Human-in-the-Loop Schema Induction

Tianyi Zhang[*1], **Isaac Tham**[*1], **Zhaoyi Hou**[*1], **Jiaxuan Ren**[1], **Liyang Zhou**[1]
Hainiu Xu[1], **Li Zhang**[1], **Lara J. Martin**[1], **Rotem Dror**[1], **Sha Li**[2], **Heng Ji**[2]
Martha Palmer[3], **Susan Brown**[3], **Reece Suchocki**[3], and **Chris Callison-Burch**[1]
[1] University of Pennsylvania, [2] University of Illinois Urbana-Champaign
[3] University of Colorado, Boulder, [*] equal contribution
{zty, joeyhou, ccb}@upenn.edu

Abstract

Schema induction builds a graph representation explaining how events unfold in a scenario. Existing approaches have been based on information retrieval (IR) and information extraction (IE), often with limited human curation. We demonstrate a human-in-the-loop schema induction system powered by GPT-3.[1] We first describe the different modules of our system, including prompting to generate schematic elements, manual edit of those elements, and conversion of those into a schema graph. By qualitatively comparing our system to previous ones, we show that our system not only transfers to new domains more easily than previous approaches but also reduces efforts of human curation thanks to our interactive interface.

1 Introduction

Event-centric natural language understanding (NLU) has been increasingly popular in recent years. Systems built from an event-centric perspective have resulted in impressive improvements in numerous tasks, including open-domain question answering (Yang et al., 2003), intent prediction (Rashkin et al., 2018), timeline construction (Do et al., 2012), text summarization, (Daumé and Marcu, 2006) and misinformation detection (Fung et al., 2021). At the heart of event-centric NLU lie event schemas, an abstract representation of how complex events typically unfold. The study for such a representation dates back to the 70s, where scripts were proposed as a series of sequential actions (Roger C. Schank, 1977). Back then, the schemas were limited to linear and temporal ones. A more recent formulation of event schemas is a graph where the vertices are event flows and the edges are temporal or hierarchical relations between those events (Du et al., 2022).
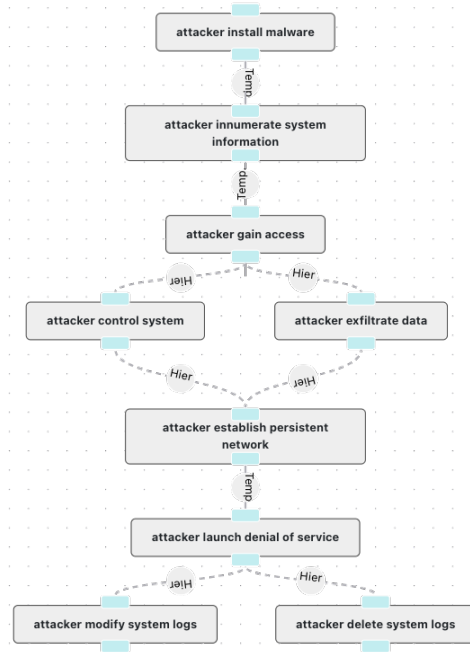

Figure 1: An example of Cyber Attack schema. The tree structure represents the temporal and hierarchical relations between the nodes.

For example, as shown in Figure 1, the event schema for a "cyber attack" could include sub-events such as "gain access", "control system", "exfiltrate files", "modify system logs", etc. The schema would also include the relationships between these sub-events. For instance, the event "gain access" would take place *before* the event "modify system logs" since a person needs access to a system before modifying it. For the same reason, "exfiltrate data" would only take place *after* "gain access". Event schemas like this encode high-level knowledge about the world and allow artificial intelligence systems to reason about unseen events (Du et al., 2022).

The DARPA Knowledge-directed Artificial Intelligence Reasoning Over Schemas (KAIROS) program[2] aims at developing schema-based AI sys-
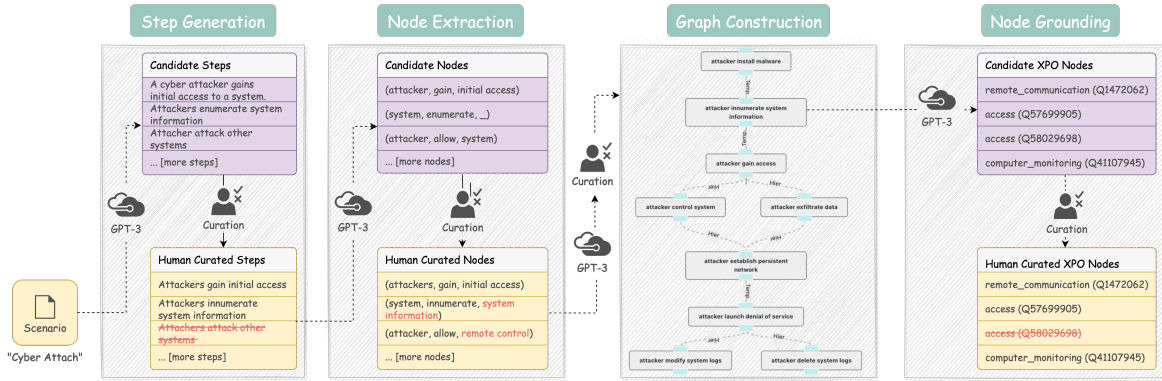

Figure 2: Our schema curation system includes four main stages: Step Generation, Node Extraction, Graph Construction and Node Grounding; Model ouput is highlighted in purple background ; Human curated output is highlighted in yellow background ; human curation is shown in red.

tems that can identify, comprehend, and forecast complex events in a diverse set of domains. To enable such a system, scalable generation of high-quality event schemas is very crucial. On one hand, fully-manual schema creation at a large scale can be inefficient, since people have diverse views about a certain concept, leading to inconsistent schema results. On the other hand, fully automated systems are scalable, but not with high-quality. In fact, the majority of existing approaches under the KAIROS program are fully-automated IR and IE systems over large collections of news articles (Li et al., 2020, 2021). Only some of limited human post-processing on schemas (Ciosici et al., 2021) have been explored. Further discussion of the advantages and limitations of existing systems can be found in Related Work.

Instead of focusing on fully-automated schema induction systems, we propose a human-in-the-loop schema induction pipeline system. Rather than using IR and IE over a large document collection, our system relies on pre-trained large language models (LLMs) and human intervention to jointly produce schemas. Our main motivation is that human-verified schemas are of higher quality. That is because human curation can filter out failure cases such as incompletness, instability, or poor domain transfer results in previous systems (Dror et al., 2022; Peng et al., 2019). With human curation, schemas are more reliable and accountable when applied to downstream tasks such as event prediction. This is significant if the downstream tasks involve safety-critical applications like epidemic prevention, where the quality of the schema matters beyond task performance numbers.

Figure 2 is a flowchart of our four-stage schema induction system: **step generation**, **node extraction**, **graph construction**, and **node grounding**. Each stage has two main components: the LLM (e.g. GPT-3) at the back-end to output predictions (the purple boxes in the figure) and an interactive interface at the front-end for human curation of the model output (the yellow boxes). The GPT-3 prompts that are used in each stage of the process are given in the Appendix A, along with example inputs and outputs.

A more comprehensive description of the implementation and functionalities of our interface can be found in Section 4. A case study is given in Section 5. It walks through each step in our pipeline system under an example scenario, cyber attack. Also, in Section 5, we provide a qualitative evaluation of five example scenarios. The summary and discussion of our system are included in Section 6.

2 Related Work

2.1 Schema Induction

Early work from Chambers and Jurafsky (2008, 2009) automatically learned a schema from newswire text based on coreference and statistical probability models. Later, Peng and Roth (2016); Peng et al. (2019) generated an event schema based on their proposed semantic language model (like an RNN structure). Their work represented the whole schema as a linear sequence of abstract verb senses like `arrest.01` from VerbNet (Schuler, 2005). Those works had two main shortcomings: first, the schema was created for a single actor (protagonist), e.g. suspect. It caused limited coverage in a more complex scenario, e.g. business change-acquisition; second, the generated schema, a simple

linear sequence, failed to consider different alternatives such as XOR.

More recently, Li et al. (2020, 2021) used transformers to handle schema generation in a complex scenario. It viewed a schema as a graph instead of a linear sequence. However, this approach was unable to transfer to new domains where the supervised event retrieval and extraction model failed. Dror et al. (2022) took GPT-3 generated documents to build a schema. Although it bypassed the event retrieval and extraction process and solved the domain transfer problem, it suffered from the incompleteness and instability of GPT-3 outputs.

Currently, neither do they offer a perfect solution for schema induction without manual post-processing, nor build a timely human correction system (Du et al., 2022). Our demonstration system develops a curation interface that can generate a comprehensive schema easily with a human curator in the loop. The curated data collected through our tool could be useful for fine-tuning and improving the models.

2.2 Human-in-the-loop Schema Curation Interface

Another area related to our work is human-in-the-loop schema generation, where annotators collaborate with computational models to create high-quality event schema. In this field, one of the closest approachs is the Machine-Assisted Script Curation (Ciosici et al., 2021) created for script induction. With a fully interactive interface, they have shown the feasibility of realtime interaction between humans and pre-trained LLMs (e.g. GPT-2 or GPT-3). The main differences are the level of automation and adaptability to other generative models. In terms of automation, our interface makes use of pre-trained LLMs to automatically generate schema content, compared to their interface which largely counts on human input. For adaptability, our interface supports the curation of the schema generated by different language models (e.g. GPT-3 models with different sizes), which makes it possible for users to evaluate the generations of different models. In contrast, there is no such possibility in their interface.

Another interface built for schema curation focuses on visualization of the schema structure, such as the temporal relations between event nodes and internal relations among entities (Mishra et al., 2021). While this interface provides a user-friendly experience when it comes to schema graph curation, it requires the user to come up with the content of event schemas in json format, which requires much more human effort compared to our interface. In addition, our interface also provides an optional grounding function after the event graph curation step, which is not presented in this interface.

3 Terminology and Problem Definition

Our work focuses on efficiently building a schema graph of a scenario using both LLMs and human input. Following the workflow of our system (see the workflow in Figure 2), a **scenario** is a general event type that an interested party will build the schema for, e.g. a 'disease outbreak'. **Steps** are a list of sub-events generated by GPT-3 according to a prompt in the step generation stage. Each step can be a phrase or a short sentence, such as 'spread to other areas', etc. **Nodes** or **tuples** are subject-verb-object pairs extracted from steps at the node extraction stage, such as '(disease, spread, to other area)'. **Graphs** are a visualization of the schema, whose edges joining the nodes represent temporal and hierarchical relations.

4 Implementation

Our pipeline system contains four sequential stages: **step generation**, **node extraction**, **graph construction**, and **node grounding**. A flowchart of the interface system is shown in Figure 2. The **step generation** stage generates steps for a scenario and the user can specify how many steps they would like to generate. The **node extraction** stage extracts nodes (subject-verb-object tuples) from the previous verbose steps. The **graph construction** stage orders the extracted nodes temporally and hierarchically. Meanwhile, modifications of the nodes are still possible. The **node grounding** stage maps node text to a node in the XPO ontology (Elizabeth Spaulding et al., In preparation) (derived from WikiData[3]). The flexible interface system allows users to either go through the entire process to create a schema from scratch or directly start at any stage to edit the model's prediction. In addition, the back-end GPT-3 models can be replaced by other user pre-trained models if deployed locally.

---

[3] https://www.wikidata.org/wiki/Wikidata:Main_Page

4.1 Step Generation

The step generation stage aims at generating steps given a scenario. At the backend, zero-shot GPT-3 incorporates a user's input into a prompt and generates ordered steps. The interface allows users to generate steps quickly with prompt templates[4] or finetune the generated steps with user-designed prompts. A typical use case of the user-designed prompts is to expand a certain step to more detailed steps. For instance, a template prompt "List the steps involved in {disease outbreak}:" may create steps such as "1. Identify the symptoms of the disease; 2. Collect data from affected individuals; ...". Then, the user can re-prompt for, e.g., the second step, "List the steps involved {step2} in detail:". Additionally, users can modify and select GPT-3 generated steps easily by clicking on them. When the 'save' button is clicked, all user selected steps will be saved in the database for the use of the node extraction stage or further fine-tuning of the step generation model. A screenshot of the step generation interface with user's operations can be seen in Figure 3.

4.2 Node Extraction

Nodes are structured representations of events in the form of a {subject, verb, object} tuple. Node extraction is to extract these nodes from the GPT-3 generated steps saved in step generation stage, which are unstructured sentences.

There are two methods, based on AllenNLP (Shi and Lin, 2019) or GPT-3, that users can choose from to extract nodes. The former uses AllenNLP's Semantic Role Labelling (SRL) model to extract nodes from the steps. The SRL model implements a BERT (Devlin et al., 2018) sequence prediction model to identify the predicates and the arguments (e.g. A0, A1) in a text. We simply choose the identified A0 as subject, A1 as object, and predicate as the verb to form a node. An optional coreference resolution model can be used to resolve referenced entities between the different steps with an AllenNLP's SpanBERT-based model (Lee et al., 2018). Here, we concatenate all the steps and replace a pronoun with its referenced entity (noun) in the original steps.

The GPT-3 node extraction method uses instructional few-shot prompting to extract {subject, verb, object} tuples from the steps. Several example sentences are given to show GPT-3 the expected syntactic and semantic output. We follow (Liu et al., 2022)'s recommendation for few-shot design by including context examples that are semantically similar to the KAIROS application environment (daily life and news). See appendix A for our few-shot prompts.

The extracted nodes are shown to the user in a table with 3 columns (subject, verb, object). For example, for "The CDC collects and analyzes data on disease outbreaks", one of the extracted nodes is "The CDC (subject) collects (verb) data (object)". Users are able to choose and edit nodes (tuples). User edits are saved and will be used for graph construction and fine-tuning of the GPT-3 node extraction model.

4.3 Graph Construction

In the graph construction stage, our system automatically adds temporal and hierarchical edges to the previously extracted nodes. The edges are created using zero-shot GPT-3 with multiple choice questions. For each pair of nodes, GPT-3 is instructed to choose between 'Before', 'After', 'Same time' or 'no relation' for temporal eges; and 'Parent', 'Child' or 'no relation' for hierarchical edges. For example, for the node pair "collect data" and "identify the signs and symptoms", GPT-3 predicts 'After' for temporal order and 'no relation' for hierarchical order, in which case we will add a temporal edge from "identify the signs and symptoms" to "collect data", and no hierarchical edge will be created. If a conflict occurs between (node1, node 2) pair and (node2, node1) pair, e.g. 'After' and 'After' for a temporal order or 'Parent' and 'Parent' for a hierarchical order, we will treat it as no relation to resolve the conflict, thus adding no new edges to the graph.

The graph construction interface allows users to modify the GPT-3 generated schema with ease. After predicting both temporal and hierarchical relations between all pairs of nodes, the interface will display the graph via the Vis-network framework[5]. It supports adding, editing, deleting graph nodes and edges. When the user clicks on a node, the detailed information including the ID and description of a node will be shown as well as the button to delete or edit the node. By clicking the edge, users can modify the edge type or delete it. Users will be

---

[4]an {event type} appended to a predefined prompt: Before, After or Sub-steps

[5]https://www.npmjs.com/package/react-vis-network-graph

able to create a new node by double clicking and a new edge by dragging and dropping an arrow from two nodes. A screenshot of our graph construction interface can be seen in figure 4.

4.4 Node Grounding

Although a schema (graph) is completely created after the previous stages, some nodes may express the same semantic information, e.g., "refugees flee" and "refugees ran away". To ensure the reliability and comparability of created schemas, our system grounds the nodes to an ontology, namely the XPO ontology, in the last stage. Each node in the XPO ontology contains a unique node ID, a node name, and a concise description (definition), and a list of similar nodes. Our system offers two ways of grounding, "name inference grounding" or "name similarity grounding". Name inference grounding maps the schema nodes to XPO nodes by predicting the XPO node's name; name similarity grounding finds the XPO nodes by comparing the similarities between the embeddings of a schema node and a XPO node's name.

In name inference grounding, given a graph node, our system first uses few-shot GPT-3 to deduce a list of possible XPO names (see few-shot prompt example in appendix A). Then, the candidate XPO names are postprocessed by dropping off the wrong prediction and adding similar XPO names to the true prediction. After that, each possible XPO name will be checked for entailment with the original graph node. The entailment model is a BART-large model fine-tuned on the MNLI dataset (Lewis et al., 2020; Williams et al., 2018). The input is the original graph node as the premise and the possible XPO name as the hypothesis, and the output is the entailment score. We sort the possible XPO node names by their entailment scores. Users can view and choose from the top-k suggested XPO nodes for the grounding of the original graph node. In name similarity grounding, the top-k related XPO nodes are retrieved by computing the cosine-similarity of the GloVe embedding between the graph node and the name of XPO nodes (Pennington et al., 2014). The above two methods are complementary to each other especially when users cannot find expected XPO nodes with one method. Human-curated data is saved in the backend database. A screenshot of node grounding can be seen in Figure 5.

5 Evaluation

5.1 A Case Study

In this section, we walk through the whole process of creating a toy schema with our interface which is much simpler than a fully developed schema. We assume the scenario is 'cyber attack'.

In the step generation stage, users can form a prompt from templates such as "list the steps involved in a cyber attack" with 'cyber attack' as the name and sub-event as the prompt type. Then, GPT-3 will generate 5 steps. For example, "1. A cyber attacker gains initial access to a system" and "5. The attacker exfiltrates data from the compromised system." Users can modify the content and choose steps to save. For example, one may change the first step to "1. A cyber attacker access a system." and save the step. See a screenshot of five steps for reference in figure 3.

Figure 3: A sample of generated steps after human-curation for scenario 'cyber attack'.

Next, in the node extraction stage, GPT-3 will be prompted to extract nodes from the selected steps. For example, GPT-3 will output {cyber attacker, access, system} for the first step. The user can change the outputs to correct any mistakes. In this sample, we extract 4 nodes, they are: {cyber attacker, access, system}, {attacker, enumerate, system information and user account}, {attacker, escalates, privileges}, {attacker, exfiltrate, data}. And we concatenate the {subject, verb, object} into a piece of text as a node for the next stage.

Thereafter, in the graph construction stage, we prompt GPT-3 to automatically build linear temporal edges on the above four nodes that users can modify. We manually add a scenario node 'cyber attack' and link with the other four existing nodes through hierarchical edges. see a screenshot of the graph in figure 4.

Finally, we can optionally ground our graph node into the XPO ontology. For example, the node "cyber attacker access system" can be mapped to choices of 'access', 'computer monitoring', 'remote communicating' using name similarity grounding. In this case, we don't get any results
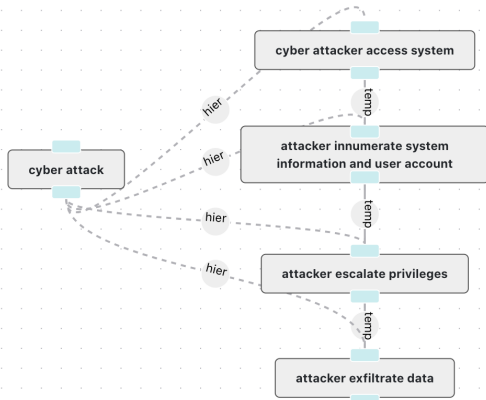
Figure 4: A sample of a constructed graph after human-curation for scenario 'cyber attack'

from name inference grounding. See a screenshot of grounding in Figure 5.

Figure 5: Top-4 XPO node choices of graph node "cyber attacker access system".

5.2 User Evaluation

We followed the evaluation methodology used by Ciosici et al. (2021) with slight modifications to assess our system. Evaluation is done by researchers in the field of NLP who have experience in hand-writing event schemas but have not used the interface before. In the step generation and node extraction stage, we count the number of human selected steps/nodes out of the total number of machine generated results as accuracy. For simplicity, we ignore users' modifications (e.g. rephrasing) at this point. In the graph construction stage, we compare how many nodes and edges are modified

	EVC	FOD	JOB	MED	MRG
Step Acc	11/12	7/8	10/10	10/10	12/12
Node Acc	13/15	10/10	11/12	12/12	12/14
Graph Node ED	1	0	0	0	0
Graph Edge ED	8	0	7	3	16
Grounding Success Rate	5/12	3/10	3/11	6/12	9/12
Self-reported time (min)	15	10	11	10	14

Table 1: User evaluation results. Acc in line 1 and 2 represents Accuracy. ED in line 3 and 4 means Editing Distance. Ciosici et al. (2021)'s approach, on average, took an hour to create the schema of a scenario.

(added or deleted) using graph edit distance. In the grounding part, the success rate is measured as successful retrieval of at least one relevant XPO node within top-3 grounding results for a given event node. We also ask users to self-report their total time of interaction. For all the evaluations, we use GPT-3 Davinci model as the language model.[6]

We follow prior work and evaluate our system on five scenarios: Evacuation (EVC), Ordering Food in a Restaurant (FOD), Finding and Starting a New Job (JOB), Obtaining Medical Treatment (MED), Corporate Merger or Acquisition (MRG).[7]

As shown in Table 1, our interactive system shows high accuracy in step and node generation phases, thanks to the richness of world knowledge from LLMs. However, the graph construction and the node grouding require more human curation, due to the difficulty of event reasoning, such as the understanding of temporal and hierarchical relationships; and the retrieval ability from large database. In those cases, we showed that human curation can step in timely and improve the quality of event schema when LLM-based models make mistakes [7]. In addition, our interface is easy to use, with much shorter time required to complete each event schema task compared to previous work (Ciosici et al., 2021).

We also report a qualitative study introducing the types of human modifications on the automated generations. At the step generation stage, GPTs aren't likely to make commonsense and grammar errors. However, if its required to generate more steps, it may be susceptible to redundancy, such as, *"A does B"* and *"A finishes doing B"*, and hu-

---

[6] https://platform.openai.com/docs/models/gpt-3

[7] Detailed evaluation results: https://joeyhou.notion.site/Human-in-the-Loop-Schema-Induction-Interface-Logs-1eb52403b05542919ccea214656f4211

man removes these steps. Then, for the node extraction, results can be simplistic and ambiguous when the original sentence contains rich information, such as location, condition, or other modifiers. For example, given the step *"waitress bring order to the kitchen"*, automatic node extraction produces *"(waitress, bring, order)"*, while human needs to add back some necessary components, e.g. the location information *"kitchen"* or constraint *"food order"*. Last, for the graph construction, current graph is often linear based on the previous nodes' order, human efforts play an essential role to elaborate on the specific relations including AND, OR. For example, *"person updates the resume"* and *"person tailors the cover letter"* are independent and can be concurrent, not sequential.

6 Conclusion

With the acknowledgements that fully depending on human annotation is expensive and inefficient, while wholly automated generations can be unreliable, we propose a human-in-the-loop schema curation interface with pre-trained large language models (LLMs) as the backbone. We use LLMs to generate candidate components of a schema and involve human as the final judge for both the content and structure of the event schema. With empirical evaluations, we show that our system can efficiently produce human-validated event schemas with minori human efforts.

Limitations

We have several limitations in our current approach. First, our current system uses zero-shot or few-shot to prompt GPT-3 without any fine tuning. In future work, we plan to fine-tune our GPT-3 with human curated data that we collect. We expect that fine-tuning will improve our models' performance. It may also be possible to use human curated data to train a policy network recommended by OpenAI (Ouyang et al., 2022). Second, we can replace GPT-3 with more robust task specific models at some stages, e.g., the pre-trained model for predicting temporal and hierarchical orders. Third, some users suggested incorporating a graph view at the other three stages, which will help users to generate based on the current graph. We will include this graph view in our next version. Forth, our current evaluation is experimental and probably subjective, we will develop more robust evaluation metrics comparing manual, Ciosici et al. (2021)'s and our

schema and test on downstream tasks in the next step.

Ethics Statement

To our knowledge, our back-end GPT-3 model was trained mainly on English web data, it may prefer events happen in an English environment. Furthermore, our test showed that it generated events specifically fit in American setting, for example, Miranda Rights for arrest, Democrats and Republicans in United States for election. These facts suggest GPT-3 may ignore the knowledge of non-American cultures or minority groups. In addition, currently, we only create schemas for scenarios that are reported in mainstream news media, e.g. conflict, communication. It excludes the schemas from other domains, such as biology, medicine.

7 Acknowledgements

References

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

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.

Manuel Ciosici, Joseph Cummings, Mitchell DeHaven, Alex Hedges, Yash Kankanampati, Dong-Ho Lee,

Ralph Weischedel, and Marjorie Freedman. 2021. Machine-assisted script curation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 8–17, Online. Association for Computational Linguistics.

Hal Daumé and Daniel Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, page 305–312, USA. Association for Computational Linguistics.

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

Quang Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 677–687, Jeju Island, Korea. Association for Computational Linguistics.

Rotem Dror, Haoyu Wang, and Dan Roth. 2022. Zero-shot on-the-fly event schema induction. *arXiv preprint arXiv:2210.06254*.

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.

Anatole Gershman Elizabeth Spaulding, Susan Brown Rosario Uceda-Sosa, Peter Anick James Pustejovsky, and Martha Palmer. In preparation. The darpa wikidata overlay: Wikidata as an ontology for natural language processing.

Yi Fung, Christopher Thomas, Revanth Gangi Reddy, Sandeep Polisetty, Heng Ji, Shih-Fu Chang, Kathleen McKeown, Mohit Bansal, and Avi Sil. 2021. InfoSurgeon: Cross-media fine-grained information consistency checking for fake news detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1683–1698, Online. Association for Computational Linguistics.

Kenton Lee, Luheng He, and L. Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *NAACL-HLT*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

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.

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.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Piyush Mishra, Akanksha Malhotra, Susan Windisch Brown, Martha Palmer, and Ghazaleh Kazeminejad. 2021. A graphical interface for curating schemas. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 159–166.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Haoruo Peng, Qiang Ning, and Dan Roth. 2019. Knowsemlm: A knowledge infused semantic language model. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 550–562.

Haoruo Peng and Dan Roth. 2016. Two discourse driven language models for semantics. *arXiv preprint arXiv:1606.05679*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference*

on empirical methods in natural language processing (EMNLP), pages 1532–1543.

Hannah Rashkin, Maarten Sap, Emily Allaway, Noah A. Smith, and Yejin Choi. 2018. Event2Mind: Commonsense inference on events, intents, and reactions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 463–473, Melbourne, Australia. Association for Computational Linguistics.

Robert P. Abelson Roger C. Schank. 1977. *Scripts, Plans, Goals, and Understanding : an Inquiry into Human Knowledge Structures*.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Hui Yang, Tat-Seng Chua, Shuguang Wang, and Chun-Keat Koh. 2003. Structured use of external knowledge for event-based open domain question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, page 33–40, New York, NY, USA. Association for Computing Machinery.

A   Appendix: Prompt and Completion

A.1   Appendix A: Prompt and example completions for Step Generation

Prompt: List the events before an attack: 1.
Completion:
The attacker gathers information about the target.
2. The attacker plans the attack.
3. The attacker gains access to the target system.
4. The attacker executes the attack.
5. The attacker covers their tracks.

Prompt: List the events after an attack: 1.
Completion:
The attacker's identity is confirmed.
2. The target is notified of the attack.
3. The attacker is placed on a watch list.
4. The attacker's device is seized.
5. The attacker is arrested.

Prompt: List the sub-events involved in an attack: 1.
Completion:
Identify the target.
2. Plan the attack.
3. Choose the weapons.
4. Assemble the team.
5. Launch the attack.
6. Evaluate the results.

A.2   Appendix B: Prompt and example completions for Node Extraction

Prompt:
/* instruction */
For each sentence, extract event verbs and their arguments, categorizing the arguments as subject or object. Write None if there is no object.
Return in [verb: _, subject: _, object: _] format.

/* few-shot examples */
For example:
Q: Isaac ate a cake today and he played football.
A: [verb: eat, subject: Isaac, object: cake], [verb: play, subject: Isaac, object: football]

Q: The teacher arrived in class and he started teaching.
A: [verb: arrive, subject: teacher, object: class], [verb: start, subject: teacher, object: teaching]

Q: Nate and Isaac ate dinner.
A: [verb: eat, subject: Nate and Isaac, object: dinner]

Q: Justin slept.
A: [verb: sleep, subject: Justin, object: None]

/* target example */
Q: The attacker gathers information about the target.
A:
Completion:
[verb: gather, subject: attacker, object: information]

/* target example */
Q: The attacker's identity is confirmed.
A:
Completion:
[verb: confirm, subject: attacker's identity, object:

None]

/* target example */
Q: The attacker is placed on a watch list.
A:
Completion:
[verb: place, subject: attacker, object: watch list]

A.3 Appendix C: Prompt and example completions for Node Grounding

Prompt:
/* few-shot examples */
List event names related to the event "People are infected with this disease":
1.infection
2.epidemic
3.pandemic

List event names related to the event "It was a robbery-related incident":
1.robbery
2.burglary
3.theft

List event names related to the event "The first case of the disease have detected and it has been reported":
1.infection
2.epidemic
3.pandemic

List event names related to the event "The disease is eventually brought under control":
1.control
2.improvement

List event names related to the event "People who are ill have serious symptoms":
1.symptoms

List event names related to the event "The pathogen begins to spread through the population":
1.transmission
2.spread

/* target example */
List event names related to the event "The attacker gathers information about the target":
Completion:
1.reconnaissance
2.surveillance

3.investigation

/* target example */
List event names related to the event "The attacker's identity is confirmed":
Completion:
1.identification
2.confirmation

/* target example */
List event names related to the event "The attacker is placed on a watch list":
Completion:
1.surveillance
2.monitoring
3.investigation