

This work was written as part of one of the author's official duties as an Employee of the United States Government and is therefore a work of the United States Government. In accordance with 17 U.S.C. 105, no copyright protection is available for such works under U.S. Law.

Public Domain Mark 1.0

<https://creativecommons.org/publicdomain/mark/1.0/>

Access to this work was provided by the University of Maryland, Baltimore County (UMBC) ScholarWorks@UMBC digital repository on the Maryland Shared Open Access (MD-SOAR) platform.

Please provide feedback

Please support the ScholarWorks@UMBC repository by emailing scholarworks-group@umbc.edu and telling us what having access to this work means to you and why it's important to you. Thank you.

SAGEViz: Schema GEneration and Visualization

Sugam Devare^{*1}, Mahnaz Koupaee^{*1}, Gautham Gunapati¹, Sayontan Ghosh¹,
Sai Vallurupalli², Yash Kumar Lal¹, Francis Ferraro², Nathanael Chambers³,
Greg Durrett⁴, Raymond Mooney⁴, Katrin Erk⁴, Niranjan Balasubramanian¹

¹ Stony Brook University, ² University of Maryland, Baltimore County

³ United States Naval Academy, ⁴ The University of Texas at Austin

¹{sdevare, mkoupaee, ggunapati, sagghosh, ylal, niranjan}@cs.stonybrook.edu

²{kolli, ferraro}@umbc.edu, ³nchamber@usna.edu,

⁴{gdurrett@cs, mooney@cs, katrin.erk@}.utexas.edu

Abstract

Schema induction involves creating a graph representation depicting how events unfold in a scenario. We present SAGEViz, an intuitive and modular tool that utilizes human-AI collaboration to create and update complex schema graphs efficiently, where multiple annotators (humans and models) can work simultaneously on a schema graph from any domain. The tool consists of two components: (1) a curation component powered by plug-and-play event language models to create and expand event sequences while human annotators validate and enrich the sequences to build complex hierarchical schemas, and (2) an easy-to-use visualization component to visualize schemas at varying levels of hierarchy. Using supervised and few-shot approaches, our event language models can continually predict relevant events starting from a seed event. We conduct a user study and show that users need less effort in terms of interaction steps with SAGEViz to generate schemas of better quality. We also include a video demonstrating the system¹.

1 Introduction

Event schemas are central to understanding and reasoning about events. They provide a way to organize and represent how complex events unfold (Schank and Abelson, 1975; Mooney and DeJong, 1985). Schema-based reasoning enables reliable and explainable prediction of next events, inference of missing events or entities (Chambers and Jurafsky, 2008, 2009; Manshadi et al., 2008; Chambers, 2013; Balasubramanian et al., 2013; Pichotta and Mooney, 2016; Weber et al., 2018; Koupaee et al., 2021; Rezaee et al., 2021), and drawing connections between events that have already occurred (Kwon et al., 2020). For example, when a disease

outbreak happens, it is likely that an investigation into the outbreak and mitigation steps will follow. One main challenge in schema-based reasoning is in acquiring the schematic knowledge at scale.

One approach to automate schema curation is to learn from manually created reference schemas. Manual creation of complex hierarchical schemas require expert annotation, which is time-consuming and not scalable. Further, supervised systems (Ji and Grishman, 2008; Lin et al., 2021) are domain-specific, of poor quality and unable to handle unseen world events².

We propose SAGEViz, a human-in-the-loop schema curation and visualization pipeline, a combined approach to producing human-verified schemas using automated acquisition strategies. We leverage Large Language Models (LLMs) for acquiring various types of events and entity knowledge automatically. We use human inputs to ensure the contextual validity of model produced outputs at various stages in the curation pipeline. SAGEViz’s visualization allows a human curator to easily navigate through the complex event schemas at various levels of granularity starting with the higher level big picture and drilling down to the lower levels or vice versa.

SAGEViz allows users to build and curate a schema from scratch or edit existing ones. SAGEViz begins from a set of domain-expert-identified seed events. For each seed event, an event language model produces a suggested set of *next* events in a systematic fashion. For each participant in the seed event, we generate next events by asking the model to predict the next events in which the participant plays agent-based or patient-based roles (*Event Expansion*).

Human curator selects a subset of valid events

^{*}Equal contribution

¹<https://github.com/sugamxp/SAGEViz>

²models built till 2019 could not reason about COVID-19 since it had not occurred yet

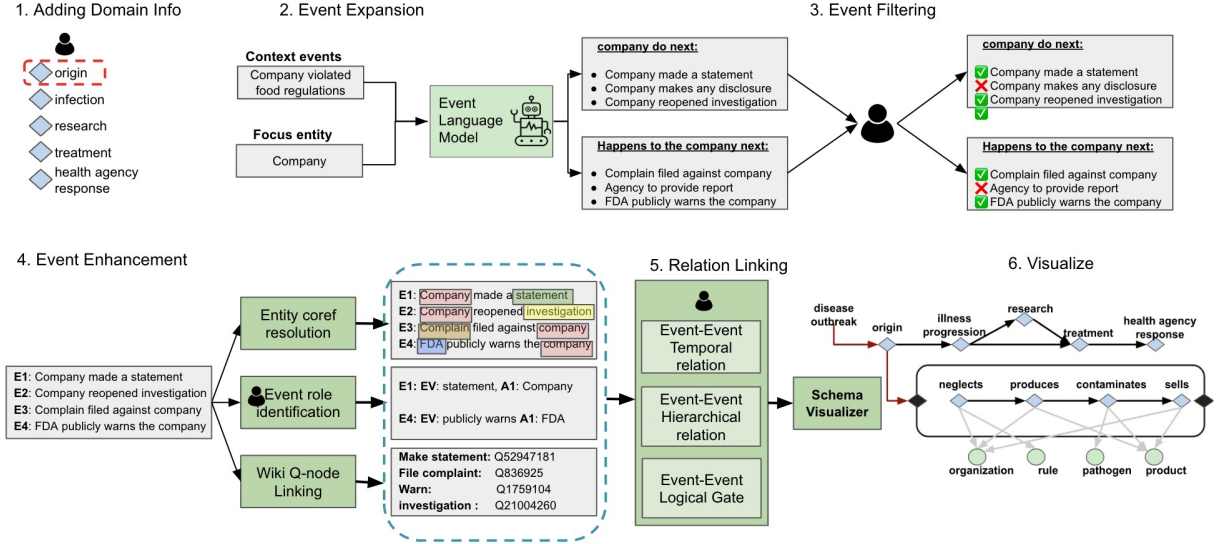


Figure 1: SAGEViz architecture capturing several curation stages and visualization. 1. Expert defines a high-level event structure for the domain 2. Based on some seed event, event language model recursively generates and expands events for the seed. 3. The candidate generated events are filtered by the expert 4. Filtered events are enhanced by adding additional event-specific information 5. Different categories of links are defined between the events 6. The schema at any point can be visualized as a multi-level graph.

from the model generated lists (*Event Filtering*) and enhances it with explanations, event-entity details such as roles, coreference and wikidata links (*Event Enhancement*) before linking event pairs based on logical, temporal and hierarchical event relationships (*Relation Linking*). Human curation in the enhancement stage ensures proper identification of entities with the goal of linking an event to multiple valid related documents. In the linking stage, we identify and validate the relationship between an event pair with the goal of building schemas with a hierarchical structure to support both specific and generic events. SAGEViz’s visualizer displays the schema as a graph at different levels of granularity, showing various event relation types with different widgets and color schemes.

SAGEViz has multiple advantages. First, multiple human annotators can provide input and curate a single complex schema in parallel. Second, it allows for iterative updating, to add, edit and delete non-conforming events at any stage. Third, with SAGEViz allows for modeling complex schemas through quick and intuitive means for generating and expanding sub-events and visualizing relations between them. Last, the back end and front end architecture of the tool allows curators to ensure global coherence of the generated schemas, across iterative updates and multiple simultaneous annotators, leading to more reliable schemas especially

suited for safety-critical downstream applications.

In summary, our contributions are as follows:

- We present a unified web-based tool to visualize and analyze complex event sequences.
- We leverage various LLMs to create and expand new and existing schemas.
- Our tool enables both human-human and human-AI collaboration for the task of schema generation.

2 System Description

In this section we provide a high-level overview and a detailed description of our human-in-the-loop schema curation framework.

2.1 Overview

A high-level overview of our system is illustrated in Figure 1. The system has two main components: (i) **Schema curator** - a system for curating the schema through LLM and expert collaboration. (ii) **Schema visualizer** - a system for visualizing the curated schema at any point during curation.

The schema curation process for a complex event starts with a domain expert creating a high-level structure of events to indicate event progression from start to end. For example, for the complex event **pandemic outbreak**, the expert creates a high level structure capturing its

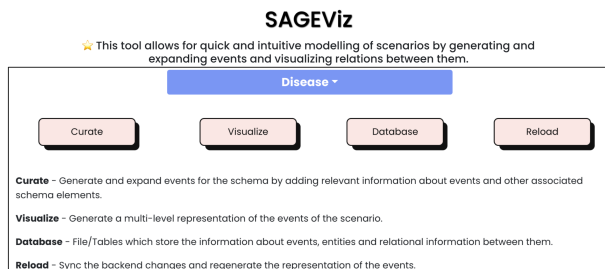


Figure 2: Main interface of SAGEViz. At this stage you can either select an existing domain from the drop-down menu to expand or edit. There are also four actions (along with their descriptions) that can be taken as shown in the figure.

progressive event stages as `pathogen_origin` → `illness_progression` → `investigation` → `mitigation` → `treatment` → `research` → `legal-actions` → `health-agency-response`. Each of the sub-events is recursively expanded until a desired level of granularity is achieved. While the initial list of sub-events is generated by a domain expert, the candidate subevents for these recursive expansions, are generated using an event language model that is trained for the next event prediction task. The relevant subevents from the candidate subevents are then selected by trained annotators. The selected events are mapped to their canonical form by linking them to corresponding Wikidata qnode entries. For each event, our system also has the provision to record text excerpts from a news article or other online documents, that act as evidence or justification for including the event in the overall schema.

There are multiple types of relationships between events where each relation type uniquely identifies how a pair of complex or sub-events and entities are related. Every event-entity, event-event and entity-entity pairs is related through one of hierarchical (parent, child), temporal (before, after) and logical (OR, AND, XOR) relationships by trained annotators after resolving event coreferences automatically. Our system supports multiple annotators annotating these structures in parallel. The schema visualizer enables the user to view the schema at different hierarchical levels with varying levels of granularity. The various relation types are displayed using different widgets and coloring schemes to easily grasp how the event sequences are related. The annotation process is incremental and iterative where annotators and models can revise an old annotation based on the current state of the overall annotation. The visualizer ensures the annotator can view the current state at all times to

ensure a coherent schema is generated.

2.2 Detailed System Description

The following sections will talk about the components of SAGEViz in more detail.

Event Generator The first and one of the most important components of the curator tool, is the event generator. These event generators can be finetuned event language models that are trained on the event sequences or can be any other large language model that can generate events (zero-shot or few-shot). The current version of the tool has two event generation models described below, however SAGEViz is designed such that incorporating new models is very easy as long as it uses a similar input/output structure.

Question-guided event language model: The first model used in the tool, is a finetuned event language model that takes a set of events and a question about an entity of interest as the context and generates the next event (Koupaei et al., 2023). The users can have control over the entities and can ask the system to generate the events with respect to the desired entities.

GPT-3.5 one-shot event generator: Given the efficacy of the large language models such as GPT models (Brown et al., 2020) for generation in zero/few shot settings, we incorporated GPT-3.5 as the second model in the tool.

Entity Coreference Resolution Since the questions or prompts to the event generator are entity-based, it is important to track entities across different events. Event generators might generate different mentions of the entities for different events (for example “they” and “attackers” referring to the same entity), however, we assign the same identifier to the various mentions across all events (this allows us to easily track all the events in which the entity “attackers” is involved in).

We incorporate the current state-of-the-art coreference resolution model SpanBERT (Joshi et al., 2019) which identifies clusters of coreferent entity mentions. We assign a unique identifier to all the entities in a cluster and use the identifier when referring to entities in the event sequence leading to more generalized schemas. Since the coreference model does not identify entities with a single mention, we identify noun phrases in the event sequence using the spaCy (Honnibal and Montani, 2017) library and assign unique identifiers to the entities that do not belong to a coreference cluster.

Figure 3: Curator section of SAGEViz user interface. This part of the UI is used to generate event sequences using different models that can be selected from the drop-down menu. Events generation, entity coreference resolution, etc. are all done on the curator section.

Event Role Labeler An event generated by our event language model consists of a subject-verb-object (SVO) format. Identifying the semantic roles of the entities enriches schematic knowledge regarding it participants. This version of SAGEViz uses manual assignment of semantic roles to entities (as existing systems do not perform well on events tuple representations) but the system design supports replacing it with an automatic system.

Q-node Linker Using the SVO formatted events, we use the entity linking model and python library BLINK (Wu et al., 2020) to identify Q-nodes for the verb (to represent the event), and Q-nodes for entities in both the subject and object. Since an entity or event could be associated with multiple Q-nodes, users can search for a Q-node from a provided list.

Schema Visualizer Once the schema is saved, all the information about the events, their relationships and the participating entities are converted to JSON format. This file is parsed into a directed graph structure, which is then ingested, visualized and presented in the visualizer, as shown in Figure 1.

2.3 User Interface

Multiple users can interact with SAGEViz (and its different components) simultaneously and curate or visualize schemas through an easy-to-use interface. Figure 2 shows the main interface of the system with its core components (curator and visualizer).

Curator Choosing the curator button will take users to another page (shown in Figure 3) where they can start with the event generator component. They can specify a seed event and an entity of interest (shown in the top section of Figure 3) and then the system provides users with a list of events specific to that entity (Generate Events box in the middle section of Figure 3). Now, the user has the option of selecting as many relevant events as they desire, edit the generated events, or add an event which they think might fit in the sequence to guide the next generation steps of the model. The events will be added to the Event Sequence box (left side of Figure 3) as the user selects them.

Once the user is done generating events for a seed, they can use the Entity identifier tab of the Event Sequence box to ask the system to identify the corefering entities and assign unique ids to them (as described earlier).

Finally, the EventDetails box (Figure 3) is where the users can search the Qnodes of the events as well as assign importance scores to the events. Once the user is satisfied with the generated sequence and after filling in all the necessary details, they can use the Save Schema tab to save it.

Visualizer The Visualizer component of the UI can represent the generated schemas in a multi-level graph representation. Once you click on Visualizer button, you will see the high-level graph structure of the (disease outbreak) schema containing the high-level stages as shown in Fig-

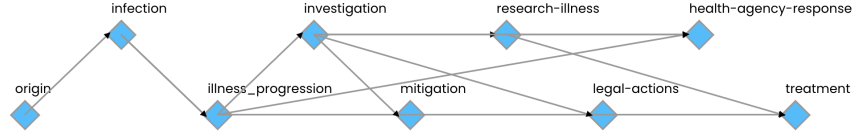


Figure 4: A high level view of the Disease Outbreak schema showing the first-level stages or sub-schemas. These sub-schemas can further be expanded into other sub-schemas or primitive events.

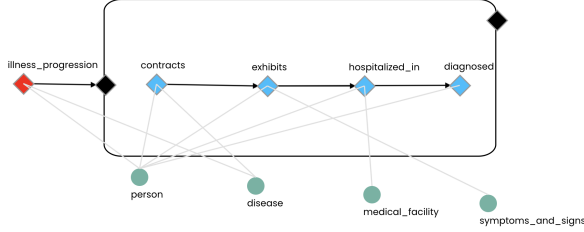


Figure 5: An expanded view of the illness_progression sub-schema. This is a sub-schema within the disease outbreak schema shown in Figure 4.

ure 4 and the temporal relations between these stages. You can further click on each node and then you can see the fine-grained representation of each stage consisting of all the events, entities, and the relations between entities and events (shown in Figure 5). You can keep getting deeper into the representation for all non-primitive events (primitive events are the atomic events that are not expanded).

2.4 Implementation Details

We now describe the technical details of the web application and how it is built in detail.

Frontend The frontend application uses React, a JavaScript library for creating user interfaces with modular components. To maintain visual consistency, React-Bootstrap is employed, providing a set of commonly used components. We utilize Webpack to generate static assets for the application and Express.js to serve the application. A significant challenge we encountered was optimizing the bundle size to enhance the application’s performance and overall user experience. To address this, we focused on developing reusable components, which allowed us to reduce redundancy and improve efficiency. Additionally, we leveraged the browser’s built-in caching capabilities to cache static assets, resulting in faster subsequent visits and reduced server load. Finally, we deployed the application on Amazon EC2 (EC2, 2023) which offers a robust, scalable, and cost-effective infrastructure.

Backend The backend APIs are developed using Flask (Grinberg, 2018), a lightweight Python web framework, which makes it an ideal choice

for hosting our Question-guided event language model. We load the language model (Koupae et al., 2023) and SpanBERT model (for coreference resolution) during application startup. For the GPT-3.5 few-shot event generator we use the OpenAI python library which gives us access to the gpt-3.5-turbo model. To optimize the usage of the gpt-3.5-turbo model and enhance overall performance, we implement a response caching mechanism. This caching system stores the responses obtained from the OpenAI API (OpenAI, 2023), preventing the need for repeated calls for the same or similar queries. By doing so, we reduce the number of API requests and consequently, minimize costs and latency. The caching strategy efficiently handles recurrent queries and ensures that if an identical request is made in the future, the corresponding response can be retrieved from the cache, avoiding unnecessary re-computation.

3 User Study

We aimed to study the effectiveness and efficiency of SAGEViz at generating schemas using the process outlined in Koupae et al. (2023) and compared it with the schema generation interactive system (CLI) used in this study. For a fair comparison, we selected only the event generator component of SAGEViz. We followed the same guidelines and used the same test data consisting of 35 seeds from 8 domains. Four different users interacted with the system spending 4 minutes on each seed, as was done in the previous study, selecting and adding events to the schema that are *sensible*, *relevant*, *unique* and *typical* for the schema context.

For each 4-minute interaction, a user started with a seed event as the schema context and a participating entity to use in the question to the event generator, and generated events using the Question Guided Event Language Model (QGELM) (Koupae et al., 2023). From the generated events, the user selected and added each event that satisfied all 4 of the selection criteria (each is a binary judgment). If any of the events was selected and added, the generation step is counted as an accepted

Metric	CLI	SAGEViz
# events ▲	8.8	12.1
% accepted steps ▲	73.3	80.0
% rejected steps ▼	26.7	20.0
total steps	12.0	7.48

Table 1: Quantitative analysis of schema generation using the our tool and the baseline. The higher the average the better a system is for metrics with ▲ whereas lower values are desired for metrics with ▼.

step and if none of the events fit the criterion, no events are added, and the step is counted as a rejected step before the user generates for the next step. Generation is stopped when the 4 minutes elapse or if no events that fit the selection criteria can be found in the generations. The participating entity (in the question to the event generator) can be changed in any step to encourage the generation of diverse events that fit the selection criteria. As shown in Table 1, SAGEViz allows a user to generate schemas more effectively and efficiently when compared to CLI (Koupae et al., 2023). On average, SAGEViz, allows a user to select more events (# events) with fewer interactions (total steps). The capability to edit generated events in SAGEViz leads to fewer regenerations (% rejected steps) with more generation steps (% accepted steps) accepted by users.

4 Related Work

4.1 Schema Induction

Chambers and Jurafsky (2008, 2009) automatically learned a schema from newswire text based on coreference and statistical probability models. Peng and Roth (2016); Peng et al. (2019) generated an event schema based on their proposed semantic language model., representing the whole schema as a linear sequence of abstract VerbNet (Schuler, 2005) verb senses. In these works, the schema was created for a single actor (protagonist). It caused limited coverage in a more complex scenario. Further, 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 treated 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 which by-

passed the event retrieval and extraction process and solved the domain transfer problem, but, it suffered from the incompleteness and instability of GPT-3 outputs. Currently, there is neither a perfect solution for schema induction without manual post-processing, nor a human correction system (Du et al., 2022). Our demonstration system develops a curation interface that can generate a comprehensive schema with a human curator in the loop.

4.2 Human-in-the-loop Schema Curation Interface

Another related area is the human-in-the-loop schema generation, where annotators collaborate with computational models to create a high-quality event schema. Machine-Assisted Script Curation (Ciosici et al., 2021) was 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 to other generative models. Our interface makes use of pre-trained LLMs to automatically generate schema content, compared to their interface which largely counts on human input. 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.

5 Conclusion

Prior work on schema induction either relied on existing information extraction pipelines to convert unstructured documents into event graphs, or required massive human effort to annotate event schemas. The former lacks the guarantee of being high-quality, while the latter is hard to scale due to its demands of time, effort and cost. We alleviated these problems with a web application that leverages the power of LLMs and the reliability provided by human intervention to create, expand, and visualize event schemas. Our tool enables users to collaboratively and seamlessly generate and update

event sequences. We believe that our human-in-the-loop tool reduces expert effort for creating new schemas and analyzing complex event sequences.

Acknowledgments

We thank the anonymous reviewers for their insightful feedback and suggestions. This material is based on research that is supported by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes.

References

- Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni, et al. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1721–1731.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- 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. 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.
- 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.
- AWS EC2. 2023. Amazon elastic compute cloud (amazon ec2) provides on-demand, scalable computing capacity in the amazon web services (aws) cloud. <https://aws.amazon.com/ec2/>.
- Miguel Grinberg. 2018. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc."
- Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.
- Heng Ji and Ralph Grishman. 2008. [Refining event extraction through cross-document inference](#). In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.
- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. [Spanbert: Improving pre-training by representing and predicting spans](#). *CoRR*, abs/1907.10529.
- Mahnaz Koupaee, Greg Durrett, Nathanael Chambers, and Niranjan Balasubramanian. 2021. Don't let discourse confine your model: Sequence perturbations for improved event language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 599–604.
- Mahnaz Koupaee, Greg Durrett, Nathanael Chambers, and Niranjan Balasubramanian. 2023. [Modeling complex event scenarios via simple entity-focused questions](#). In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2468–2483, Dubrovnik, Croatia. Association for Computational Linguistics.
- Heeyoung Kwon, Mahnaz Koupaee, Pratyush Singh, Gargi Sawhney, Anmol Shukla, Keerthi Kumar Kallur, Nathanael Chambers, and Niranjan Balasubramanian. 2020. Modeling preconditions in text with a crowd-sourced dataset. *arXiv preprint arXiv:2010.02429*.

- 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.
- Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. [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.
- Mehdi Manshadi, Reid Swanson, and Andrew S Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *FLAIRS Conference*, pages 159–164.
- 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.
- Raymond J Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *IJCAI*, pages 681–687.
- OpenAI. 2023. Openai api provides a general-purpose “text in, text out” interface. <https://platform.openai.com/overview>.
- Haoruo Peng, Qiang Ning, and Dan Roth. 2019. Knowsemmlm: 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*.
- Karl Pichotta and Raymond J Mooney. 2016. Learning statistical scripts with lstm recurrent neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*.
- Mehdi Rezaee, Francis Ferraro, et al. 2021. Event representation with sequential, semi-supervised discrete variables. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.
- Roger C. Schank and Robert P. Abelson. 1975. Scripts, plans, and knowledge. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI’75*, page 151–157, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania.
- Noah Weber, Leena Shekhar, Niranjan Balasubramanian, and Nathanael Chambers. 2018. Hierarchical quantized representations for script generation. *arXiv preprint arXiv:1808.09542*.
- Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Zero-shot entity linking with dense entity retrieval. In *EMNLP*.