A Graphical Interface for Curating Schemas

Piyush Mishra[1,2] **Akanksha Malhotra**[1,2] **Susan W. Brown**[1,2]

Martha Palmer[1,2] **Ghazaleh Kazeminejad**[1,2]

[1]: University of Colorado Boulder
[2]: firstname.lastname@colorado.edu

Abstract

Much past work has focused on extracting information like events, entities, and relations from documents. Very little work has focused on analyzing these results for better model understanding. In this paper, we introduce a curation interface that takes an Information Extraction (IE) system's output in a pre-defined format and generates a graphical representation of its elements. The interface supports editing while curating schemas for complex events like Improvised Explosive Device (IED) based scenarios. We identify various schemas that either have linear event chains or contain parallel events with complicated temporal ordering. We iteratively update an induced schema to uniquely identify events specific to it, add optional events around them, and prune unnecessary events. The resulting schemas are improved and enriched versions of the machine-induced versions.

1 Introduction

Understanding events, how they progress, and who is involved in them is fundamental to our knowledge of the world and our ability to anticipate future events. Human beings have mental representations of typical scenarios at various levels of granularity. Defining such scenarios or templates for use in information extraction, knowledge base construction, and narrative prediction has a long history. As these fields have progressed, the complexity of the events and sequences being represented has increased. Any machine-readable format capable of representing multiple events, tracking their participants across the events, and delineating the temporal and causal relations between the events will be extremely difficult for a person to read and review. Because the extraction and construction of such complex schemas has accelerated in recent years (Li et al., 2020; Zhang et al., 2020), the need for a way for people to easily review them has increased. In this paper we will describe a tool designed to take complex event schemas in a json format and render them graphically for human review and revision.

The complex schemas handled by our tool include multiple levels of intersecting information. For example, imagine a typical emergency medical intervention. We know this includes a Victim who is injured or ill and usually begins with communication by the Victim or a Bystander to an emergency Dispatcher, then progresses to communication from the Dispatcher to Medical Personnel, travel by Medical Personnel to the Victim, immediate medical assessment of the Victim, and, finally, possible transportation of the Victim to a Medical facility. A more complete schema would include some alternative or parallel events, such as the possibility of Medical personnel already being on site or the death of the Victim at any point in the sequence of subevents. As we will describe in more detail, the tool uses distinctive nodes and edges to represent these subevents and participants and their relationships to each other. Left to right progression of subevents across the visual field represents temporal progression, and types of edges further indicate specific temporal and causal relations. Users can zoom in to specific subevents to see participants and their relations. In addition, our tool allows for simultaneous visualization of a complex schema and direct revision of the underlying json file.

Our paper begins by describing the background and motivation for our schema editing tool in Section 2. Section 3 gives detailed description of the tool implementation, while Sections 4 and 5 provide examples of its use, and discuss general issues with respect to editing schemas. We conclude in Section 6.

Proceedings of the Joint Conference 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, August 1st - August 6th, 2021.

2 Background

The idea of typical event scenarios being stored by people as abstract mental frames or schemas with slots for particular participants has a long history in psychology (Bobrow and Norman, 1975), linguistics (Fillmore, 1976) and artificial intelligence (Schank and Abelson, 2013). Prototypical sequences of events, such as the medical intervention sequence described above, can be made explicit in scripts that detail the usual subevents, their typical sequence, the people and objects generally involved in those events, and the progression of those participants through the subevents.

Proving useful for both linguistic analysis and natural language processing, repositories of defined schemas and event representations were manually developed, such as FrameNet (Fillmore et al., 2002), PropBank (Kingsbury and Palmer, 2002) and VerbNet (Kipper et al., 2008). Most longstanding repositories of such frames or scripts were built at the level of single events and their participants, although some have made sparse connections between these simple event descriptions (e.g., FrameNet's Uses and Precedes relations). The desire to extend these event representations to include more complex relations led to the development of systems of temporal, causal and other semantic relations, such as TimeML (Pustejovsky et al., 2005), Richer Event Descriptions (O'Gorman et al., 2016), Reference Event Ontology (Brown et al., 2017), and Abstract Meaning Representations (Banarescu et al., 2013).

Because of the substantial effort involved in manually creating schemas, automatically inducing schemas from textual and visual data has become a priority. Automatically generated schemas have advanced from schemas for single events (Balasubramanian et al., 2013; Chambers and Jurafsky, 2009; Chambers, 2013; Cheung et al., 2013; Huang et al., 2016; Nguyen et al., 2015) to complex, multistep schemas (Li et al., 2020; Zhang et al., 2020) However, for optimal usefulness, these generated schemas still benefit from human revision. To our knowledge, no open-source interface for complex schema visualization and editing has previously been developed.

One large-scale effort to create a repository of complex event schemas is the DARPA Knowlege-directed Artificial Intelligence Reasoning Over Schemas (KAIROS) program. KAIROS relies on the assumption discussed above, that humans make sense of events by organizing them into frequently occurring narrative structures that in this context are called schemas. The goal of the program is to develop schema-based AI systems that can identify, link and temporally sequence complex events and their subsidiary elements and participants. The program is set up with separate tasks. Task 1 involves inducing schemas from large amounts of text, followed by careful hand curation, with the goal of creating a schema library. Task 2 is aimed at finding schema instances that match schemas from the schema library in streaming news feeds. In order to evaluate system performance for both of these tasks, a common, agreed upon KAIROS Schema Format is needed. This also allows one DARPA team to try to instantiate schemas from another team's Schema Library and vice versa. The KAIROS Schema Format (KSF) stores representations of real-world complex events in a systematic JSON-LD format containing primitive events, their participants, possible entities acting as the participants, and the relations between these events and entities. Our tool takes these JSON files as input and assumes that all the schemas are validated and tested for format consistency before use. An additional complexity in this beginning phase of the program is a restriction against hierarchical schemas, in which subschemas could be collapsed into a single parent node. The schemas are orderings of all individual events, that can therefore get quite lengthy.

3 Schema Curation Interface

The Schema Curation Interface[1][2] is a web application designed for interpreting induced schemas. It provides a visual representation of the schema to understand the underlying structure, reflects relations between events and entities, and allows correction of potential flaws. The interface accepts KSF-validated schemas as input, extracts the events and participants as nodes and relations as edges, and visualizes them as a graph on a canvas. These graphs, in turn, can be corrected at the discretion of the curators. The interface is an open-source project that is accessible from *'cu-clear'* GitHub repository.

We use React.js and Flask for designing the web application. React.js is a JavaScript library to build interactive user interfaces (UI). It allows

[1]GitHub: https://github.com/cu-clear/schema-interface
[2]Demo: https://youtu.be/J9yox50gZUU

encapsulated component building and easy debugging, which makes new features easy to integrate. Flask is a micro web framework written in Python that acts as the web server for receiving requests from the user and sending a response. React passes the schema, uploaded by a user, to the Flask web server, which in turn extracts the possible nodes and edges between them and returns them to the UI for rendering. Cytoscape.js (Franz et al., 2015) uses these nodes and links to generate the graph on a canvas. Cytoscape.js is a fully featured graph library written in JavaScript that allows users to display and manipulate rich, interactive graphs. In the curation interface, it controls the positioning and layout. Explicit configuration constrains Cytoscape.js to orient the representation from left to right within the canvas, which preserves any possible temporal ordering and parallel events. It also has standard gestures like dragging and zooming on desktops as well as touch devices. Besides the canvas, a JSON-viewer provides a JSON view of the uploaded schema. It allows the user to edit the schema and dynamically update the graph structure. Add, edit, and delete are the three operations the "react-json-view" library allows for manipulating schemas. "react-json-view" is a React component for displaying and editing JavaScript arrays and JSON objects.

The interface is currently accessible from the web using Google Cloud Platform (GCP). We use Docker and Kubernetes in this process. Docker enables the packing, shipping, and running of our application as a portable and self-sufficient container, which can run virtually anywhere. Kubernetes runs and coordinates these containerized applications across a cluster of machines, automating the deployment, scaling, and management process. Kubernetes' load balancing configuration keeps at least one instance always available to a user. Since the user count is small, six replicas serve the purpose. But as the users increase, we can scale it up. A separate log server keeps track of any issue or error that occurs while parsing the schemas using RabbitMQ. It helps in debugging and analyzing the usage of the interface.

The representation consists of nodes and edges. The nodes signify an event (referred to in the schema as "step"), entity (as "participant" or "slot"), or filler (as "value" for the mentioned "participant"). Shapes like ellipse, round-rectangle, and round-pentagon distinguish one node type from another. The edges signify temporal relations between any two events, an entity's participation in an event, or co-reference between two entities.

The current implementation allows a dual-layer view. The first (default) view (Figure 2) shows only the representation of events and their temporal relations. Selecting an event opens the second view (Figure 3), consisting of entities and values for that event. As a result, all the entities and fillers remain hidden in the default view and reduce the clutter in the visualization, increasing readability.

While the dual-layer view allows a cleaner visualization of schemas, having many events in a schema (currently constrained to not make use of hierarchical structure) can still clutter the representation. Since the layout is from left to right, a complex event with a long sequential chain of events becomes partially hidden on reaching the screen width. Cytoscape.js reacts by zooming out the canvas to keep the graph in view as much as possible. However, in doing so, the schema graph becomes illegible. The only solution is manually arranging the nodes so they are in the scope and are using an appropriate zoom setting.

4 Interface Use

Approximately 10 people collaborating across three institutions used the interface heavily to induce schemas and manually curate them. The schemas were induced from shared data using information extraction systems. After evaluating a system's output, the inducers shared the schema draft with the human curators for editing. The interface facilitated the interpretation of the schemas and simplified the identification of needed changes to events, entities, values, or relations between them. We then created a new set of visualizations for comparison with the pre-curated version. The schema library inducers, after discussion, used these curated sets of schemas for improved inductions, which were passed back for further curation. We repeated this process until we reached a version where the schema had the highest coverage over the data for each complex event.

The human curators worked with two sets of automatically induced schemas, from two different institutions. One set of schemas consisted of events in a linear chain, lacking parallel events. The other set contained parallel events with more complex temporal ordering. By comparing the visualizations of these schemas, curators were able to

identify which schemas covered similar events and could select the best parts of each to create a single, comprehensive schema. In addition, we aimed to include all possible events in a scenario from planning stages to results. Every induced schema was missing events or had extraneous events, which were easy to identify and fix with the curation interface. However, the greatest improvement to curation came from the ease of visualizing parallel and optional events and of tracking entities across multiple subevents, all of which are obscured in the necessarily linear presentation of a JSON file.

Label	Life.Die
Description	The death of a person

Slot Role	Slot Argument Constraints
Victim	per
Place	loc, gpe, fac

Temporal	
Start and End	(times specific to event)
Duration	1 minute through 1 day expected

Figure 1: Life.Die Event Primitive

KAIROS Event Primitives are the backbone of the schemas. KAIROS has defined event primitives to be elemental single events that are unlikely to be decomposed into subevents, but could themselves form crucial elements of more complex events. The event primitives are comprised of the event definition, the roles associated with the event and their corresponding constraints, as well as temporal information about the event. Figure 1 provides an example of a Life.Die Event Primitive.

Events in the schema are automatically generated in temporal order from left to right in the curation interface. Parallel events originate from the same source and can occur simultaneously or independently of the sibling events. A gray rectangle labeled START indicates the beginning of the schema. Individual events are green ellipses. The general attack schema (Figure 2) can be used to represent any attack. We include a demonstration event as an instigating event that could motivate an attack. Alternative instigating events will be added in future work. After the instigating event, the attack event happens, leading to three simultaneously occurring parallel events - death, injury, and damages due to the attack, and one independently occurring parallel event - an investigation. Events related to medical intervention, an arrest, a trial, and sentencing follow.

By clicking on individual events, the event's arguments are revealed, along with various relationships between them. The entity nodes are peach-colored pentagons. Coreference relationships are indicated using the relation "Same as." In Figure 3, these coreference relations indicate that the attacker is the same as the agent of the Death event and of the Damage event. On the right side of the schema, the interface shows the schema in the JSON format (see Figure 2), which is directly editable. Therefore, if one sees that the order of events is wrong, or deletion or addition of events is necessary, these changes can be made within the interface.

We can also see details of the events and participant entities more legibly on the screen's left side by right-clicking on any specific event or entity, as shown in Figure 3.

The Drone IED (Figure 4) is an expansion of the general attack schema. We added more events specific to Drone IED's, such as buying drones, buying parts to make an IED, moving both to a common place, and assembling them. We added options like a drone crash, and a detonate event in place of the attack event from the general attack schema. We also added more event primitives related to damage and destruction. All other events are similar to the general attack schema.

5 Iterative Schema Updates

As mentioned above, there were several rounds of schema curation. One of our goals was identifying distinctive schema events. For example, in a drone-based IED schema, acquiring a drone is an essential step; and in a vehicle-based IED schema, acquiring a vehicle is an essential step. The final step was checking the proper temporal ordering between the events. We went through several wiki articles about various IED attacks to determine generic event types and the correct temporal ordering for all the steps. The visualization in the interface facilitated these tasks by allowing us to see generalities across schemas and to easily spot gaps in the temporal ordering.

In the subsequent rounds, we focused on robustness. We introduced new optional events to schemas, such as an acquittal event, which better encompassed the possible outcomes of a trial. We also introduced additional phases to the schemas, such as instigating events like conflict between the terrorist organization and a government. We also introduced retaliatory events to describe what ac-

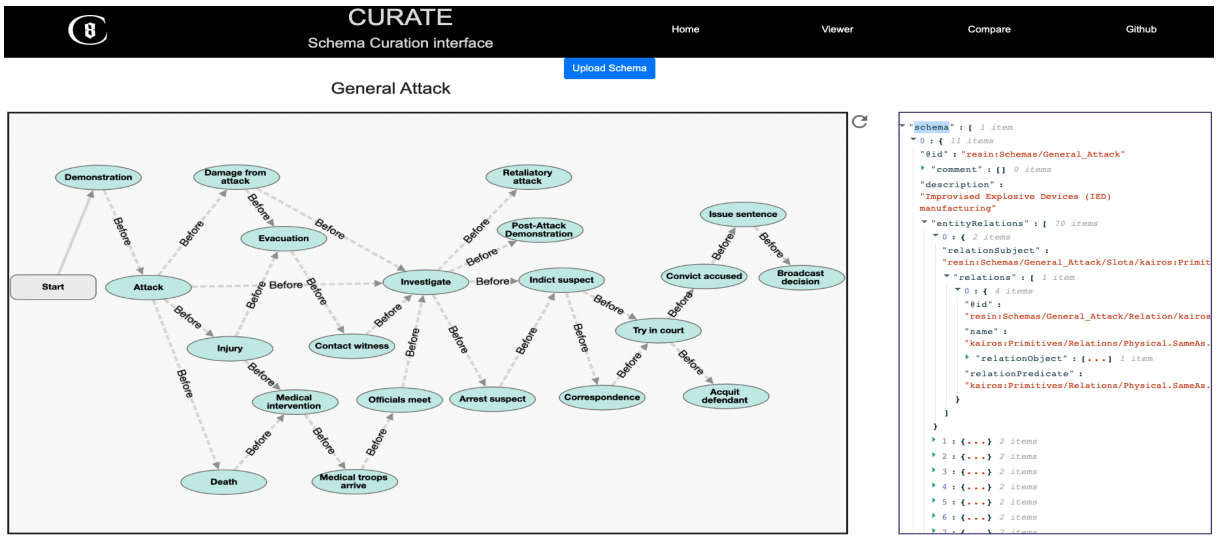


Figure 2: Editable JSON for schema curation

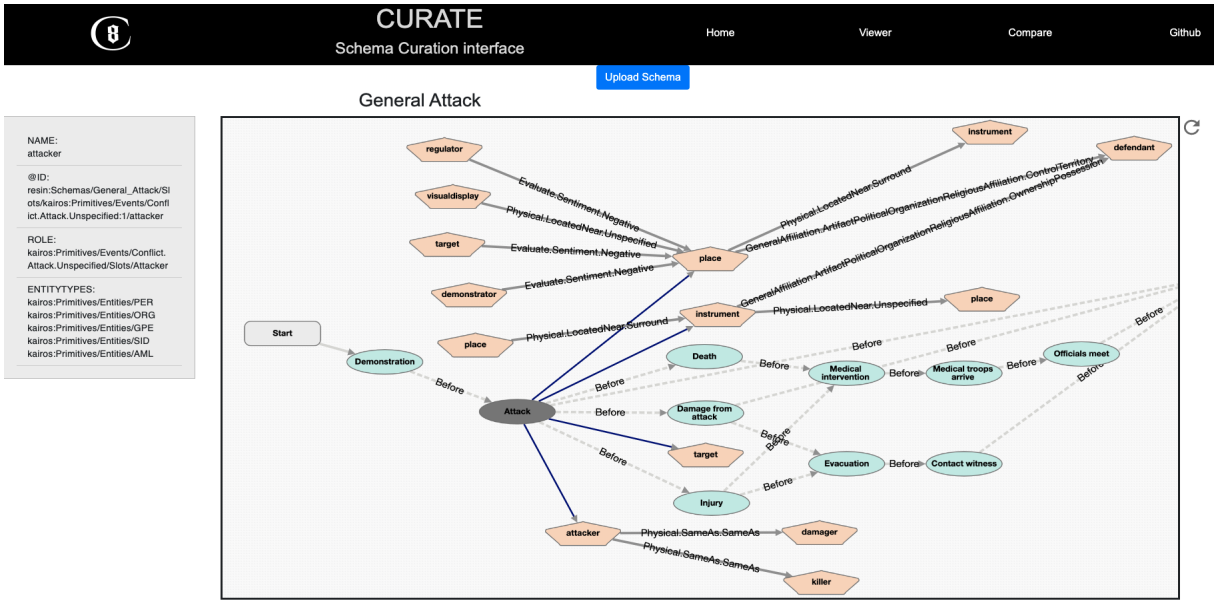


Figure 3: Entity associated to event Attack

tions were taken after the attack happened, such as demonstrations against the IED attacks or retaliatory actions taken by the army.

To demonstrate a typical editing task, the following example walks through the steps needed to make a new temporal connection between events. This example is also illustrated in our demo video.

In Figure 5a, there is a missing link between the events Injury and Medical Intervention. To add the link, we get the IDs of the Injury and Medical Intervention events by right-clicking on these events, and then create a new entry in the order key provided in the JSON editor on the right side of the screen by clicking on the plus sign next to the order key. This creates a new order step with NULL value. As shown in Figure 5b, within this box, we write the ID of the Injury event as the value to the "before" key, and the ID of the Medical Intervention event as the value to the "after" key. This signifies that an injury event needs to happen before medical intervention. We can also add flags like optional or precondition. The flag name is displayed on the arrow connecting the events. If no flags are given, "Before" is displayed on the arrow. We also need to provide a unique ID to this order step. After saving the JSON, the changes are automatically reflected in the visualization. Figure 5c shows the curated schema, with a link present between the

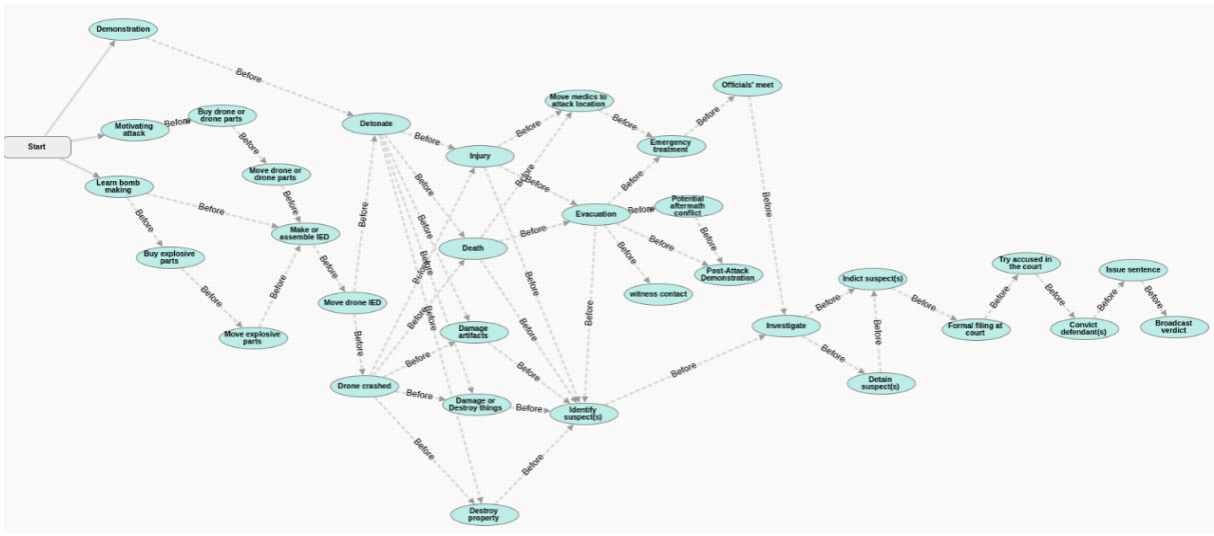
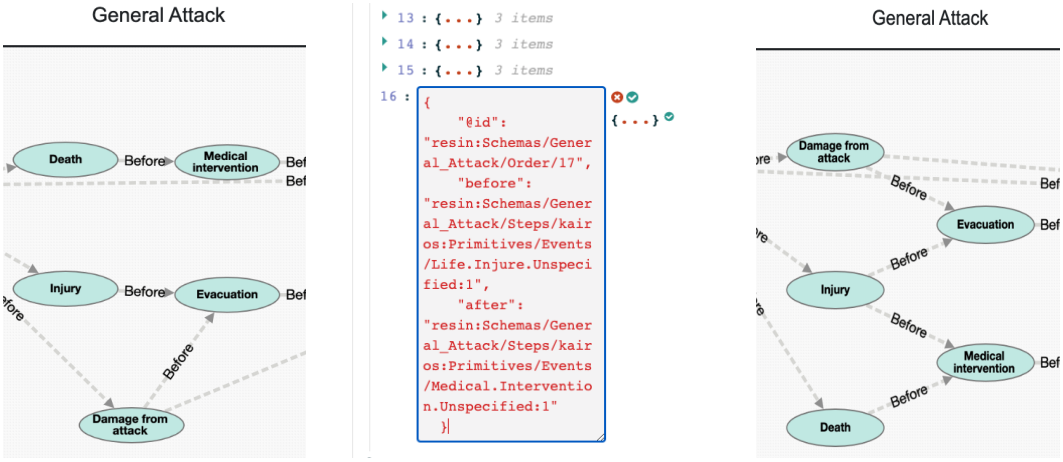


Figure 4: Drone IED



(a) No temporal link between Injury and Medical Intervention

(b) Editing JSON

(c) Temporal Link Added

Figure 5: Schema Curation Demo

Injury and Medical Intervention Event. Similarly, we can delete the events or link them. An event can be deleted by removing it from the step key in the JSON, and links can be deleted by deleting the order ID containing the mentioned events.

6 Conclusion and Future Work

In this paper, we introduced an interface that assists human curators in refining induced schemas. These schemas contain events, entities, and the relations between them. The curation interface extracts these elements in the form of nodes and edges and represents them in a graphical structure. The visualization enables the curators to better understand the ordering of events and the relations between entities, resulting in an improved and enriched schema. We also discussed leveraging the attributes of two structurally different induced schemas to design a single unified schema. This schema captures the salient events from its parents while reducing the schema size. We explored various IED-based schemas, General Attack, Medical Intervention, and Disease outbreak schemas using the curation interface.

The schemas currently include a small set of primitive events, limiting the scope of an induced complex event. In the future, a schema can have a hierarchical composition, meaning a combination of complex and primitive events within a single schema. Future work will focus on improving the handling of hierarchical schemas, provide dynamicity to the visualization, and comparison of two

schemas. It will also allow changes, like editing or deleting nodes and edges, on the graph beside the JSON editor.

Acknowledgements

We gratefully acknowledge the support of the DARPA KAIROS Program (contract FA8750-19-2-1004-A20-0047-S005, sub from RPI) for RESIN: Reasoning about Event Schemas for Induction of kNowledge, 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.

References

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1721–1731, Seattle, Washington, USA. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

Daniel G Bobrow and Donald A Norman. 1975. Some principles of memory schemata. In *Representation and understanding*, pages 131–149. Elsevier.

Susan Brown, Claire Bonial, Leo Obrst, and Martha Palmer. 2017. The rich event ontology. In *Proceedings of the Events and Stories in the News Workshop*, pages 87–97.

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

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.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. *arXiv preprint arXiv:1302.4813*.

Charles J. Fillmore. 1976. Frame semantics and the nature of language*. *Annals of the New York Academy of Sciences*, 280(1):20–32.

Charles J Fillmore, Collin F Baker, and Hiroaki Sato. 2002. The framenet database and software tools. In *LREC*.

Max Franz, Christian T. Lopes, Gerardo Huck, Yue Dong, Onur Sumer, and Gary D. Bader. 2015. Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics*, 32(2):309–311.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 258–268, Berlin, Germany. Association for Computational Linguistics.

Paul R Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*, pages 1989–1993. Citeseer.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation*, 42(1):21–40.

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.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. 2015. Generative event schema induction with entity disambiguation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 188–197.

Tim O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer event description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*, pages 47–56.

James Pustejovsky, Robert Ingria, Roser Sauri, José M Castaño, Jessica Littman, Robert J Gaizauskas, Andrea Setzer, Graham Katz, and Inderjeet Mani. 2005. The specification language timeml.

Roger C Schank and Robert P Abelson. 2013. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Artificial Intelligence Series. Psychology Press.

Hongming Zhang, Muhao Chen, Haoyu Wang, Yangqiu Song, and Dan Roth. 2020. Analogous process structure induction for sub-event sequence prediction. *arXiv preprint arXiv:2010.08525*.