Open-Domain Hierarchical Event Schema Induction by Incremental Prompting and Verification

Sha Li[1], **Ruining Zhao**[1], **Manling Li**[1], **Heng Ji**[1], **Chris Callison-Burch**[2], **Jiawei Han**[1]
[1]University of Illinois at Urbana-Champaign
[2] University of Pennsylvania
{shal2, ruining9, manling2, hengji, hanj}@illinois.edu
ccb@seas.upenn.edu

Abstract

Event schemas are a form of world knowledge about the typical progression of events. Recent methods for event schema induction use information extraction systems to construct a large number of event graph instances from documents, and then learn to generalize the schema from such instances. In contrast, we propose to treat event schemas as a form of commonsense knowledge that can be derived from large language models (LLMs). This new paradigm greatly simplifies the schema induction process and allows us to handle both hierarchical relations and temporal relations between events in a straightforward way. Since event schemas have complex graph structures, we design an *incremental prompting and verification* method INCSCHEMA to break down the construction of a complex event graph into three stages: event skeleton construction, event expansion, and event-event relation verification. Compared to directly using LLMs to generate a linearized graph, INCSCHEMA can generate large and complex schemas with 7.2% F1 improvement in temporal relations and 31.0% F1 improvement in hierarchical relations. In addition, compared to the previous state-of-the-art closed-domain schema induction model, human assessors were able to cover ∼10% more events when translating the schemas into coherent stories and rated our schemas 1.3 points higher (on a 5-point scale) in terms of readability.

1 Introduction

Schemas, defined by (Schank and Abelson, 1975) as "a predetermined, stereotyped sequence of actions that defines a well-known situation", are a manifestation of world knowledge. With the help of schemas, a model can then infer missing events such as a person must have "been within contact with a pathogen" before the event "the person was sent to the hospital for treatment" and also predict that if a large-scale incident happened, this might trigger an "investigation of the source of the pathogen".

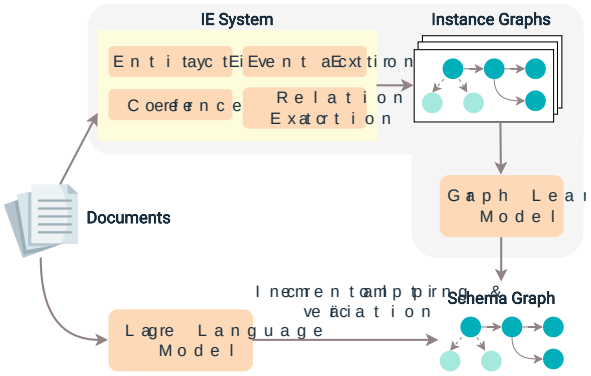


Figure 1: A comparison between the instance-based schema induction pipeline (gray background) and our INCSCHEMA approach. By directly prompting LLMs to construct the schema graph, our framework is conceptually simpler, open-domain, extensible, and more interpretable.

To automate schema creation, two mainstream approaches are to learn from manually created reference schemas or learn from large amounts of event instances automatically extracted from documents. Manual creation of complex hierarchical schemas requires expert annotation, which is not scalable[1]. On the other hand, instance-based schema induction methods (Li et al., 2020, 2021; Jin et al., 2022; Dror et al., 2022) rely on complicated preprocessing[2] to transform documents into instance graphs for learning. Moreover, supervised information extraction systems (Ji and Grishman, 2008; Lin et al., 2021c) are domain-specific and suffer from error propagation through multiple components, which makes the downstream schema induction model closed-domain and low in quality.

[1]Online crowdsourced resources such as WikiHow only contain simple linear schemas.

[2]To create an event instance graph, typical steps include entity extraction, entity-entity relation extraction, event extraction, coreference resolution, and event-event relation extraction.

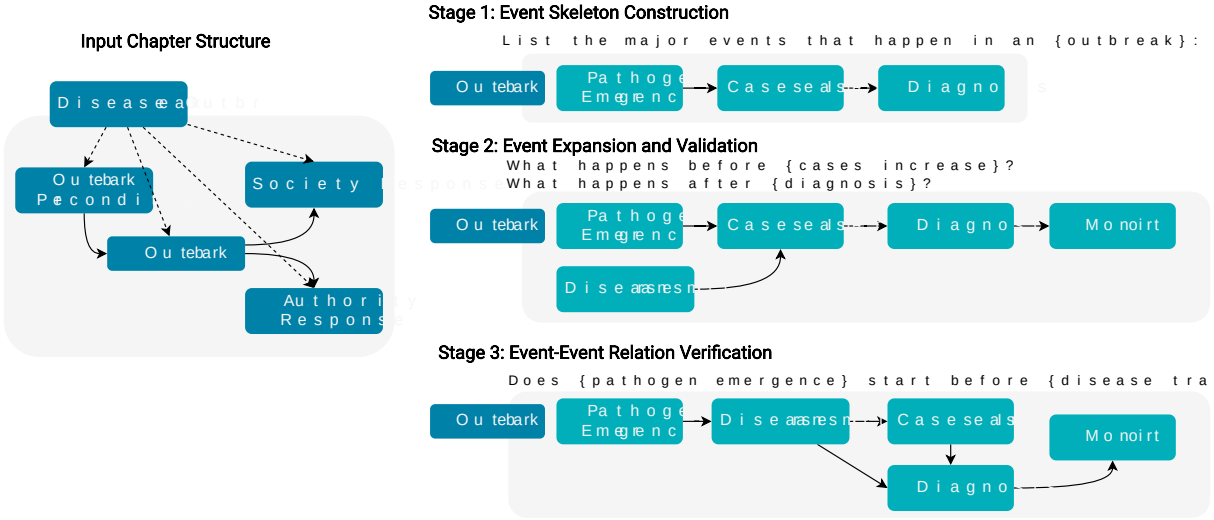


Figure 2: To create the schema for a given scenario, our model follows 3 rounds of operation: (1) *event skeleton construction* where we ask the LLM to list the important events; (2) *event expansion* to discover more related events for each existing event; *event-event relation verification* where we update the event-event relations based on the LLM's answers to questions about each event pair.

Tracing back to the original definition of schemas, we observe that "stereotyped sequences of events" or "the typical progression of events" can be viewed as *commonsense knowledge* that can be implicitly learned by training on large corpora. Through the language modeling objective, models can pick up which events statistically frequently co-occur and how their relationship is typically described. More recently, large language models (LLMs) such as GPT3 (Brown et al., 2020) and PaLM (Chowdhery et al., 2022) have shown impressive zero-shot performance on closely-related commonsense reasoning tasks such as goal-step reasoning (Zhang et al., 2020) and temporal reasoning[3].

By utilizing LLMs to directly prompt for schematic knowledge, our approach is *open-domain*, *extensible* and *more interpretable* for humans. Given a new scenario name, our model only requires lightweight human guidance in providing some top-level chapter structure (as shown in the left of Figure 2) and can produce the entire schema in under an hour whereas instance-based methods require months to collect the data and retrain the IE system for new domains. Our model is *extensible* and can support new types of event-event relations by adding new prompt templates. To showcase this, in addition to the temporal relation between events which is the focus of prior work, we also account for the different event granularities by supporting hierarchical relations between events (for example, a `physical conflict` could happen as a part of a `protest`). Finally, by representing events with free-form text instead of types and organizing them into a hierarchy, our generated schemas are considered more interpretable.

We find that directly asking LLMs to generate linearized strings of schemas leads to sub-optimal results due to the size and complexity of the graph structure. To solve this problem, we design an *incremental prompting and verification* scheme to break down the construction of a complex event graph schema into three major stages: event skeleton construction, event expansion, and event-event relation verification. As shown in Figure 2, each stage utilizes templated prompts (`What happens before cases increase?`) which can be instantiated either with the scenario name or the name of a previously generated event.

The key contributions of this paper are:

- We propose a framework INCSCHEMA for inducing complex event schemas by treating the task as knowledge probing from LLMs. Compared to previous approaches that rely on the creation of event instance graphs, our method greatly simplifies the process and as a result, is not confined to the working domain of any IE system.

[3] https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/temporal_sequences

- We extend the expressive power of event schemas by inducing hierarchical relations and temporal relations between events at the same time. Our modularized prompting framework allows us to support a new type of event-event relation easily, whereas prior work (Zhou et al., 2022b; Dror et al., 2022) required specialized pipelines or components.

- We verify the effectiveness of our framework on two complex schema datasets: ODIN, an Open-Domain Newswire schema library, and RESIN-11 (Du et al., 2022). Compared to directly generating the schema using a linearized graph description language (Sakaguchi et al., 2021), INCSCHEMA shows 7.2% improvement in temporal relation F1 and 31.0% improvement in hierarchical relation F1.

2 Task Overview

Given a scenario name, a schema depicts the *general progression* of events within that scenario.

Following (Li et al., 2021), we consider the schema to be a graph structure of events. We output a schema graph of event nodes and event-event relation edges, including temporal relations and hierarchical relations.

Since our algorithm is designed to be open-domain, we represent each event e with a description string such as "`A person shows early symptoms of the disease`" instead of a type from a restricted ontology (e.g., `Illness`). Description strings are more flexible in representing different granularities of events and are more informative. It is noteworthy that event descriptions in a schema should be general, instead of a specific instance, such as "`John had a mild fever due to COVID`".

In addition, we support the representation of chapters, which are "a collection of events that share the same theme and are connected in space-time". When a high-level chapter structure G_c (as shown in the left side of Figure 2) is available, we condition on the given chapters to guide the schema generation process. Chapters are also treated as events and can potentially have temporal relations between them. Every other event must be a descendant of a chapter event. If no chapter structure is available, we create a single chapter from the scenario name.

3 Our Approach

Leveraging LLMs to directly generate the full schema graph is challenging due to the size and complexity of schemas. Thus, we divide our schema induction algorithm INCSCHEMA into three stages as depicted in Figure 2. Starting from the scenario node or one of the chapter nodes, the skeleton construction stage first produces a list of major events that are subevents of the scenario (chapter) following sequential order. For each generated event, we expand the schema graph to include its temporally-related neighbors and potential children in the event expansion stage. For each pair of events, we further rescore their temporal and hierarchical relation probability in the relation verification stage to enrich the relations between events.

3.1 Retrieval-Augmented Prompting

To make the model more informed of how events are typically depicted in news, we introduce a retrieval component to guide LLMs to focus on scenario-related passages. The key difficulty of schema induction is to generalize from multiple passages and reflect the "stereotyped sequence of events" instead of providing concrete and specific answers. We, therefore, retrieve multiple passages each time and ask the model to provide a generalized answer that is suitable for all passages.

To build a document collection containing typical events of the given scenario, we leverage its Wikipedia category page and retrieve the reference news articles of each Wikipedia article under the category, as detailed in Appendix A. With such a document collection, for each prompt, we are able to use the description of the event as the query and retrieve $k = 3$ passages based on state-of-the-art document retrieval system TCT-ColBERT (Lin et al., 2021b). The input to the LM is structured as follows:

Retrieval-Augmented Prompt

Based on the following passages `{retrieved passages}`,
`{prompt}`

Providing more than one passage is critical as we want the model to produce a *generalized* response instead of a specific response that only pertains to one event instance.

3.2 Event Skeleton Construction

We use the following prompt to query the LM about events that belong to the chapter c:

Event Skeleton Prompt

{evt.name} is defined as "{evt.description}". List the major events that happen in the {evt.name} of a {scenario}:

This typically gives us a list of sentences, which is further translated into a linear chain of event nodes by treating each sentence as an event description and regarding the events as listed in temporal order. To assign a name to each event for easier human understanding, we leverage the LLM again with in-context learning using 10 {description, name} pairs such as {`Disinfect the area to prevent infection of the disease, Sanitize`} (the complete list of in-context examples is in Appendix D).

3.3 Event Expansion and Validation

Given an event e (such as *Cases Increase* in Figure 2), we expand the schema by probing for its connected events in terms of temporal and hierarchical relations using prompts as below:

Event Expansion Prompt

What happened during "{evt.description}"? List the answers:

(See Appendix D for a full list of prompts used.) Every sentence in the generated response will be treated as a candidate event.

For every candidate event e' (such as *Disease-Transmit* in Figure 2), we perform a few validation tests as listed below. The event is only added to the schema when all the tests pass.

Duplication Test To check if a new event is a duplicate of an existing event, we use both embedding similarity computed through cosine similarity of SBERT embeddings (Reimers and Gurevych, 2019)[4] and string similarity using Jaro-Winkler similarity (Winkler, 1990). If the event description, event name, or the embedding of the event description is sufficiently similar to an existing event in the schema, we will discard the new event. [5]

Specificity Test When we augment the prompt with retrieved documents, at times the model will answer the prompt with details that are too specific to a certain news article, for instance, include the time and location of the event. The specificity test seeks to remove such events. We implement this by asking the LLM "Does the text contain any specific names, numbers, locations, or dates?" and requesting a yes-no answer. We use 10 in-context examples to help the LLM adhere to the correct answer format and understand the instructions.

Chapter Test For the chapter assignment test, we present the name and the definition of the chapter event c and the target event e' respectively, then ask "Is e' a part of c? ". If the answer is "yes", we keep the event e'.

If a new event e' passes validation, we assign a name to the event following the same procedure as in Section 3.2.

3.4 Event-Event Relation Verification

Although the prompts from the previous step naturally provide us with some relations between events (the answer to "What are the steps in e?" should be subevents of e), such relations may be incomplete or noisy. To remedy this problem, for every pair of events (e_1, e_2) in the same chapter, we verify their potential temporal/hierarchical relation.

A straightforward way to perform verification would be to ask questions such as "Is e_1 a part of e_2?" and "Does e_1 happen before e_2?". Our pilot experiments show that this form of verification leads to sub-optimal results in two aspects: (1) *relation confusion*: the language model will predict both $e_2 \prec e_1$ and $e_1 \subset e_2$; and (2) *order sensitivity*: the language model tends to return "yes" for both "Does e_1 happen before e_2?" and "Does e_1 happen after e_2?".

To solve this *relation confusion* problem, inspired by Allen interval algebra (Allen, 1983) and the neural-symbolic system in (Zhou et al., 2021), we decompose the decision of a temporal relation into questions about start time, end time, and duration. In addition, following HiEve (Glavaš et al., 2014), we define the hierarchical relation as spatial-temporal containment. Thus a necessary condition for a hierarchical relation to hold between e_1 and e_2 is that the time period of e_1 contains e_2. This allows us to make decisions about temporal relations

[4] We use the `all-MiniLM-L6-v2` model.

[5] This threshold is determined empirically, and we set it to 0.9 for Jaro-Winkler string similarity, 0.85 for embedding cosine similarity.

Relation	Allen's base relations	e_1 starts before e_2?	e_1 ends before e_2?	Is the duration of e_1 longer than e_2?
$e_1 \prec e_2$	e_1 precedes e_2, e_1 meets e_2	Yes	Yes	-
$e_1 \succ e_2$	e_1 is preceded by e_2, e_1 is met by e_2	No	No	-
$e_1 \subset e_2$	e_1 starts e_2, e_1 during e_2, e_1 finishes e_2	No	Yes	No
$e_1 \supset e_2$	e_1 is started by e_2, e_1 contains e_2, e_1 is finished by e_2	Yes	No	Yes
$e_1 \parallel e_2$	e_1 overlaps with e_2, e_1 is equal to e_2	Yes	No	No
$e_1 \parallel e_2$	e_1 is overlapped by e_2	No	Yes	Yes

Table 1: Schema event-event relations, the correspondence to Allen's interval algebra, and the related relation questions. In the case of temporal overlap (last two rows), we refrain from adding an edge between the two events.

and hierarchical relations jointly using the three questions as shown in Table 1.

Relation Verification Prompt

Does "{e1.description}" start before "{e2.description}"?
Answer yes, no, or unknown.

For each question, to obtain the probability of the answers, we take the log probability of the top 5 tokens[6] in the vocabulary and check for the probability predicted for "yes", "no" and "unknown" tokens.

To handle the order sensitivity, we average the scores obtained from the different orderings ("Does e_1 start before e_2?" and "Does e_2 start before e_1") and different prompts ("Does e_1 start before e_2?" and "Does e_2 start after e_1?").

After obtaining the response for start time, end time, and duration questions, we only keep edges that have scores higher than a certain threshold for all of the three questions.

Since our temporal edges were only scored based on the descriptions of the event pair, we need to remove loops consisting of more than 2 events, ideally with minimal changes, to maintain global consistency. This problem is equivalent to the problem of finding the *minimal feedback arc set*, which is shown to be NP-hard. We adopt the greedy algorithm proposed in (Eades et al., 1993) using the previously predicted probabilities as edge weights to obtain a node ordering. Based on this ordering we can keep all edges directionally consistent. The detailed algorithm is provided in Appendix B. Finally, to simplify the schema, we perform transitive reduction on the relation and hierarchy edges respectively.

[6]At the time of writing, OpenAI API only supports returning the log probability of a maximum of 5 tokens.

4 Experiments

We design our experiments based on the following three research questions:

Q1: Hierarchical Schema Quality Can our model produce high-quality event graph schemas with both temporal and hierarchical relations?

Q2: Interpretability Is our model's output more interpretable than prior instance-based schema induction methods?

Q3: Model Generalization Can our model also be applied to everyday scenarios as in (Sakaguchi et al., 2021)?

4.1 Dataset

RESIN-11 (Du et al., 2022) is a schema library targeted at 11 newsworthy scenarios and includes both temporal and hierarchical relations between events. However, RESIN-11 is still quite heavily focused on attack and disaster-related scenarios, so we expand the coverage and create a new Open-Domain Newswire schema library ODIN which consists of 18 new scenarios, including coup, investment, and health care. The complete list of scenarios is in Appendix C.

Upon selecting the scenarios, we collected related documents from Wikipedia (following the procedure described in Section 3.1) and create the ground truth reference schemas by asking human annotators to curate the schemas generated by our algorithm by referring to the news reports of event instances. Human annotators used a schema visualization tool [7] to help visualize the graph structure while performing curation. Curators were encouraged to (1) add or remove events; (2) change the event names and descriptions; (3) change the temporal ordering between events; and (4) change the hierarchical relation between events. After the curation, the schemas were examined by linguistic

[7]https://schemacuration.colorado.edu/

experts. We present the statistics of ODIN along with RESIN-11 and ProScript in Table 2.

4.2 Evaluation Metrics

For automatic evaluation of the schema quality against human-created schemas, we adopt **Event F1** and **Relation F1** metrics. Event F1 is similar to the Event Match metric proposed in (Li et al., 2021) but since here we are generating event descriptions instead of performing classification over a fixed set of event types, we first compute the similarity score s between each generated event description and ground truth event description using cosine similarity of SBERT embeddings (Reimers and Gurevych, 2019). Then we find the maximum weight matching assignment ϕ between the predicted events $\hat{E}$ and the ground truth events E by treating it as an assignment problem between two bipartite graphs[8].

Based on the event mapping ϕ, we further define **Relation F1** metrics for temporal relations and hierarchical relations respectively. Note that this metric only applies to events that have a mapping.

4.3 Implementation Details

For both our model and the baseline, we use the GPT3 model `text-davinci-003` through the OpenAI API. We set the temperature to 0.7 and `top_p` to 0.95.

For INCSCHEMA we set the minimum number of events within a chapter to be 3 and the maximal number of events to be 10. During the event skeleton construction stage, if the response contains less than 3 sentences, we will re-sample the response. Once the number of events within a chapter reaches the maximal limit, we will not add any more new events through the event expansion stage.

We set the threshold for the duplication test to be 0.9 for Jaro-Winkler string similar OR 0.85 for cosine similarity between SBERT embeddings. For the shorter event name, we also check if the Levenshtein edit distance is less than 3. For the event-event relation verification, we set the threshold for the start time and end time questions to be 0.2 and the threshold for the duration question to be 0.7.

[8] https://en.wikipedia.org/wiki/Assignment_problem, we use the implementation from scipy(https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linear_sum_assignment.html)

Dataset	# Scenarios	# Event	# Temp.	# Hier.
RESIN-11	11	579	381	603
ODIN	18	593	398	569
ProScript	2077	14997	13946	0

Table 2: Dataset statistics. RESIN-11 and ODIN both focus on hierarchical schemas for newsworthy scenarios. ProScript is a collection of small-sized schemas for everyday scenarios.

4.4 Q1: Hierarchical Schema Quality

We test our algorithm's ability to induce complex hierarchical schemas for news scenarios in RESIN-11 (Du et al., 2022) and our new Open-Domain Newswire schema library ODIN.

We compare our model against a different prompt formulation method using the DOT graph description language as purposed by (Sakaguchi et al., 2021) (**GPT-DOT**). This method requires the LLM to generate all events and event-event relations in a single pass. To inform the model of the DOT language format, we use one in-context example converted from the Chemical Spill ground truth schema (the prompt is shown in Appendix D). During inference, we will input the scenario name and the chapter structure.

We show our results on the RESIN-11 dataset in Table 3 and the results for ODIN in Table 4 [9].

Compared to our incremental prompting procedure, GPT-DOT generally outputs fewer events (10.11 events for GPT-DOT VS 52.6 events for INCSCHEMA on ODIN), which leads to high precision but low recall. While the generated events from GPT-DOT are still reasonable, the real deficiency of this formulation is its inability to identify hierarchical relations, especially when hierarchical relations co-exist with temporal relations.

To test if using an in-context learning prompt is the reason for low performance, we also experiment with an instruction-style prompt (GPT-DOT-Instruct) that explains the task and output format in detail and a step-by-step reasoning prompt (GPT-DOT-StepByStep) that allows the model to output parts of the schema separately (and we will merge them together). For the ODIN dataset, we find that the different prompt styles do not vary much except for improved temporal relation F1 when we use the step-by-step formulation.

Compared with the variants of our model, we

[9] GPT results in Table are shown by averaging score of 5 runs.

Model	Event			Temp. Relation			Hier. Relation		
	Prec	Recall	F1	Prec	Recall	F1	Prec	Recall	F1
GPT-DOT	80.0	30.7	**41.8**	24.7	8.31	11.2	11.1	13.7	12.0
INCSCHEMA	39.7	49.3	41.7	13.8	14.8	**13.5**	39.3	38.7	**38.9**

Table 3: Schema induction evaluation on RESIN-11 scenarios. Results are shown in %.

Model	Event			Temp. Relation			Hier. Relation		
	Prec	Recall	F1	Prec	Recall	F1	Prec	Recall	F1
GPT-DOT	85.2	35.2	47.4	34.0	18.0	20.9	15.4	19.3	17.5
GPT-DOT-Instruct	85.4	34.4	46.7	33.8	17.8	20.8	16.7	18.4	16.3
GPT-DOT-StepByStep	95.3	27.1	41.2	49.5	21.8	25.9	13.0	18.1	14.9
INCSCHEMA	45.1	72.1	**53.3**	27.5	29.6	28.1	49.3	48.0	48.5
- No retrieval	41.6	73.2	50.7	28.5	29.3	**28.3**	48.2	45.6	46.7
- No decompose	44.8	71.1	52.4	25.6	22.7	23.5	49.6	50.2	**49.7**

Table 4: Schema induction evaluation on ODIN scenarios. Results are shown in %.

can see that the retrieval component helps improve event generation quality and the question decomposition strategy can greatly improve temporal relation F1.

Since RESIN-11 schemas were created without referencing any automatic results, the scores on RESIN-11 are generally lower than that of ODIN. However, on both datasets, our method can generally outperform GPT-DOT.

4.5 Q2: Schema Interpretability

To be able to compare our schemas side-by-side with previous work that assumed a limited ontology, we conduct a human evaluation that focuses on the interpretability of the induced schemas.

Human assessors are presented with the scenario name and a subgraph from the schema induction algorithm's output. We then ask the assessor to write a coherent short story by looking at the graph and indicate which events were included in their story. An example of the subschema and the story is shown in Figure 3. After they complete the story writing task, they will be asked to rate their experience from several aspects on a 5-point Likert scale. The human assessment interface is shown in Appendix F.

We compare against the state-of-the-art closed-domain schema induction method **DoubleGAE** (Jin et al., 2022). DoubleGAE is an example of the instance-based methods that rely on IE: the nodes in the schema graph are typed instead of described with text.

In Table 5 we show the results for the story writing task. We observe that human assessors are able to compose a longer story with higher event coverage when presented with our schemas while taking roughly the same time.

Model	Coverage↑	Len(words)↑	Time(mins)↓
Double-GAE	79.8	9.62	0.998
INCSCHEMA	89.7	15.53	1.137

Table 5: Human performance on the storytelling task using schemas from DoubleGAE and our algorithm INCSCHEMA. The length and time measurements are averaged over the number of events in the schemas. Coverage is shown in percentage.

In the post-task questionnaire, as shown in Figure 4, the human assessors on average strongly agreed that the event names and event descriptions produced by our model were helpful and thought that our schemas were easier to understand compared to the baseline (4.50 vs 3.20 points). Both schemas contained events that were highly relevant to the scenario and the temporal ordering in the schemas was mostly correct.

4.6 Q3: Model Generalization

For this experiment, we use Proscript (Sakaguchi et al., 2021) as our dataset. Schemas in Proscript are typically short (5.45 events on average) and describe everyday scenarios. Proscript schemas only contain temporal relations and have no chapter structure, so we include the first two events (by topological sorting) as part of the prompt. We show the results in Table 6.

For our algorithm INCSCHEMA we omitted the event expansion stage since the event skeleton construction stage already generated enough events. In the event-event relation verification stage, we continue to add temporal relations among events

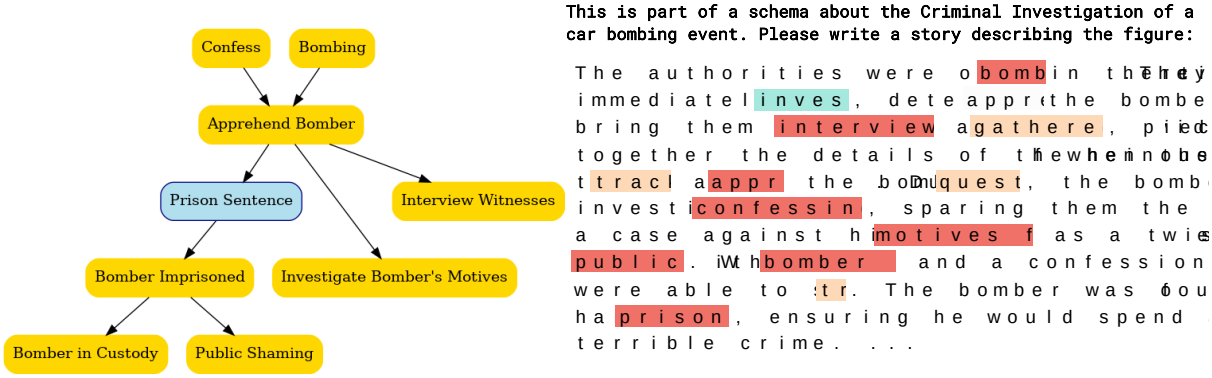


Figure 3: One example response from the schema interpretability human assessment. On the left we show the subevents of the `Criminal Investigation` chapter produced by our model. On the right is the human-written story describing the schema. We highlight the events that match the chapter in blue, events that appear in the schema in red and additional events in pink.

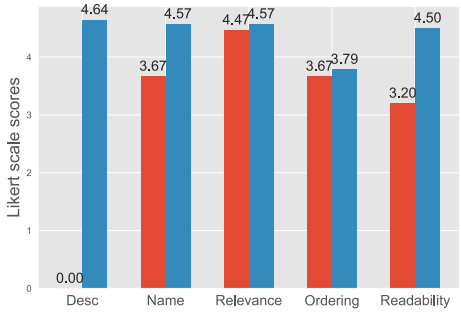


Figure 4: Human assessment of schema quality and interpretability from different aspects. Results for Double-GAE are shown in red and our approach INCSCHEMA in blue. Double-GAE does not produce event descriptions thus we omit the description helpfulness question.

Model	Event			Temp. Relation		
	Prec	Recall	F1	Prec	Recall	F1
GPT3-DOT	61.8	59.5	59.3	25.9	23.3	**23.7**
INCSCHEMA	58.4	69.6	**61.1**	22.4	25.8	22.7

Table 6: Evaluation on Proscript's everyday scenarios. These schemas are small in size and do not contain hierarchical relations. We compare against directly generating the linearized graph as DOT language.

based on their verification score beyond the threshold until the graph is connected.

On these small-scaled schemas with only temporal relations, we see that directly generating the full schema and incrementally prompting the schema lead to comparable results. This shows that GPT3 can indeed understand and generate valid graph description DOT language and the gap that we observe in Table 4 is mainly due to the difficulty of capturing long-range dependencies in large schemas and the confusion between temporal and hierarchical relations.

5 Related Work

Event Schema Induction Event schema induction, or script induction, is the task of inducing typical event-event relation structures for given scenarios/situations[10]. A large fraction of work considers event schemas as narrative chains (Chambers and Jurafsky, 2008, 2009; Jans et al., 2012; Pichotta and Mooney, 2014, 2016; Rudinger et al., 2015a; Ahrendt and Demberg, 2016; Granroth-Wilding and Clark, 2016; Wang et al., 2017; Weber et al., 2018), limiting the structure to include only sequential temporal relations. More recently, non-sequential, partially ordered temporal relations have been taken into consideration (Li et al., 2018, 2020, 2021; Sakaguchi et al., 2021; Jin et al., 2022) but they do not consider the different scales of events and the potential hierarchical relations. In terms of schema expressiveness, (Dror et al., 2022) is the most similar to ours as they also consider both partial temporal order and hierarchical relations.

Our work also resembles a line of recent research on inducing schema knowledge from pre-trained language models. Our schema induction process can be seen as a super-set of the post-processing in (Sancheti and Rudinger, 2022), which comprises irrelevant event removal, de-duplication, and temporal relation correction. We compare our incre-

[10]There exists some work that refer to the task of "inducing roles of events" as schema induction, but their scope is distinct from ours.

mental prompting approach with the end-to-end approach proposed in (Sakaguchi et al., 2021) in Section 4. The work of (Dror et al., 2022) is orthogonal to ours as they use LLMs for data generation instead of probing for schema knowledge.

Language Model Prompting Prompting has been the major method of interaction with billion-scale language models (Brown et al., 2020; Rae et al., 2021; Wei et al., 2022; Chowdhery et al., 2022). Prompting can either be used to inform the model of the task instructions (Wei et al., 2022), provide the model with task input/output examples (Brown et al., 2020), or guide the model with explanations (Lampinen et al., 2022) and reasoning paths (Wang et al., 2022). In this work, we explore how a complex knowledge structure such as an event graph schema can be induced using LLMs by decomposing the task through incremental prompting.

6 Conclusions and Future Work

Prior work on schema induction has either relied on existing information extraction pipelines to convert unstructured documents into event graphs, or require massive human effort in annotating event schemas. We propose to view schema induction as a type of *event-oriented commonsense* that can be implicitly learned with large language models. However, since schemas are complex graph structures, instead of directly querying for schemas, we design an incremental prompting and verification framework INCSCHEMA to decompose the schema induction task into a series of simple questions. As a result, our model is applicable to the open-domain and can jointly induce temporal and hierarchical relations between events.

For future work, we plan to cover more aspects of schemas, including accounting for entity coreference, entity relations and entity attributes. While this work is focused on the task of schema induction, we hope to show the possibility of using LLMs for constructing complex knowledge structures.

7 Limitations

The event schemas generated by our model are not directly comparable to those generated by previous work that utilized a close-domain ontology. As a result, we were unable to adopt the same metrics and evaluate our schemas on type-level event prediction tasks as in (Li et al., 2021; Jin et al., 2022; Dror et al., 2022). Grounding the events generated by the LLM into one of the types in the ontology could be added as a post-processing step to our model, but this would require some ontology-specific training data, which goes against our principles of designing an *open-domain, portable* framework.

Our event schema does not explicitly represent entity coreference, entity relations, and entity attributes. The current schemas that we produce focus on events and their relations, with entity information captured implicitly through the event descriptions. For instance, the `See Medical Professional` event is described as "The patient is seen by a doctor or other medical professional" and the proceeding `Obtain Medical History` event is described as "The medical professional obtains a medical history from the patient". The "medical professional" and "patient" are implied to be coreferential entities in this case, but not explicitly connected in the schema graph.

Our approach is also quite distinct from prior work (Rudinger et al., 2015b; Wang et al., 2017; Li et al., 2021; Jin et al., 2022) that consider a probabilistic model as an implicit schema where the schema graph, or event narrative chain can be sampled from. Probabilistic schema models have the advantage of being adaptive and can be conditioned on partially observed event sequences, but are hard to interpret. We make the conscious design decision to generate explicit, human-readable schema graphs instead of black-box schema models.

Finally, our model relies on the usage of LMs, which have been observed to sometimes show inconsistent behavior between different runs or when using different prompts with the same meaning (Elazar et al., 2021; Zhou et al., 2022a). However, quantification of consistency has only been done for factual probing tasks while schema generation is a more open-ended task. For example, in our experiments on everyday scenarios, we observe that the model could generate distinct schemas for `Buying a (computer) mouse` based on whether the purchase was done online or in person. This variance is often benign and we leave it to future work to take advantage of such variance and possibly aggregate results over multiple runs.

Acknowledgement

We thank the anonymous reviewers for their helpful suggestions. This research is based upon work

supported by U.S. DARPA KAIROS Program No. FA8750-19-2-1004, the DARPA LwLL Program (contract FA8750-19-2-0201), the IARPA HIATUS Program (contract 2022-22072200005), and the NSF (Award 1928631). Approved for Public Release, Distribution Unlimited. 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

Simon Ahrendt and Vera Demberg. 2016. Improving event prediction by representing script participants. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 546–551, San Diego, California. Association for Computational Linguistics.

James F. Allen. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26:832–843.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Neurips*.

Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. 2016. Ms marco: A human generated machine reading comprehension dataset. *ArXiv*, abs/1611.09268.

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.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek B Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Oliveira Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311.

Rotem Dror, Haoyu Wang, and Dan Roth. 2022. Zero-shot on-the-fly event schema induction.

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.

Peter Eades, Xuemin Lin, and William F Smyth. 1993. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, 47(6):319–323.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Goran Glavaš, Jan Šnajder, Marie-Francine Moens, and Parisa Kordjamshidi. 2014. HiEve: A corpus for extracting event hierarchies from news stories. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3678–3683, Reykjavik, Iceland. European Language Resources Association (ELRA).

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *AAAI*.

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.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through unsupervised cross-document inference. In *In Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL 2008). Ohio, USA*.

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.

O. Khattab and Matei A. Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Andrew Kyle Lampinen, Ishita Dasgupta, Stephanie C. Y. Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L. McClelland, Jane X. Wang, and Felix Hill. 2022. Can language models learn from explanations in context? *ArXiv*, abs/2204.02329.

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. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5203–5215, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Zhongyang Li, Xiao Ding, and Ting Liu. 2018. Constructing narrative event evolutionary graph for script event prediction. In *IJCAI*.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021a. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 2356–2362.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021b. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 163–173, Online. Association for Computational Linguistics.

Ying Lin, Han Wang, Heng Ji, Premkumar Natarajan, and Yang Liu. 2021c. Personalized entity resolution with dynamic heterogeneous knowledge graph representations. In *Proc. ACL-IJCNLP2021 Workshop on e-Commerce and NLP*.

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 Mooney. 2016. Statistical script learning with recurrent neural networks. In *Proceedings of the Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*, pages 11–16, Austin, TX. Association for Computational Linguistics.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John F. J. Mellor, Irina Higgins, Antonia Creswell, Nathan McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, L. Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, N. K. Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Tobias Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew G. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem W. Ayoub, Jeff Stanway, L. L. Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling language models: Methods, analysis & insights from training gopher. *ArXiv*, abs/2112.11446.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

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

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015b. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1681–1686.

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. proScript: Partially ordered scripts generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2138–2149, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Abhilasha Sancheti and Rachel Rudinger. 2022. What do large language models learn about scripts? In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*, pages 1–11, Seattle, Washington. Association for Computational Linguistics.

Roger C Schank and Robert P Abelson. 1975. Scripts, plans, and knowledge. In *IJCAI*, volume 75, pages 151–157.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Rationale-augmented ensembles in language models. *ArXiv*, abs/2207.00747.

Zhongqing Wang, Yue Zhang, and Ching-Yun Chang. 2017. Integrating order information and event relation for script event prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 57–67, Copenhagen, Denmark. Association for Computational Linguistics.

Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. 2018. Event representations with tensor-based compositions. In *AAAI*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. *ICLR*.

William E Winkler. 1990. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage.

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.

Ben Zhou, Kyle Richardson, Qiang Ning, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2021. Temporal reasoning on implicit events from distant supervision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1361–1371, Online. Association for Computational Linguistics.

Chunting Zhou, Junxian He, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022a. Prompt consistency for zero-shot task generalization. *ArXiv*, abs/2205.00049.

Shuyan Zhou, Li Zhang, Yue Yang, Qing Lyu, Pengcheng Yin, Chris Callison-Burch, and Graham Neubig. 2022b. Show me more details: Discovering hierarchies of procedures from semi-structured web data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2998–3012, Dublin, Ireland. Association for Computational Linguistics.

A Retrieval Component

To build our document collection, we first search for the scenario name on Wikipedia, find its corresponding category page[11] and then for each Wikipedia article listed under the category, we follow the external reference links to news sources under the Wikipedia pages to retrieve the original news articles. We only keep English articles and filter out articles that have fewer than 4 sentences. Then we split the articles into overlapping segments of 5 sentences with 1 sentence overlap for indexing.

Our retrieval model is based on TCT-ColBERT (Lin et al., 2021b), specifically, the implementation provided by Pyserini (Lin et al., 2021a) and pretrained on the MSMARCO dataset (Campos et al., 2016). TCT-ColBERT is a distillation of ColBERT (Khattab and Zaharia, 2020) which is a late-interaction bi-encoder model. It encodes the query and the document separately into multiple vectors offline and then employs an interaction step to compute their similarity.

B Algorithm for Removing Temporal Loops

The key observation for finding the minimum feedback arc set is to convert the problem into finding an ordering $v_1 v_2 \cdots v_n$ among the vertices of graph G, then all of the edges $v_i v_j$ that violate this ordering by having $i > j$ will be feedback arcs.

[11] A category page is a curated list of Wikipedia pages organized by topic, such as https://en.wikipedia.org/wiki/Category:Disease_outbreaks.

To create a good ordering (with a small number of feedback arcs), we maintain two lists s_1 and s_2 which correspond to the head and tail of the vertex ordering. We first remove the source and sink nodes from the graph recursively by adding the source nodes to s_1 and the sink nodes to s_2.

For the remaining nodes, we compute a $\delta(v)$ score for each node which is the difference between the weights of its outgoing edges and incoming edges. Then the node with the maximal $\delta(v)$ will be appended to the end of s_1 and removed from the graph. This step is also done recursively and δ needs to be recomputed after removing nodes. Finally the ordering is obtained by concatenating s_1 and s_2. The complete algorithm is shown in Algorithm 1.

This ordering will divide the edges in graph G into 2 sets: the set of edges (v_i, v_j) that follow the ordering $i < j$ and the set of edges that go against the ordering $j > i$. The feedback arc set will be whichever of these two sets that have lesser edges.

Algorithm 1 A greedy algorithm for finding the minimal feedback arc set (Eades et al., 1993)

Require: Graph G
  $s_1 \leftarrow \emptyset$, $s_2 \leftarrow \emptyset$
  while $G \neq \emptyset$ **do**
    while G contains sink node v **do**
      $G \leftarrow G - v$
      $s_2 \leftarrow v s_2$
    end while
    while G contains source node v **do**
      $G \leftarrow G - v$
      $s_1 \leftarrow s_1 v$
    end while
    $v = \arg\max_G \delta(v)$
    $G \leftarrow G - v$
  end while
  $s \leftarrow s_1 s_2$

C List of Scenario Names

We show the complete list of scenarios in RESIN-11 and ODIN in Table 7.

All of our scenario documents and schemas are in English.

Dataset	Scenarios
RESIN-11	Business change
	Election
	General IED
	Kidnapping
	Mass shooting
	Natural disaster and rescue
	Sports
	Disease outbreak
	Civil unrest (Protest)
	Terrorist attack
	International conflict
ODIN	Chemical spill
	Chemical warfare
	Coup
	Cyber attack
	Health care
	Infrastructure disaster
	International aggression
	Investment
	Medical procedure
	Medical research
	Nuclear attack
	Political corruption
	Recession
	Refugee crisis
	Trading
	Transport accident
	Violent crime
	Warfare

Table 7: The complete list of scenarios that were used in our experiments. RESIN-11 provides many variants of the IED scenario, we kept the `General IED` scenario.

D List of Prompts and In-Context Examples

D.1 Templated Prompts in INCSCHEMA

Below is the prompt that we use for the *event skeleton construction* stage:

```
{evt.name} is defined as "{evt.description}".
List the major events that happen in the {evt.name}
↪ of a {scenario}:
```

`scenario` is the scenario name and `evt` can be filled in with the chapters that are provided as part of the input.

To assign names to the events, we use the following prompt with 10 in-context examples:

```
Give names to the described event.
Description: Disinfect the area to prevent
↪ infection of the disease. Name: Sanitize
Description: A viral test checks specimens from
↪ your nose or your mouth to find out if you are
↪ currently infected with the virus. Name: Test
↪ for Virus
Description: If the jury finds the defendant
↪ guilty, they may be sentenced to jail time,
↪ probation, or other penalties. Name: Sentence
```

```
Description: The police or other law enforcement officials arrive at the scene of the bombing. Name: Arrive at Scene
Description: The attacker parks the vehicle in a location that will cause maximum damage and casualties.  Name: Park Vehicle
Description: The government declares a state of emergency. Name: Declare Emergency
Description: The government mobilizes resources to respond to the outbreak. Name: Mobilize Resources
Description: The liable party is required to pay damages to the affected parties. Name: Pay Damages
Description: People declare candidacy and involve in the campaign for party nomination. Name: Declare Candidacy
Description: Assessing the damage caused by the disaster and working on a plan to rebuild. Name: Assess Damage
Description:{evt.description} Name:
```

For the second *event expansion* stage, we use the following 6 prompts:

```
What happened during "{evt.description}"? List the answers:
What are the steps in "{evt.description}"? List the answers:
What happens after "{evt.description}"? List the answers:
What happened before "{evt.description}"? List the answers:
List the consequences of "{evt.description}:
List the possible causes of "{evt.description}":
```

The prompt for the specificity test containing 10 in-context examples is:

```
Does the text contain any specific names, numbers, locations or dates? Answer yes or no.

Text: The UN Strategy for Recovery is launched in an attempt to rebuild the areas most affected by the Chernobyl disaster. Answer: Yes
Text: More than 300 teachers in the Jefferson County school system took advantage of counseling services. Answer: Yes
Text: The police or other law enforcement officials will interview witnesses and potential suspects. Answer: No
Text: The IHT will establish a Defense Office to ensure adequate facilities for counsel in the preparation of defense cases. Answer: Yes
Text: Helping people to recover emotionally and mentally from the trauma of the disaster. Answer: No
Text: The area is cleaned up and any contaminated materials are removed. Answer: No
Text: About 100,000 people evacuated Mariupol. Answer: Yes
Text: Gabriel Aduda said three jets chartered from local carriers would leave the country on Wednesday. Answer: Yes
Text: The party attempting the coup becomes increasingly frustrated with the ruling government. Answer: No
Text: The international community condemns the war and calls for a peaceful resolution: Answer: No
Text: {evt.description} Answer:
```

Examples of events that **did not** pass the specificity test:

```
Reporting of the first suspected cases in the limits of Union Council 39 of Tehkal Bala area
Focus groups with members of the public from 5 provinces were conducted to identify major factors influencing public knowledge, perceptions and behaviours during COVID
The PLO sent an encrypted message to the Iraqi Foreign Ministry in Baghdad
```

The prompt for the chapter test is

```
{chapter_evt.name} is defined as "{chapter_evt.description}"
{evt.name} is defined as "{evt.description}"
Is {evt.name} a part of {chapter_evt.name}? Answer yes or no.
```

For the *event-event relation verification* stage, we use the following questions:

```
Does "{e1.description}" start before "{e2.description}"? Answer yes, no or unknown.
Does "{e1.description}" end before "{e2.description}"? Answer yes, no or unknown.
Is the duration of {e1.description} longer than {e2.description}? Answer yes or no.
```

Note that in the verification stage, we use the probabilities assigned to the "yes", "no", and "unknown" tokens instead of directly taking the generated text as the answer.

D.2 In-Context Example for GPT3-DOT

The in-context example follows the DOT language specifications (https://graphviz.org/doc/info/lang.html) to linearize a graph. Here we only list a few events and relations due to length considerations.

```
List relevant events and edges in "chemical spills":
events:
0: chemical spills news story.
1: The accumulation of the chemical that is leaked later.
2: Chemicals are spilled into the environment.
3: The spill causes other hazards such as fire.
...
edges:
1->2[label='temporal']
2->3[label='temporal']
```

```
3->4[label='temporal']
...
```

E Schema Examples

We show an example of a schema generated by GPT3-DOT in Figure 5 and an example schema generated by INCSCHEMA in 6. In the visualization, blue nodes are events with subevents (children nodes) and yellow nodes are primitive events (leaf nodes). Blue edges represent hierarchical relations and go from parent to child. Black edges represent temporal edges and go from the previous event to the proceeding event. Schemas generated by GPT3-DOT are typically much smaller in size and confuse hierarchical relations with temporal relations.

F Human Assessment Details

We designed and distributed our human assessment task with Qualtrics[12]. We recruited 15 graduate students as our human assessors (all of which are paid as research assistants). The assessors had basic knowledge of what a schema is, but were not involved in the development of our model. Assessors were informed of the purpose of the study. Before they begin to work on the story-writing task, they were presented with task instructions (Figure 7 and an example response. We did not collect any personal identifiers during the assessment. The order of the schema graphs is randomized both in terms of the schema induction algorithm and the scenario. We show two screenshots of the interface in Figure 8 and Figure 9. Additionally, we show a figure of the schema generated by Double-GAE and a human response corresponding to the schema in Figure 10.

[12] https://www.qualtrics.com/

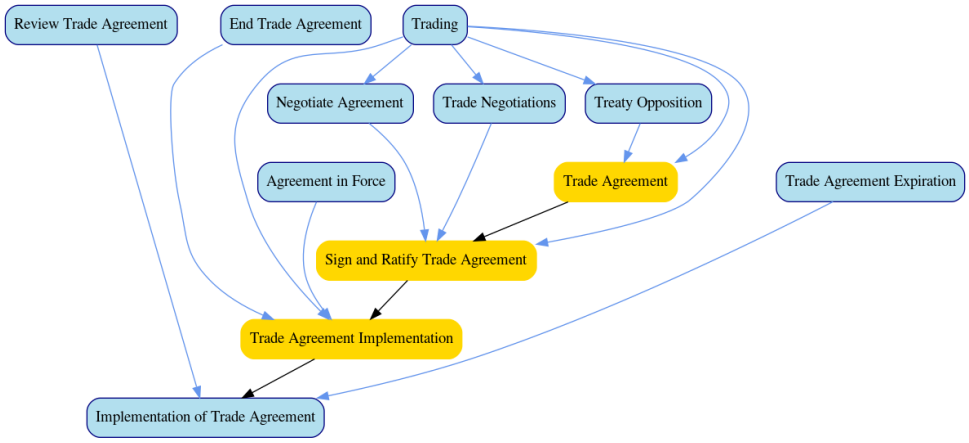


Figure 5: The Trading schema generated by GPT3-DOT.

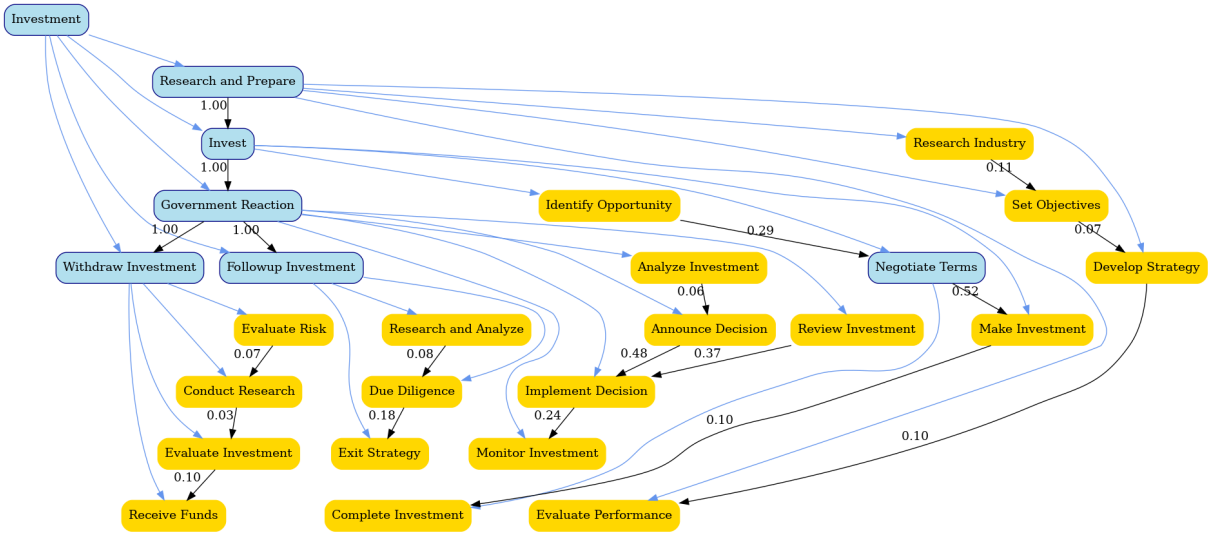


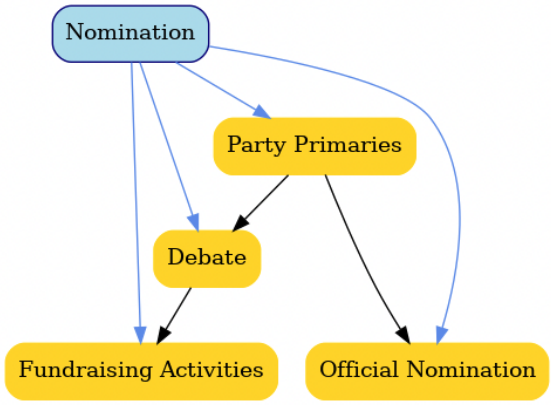
Figure 6: The schema for Investment generated by INCSCHEMA.

1. You will be shown a schema figure and a corresponding description. You will be asked to write a story that describes the event progression shown in the figure.

2. After finishing part 1, you will be asked several questions about your experience in completing the first task.

Detailed instructions:
- Each square node represents one event.
- There are two types of edges, blue edges indicate event-subevent relations and black edges indicate temporal relations. The numbers on the black edges indicate edge weight, which is correlated to the probability of one event following another.
- You are required to write a short story about the events shown in the graph. The story does not need to include concrete names or entities.
- The length of the story should be around 1-2 sentences per node.
- The story should be plausible and coherent, while trying to resemble the graph as closely as possible.
- Typical cases where the story can differ from the graph: (1) the temporal ordering in the graph does not make sense so the events in the story are reordered to make it plausible; (2) some events do not belong to the chapter or are completely irrelevant to the scenario, so the story can omit such events.
- After you write the story, you will be asked to indicate which events were included in your story. The events are roughly listed from left to right. If there are multiple events with the same name, we will add suffixes "Left, Middle, Right" to indicate the node's position in the graph.

Figure 7: Full set of instructions shown to assessors.



This is a schema for the Primary Election stage of an election event.

Nomination: People declare candidacy and involve in the campaign for party nomination.
Official Nomination: The candidate who wins the majority of delegates at the party's nominating convention is officially nominated as the party's candidate for the general election.
Debate: Candidates participate in debates and other forums to discuss their platforms and policies.
Fundraising Activities: Candidates engage in fundraising activities to support their campaign.
Party Primaries: Party primaries or caucuses are held in various states to determine which candidate will receive the party's nomination.

Please write a short story describing this figure:

Figure 8: The interface for human assessment. The assessor is shown a figure of the schema and descriptions of the events (if applicable).

Choose the events that were included in your story:

- Official Nomination
- Debate
- Fundraising Activities
- Party Primaries

Figure 9: After the assessors write the story for the schema, they will be asked to choose which events were included in the story.

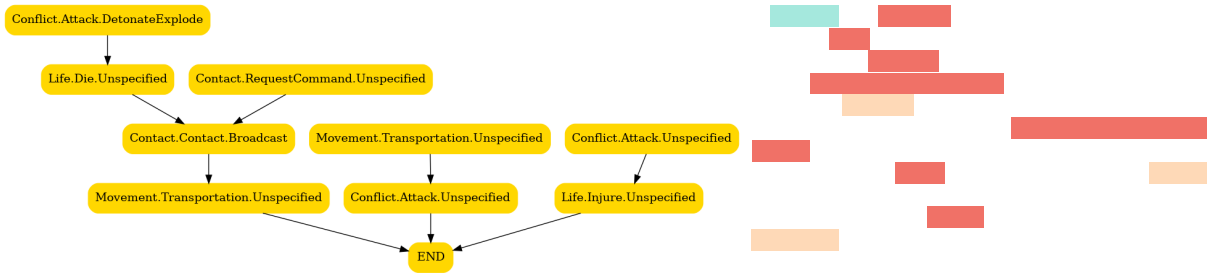


Figure 10: An example schema from Double-GAE (Jin et al., 2022) and human response for the interpretability assessment.