

In Figure 5, we observe that our Instance Graph Expansion module successfully generates a schema that encompasses a broader range of events and exhibits more comprehensive temporal relationships within complex events. This outcome support the effectiveness of leveraging object-oriented coding to distill knowledge from LLMs. Additionally, we provide a case study showcasing the diffusion process on the "Suicide-IED" dataset in Figure 6 in Appendix.
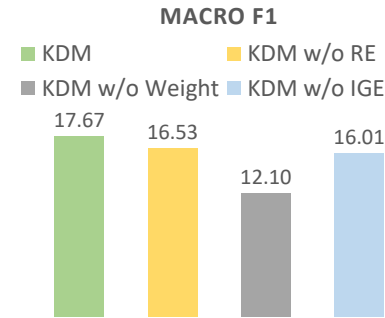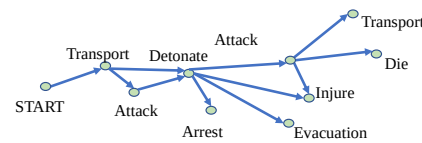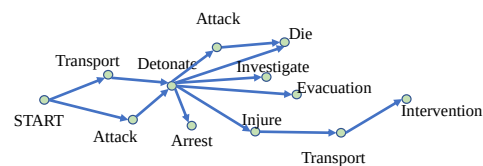

Figure 4: The entity predictor ablation experiment. "w/o RE" denoted as trained without fixed role embedding features and event embedding features; "w/o weight" denoted as trained without hyperparameter weight; "w/o IGE" denoted as trained without Instance Graph Expansion module.

(a) the event schema skeleton without Instance Graph Expansion module

(b) the event schema skeleton with Instance Graph Expansion module

Figure 5: Case Study on event schema skeleton

5 Related Work

Event Schema Event schema induction is a comprehensive graphical pattern composed of temporal and multi hop argument relationships (Li et al., 2020; Jin et al., 2022). It is actually a combination of atomic schema induction (Chambers, 2013; Yuan et al., 2018; Du and Ji, 2022; Wang et al., 2021) and script learning (Rudinger et al., 2015; Granroth-Wilding and Clark, 2016; Weber et al., 2018). Clearly, the event schema induction has broad application significance. For example, the event schema facilitates analysis and prediction of future events, aiding in the development of reaction plans for relevant scenarios (Li et al., 2021; Dror et al., 2023; Pan et al., 2021). Event schemas can be used as guidance information in information extraction, which helps people understand the internal logic of events (Wen et al., 2021).

Diffusion models Diffusion model (Sohl-Dickstein et al., 2015; Ho et al., 2020) has achieved impressive results on image, text and audio generation (Rombach et al., 2022; Shen et al., 2023; Li et al., 2022; Gong et al., 2022; Kong et al., 2020; Yuan et al., 2022). Recently, Vignac et al. (2022) have shown great potential in graph generation field. Previous graph diffusion models embedded graphs in a continuous space by adding Gaussian noise to the nodes and edges feature (Niu et al., 2020; Jo et al., 2022). However, this approach destroys the graph's sparsity and makes it hard to capture the node connections (Vignac et al., 2022). Discrete diffusion model (Austin et al., 2021; Yang et al., 2023; Vignac et al., 2022; Johnson et al., 2021) overcomes this problem by utilizing Markov process that can occur independently on each node or edge.

6 Conclusion

In this work, we identify the limitations of previous works and proposed a Knowledge-enriched discrete diffusion model. To enhance the quality and coherence of the generated schemas, we harness the potential and rich knowledge present in LLMs by utilizing them for Instance Graph Expansion. Our model leverages a discrete diffusion process to learn and generate event skeletons, while incorporating an entity relationship predictor to predict the relationships between event arguments. Additionally, we propose a conditional diffusion model with the purpose of generating schemas for multiple diverse topics. We achieved the best results among multiple different evaluation indicators.

Limitations

We only consider the temporal relationship between events here and do not consider the hierarchical structure of the event schema, which may result in not perfect event schemas generated by us.

Due to the limited availability of datasets, our conditional diffusion model **KDMall** has only undergone unified training and testing on three highly related explosive events, requiring more categories and quantities of data for the comprehensive ability testing of the model.

Ethics Statement

We use a discrete diffusion model to generate event skeletons and design an entity relationship predictor. At the same time, we have fully explored the potential rich knowledge in LLM for knowledge expansion. Our work has improved the effectiveness of event schema induction, helping people better summarize the logic and ontology knowledge of events, making contributions to this field.

Acknowledgements

References

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Seattle, Washington, USA. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794.

Rotem Dror, Haoyu Wang, and Dan Roth. 2023. Zeroshot on-the-fly event schema induction. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 705–725, Dubrovnik, Croatia. Association for Computational Linguistics.

Xinya Du and Heng Ji. 2022. Retrieval-augmented generative question answering for event argument extraction. *arXiv preprint arXiv:2211.07067*.

Xinya Du, Zixuan Zhang, Sha Li, Pengfei Yu, Hongwei Wang, Tuan Lai, Xudong Lin, Ziqi Wang, Iris Liu, Ben Zhou, Haoyang Wen, Manling Li, Darryl Hannan, Jie Lei, Hyounghun Kim, Rotem Dror, Haoyu Wang, Michael Regan, Qi Zeng, Qing Lyu, Charles Yu, Carl Edwards, Xiaomeng Jin, Yizhu Jiao, Ghazaleh Kazeminejad, Zhenhailong Wang, Chris Callison-Burch, Mohit Bansal, Carl Vondrick, Jiawei Han, Dan Roth, Shih-Fu Chang, Martha Palmer, and Heng Ji. 2022. RESIN-11: Schema-guided event prediction for 11 newsworthy scenarios. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: System Demonstrations*, pages 54–63, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2727–2733. AAAI Press.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465.

Xiaomeng Jin, Manling Li, and Heng Ji. 2022. Event schema induction with double graph autoencoders. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2013–2025, Seattle, United States. Association for Computational Linguistics.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. 2022. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, pages 10362–10383. PMLR.

Daniel D Johnson, Jacob Austin, Rianne van den Berg, and Daniel Tarlow. 2021. Beyond in-place corruption: Insertion and deletion in denoising probabilistic models. *arXiv preprint arXiv:2107.07675*.

Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. 2022. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2426–2435.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2020. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*.

Manling Li, Sha Li, Zhenhailong Wang, Lifu Huang, Kyunghyun Cho, Heng Ji, Jiawei Han, and Clare Voss. 2021. The future is not one-dimensional: Complex event schema induction by graph modeling for event prediction. *arXiv preprint arXiv:2104.06344*.

Manling Li, Qi Zeng, Ying Lin, Kyunghyun Cho, Heng Ji, Jonathan May, Nathanael Chambers, and Clare Voss. 2020. Connecting the dots: Event graph schema induction with path language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 684–695, Online. Association for Computational Linguistics.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. 2015. Generative event schema induction with entity disambiguation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International*

Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 188–197.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. 2020. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Jeff Z Pan, Elspeth Edelstein, Patrik Bansky, and Adam Wyner. 2021. A knowledge graph based approach to social science surveys. *Data Intelligence*, 3(4):477–506.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script iuction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1681–1686, Lisbon, Portugal. Association for Computational Linguistics.

Yong Rui, Vicente Ivan Sanchez Carmona, Mohsen Pourvali, Yun Xing, Wei-Wen Yi, Hui-Bin Ruan, and Yu Zhang. 2022. Knowledge mining: a cross-disciplinary survey. *Machine Intelligence Research*, 19(2):89–114.

Chandramouli Sastry, Sri Harsha Dumpala, and Sageev Oore. 2023. Training diffusion classifiers with denoising assistance. *arXiv preprint arXiv:2306.09192*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Diffusionner: Boundary diffusion for named entity recognition. *arXiv preprint arXiv:2305.13298*.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. 2022. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*.

Haitao Wang, Tong Zhu, Mingtao Wang, Guoliang Zhang, and Wenliang Chen. 2021. A prior information enhanced extraction framework for document-level financial event extraction. *Data Intelligence*, 3(3):460–476.

Xiao Wang, Guangyao Chen, Guangwu Qian, Pengcheng Gao, Xiao-Yong Wei, Yaowei Wang, Yonghong Tian, and Wen Gao. 2023. Large-scale multi-modal pre-trained models: A comprehensive survey. *Machine Intelligence Research*, pages 1–36.

Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. 2018. Event representations with tensor-based compositions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Haoyang Wen, Ying Lin, Tuan Lai, Xiaoman Pan, Sha Li, Xudong Lin, Ben Zhou, Manling Li, Haoyu Wang, Hongming Zhang, Xiaodong Yu, Alexander Dong, Zhenhailong Wang, Yi Fung, Piyush Mishra, Qing Lyu, Dídac Surís, Brian Chen, Susan Windisch Brown, Martha Palmer, Chris Callison-Burch, Carl Vondrick, Jiawei Han, Dan Roth, Shih-Fu Chang, and Heng Ji. 2021. RESIN: A dockerized schema-guided cross-document cross-lingual cross-media information extraction and event tracking system. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 133–143, Online. Association for Computational Linguistics.

Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. 2023. Diffsound: Discrete diffusion model for text-to-sound generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

C Ying, T Cai, S Luo, S Zheng, G Ke, D He, Y Shen, and TY Liu. 2021. Do transformers really perform bad for graph representation? arxiv 2021. *arXiv preprint arXiv:2106.05234*.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. 2022. Seqdiffuseq: Text diffusion with encoder-decoder transformers. *arXiv preprint arXiv:2212.10325*.

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. In *Proceedings of the 27th ACM International Conference on Information and*

Knowledge Management, CIKM '18, page 587–596, New York, NY, USA. Association for Computing Machinery.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223.*

A   Data Preprocessing

In the data preprocessing stage. Firstly, for each complex event, we constructed an instance graph by merging coreferential events or entities. Isolated events were excluded from the instance graphs during the graph construction process.

Specifically, we followed the cleaning strategy outlined in Jin et al. (2022). We deleted links with the same start and end types, as well as event-event links such as (DIE, INJURE), (ARRESTJAILDETAIN, ATTACK), (ENDPOSITION, STARTPOSITION), (DEFEAT, EXCHANGEBUYSELL), (SENTENCE, DIE), (ENDPOSITION, SENTENCE), and (THREATENCOERCE, RELEASEPAROLE) from the instance graphs. The maximum number of graph nodes m is set to 50.

B   Training And Evaluation Details

In our event skeleton induction process, we utilize a 12-layer Transformer model. Additionally, we employ a 3-layer Transformer as our entity relation predictor. To balance the trade-off between nodes and edges, we set λ to 3. The learning rate is set to 1e-4, and the number of diffusion training epochs is set to 2500. The scalar function $H(\hat{r}_{ij})$ is set to 0.1 if r_{ij} indicates "No-Relation", otherwise the function is set to 0.9. We conduct evaluations using 500 randomly generated event schemas for each performance metric. The node number is sampled from a range of 25 to 35. We choose the model checkpoint from the last epoch for evaluation.

In the Instance Graph Expansion process, we select the top 10 frequently occurring event sequences from the training data as inputs for ChatGPT. Each event sequence is input to ChatGPT 10 times to obtain the final result. Furthermore, we use a hyperparameter K of 3 to filter out events generated by ChatGPT that occur less frequently.

In the Entity Relation Prediction module, each event has a predetermined set of argument roles. For example, the "Injure" event may have the argument role "Victim" limited to entity types "PER" and "AML". We count the occurrences of entity categories for each role in all instance graphs. The entity category with the highest occurrence in the corresponding role is then inserted into the event skeleton.

To modify the test data, we made the following modifications: 1. Merge the same path: For all subsequent nodes of each event node, merge event

nodes with the same type, starting from the START node and merging in the order of the BFS algorithm. 2. Supplementary event nodes: Based on human judgment, randomly add possible missing events that may occur in the schema.

C Conditional Discrete Diffusion Model

Transition probability matrix in forward process. In discrete diffusion model, a transition probability matrix $\boldsymbol{Q}$ is defined to corrupt data for each step. Here $\boldsymbol{Q}_t = \alpha^t \mathbf{I} + (1 - \alpha^t)\mathbf{1}(\mathbf{1})^T/K$, where $\mathbf{1}$ is a column vector of all ones, α^t varies from 1 to 0 making sure the node and edge feature sampled from is a uniform distribution at time T (Hoogeboom et al., 2021; Yang et al., 2023). $\beta^t = (1 - \alpha^t)/K$ and the transition matrix can be represent as:

$$
\boldsymbol{Q}_t =
\begin{bmatrix}
\alpha^t + \beta^t & \beta^t & \dots & \beta^t \\
\beta^t & \alpha^t + \beta^t & \dots & \beta^t \\
\cdot & \cdot & \dots & \cdot \\
\cdot & \cdot & \dots & \cdot \\
\beta^t & \beta^t & \dots & \alpha^t + \beta^t
\end{bmatrix}
$$

we can calculate $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ according to following formula:

$$
\boldsymbol{x}_0\bar{\boldsymbol{Q}}_t = \bar{\alpha}^t\boldsymbol{x}_0 + \bar{\beta}^t. \tag{20}
$$

as t is enough large, α^t is close to 0, the graph distribution $\mathbf{G}_t$ is confirm to uniform distribution.

Reverse discrete diffusion process, we convert the noise $\boldsymbol{G}_T$ into $\boldsymbol{G}$, whose joint probability having a Markovian structure follows (Vignac et al., 2022):

$$
\log p_\theta(\boldsymbol{G}_{0:T}) = \log p(\boldsymbol{G}_T) \sum_{t=1}^{T} \log p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t)
$$

where p_θ is the process of the reverse with learnable parameters θ. and for each discrete elements x in graph $\boldsymbol{G}_{0:T}$ posterior probability is :

$$
\begin{aligned}
p_\theta(x_{t-1}|x_t) &= \sum_x q(x_{t-1}|x_t, x)p(x) \\
&= \sum_{x_0} q(x_{t-1}|x_t, x_0)p_\theta(x_0|x_t)
\end{aligned}
$$

The posterior $q(x_{t-1}|x_t, x_0)$ can be derived according to Bayes formula as follows (Austin et al.,

2021):

$$
\begin{aligned}
q(x_{t-1}|x_t, x_0) &= \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)} \\
&= \frac{x_t(\boldsymbol{Q}_t)^T \odot x_0\bar{\boldsymbol{Q}}_{t-1}}{x_0\bar{\boldsymbol{Q}}_t(x_t)^T}
\end{aligned}
$$

To train the discrete diffusion process, we minimize the negative logarithmic likelihood of the predicted distribution of the model using variational lower bound (VLB), We use $\boldsymbol{G}_0$ here to represent $\boldsymbol{G}$:

$$
\begin{aligned}
\mathcal{L} &= -E_{q(\boldsymbol{G}_0)}[\log p_\theta(\boldsymbol{G}_0)] \\
&\leq \mathcal{L}_{VLB} \\
&= -E_{q(\boldsymbol{G}_{0:G})}[\log p_\theta(\boldsymbol{G}_0)] \\
&= \mathbb{E}_{q(\boldsymbol{G}_{0:T})}[\underbrace{D_{KL}(q(\boldsymbol{G}_T|\boldsymbol{G}_0) \parallel p_\theta(\boldsymbol{G}_T))}_{L_T} \\
&\quad + \sum_{t=2}^{T} \underbrace{D_{KL}(q(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t, \boldsymbol{G}_0) \parallel p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t))}_{L_{t-1}} \\
&\quad \underbrace{- \log p_\theta(\boldsymbol{G}_0|\boldsymbol{G}_1)]}_{L_0} \\
&= \mathbb{E}_{q(\boldsymbol{G}_{0:T})}[\underbrace{D_{KL}(q(\boldsymbol{G}_T|\boldsymbol{G}_0) \parallel p_\theta(\boldsymbol{G}_T|\boldsymbol{G}_n, \boldsymbol{G}_d))}_{L_T} \\
&\quad + \log p_\theta(\boldsymbol{G}_n, \boldsymbol{G}_d) \\
&\quad + \sum_{t=2}^{T} \underbrace{D_{KL}(q(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t, \boldsymbol{G}_0) \parallel p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t))}_{L_{t-1}} \\
&\quad \underbrace{- \log p_\theta(\boldsymbol{G}_0|\boldsymbol{G}_1)]}_{L_0}
\end{aligned}
$$

Please note that $\boldsymbol{G}_t$ is sampled from the node number distribution $\boldsymbol{G}_n$ and the corresponding depth distribution $\boldsymbol{G}_d$. Therefore, the probability $p_\theta(\boldsymbol{G}_T)$ can be expressed as $p_\theta(\boldsymbol{G}_T) = p_\theta(\boldsymbol{G}_T|\boldsymbol{G}_n, \boldsymbol{G}_d)p_\theta(\boldsymbol{G}_n, \boldsymbol{G}_d)$.

The terms L_T and L_{t-1} represent the Kullback-Leibler (KL) divergences between graph categorical distributions, while L_0 represent the predicted probabilities of the graph $\boldsymbol{G}_0$ based on the noisy graph $\boldsymbol{G}_1$. The algorithms 1 and 2 is the training and generating algorithms about **KDM**.

D Supplement Experiment

As presented in Table 5, we evaluate our model on the original testing data used by (Jin et al., 2022) and observe consistent outperformance of our model compared to the baselines specially in sequence match. This result highlights the strong capability of our discrete diffusion model in generating high-quality event schemas. Interestingly, we

(a) t=500

(b) t=250

(c) t=0

Figure 6: Case Study on diffusion process. During the reverse process, t gradually changes from 500 to 0, and the corresponding schema for t at different times corresponds to the above.

note that models without knowledge expansion exhibited better performance in this evaluation. This finding also suggests that the original test data may not effectively measure the quality of the generated schemas.

Algorithm 1: Training **KDM**

1 **for** G *in* $\mathcal{D}$ **do**
2   sample t from Uniform(T);
3   sample $\mathcal{G}_t$ from distribution
        $(\boldsymbol{N}\bar{\boldsymbol{Q}}_t^N, \boldsymbol{E}\bar{\boldsymbol{Q}}_t^E)$;
4   estimate distribution $\boldsymbol{G}_p^t = \phi_\theta(\mathcal{G}_t)$;
5   calculate loss $\mathcal{L}_{CE}(\boldsymbol{G}_p^t, \boldsymbol{G})$;
6   update network ϕ_θ
7 **end**

Algorithm 2: Sampling from **KDM**

1 sample $\mathcal{G}_T$ from Uniform distribution;
2 **for** $t = T$ *to 1* **do**
3   convert $\mathcal{G}_t$ to distribution $\boldsymbol{G}_t$;
4     $\boldsymbol{G}_p^t = \phi_\theta(\boldsymbol{G}_t)$;
5   estimate distribution $p_\theta(\boldsymbol{G}_{t-1}|\boldsymbol{G}_t)$;
6   sample $\mathcal{G}_{t-1}$ from distribution;
7 **end**

E   Instance Graph Expansion Details

The process of the Instance Graph Expansion is shown in Figure 7, we also present our event knowledge expansion prompt in Figure 8, temporal relation expansion prompt in Figure 10 and entity relation expansion prompt in Figure 9.

Dataset	Model	Event type Match(F1)	Event sequence match(F1)		KL divergence	
			l=2	l=3	Node-type	Edge-type
General-IED	FBS	0.617	0.149	0.064	1.88	4.32
	DoubleGAE	0.697	0.273	0.128	**1.66**	4.96
	KDM	0.663	0.318	0.132	3.01	5.08
	KDM w/o IGE	**0.698**	**0.327**	**0.138**	2.52	**4.87**
Car-IED	FBS	0.542	0.126	0.038	4.12	6.37
	DoubleGAE	0.674	0.259	0.081	2.14	5.42
	KDM	0.702	0.371	0.180	2.07	4.47
	KDM w/o IGE	**0.757**	**0.411**	**0.185**	**1.42**	**4.24**
Suicide-IED	FBS	0.642	0.164	0.048	2.39	6.36
	DoubleGAE	0.709	0.290	0.095	1.76	5.91
	KDM	0.688	0.415	0.217	2.13	3.75
	KDM w/o IGE	**0.738**	**0.467**	**0.268**	**1.61**	**3.50**

Table 5: Schema matching score (%) by checking the intersection of induced schemas and test schemas. both the training and testing instances are not through Knowledge Expansion. The baseline's results are provided by previous work.
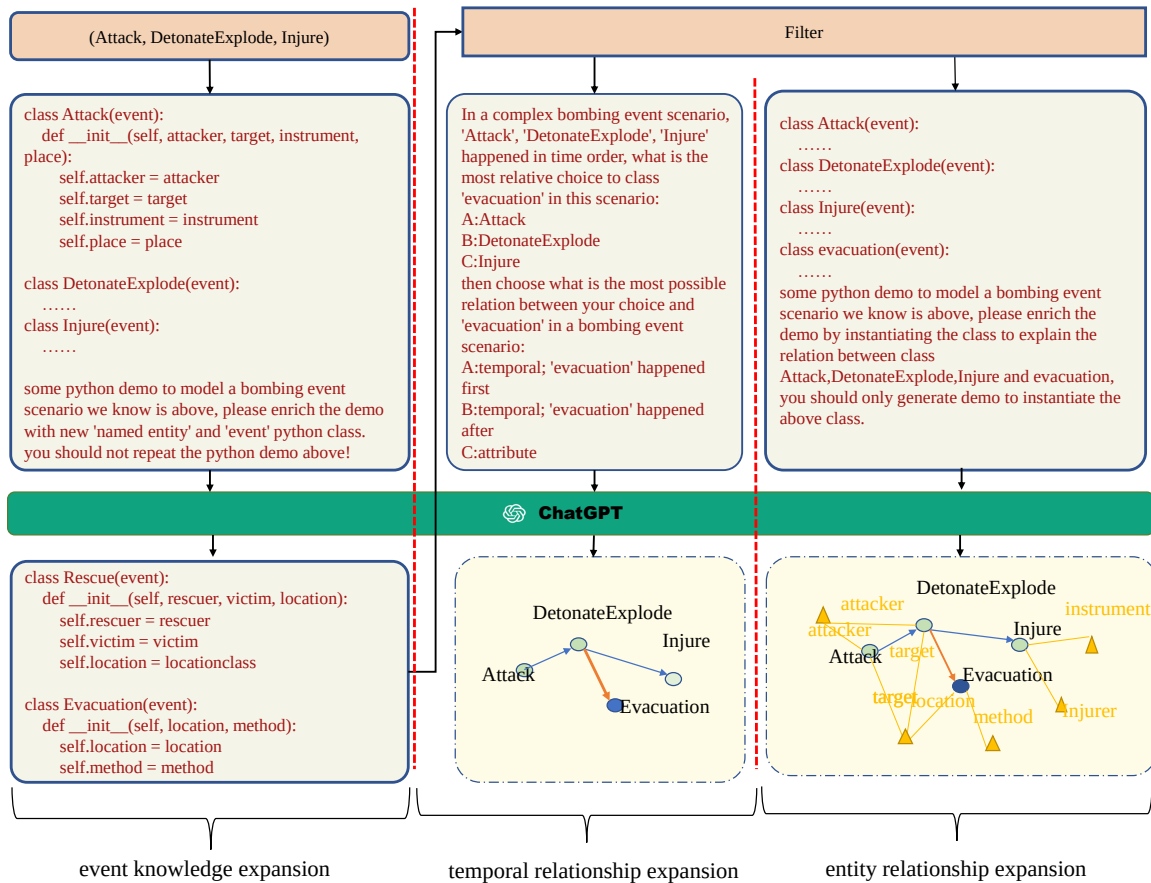

Figure 7: This figure illustrates the Instance Graph Expansion process. The event knowledge expansion process write selected event sequences into Python code and complete them by ChatGPT to generate new events; The temporal relationship expansion process utilize choice question to link new event to sequence; The entity relationship expansion process instantize the event class to get argument relations.

Input:

```
class Attack(event):
    def __init__(self, attacker, target, instrument, place):
        self.attacker = attacker
        self.target = target
        self.instrument = instrument
        self.place = place
class DetonateExplode(event):
    def __init__(self, attacker, target, instrument, explosive_device, place):
        self.attacker = attacker
        self.target = target
        self.instrument = instrument
        self.explosive_device = explosive_device
        self.place = place
class Injure(event):
    def __init__(self, victim, injurer, instrument):
        self.victim = victim
        self.injurer = injurer
        self.instrument = instrument
```

some python demo to model a bombing event scenario we know is above, please enrich the demo with new 'named entity' and 'event' python class. you should not repeat the python demo above!you should not repeat the python demo above!you should not repeat the python demo above!

Answer:

```
class Bombing(event):
    def __init__(self, attacker, target, instrument, place, casualties):
        self.attacker = attacker
        self.target = target
        self.instrument = instrument
        self.place = place
        self.casualties = casualties
class Evacuation(event):
    def __init__(self, location, evacuees, reason):
        self.location = location
        self.evacuees = evacuees
        self.reason = reason
class Investigation(event):
    def __init__(self, location, investigators, start_date):
        self.location = location
        self.investigators = investigators
        self.start_date = start_date
```

Figure 8: Instance of event knowledge expansion process

Input:

In a complex bombing event scenario, 'Attack', 'DetonateExplode', 'Injure' happened in time order, what is the most relative choice to class 'evacuation' in this scenario, please choose your answer from:
A:Attack
B:DetonateExplode
C:Injure

then choose what is the most possible relation between your choice and 'evacuation' in a bombing event scenario, please choose your answer from:
A:temporal; 'evacuation' happened first
B:temporal; 'evacuation' happened after
C:attribute
(attention you can only answer the above two questions only with A or B or C)

Answer:

B: DetonateExplode
B: Temporal; 'evacuation' happened after

Figure 9: Instance of temporal relation expansion process

Input:

```
class Attack(event):
    def __init__(self, attacker, target, instrument, place):
        self.attacker = attacker
        self.target = target
        self.instrument = instrument
        self.place = place
class DetonateExplode(event):
    def __init__(self, attacker, target, instrument, explosive_device, place):
        self.attacker = attacker
        self.target = target
        self.instrument = instrument
        self.place = place
class Injure(event):
    def __init__(self, victim, injurer, instrument):
        self.victim = victim
        self.injurer = injurer
        self.instrument = instrument
class Evacuation(event):
    def __init__(self, location, method):
        self.location = location
        self.method = method
```

some python demo to model a bombing event scenario we know is above, please enrich the demo by instantiating the class to explain the relation between class Attack,DetonateExplode,Injure and Bombing, you should only generate demo to instantiate the above class.
one demo example is below:

```
# Instantiate Transport event
transport_event = Transport(Vehicle ="car", Origin ="park", Destination ="Building)
# Instantiate DetonateExplode event
detonate_explode_event = DetonateExplode(attacker="John", target="Building", instrument="Explosive",
explosive_device="C4", place="City Center")
# Instantiate Investigate event
investigate_event = Investigate(location="City Center", investigator="Detective Smith")
```

Answer:

```
# Instantiate Attack event
attack_event = Attack(attacker="John", target="Building", instrument="Explosive", place="City Center")
# Instantiate DetonateExplode event
detonate_explode_event = DetonateExplode(attacker="John", target="Building", instrument="Explosive",
explosive_device="C4", place="City Center")
# Instantiate Injure event
injure_event = Injure(victim="Jane", injurer="John", instrument="Explosive")
# Instantiate Evacuation event
evacuation_event = Evacuation(location="City Center", method="Emergency evacuation")
```

Figure 10: Instance of entity relation expansion process