The Future is not One-dimensional: Complex Event Schema Induction by Graph Modeling for Event Prediction

Manling Li[1], **Sha Li**[1], **Zhenhailong Wang**[1], **Lifu Huang**[2],
Kyunghyun Cho[3], **Heng Ji**[1], **Jiawei Han**[1], **Clare Voss**[4]
[1]UIUC [2]Virginia Tech [3]NYU [4]US ARL
{manling2,shal2,wangz3,hengji,hanj}@illinois.edu,
lifuh@vt.edu, kyunghyun.cho@nyu.edu, clare.r.voss.civ@mail.mil

Abstract

Event schemas encode knowledge of stereotypical structures of events and their connections. As events unfold, schemas are crucial to act as a scaffolding. Previous work on event schema induction focuses either on atomic events or linear temporal event sequences, ignoring the interplay between events via arguments and argument relations. We introduce a new concept of *Temporal Complex Event Schema*: a graph-based schema representation that encompasses events, arguments, temporal connections and argument relations. In addition, we propose a *Temporal Event Graph Model* that predicts event instances following the temporal complex event schema. To build and evaluate such schemas, we release a new schema learning corpus containing 6,399 documents accompanied with event graphs, and we have manually constructed gold-standard schemas. Intrinsic evaluations by *schema matching* and *instance graph perplexity*, prove the superior quality of our probabilistic graph schema library compared to linear representations. Extrinsic evaluation on *schema-guided future event prediction* further demonstrates the predictive power of our event graph model, significantly outperforming human schemas and baselines by more than 23.8% on HITS@1. [1]

1 Introduction

The current automated event understanding task has been overly simplified to be local and sequential. Real world events, such as disease outbreaks and terrorist attacks, have multiple actors, complex timelines, intertwined relations and multiple possible outcomes. Understanding such events requires knowledge in the form of a library of event schemas, capturing the progress of time, and performing global inference for event prediction. For example, regarding the 2019 protest in Hong Kong International Airport, a typical question from analysts would be "*How long will the flights being canceled?*" This requires an event understanding system to match events to schema representations and reason about what might happen next. The *airport protest* schema would be triggered by "protest" and "flight cancellation", and evidence of protesters (e.g., the number of protesters, the instruments being used, etc) will suggest a CEO resignation event, or a flight rescheduling event, or continuous flight cancellation events with respective probabilities.

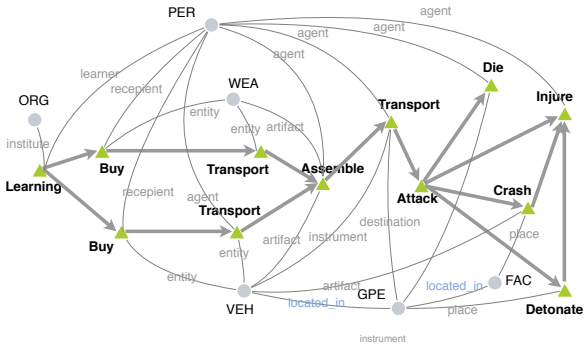

Figure 1: The example schema of the complex event type *car-bombing*. A person learned to make bombs and bought materials as well as a vehicle. Then the bomb was assembled to the vehicle, and then the attacker drove it to attack people. People can be hurt by the vehicle, or by the explosion of the bomb, or by the crash of the vehicle.

Comprehending such a news story requires following a timeline, identifying key events and tracking characters. We refer to such a "story" as a **complex event**, e.g., the *Kabul ambulance bombing* event. Its complexity comes from the inclusion of multiple atomic events (and their arguments), relations and temporal order. A **complex event schema** can be used to define the typical structure of a particular type of complex event, e.g., *car-bombing*. This leads us to the new task that we address in this paper: **temporal complex event**

[1]The programs, data and resources are made publicly available for research purpose in https://github.com/limanling/temporal-graph-schema.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 5203–5215
November 7–11, 2021.

schema induction. Figure 1 shows an example schema about *car-bombing* with multiple temporal dependencies between events. Namely, the occurrence of one event may depend on multiple events. For example, the ASSEMBLE event happens after buying both the bomb materials and the vehicle. Also, there may be multiple events following an event, such as the multiple consequences of the ATTACK event in Figure 1. That is to say, "the future is not one-dimensional". Our automatically induced probabilistic complex event schema can be used to forecast event abstractions into the future and thus provide a comprehensive understanding of evolving situations, events, and trends.

For each type of complex event, we aim to induce a schema library that is probabilistic, temporally organized and semantically coherent. Low level atomic event schemas are abundant, and can be part of multiple, sparsely occurring, higher-level schemas. We propose a **Temporal Event Graph Model**, an auto-regressive graph generation model, to reach this goal. Given a currently extracted event graph, we generate the next event type node with its potential arguments, such as the ARREST event in Figure 2, and then propagate edge-aware information following temporal orders. After that, we employ a copy mechanism to generate coreferential arguments, such as the DETAINEE argument is the ATTACKER of the previous ATTACK event, and build relation edges for them, e.g., PART_WHOLE relation between the PLACE arguments. Finally, temporal dependencies are determined with argument connections considered, such as the temporal edge showing that ARREST is after ATTACK.

Our generative model serves as both a schema library and a predictive model. Specifically, we can probe the model to generate event graphs unconditionally to obtain a set of schemas. We can also pass partially instantiated graphs to the model and "grow" the graph either forward or backward in time to predict missing events, arguments or relations, both from the past and in the future. We propose a set of schema matching metrics to evaluate the induced schemas by comparing with human-created schemas and show the power of the probabilistic schema in the task of *future event prediction* as an extrinsic evaluation, to predict event types that are likely to happen next.

We make the following novel contributions:

- This is the first work to induce probabilistic temporal graph schemas for complex events across documents, which capture temporal dynamics and connections among individual events through their coreferential or related arguments.
- This is the first application of graph generation methods to induce event schemas.
- This is the first work to use complex event schemas for event type prediction, and also produce multiple hypotheses with probabilities.
- We have proposed a comprehensive set of metrics for both intrinsic and extrinsic evaluations.
- We release a new data set of 6,399 documents with gold-standard schemas annotated manually.

Symbol	Meaning
$G \in \mathcal{G}$	Instance graph of a complex event
$S \in \mathcal{S}$	Schema graph of a complex event type
$e \in \mathcal{E}$	Event node in an instance graph
$v \in \mathcal{V}$	Entity node in an instance graph
$\langle e_i, e_l \rangle$	Temporal ordering edge between events e_i and e_l, indicating e_i is before e_l
$\langle e_i, a, v_j \rangle$	Argument edge, indicating v_j plays argument role a in the event e_i
$\langle v_j, r, v_k \rangle$	Relation edge between entities v_j and v_k, and r is the relation type
$\mathcal{A}(e)$	Argument role set of event e, defined by the IE ontology
$\Phi_{\mathcal{E}}$	The type set of events
$\Phi_{\mathcal{V}}$	The type set of entities
$\phi(\cdot)$	A mapping function from a node to its type
$G_{<i}$	Subgraph of G containing events before e_i and their arguments

Table 1: List of symbols

2 Problem Formulation

From a set of documents describing a complex event, we construct an **instance graph** G which contains event nodes E and entity nodes (argument nodes) V. There are three types of edges in this graph: (1) event-event edges $\langle e_i, e_l \rangle$ connecting events that have direct temporal relations; (2) event-entity edges $\langle e_i, a, v_j \rangle$ connecting arguments to the event; and (3) entity-entity edges $\langle v_j, r, v_k \rangle$ indicating relations between entities. We can construct instance graphs by applying Information Extraction (IE) techniques on an input text corpus. In these graphs, the relation edges do not have directions but temporal edges between events are directional, going from the event before to the event after.

For each complex event type, given a set of instance graphs $\mathcal{G}$, the goal of schema induction is to generate a schema library $\mathcal{S}$. In each **schema graph** S, the nodes are abstracted to the types of events and entities. Figure 1 is an example

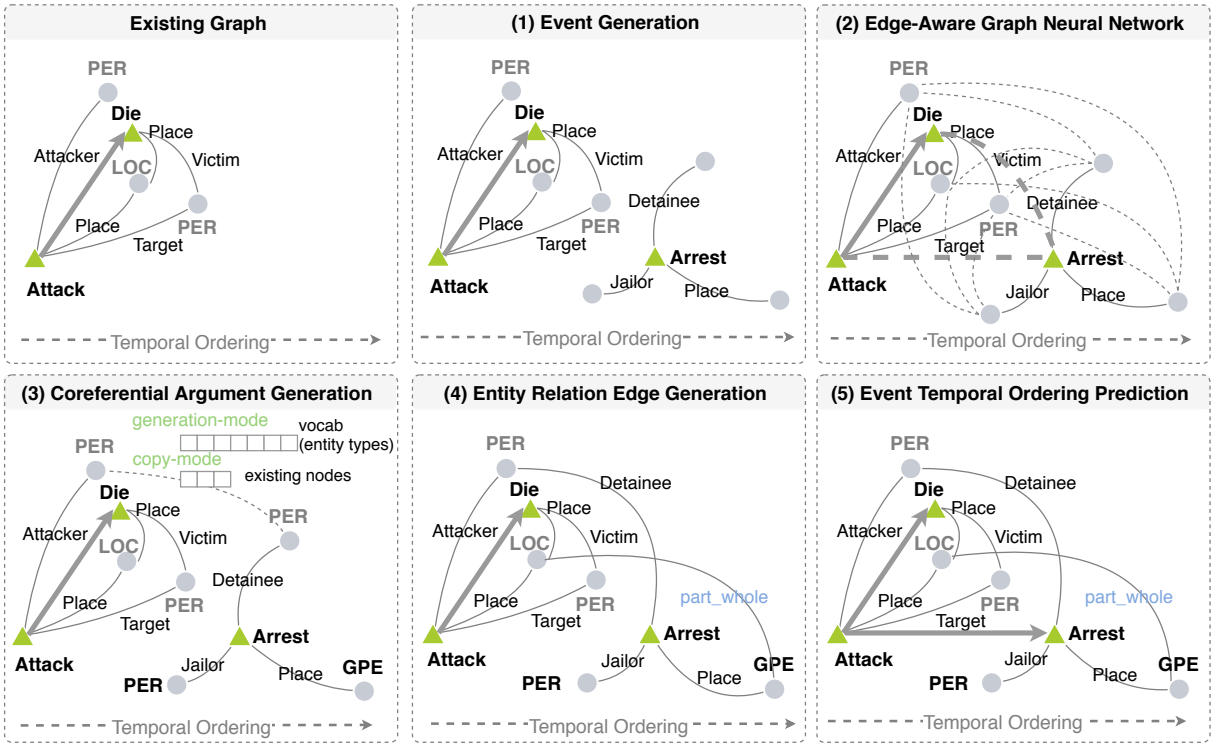

Figure 2: The generation process of Temporal Event Graph Model.

of schema[2] for complex event type *car-bombing*. Schema graphs can be regarded as a summary abstraction of instance graphs, capturing the reoccurring structures.

3 Our Approach

3.1 Instance Graph Construction

To induce schemas for a complex event type, such as *car-bombing*, we construct a set of instance graphs, where each instance graph is about one complex event, such as *Kabul ambulance bombing*.

We first identify a cluster of documents that describes the same complex event. In this paper, we treat all documents linked to a single Wikipedia page as belonging to the same complex event, detailed in §4.1.

We use OneIE, a state-of-the-art Information Extraction system (Lin et al., 2020), to extract entities, relations and events, and then perform cross-document entity (Pan et al., 2015, 2017) and event coreference resolution (Lai et al., 2021) over the document cluster of each complex event. We further conduct event-event temporal relation extraction (Ning et al., 2019; Wen et al., 2021b) to determine the order of event pairs. We run the entire pipeline following (Wen et al., 2021a) [3], and the detailed extraction performance is reported in the paper.

After extraction, we construct one instance graph for each complex event, where coreferential events or entities are merged. We consider the isolated events as irrelevant nodes in schema induction, so they are excluded from the instance graphs during graph construction. Considering schema graphs focus on type-level abstraction, we use type label and node index to represent each node, ignoring the mention level information in these instance graphs.

3.2 Temporal Event Graph Model Overview

Given an instance graph $\mathcal{G}$, we regard the schema as the hidden knowledge to guide the generation of these graphs. To this end, we propose a temporal event graph model that maximizes the probability of each instance graph, parameterized by $\prod_{G \in \mathcal{G}} p(G)$. At each step, based on the previous graph $G_{<i}$, we predict one event node e_i with its arguments to generate the next graph G_i,

$$p(G) = \prod_{i=0}^{|\mathcal{E}|} p(G_i | G_{<i}).$$

[2]For simplification purposes, we mention "schema graphs" as "schemas", and "events" in schemas are only "event types".

[3]https://github.com/RESIN-KAIROS/RESIN-pipeline-public

We factorize the probability of generating new nodes and edges as:

$$p(G_i|G_{<i}) = p(e_i|G_{<i}) \prod_{a_j \in \mathcal{A}(e_i)} p(\langle e_i, a_j, v_j \rangle | e_i, a_j)$$
$$\prod_{v_k \in G_{<i}} p(\langle v_j, r, v_k \rangle | v_j, v_k) \prod_{e_l \in G_{<i}} p(\langle e_i, e_l \rangle | e_i, e_l). \quad (1)$$

As shown in Figure 2, an event node e_i is generated first according to the probability $p(e_i|G_{<i})$. We then add argument nodes based on the IE ontology. We also predict relation $\langle v_j, r, v_k \rangle$ between the newly generated node v_j and the existing nodes $v_k \in G_{<i}$. After knowing the shared and related arguments, we add a final step to predict the temporal relations between the new event e_i and the existing events $e_l \in G_{<i}$.

In the traditional graph generation setting, the order of node generation can be arbitrary. However, in our instance graphs, event nodes are connected through temporal relations. We order events as a directed acyclic graph (DAG). Considering each event may have multiple events both "before" and "after", we obtain the generation order by traversing the graph using Breadth-First Search.

We also add dummy START/END event nodes to indicate the starting/ending of the graph generation. At the beginning of the generation process, the graph G_0 has a single start event node $e_{[\text{SOG}]}$. We generate $e_{[\text{EOG}]}$ to signal the end of the graph.

3.3 Event Generation

To determine the event type of the newly generated event node e_i, we apply a graph pooling over all events to get the current graph representation $\boldsymbol{g}_i$,

$$\boldsymbol{g}_i = \text{Pooling}(\{\boldsymbol{e}_0, \cdots, \boldsymbol{e}_{i-1}\}).$$

We use bold to denote the latent representations of nodes and edges, which will be initialized as zeros and updated at each generation step via message passing in § 3.4. We adopt a mean-pooling operation in this paper. After that, the event type is predicted through a fully connected layer,

$$p(e_i|G_{<i}) = \frac{\exp(\boldsymbol{W}_{\phi(e_i)}\boldsymbol{g}_i)}{\sum_{\phi' \in \Phi_{\mathcal{E}} \cup [\text{EOG}]} \exp(\boldsymbol{W}_{\phi'}\boldsymbol{g}_i)}.$$

Once we know the event type of e_i, we add all of its arguments in $\mathcal{A}(e_i)$ defined in the IE ontology as new entity nodes. For example, in Figure 2, the new event e_i is an ARREST event, so we add three argument nodes for DETAINEE, JAILOR, and PLACE respectively. The edges between these arguments and event e_i are also added into the graph.

3.4 Edge-Aware Graph Neural Network

We use a Graph Neural Network (GNN) (Kipf and Welling, 2017) to update node embeddings following the graph structure. Before we run the GNN on the graph, we first add *virtual edges* between the newly generated event and all previous events, and between new entities and previous entities, shown as dashed lines in Figure 2. The virtual edges enable the representations of new nodes to aggregate the messages from previous nodes, which has been proven effective in (Liao et al., 2019).

To capture rich semantics of edge types, we pass edge-aware messages during graph propagation. An intuitive way is to encode different edge types with different convolutional filters, which is similar to RGCN (Schlichtkrull et al., 2018). However, the number of RGCN parameters grows rapidly with the number of edge types and easily becomes unmanageable given the large number of relation types and argument roles in the IE ontology.[4] Instead, we learn a vector representation for each relation type r and argument role a. The message passed through each argument edge $\langle e_i, a, v_j \rangle$ is:

$$\boldsymbol{m}_{i,j} = \text{ReLU}\left(\boldsymbol{W}_a((\boldsymbol{e}_i - \boldsymbol{v}_j) \parallel \boldsymbol{a})\right),$$

where $\parallel$ denotes concatenation operation. Similarly, the message between two entities v_j and v_k is:

$$\boldsymbol{m}_{j,k} = \text{ReLU}\left(\boldsymbol{W}_r((\boldsymbol{v}_j - \boldsymbol{v}_k) \parallel \boldsymbol{r})\right).$$

Considering that the direction of the temporal edge is important, we parametrize the message over this edge by assigning two separate weight matrices to the outgoing and incoming vertices:

$$\boldsymbol{m}_{i,l} = \text{ReLU}\left(\boldsymbol{W}_{\text{bfr}}\boldsymbol{e}_i - \boldsymbol{W}_{\text{aft}}\boldsymbol{e}_l\right).$$

We aggregate the messages using edge-aware attention following (Liao et al., 2019):[5]

$$\alpha_{i,j} = \sigma(\text{MLP}(\boldsymbol{e}_i - \boldsymbol{e}_j)),$$

where σ is the sigmoid function, and MLP contains two hidden layers with ReLU nonlinearities.

The event node representation $\boldsymbol{e}_i$ is then updated using the messages from its local neighbors $\mathcal{N}(e_i)$, similar to entity node representations:

$$\boldsymbol{e}_i \leftarrow \text{GRU}\left(\boldsymbol{e}_i \parallel \sum\nolimits_{j \in \mathcal{N}(e_i)} \alpha_{i,j}\boldsymbol{m}_{i,j}\right).$$

[4] There are 131 edge types according to the fine-grained LDC Schema Learning Ontology.

[5] Compared to (Liao et al., 2019), we do not use the positional embedding mask because the newly generated nodes have distinct roles.

3.5 Coreferential Argument Generation

After updating the node representations, we detect the entity type of each argument, and also predict whether the argument is coreferential to existing entities. Inspired by copy mechanism (Gu et al., 2016), we classify each argument node v_j to either a new entity with entity type $\phi(v_j)$, or an existing entity node in the previous graph $G_{<i}$. For example, in Figure 2, the DETAINEE should be classified to the existing ATTACKER node, while JAILOR node is classified as PERSON. Namely,

$$p(\langle e_i, a_j, v_j\rangle | e_i, a_j) = \begin{cases} p(\langle e_i, a_j, v_j\rangle, \mathsf{g} | e_i, a_j) & \text{if } v_j \text{ is new,} \\ p(\langle e_i, a_j, v_j\rangle, \mathsf{c} | e_i, a_j) & \text{otherwise,} \end{cases}$$

where $p(\langle e_i, a_j, v_j\rangle, \mathsf{g} | e_i, a_j)$ is the generation probability, classifying the new node to its entity type $\phi(v_j)$:

$$p(\langle e_i, a_j, v_j\rangle, \mathsf{g} | e_i, a_j) = \exp(\boldsymbol{W}_{\phi(v_j)} \boldsymbol{v}_j)/Z$$

The copy probability $p(\langle e_i, a_j, v_j\rangle, \mathsf{c} | e_i, a_j)$ selects the coreferential entity v from the entities in existing graph, denoted by $V_{<i}$,

$$p(\langle e_i, a_j, v_j\rangle, \mathsf{c} | e_i, a_j) = \exp(\boldsymbol{W}_{v} \boldsymbol{v}_j)/Z.$$

Here, Z is the shared normalization term,

$$Z = \sum\nolimits_{\phi' \in \Phi_{\mathcal{V}}} \exp(\boldsymbol{W}_{\phi'} \boldsymbol{v}_j) + \sum\nolimits_{v' \in V_{<i}} \exp(\boldsymbol{W}_{v'} \boldsymbol{v}_j)$$

If determined to copy, we merge coreferential entities in the graph.

3.6 Entity Relational Edge Generation

In this phase, we determine the *virtual edges* to be kept and assign relation types to them, such as PARTWHOLE relation in Figure 2. We model the relation edge generation probability as a categorical distribution over relation types, and add [O] (OTHER) to the typeset $\mathcal{R}$ to represent that there is no relation edge:

$$p(\langle v_j, r, v_k\rangle | v_j, v_k) = \frac{\exp(\text{MLP}_r(\boldsymbol{v}_j - \boldsymbol{v}_k))}{\sum\limits_{r' \in \mathcal{R} \cup [\text{O}]} \exp(\text{MLP}_{r'}(\boldsymbol{v}_j - \boldsymbol{v}_k))}$$

We use two hidden layers with ReLU activation functions to implement the MLP.

3.7 Event Temporal Ordering Prediction

To predict the temporal dependencies between the new events and existing events, we connect them through temporal edges, as shown in Figure 2. These edges are critical for message passing in predicting the next event. We build temporal edges in the last phase of generation, since it relies on the shared and related arguments. Considering that temporal edges are interdependent, we model the generation probability as a mixture of Bernoulli distributions following (Liao et al., 2019):

$$p(\langle e_i, e_l\rangle | e_i, e_l) = \sum\nolimits_b \gamma_b \, \theta_{b,i,l},$$
$$\gamma_1, \cdots, \gamma_B = \text{Softmax}\left(\sum\nolimits_{i,l} \text{MLP}(\boldsymbol{e}_i - \boldsymbol{e}_l)\right),$$
$$\theta_{1,i,l}, \cdots, \theta_{B,i,l} = \sigma\left(\text{MLP}_\theta(\boldsymbol{e}_i - \boldsymbol{e}_l)\right),$$

where B is the number of mixture components. When $B = 1$, the distribution degenerates to factorized Bernoulli, which assumes the independence of each potential temporal edge conditioned on the existing graph.

3.8 Training and Schema Decoding

We train the model by optimizing the negative log-likelihood loss,

$$\mathcal{L} = \sum\nolimits_{G \in \mathcal{G}_{\text{train}}} -\log_2 p(G).$$

To compose the schema library for each complex event scenario, we construct instance graphs from related documents to learn a graph model, and then obtain the schema using greedy decoding.

4 Evaluation Benchmark

4.1 Dataset

We conduct experiments on two datasets for both the general scenario and a more specific scenario. We adopt the DARPA KAIROS[6] ontology, a newly defined fine-grained ontology for Schema Learning, with 24 entity types, 46 relation types, 67 event types, and 85 argument roles. [7] Our schema induction method does not rely on any specific ontology, only the IE system is trained on a given ontology to create the instance event graphs.

General Schema Learning Corpus: The Schema Learning Corpus, released by LDC (LDC2020E25), includes 82 types of complex events, such as *Disease Outbreak*, *Presentations* and *Shop Online*.

[6] https://github.com/NextCenturyCorporation/kairos-pub/tree/master/data-format/ontology

[7] The ontology has been released in LDC2020E25.

Each complex event is associated with a set of source documents. This data set also includes ground-truth schemas created by LDC annotators, which were used for our intrinsic evaluation.

Dataset	Split	#doc	#graph	#event	#arg	#rel
General	Train	451	451	6,040	10,720	6,858
	Dev	83	83	1,044	1,762	1,112
	Test	83	83	1,211	2,112	1,363
IED	Train	5,247	343	41,672	136,894	122,846
	Dev	575	42	4,661	15,404	13,320
	Test	577	45	5,089	16,721	14,054

Table 2: Data statistics. Each instance graph is about one complex event.

IED Schema Learning Corpus: The same type of complex events may have many variants, which depends on the different types of conditions and participants. In order to evaluate our model's capability at capturing uncertainty and multiple hypotheses, we decided to dive deeper into one scenario and chose the *improvised explosive device (IED)* as our case study. We first collected Wikipedia articles that describe 4 types of complex events, i.e., *Car-bombing IED*, *Drone Strikes IED*, *Suicide IED* and *General IED*. Then we followed (Li et al., 2021) to exploit the external links to collect the additional news documents with the corresponding complex event type.

The ground-truth schemas for this IED corpus are created manually, through a schema curation tool (Mishra et al., 2021). Only one human schema graph was created for each complex event type, resulting in 4 schemas. In detail, for each complex event type, we presented example instance graphs and the ranked event sequences to annotators to create human (ground truth) schemas. The event sequences are generated by traversing the instance graphs, and then sorted by frequency and the number of arguments. Initially we assigned three annotators (IE experts) to each create a version of the schema and then the final schema was merged through discussion. After that, two annotators (linguists) performed a two-pass revision. Human curation focuses on merging and trimming steps by validating them using the reference instance graphs. Also, temporal dependencies between steps were further refined, and coreferential entities and their relations were added during the curation process. To avoid bias from the event sequences, linguists in the second round revision were not presented with the event sequences. All annotators were trained and disagreements were resolved through discussion.

4.2 Schema Matching Evaluation

We compare the generated schemas with the ground truth schemas based on the overlap between them. The following evaluation metrics were employed:[8]

Event Match: A good schema must contain the events crucial to the complex event scenario. *F-score* is used to compute the overlap of event nodes.

Event Sequence Match: A good schema is able to track events through a timeline. So we obtain event sequences following temporal order, and evaluate *F-score* on the overlapping sequences of lengths $l = 2$ and $l = 3$.

Event Argument Connection Match: Our complex event graph schema includes entities and their relations and captures how events are connected through arguments, in addition to their temporal order. We categorize these connections into three categories: (1) two events are connected by shared arguments; (2) two events have related arguments, i.e., their arguments are connected through entity relations; (3) there are no direct connections between two events. For every pair of overlapped events, we calculate *F-score* based on whether these connections are predicted correctly. The human schemas of the General dataset do not contain arguments and the relations between arguments, so we only compute this metric for the IED dataset.

4.3 Instance Graph Perplexity Evaluation

To evaluate our temporal event graph model, we compute the *instance graph perplexity* by predicting the instance graphs in the test set,

$$\text{PP} = 2^{-\frac{1}{|\mathcal{G}_{\text{test}}|}\sum_{G\in\mathcal{G}_{\text{test}}}\log_2 p(G)}. \quad (1)$$

We calculate the *full perplexity* for the entire graph using Equation (1), and *event perplexity* using only event nodes, emphasizing the importance of correctly predicting events.

4.4 Schema-Guided Event Prediction

To explore schema-guided probabilistic reasoning and prediction, we perform an extrinsic evaluation of event prediction. Different from traditional event prediction tasks, the temporal event graphs contain arguments with relations, and there are

[8] We cannot use graph matching to compare between baselines and our approach due to the difference in the graph structures being modeled.

type labels assigned to nodes and edges. We create a graph-based event prediction dataset using our testing graphs. The task aims to predict ending events of each graph, i.e., events that have no future events after it. An event is predicted correctly if its event type matches one of the ending events in the graph. Considering that there can be multiple ending events in one instance graph, we rank event type prediction scores and adopt *MRR (Mean Reciprocal Rank)* and *HITS@1* as evaluation metrics.

5 Experiments

5.1 Experiment Setting

Baseline 1: Event Language Model (Rudinger et al., 2015; Pichotta and Mooney, 2016) is the state-of-the-art event schema induction method. It learns the probability of temporal event sequences, and the event sequences generated from event language model are considered as schemas.

Baseline 2: Sequential Pattern Mining (Pei et al., 2001) is a classic algorithm for discovering common sequences. We also attach arguments and their relations as extensions to the pattern. Considering that the event language model baseline cannot handle multiple arguments and relations, we add sequential pattern mining for comparison. The frequent patterns mined are considered as schemas.

Reference: Human Schema is added as a baseline in the extrinsic task of event prediction. Since human-created schemas are highly accurate but not probabilistic, we want to evaluate their limits at predicting events in the extrinsic task. We match schemas to instances and fill in the matched type.

Ablation Study: Event Graph Model w/o Argument Generation is included as a variant of our model in which we remove argument generation (§3.5 and §3.6). It learns to generate a graph containing only event nodes with their temporal relations, aiming to verify whether incorporating argument information helps event modeling.

5.2 Implementation Details

Training Details. For our event graph model, the representation dimension is 128, and we use a 2-layer GNN. The value of B is 2. The number of mixture components in temporal classifier is 2. The learning rate is 1e-4. To train event language model baseline, instead of using LSTM-based architecture following (Pichotta and Mooney, 2016), we adopt the state-of-the-art auto-regressive language XLNet (Yang et al., 2019). In detail, we first linearize the graph using topological sort, and then train XLNet[9] using the dimension of 128 (the same as our temporal event graph model), and the number of layers is 3. The learning rate is 1e-4. We select the best model on the validation set. Both of our model and event language model baseline are trained on one Tesla V100 GPU with 16GB DRAM. For sequential pattern mining, we perform random walk, starting from every node in instance graphs and ending at sink nodes, to obtain event type sequences, and then apply PrefixSpan (Pei et al., 2001) [10] to rank sequential patterns.

Evaluation Details. To compose the schema library, we use the first ranked sequence as the schema for these two models. To perform event prediction using baselines, we traverse the input graph to obtain event type sequences, and conduct prediction on all sequences to produce an averaged score. For human schemas, we first linearize them and the input graphs, and find the longest common subsequence between them.

5.3 Results and Analysis

Intrinsic Evaluation. In Table 3, the significant gain on *event match* demonstrates the ability of our graph model to keep salient events. On *sequence match*, our approach achieves larger performance gain compared to baselines when the path length l is longer. It implies that the proposed model is capable of capturing longer and wider temporal dependencies. In the case of *connection match*, only sequential pattern mining in the baselines can predict connections between events. When compared against sequential pattern mining, our generation model significantly performs better since it considers the inter-dependency of arguments and encodes them with graph structures.

Extrinsic Evaluation. On the task of schema-guided event prediction, our graph model obtains significant improvement (see Table 4.) The low performance of human schema demonstrates the importance of probabilistically modeling schemas to support downstream tasks. Take Figure 3 as an example. Human schemas produce incorrect event types such as TRAILHEARING, since it matches the sequence ATTACK→DIE→TRAILHEARING, incapable of capturing the inter-dependencies between sequences. However, our model is able to customize the prediction to the global context of the input

[9] https://github.com/huggingface

[10] https://github.com/chuanconggao/PrefixSpan-py

Dataset	Model	Event Match	Sequence Match $l=2$	Sequence Match $l=3$	Connection Match	Event Perplexity	Full Perplexity
General	Event Language Model	54.76	22.87	8.61	-	-	-
	Sequential Pattern Mining	49.18	20.31	7.37	-	-	-
	Event Graph Model	**58.15**	**24.79**	**9.18**	-	**24.25**	137.18
	w/o ArgumentGeneration	56.96	22.47	8.21	-	68.59	-
IED	Event Language Model	49.15	17.77	5.32	-	-	-
	Sequential Pattern Mining	47.91	18.39	4.79	5.41	-	-
	Event Graph Model	**59.73**	**21.51**	**7.81**	**10.67**	**39.39**	168.89
	w/o ArgumentGeneration	55.01	18.24	6.67	-	51.98	-

Table 3: Intrinsic evaluation results, including schema matching F1 score (%) and instance graph perplexity.

Dataset	Model	MRR	HITS@1
General	Event Language Model	0.367	0.497
	Sequential Pattern Mining	0.330	0.478
	Human Schema	0.173	0.205
	Event Graph Model	**0.401**	**0.520**
	w/o ArgumentGeneraion	0.392	0.509
IED	Event Language Model	0.169	0.513
	Sequential Pattern Mining	0.138	0.378
	Human Schema	0.072	0.222
	Event Graph Model	**0.224**	**0.741**
	w/o ArgumentGeneraion	0.210	0.734

Table 4: Schema-guided event prediction performance.

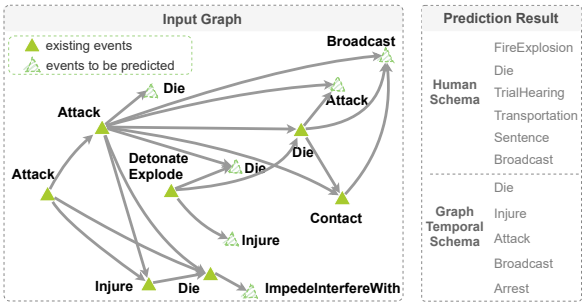

Figure 3: An event prediction example (IED scenario).

graph, and take into account that there is no ARREST event or justice-related events in the input graph. Also, the human schema fails to predict INJURE and ATTACK, because it relies on the exact match of event sequences of lengths $l \geq 2$, and cannot handle the variants of sequences. This problem can be solved by our probabilistic schema, via modeling the prediction probability conditioned on the existing graph. For example, even though ATTACK mostly happens before DIE, we learn that ATTACK might repeat after DIE event if there are multiple ATTACK and DETONATE in the existing graph, which means the complex event is about a series of conflict events.

Ablation Study. Removing argument generation ("w/o ArgumentGeneration") generally lowers the performance on all evaluation tasks, since it ignores the coreferential arguments and their relations, but relies solely on the overly simplistic temporal order to connect events. This is especially apparent from the instance graph perplexity in Table 3.

Learning Corpus Size. An average of 113 instance graphs is used for each complex event type in the IED scenario, and 383 instance graphs to learn the schema model in the General scenario. The better performance on the IED dataset in Table 3 shows that the number of instance graphs increases the schema induction performance.

Effect of Information Extraction Errors. Based on the error analysis for schemas induced in Table 1, the effect of extraction errors can be categorized into: (1) temporal ordering errors: 43.3%; (2) missing events: 34.4%; (3) missing coreferential events: 8.8%; (4) incorrect event type: 7.7%; (5) missing coreferential arguments: 5.5%. However, even on automatically extracted event graphs with extraction errors, our model significantly performs better on event prediction compared to human-constructed schemas, as shown in Table 4. It demonstrates that our schema induction method is robust and effective to support downstream tasks, even when only provided with noisy data with extraction errors.

6 Related Work

The definition of a *complex event schema* separates us from related lines of work, namely *schema induction* and *script learning*. Previous work on *schema induction* aims to characterize

event triggers and participants of individual atomic events (Chambers, 2013; Cheung et al., 2013; Nguyen et al., 2015; Sha et al., 2016; Yuan et al., 2018), ignoring inter-event relations. Work on *script learning*, on the other hand, originally limited attention to event chains with a single protagonist (Chambers and Jurafsky, 2008, 2009; Rudinger et al., 2015; Jans et al., 2012; Granroth-Wilding and Clark, 2016) and later extended to multiple participants (Pichotta and Mooney, 2014, 2016; Weber et al., 2018). Recent efforts rely on distributed representations encoded from the compositional nature of events (Modi, 2016; Granroth-Wilding and Clark, 2016; Weber et al., 2018, 2020; Zhang et al., 2020), and language modeling (Rudinger et al., 2015; Pichotta and Mooney, 2016; Peng and Roth, 2016). All of these methods still assume that events follow linear order in a single chain. They also overlook the relations between participants which are critical for understanding the complex event. However, we induce a comprehensive event graph schema, capturing both the temporal dependency and the multi-hop argument dependency across events.

Recent work on event graph schema induction (Li et al., 2020) only considers the connections between a pair of two events. Similarly, their event prediction task is designed to automatically generate a missing event (e.g., a word sequence) given a single or a sequence of prerequisite events (Nguyen et al., 2017; Hu et al., 2017; Li et al., 2018b; Kiyomaru et al., 2019; Lv et al., 2019), or predict a pre-condition event given the current events (Kwon et al., 2020). In contrast, we leverage the automatically discovered temporal event schema as guidance to forecast the future events.

Existing script annotations (Chambers and Jurafsky, 2008, 2010; Modi et al., 2016; Wanzare et al., 2016; Mostafazadeh et al., 2016a,b; Kwon et al., 2020) cannot support a comprehensive graph schema induction due to the missing of critical event graph structures, such as argument relations. Furthermore, in real-world applications, complex event schemas are expected to be induced from large-scale historical data, which is not feasible to annotate manually. We propose a data-driven schema induction approach, and choose to use IE systems instead of using manual annotation, to induce schemas that are robust and can tolerate extraction errors.

Our work is also related to recent advances in modeling and generation of graphs (Li et al., 2018a; Jin et al., 2018; Grover et al., 2019; Simonovsky and Komodakis, 2018; Liu et al., 2019; Fu et al., 2020; Dai et al., 2020; You et al., 2018; Liao et al., 2019; Yoo et al., 2020; Shi et al., 2020). We are the first to perform graph generation on event graphs.

7 Conclusions and Future Work

We propose a new task to induce *temporal complex event schemas*, which are capable of representing multiple temporal dependencies between events and their connected arguments. We induce such schemas by learning an *event graph model*, a deep auto-regressive model, from the automatically extracted instance graphs. Experiments demonstrate the model's effectiveness on both intrinsic evaluation and the downstream task of schema-guided event prediction. These schemas can guide our understanding and ability to make predictions with respect to what might happen next, along with background knowledge including location-, and participant-specific and temporally ordered event information. In the future, we plan to extend our framework to hierarchical event schema induction, as well as event and argument instance prediction.

Acknowledgement

This research is based upon work supported by U.S. DARPA KAIROS Program Nos. FA8750-19-2-1004 and Air Force No. FA8650-17-C-7715. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

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.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610, Suntec, Singapore. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2010. A database of narrative schemas. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 837–846, Atlanta, Georgia. Association for Computational Linguistics.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. 2020. Scalable deep generative modeling for sparse graphs. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2302–2312. PMLR.

Dongqi Fu, Dawei Zhou, and Jingrui He. 2020. Local motif clustering on time-evolving graphs. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 390–400. ACM.

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, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2727–2733. AAAI Press.

Aditya Grover, Aaron Zweig, and Stefano Ermon. 2019. Graphite: Iterative generative modeling of graphs. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2434–2444. PMLR.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Linmei Hu, Juanzi Li, Liqiang Nie, Xiaoli Li, and Chao Shao. 2017. What happens next? future subevent prediction using contextual hierarchical LSTM. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3450–3456. AAAI Press.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 336–344, Avignon, France. Association for Computational Linguistics.

Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. 2018. Junction tree variational autoencoder for molecular graph generation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2328–2337. PMLR.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Hirokazu Kiyomaru, Kazumasa Omura, Yugo Murawaki, Daisuke Kawahara, and Sadao Kurohashi. 2019. Diversity-aware event prediction based on a conditional variational autoencoder with reconstruction. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 113–122, Hong Kong, China. Association for Computational Linguistics.

Heeyoung Kwon, Mahnaz Koupaee, Pratyush Singh, Gargi Sawhney, Anmol Shukla, Keerthi Kumar Kallur, Nathanael Chambers, and Niranjan Balasubramanian. 2020. Modeling preconditions in text with a crowd-sourced dataset. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3818–3828, Online. Association for Computational Linguistics.

Tuan Lai, Heng Ji, Trung Bui, Quan Hung Tran, Franck Dernoncourt, and Walter Chang. 2021. A context-dependent gated module for incorporating symbolic semantics into event coreference resolution. In *Proc. The 2021 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT2021)*.

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.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 894–908.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. 2018a. Learning deep generative models of graphs. In *Proceedings of the 35 th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80*.

Zhongyang Li, Xiao Ding, and Ting Liu. 2018b. Constructing narrative event evolutionary graph for script event prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4201–4207. ijcai.org.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard S. Zemel. 2019. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4257–4267.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. 2019. Graph normalizing flows. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13556–13566.

Shangwen Lv, Wanhui Qian, Longtao Huang, Jizhong Han, and Songlin Hu. 2019. Sam-net: Integrating event-level and chain-level attentions to predict what happens next. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6802–6809. AAAI Press.

Piyush Mishra, Akanksha Malhotra, Susan Windisch Brown, Martha Palmer, and Ghazaleh Kazeminejad. 2021. Schema curation interface: Enhancing user experience in curation tasks. In *Proc. The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021) Demo Track*.

Ashutosh Modi. 2016. Event embeddings for semantic script modeling. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 75–83, Berlin, Germany. Association for Computational Linguistics.

Ashutosh Modi, Tatjana Anikina, Simon Ostermann, and Manfred Pinkal. 2016. InScript: Narrative texts annotated with script information. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3485–3493, Portorož, Slovenia. European Language Resources Association (ELRA).

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016a. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016b. CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the Fourth Workshop on Events*, pages 51–61, San Diego, California. Association for Computational Linguistics.

Dai Quoc Nguyen, Dat Quoc Nguyen, Cuong Xuan Chu, Stefan Thater, and Manfred Pinkal. 2017. Sequence to sequence learning for event prediction. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 37–42, Taipei, Taiwan. Asian Federation of Natural Language Processing.

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, Beijing, China. Association for Computational Linguistics.

Qiang Ning, Sanjay Subramanian, and Dan Roth. 2019. An improved neural baseline for temporal relation extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6203–6209, Hong Kong, China. Association for Computational Linguistics.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proc. the 2015 Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL-HLT 2015)*.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. 2001. Prefixspan: Mining sequential patterns by prefix-projected growth. In *Proceedings of the 17th International Conference on Data Engineering*, pages 215–224.

Haoruo Peng and Dan Roth. 2016. Two discourse driven language models for semantics. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 290–300, Berlin, Germany. Association for Computational Linguistics.

Karl Pichotta and Raymond Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–229, Gothenburg, Sweden. Association for Computational Linguistics.

Karl Pichotta and Raymond J. Mooney. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2800–2806. AAAI Press.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction 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.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Joint learning templates and slots for event schema induction. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 428–434, San Diego, California. Association for Computational Linguistics.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. 2020. Graphaf: a flow-based autoregressive model for molecular graph generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Martin Simonovsky and Nikos Komodakis. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer.

Lilian DA Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2016. A crowdsourced database of event sequence descriptions for the acquisition of high-quality script knowledge.

Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. 2018. Event representations with tensor-based compositions. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4946–4953. AAAI Press.

Noah Weber, Rachel Rudinger, and Benjamin Van Durme. 2020. Causal inference of script knowledge. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7583–7596, Online. Association for Computational Linguistics.

Haoyang Wen, Ying Lin, Tuan Lai, Xiaoman Pan, Sha Li, Xudong Lin, Ben Zhou, Manling Li, Haoyu Wang, Hongming Zhang, et al. 2021a. 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.

Haoyang Wen, Yanru Qu, Heng Ji, Qiang Ning, Jiawei Han, Avi Sil, Hanghang Tong, and Dan Roth. 2021b. Event time extraction and propagation via graph attention networks. In *Proc. The 2021 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT2021)*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Sanghyun Yoo, Young-Seok Kim, Kang Hyun Lee, Kuhwan Jeong, Junhwi Choi, Hoshik Lee, and Young Sang Choi. 2020. Graph-aware transformer: Is attention all graphs need? *arXiv preprint arXiv:2006.05213*.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Graphrnn: Generat-

ing realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5694–5703. PMLR.

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 2018, Torino, Italy, October 22-26, 2018*, pages 587–596. ACM.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020. Reasoning about goals, steps, and temporal ordering with WikiHow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4630–4639, Online. Association for Computational Linguistics.