Schema Curation via Causal Association Rule Mining

Noah Weber*[1], Anton Belyy*[1], Nils Holzenberger [1], Rachel Rudinger[2], Benjamin Van Durme[1]
[1]Johns Hopkins University
[2]University of Maryland, College Park

Abstract

Event schemas are structured knowledge sources defining typical real-world scenarios (e.g., *going to an airport*). We present a framework for efficient human-in-the-loop construction of a *schema library*, based on a novel mechanism for schema induction and a well-crafted interface that allows non-experts to "program" complex event structures. Associated with this work we release a machine readable resource (schema library) of 232 detailed event schemas, each of which describe a distinct typical scenario in terms of its relevant sub-event structure (*what* happens in the scenario), participants (*who* plays a role in the scenario), fine-grained typing of each participant, and the implied relational constraints between them. Our custom annotation interface, SchemaBlocks, and the event schemas are available online.[1] [2]

1 Introduction

What is implied by the invocation of a real-world scenario such as, say, a *criminal trial*? From one's knowledge of the world, one makes a myriad of inferences: that the scenario typically starts with the *defendant* being accused and brought to court, that it likely contains events such as the presentation of evidence by a *prosecutor*, and that it ends with the *judge* announcing the final verdict.

Though this type of scenario-level knowledge is recognized as being vital for text understanding (Schank and Abelson, 1977; Minsky, 1974; Bower et al., 1979; Abbott et al., 1985), explicitly annotating this knowledge in a way useful to language processing systems has proven to be a difficult task, subject to a seemingly fundamental trade-off. At one end, one may try to hand engineer this knowledge in a richly detailed format (DeJong, 1983; Mooney and DeJong, 1985; Mueller, 1999). While this facilitates quite impressive inferences, it requires an onerous annotation effort carried out by experts, and hence tends to be difficult to scale. On the other end, one may employ data-driven methods to automatically induce this knowledge (Chambers and Jurafsky, 2009, 2010; Balasubramanian et al., 2013; Rudinger et al., 2015), at the price of noise, and a severe loss of detail in the type of knowledge extracted.

One may also try to take a semi-automatic approach, taking advantage of both automatic and annotator driven components. This is the approach taken in Wanzare et al. (2016) (see also Regneri et al. (2010)) who use an initial human annotation to obtain various high quality event sequence descriptions for a target scenario, before using semi-supervised clustering to aggregate these annotations (Wanzare et al., 2017).

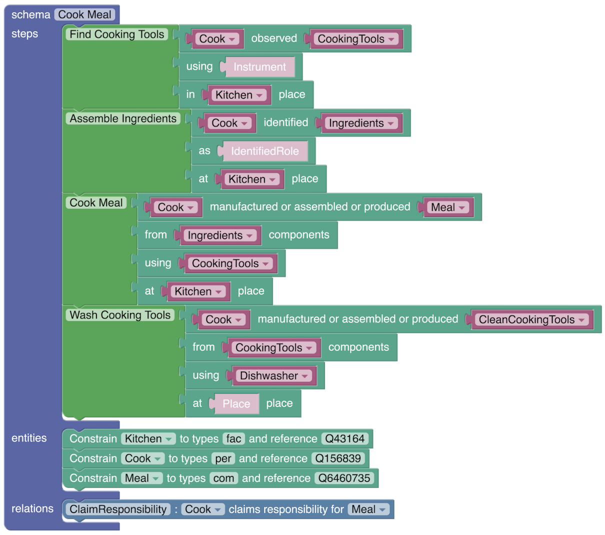

Figure 1: An example schema from our schema library, induced from a skeleton mined by Causal ARM (Section 3) and fully fleshed out by an annotator using our SchemaBlocks annotation interface (Section 4).

*Equal contribution. Order decided via wheel
[1]https://nlp.jhu.edu/demos/sb
[2]https://bit.ly/3gwpx2X

In this paper we also adopt a semi-automatic approach in order to facilitate the creation of a new annotated resource of structured, machine readable *event schemas*. Each event schema characterizes a real-world scenario, describing the events the scenario typically involves, the participants of these events (as well as their role and typing information), and the implied relations between these participants. An example visualization of an event schema is given in Figure 1. Our workflow follows two main steps:

1. Automatically induce what we term as *skeleton schemas*: argumentless event sequences that form the outline of an event schema. This is done via a new automatic script induction method which leverages and combines two recent advances in automatic script induction: an Association Rule Mining-based approach proposed in Belyy and Van Durme (2020), and a causal inference based scoring metric proposed in Weber et al. (2020).

2. Using our SchemaBlocks interface, have human annotators "flesh out" the manually selected skeleton schemas by adding argument, role, typing, and relational information, in addition to a name and description of the scenario the schema describes.

Via this process, we create a resource of 232 schemas, 150 of which are semi-automatically induced, with the rest being annotated from scratch from textual scenarios directly with SchemaBlocks. The content and format of this dataset will form the rest of the paper below.

2 The Anatomy of a Schema

Conceptualizations of what constitutes a "schema" differ across the literature. A schema in our resource is constructed from three basic elements: events, the entities that participate in these events, and the relations between these entities. The atomic types of events, entities, and relations in this work are defined by the DARPA KAIROS Phase 1 (v3.0) ontology. It consists of 67 event types, 24 coarse-grained entity types, and 46 relation types.[3]

Events The backbone for the meaning of a schema in this work is the temporally ordered chain of events that it describes. The individual events that make up this chain are drawn from a custom structured ontology which defines a taxonomy of events types (eg. an *Acquit* event, a *Transportation* event), in addition to event-specific slot types to be filled by arguments (eg. the *Defendant* or *Transporter* slot). While we use the term "chain" to describe the sequence of events defined in a schema, the schemas presented here need not always be ordered as a linear chain. In our schemas, subsequences of events in the schema may be marked either as occurring in a linear temporal order, in an arbitrary temporal order, or as forming mutually exclusive "branches."

Entities Entities fill the slots specified by each event defined in the schema. The same entity can (and usually will) be used to fill different slots across different events in order to indicate a coreferring relationship. All entities may also take on types, either coarse grained types defined in the KAIROS ontology (including types such as *person*, *commercial item*, etc), or fine grained types defined by the Wikidata qnodes. Our annotated schemas utilize both typing ontologies.

Relations Relations between participating entities are the last ingredient of the schemas defined here. These relations are also drawn from the KAIROS ontology. As of now, all relations are defined between two entities, each of which participate in at least one event defined in the schema.

3 Automatic Induction of Skeleton Schemas

Our system first automatically induces what we term as *skeleton schemas*: argumentless event sequences which form an outline of a potential event schema. It is a selected group of these skeleton schemas which are then passed forward to annotators to manually extend into full event schemas.

By starting the schema creation with an automatic, data-driven step, we allow the data to "speak for itself" with regards to what kinds of topics and scenarios we might want to target given some specified domain. The fact that the base of the schemas has some connection to our targeted domain gives at least some assurance that the final schemas will be applicable towards making common sense inferences in our domain when used in real-world applications.

The automatic system for skeleton schema induction combines two recent advances in schema

[3]The full ontology definition can be accessed at this link: https://bit.ly/3mIWJoN

induction: (1) an Association Rule Mining (ARM) based algorithm presented in Belyy and Van Durme (2020) which efficiently finds all event subsequences which have enough support in the data, and (2) a script compatibility scoring model presented in Weber et al. (2020) which finds high quality subsequences output by the ARM method and combines them together to form full skeleton schemas. We give a brief overview of each of these approaches and how they are used in our system below.

3.1 Association Rule Mining for Script Induction

Belyy and Van Durme (2020) show how prior classic work in automatic script induction (primarily the line of work following Chambers and Jurafsky (2008)) can be better recast as a problem of Association Rule Mining. ARM works with a dataset where each datapoint is a set of *items*. In the script induction setting, an item is an event and a datapoint is the set of events appearing in a document and sharing some co-referring argument. The ARM approach consist of two distinct stages:

1. **Frequent Itemset Mining**: This step searches for subsequences of events which have enough support in the dataset. What is considered "enough" is defined by a user set hyperparameter. To do this efficiently, Belyy and Van Durme (2020) make use of the FP-growth (Han et al., 2000) algorithm.

2. **Rule Mining**: This step uses the frequent itemsets mined from the previous step in order to define rules in a form similar to Horn clauses that can be used in downstream tasks.

In our system, we make use of only step 1 of the process defined above, mining event subsequences which have enough support in our targeted domain data. The output of this step is a large set of potentially interesting event subsequences.

3.2 Building Schemas with a Causal Scorer

The step presented in the previous section leaves us with fairly large inventory of event subsequences, not all of which may be useful or relevant for the creation of schemas. There are, hence, two problems at hand: (1) how to filter out lower quality subsequences and (2) how to create skeleton schemas from the filtered inventory of event subsequences. Both of these problems are handled via the causal inference based scoring approach presented in Weber et al. (2020).

The approach developed in Weber et al. (2020) defines a scoring function, $\text{cscore}(\cdot, \cdot)$ which, taking in two events e_1 and e_2, outputs a score proportional to the aptness of e_2 following e_1 in a script. The approach builds upon reasonable assumptions on the data generating process to overcome conceptual weaknesses in prior approaches and was shown to output scores more in line with human judgments of script knowledge. We refer readers to the paper for details.

In order to create our skeleton schemas, we first use the trained scoring module from Weber et al. (2020) to score all subsequences obtained via the process described in Section 3.1. Since the causal scoring module is only defined pairwise, we take the following average as the assigned score for a subsequence, $S = (e_1, ..., e_N)$, of length N:

$$\text{score}(S) = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \text{cscore}(e_i, e_j)$$

We take the top 100,000 of these subsequences. To ensure that a diverse set of event types are selected in the subsequences, we remove event subsequences in which all event types in the sequence have been used at least 50 times by higher scoring subsequences.

The last step involves joining together subsequences to create larger chains. For each of these 100,000 subsequences, we find the highest scoring event that may be appended to the subsequence. We then find other subsequences that start with this event, and append the highest scoring one to the existing subsequence. The top 1,000 of these larger subsequences are then given to a curator (one of the authors), who manually selects chains to be passed to human annotators as skeleton schemas. This is done as an expedient to ensure both the diversity and quality of the resulting schema annotations. Further work may be done to automate this process.

4 Manual Component

In this section we describe the annotation process. This includes a description of the SchemaBlocks interface, the annotation guide, and some relevant statistics on the annotation process.

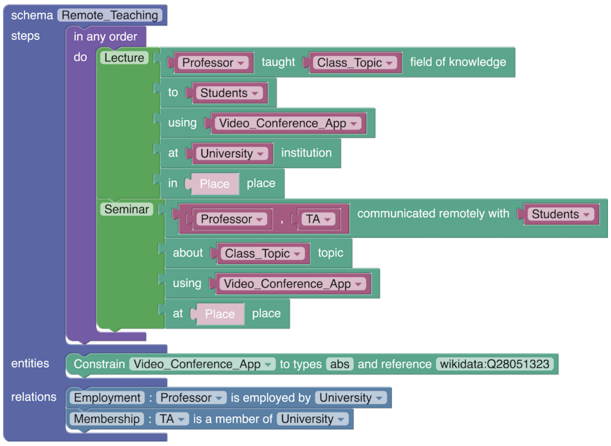

Figure 2: Hypothetical schema from the SchemaBlocks interface, featuring: 1) two events (steps) with the additional "any order" marker, 2) all of the entities have their types specified by the ontology, and the `Video_Conference_App` entity type is additionally narrowed down to `abs` and a link to Wikidata is provided, 3) two relations, `Employment` and `Membership`, are specified for three schema entities (`Professor`, `TA`, and `University`). Note also that the `Place` event slots for both events are omitted, being non-essential for this schema, and one slot is filled with two entities, since the Communication event is performed by both `Professor` and `TA`.

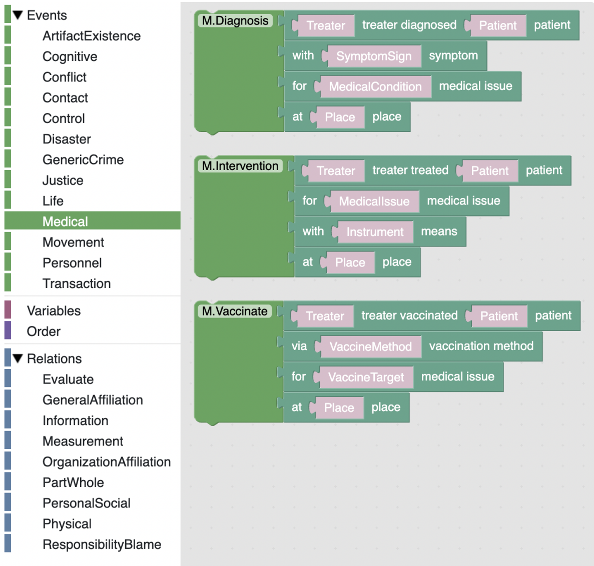

Figure 3: SchemaBlocks dashboard, displaying the high-level event and relation types from KAIROS ontology, with the "Medical" event category further expanded to show its subtypes. Other special block types, "Variables" and "Order", allow to fill multiple entities into one slot within an argument of an event and specify the ordering of events, respectively.

4.1 SchemaBlocks Annotation Interface

SchemaBlocks is a Web-based tool [4] that provides a way to display and modify the contents of a schema by representing its units – events and arguments, entity relations and types – as *blocks*, that can be stacked and nested. An example schema is shown on Figure 2. In addition to capturing schema events, participants, and their relations, the interface also allows to represent entity coreference, event ordering, and mutually exclusive events.

To get started, an annotator needs to familiarize themselves with the schema ontology, which defines the vocabulary of blocks they can use to build schemas. In the interface, this is displayed as the dashboard, organized hierarchically for convenience. Figure 3 shows all levels of the ontology hierarchy for the "Medical" event category of the KAIROS ontology used in this work. The block interface is flexible and could be adapted to a similar event ontology, such as FrameNet (Baker et al., 1998), ACE (Doddington et al., 2004) and ERE (Song et al., 2015).

SchemaBlocks is primarily based on the Google Blockly library. On top of the UI primitives provided by Blockly, we implement ontology-to-blocks and blocks-to-JSON converters to make user-created schemas usable in downstream tasks. We also make sure the schema entities have correct types by implementing continuous type checking and type inference in the UI. If a user breaks entity type constraints specified by the ontology, they will be notified and the relevant entity blocks will be highlighted until the error is fixed.

Our choice of block-based representation is inspired by Scratch (Resnick et al., 2009), a prominent tool that engages children to learn the basics of programming. By allowing annotators to program schemas using ontology-specific blocks, as opposed to general-purpose text formats such as JSON or XML, we are also able to engage more experts with non-programming backgrounds and annotate schemas at a faster rate. The annotators in our study (undergraduate students with non-CS majors) found the interface easy-to-use and left overall positive feedback.

To familiarize annotators with the interface, we provided them with the following guide prior to running the annotation: `https://bit.ly/3sBfAnf`.

[4] The repository with the UI can be accessed at this link: `https://github.com/AVBelyy/SchemaBlocks`

4.2 Annotating Schemas from Scratch

In the first annotation round, annotators were provided with 82 textual descriptions of schemas from an LDC developed resource.[5] The resource contains textual definitions for 82 schemas (called *complex events*). Each schema in the resource is given a title, a 2-3 sentence long description, specifications of the scope of the complex event (ie when and where the complex event should be considered initiated or finished), and the series of steps that defines the complex event/schema. Each step is defined with a title that specifies the event type of the step (in natural text, no event ontology is used), a short one sentence description, and expected high level event types that may happen as subevents.

The annotators are then tasked with "translating" these textual descriptions of schemas into a machine readable form via our SchemaBlocks interface. Relations and entity types are not specified in the textual descriptions, so annotators are instructed to annotate for relations that must be true throughout all steps of the schemas, as well as provide specific entity types and links to Wikidata. Annotators reported an average time of 30 minutes per schema to fully annotate, with 82 schemas being the product of this annotation task. The number of events in each of 82 schemas ranges from 2 to 10, with 6 being the median number of events.

4.3 Fleshing out Skeleton Schemas

In the second annotation round, annotators were tasked with fleshing out the skeleton schemas automatically induced via the method presented in Section 3. Given a skeleton schema, we import it into SchemaBlocks as a partially filled out schema of which only its events have been specified. We then present these partially filled out schemas to annotators and task them with:

- Determining what scenario the partially filled out schema is describing. This includes determining a name for the schema, as well as a brief textual description on what it is about.

- Determining what entities fill the slots of the given events in the schema, what types (coarse and fine-grained) they take on, and which slots are filled with co-referring entities.

- Determining what relations hold between the above defined entities. The criteria for annotating relations here is the same as before.

[5] LDC2020E25

Given this annotation is designed to be similar to the one presented in Section 4.2, annotators who participated in the previous annotation effort (all of them) required little extra training to complete this annotation, only a single one-hour training session. Again, annotators reported around a 30 minute average to annotate a schema. The end result of this fleshing out process is an additional 150 schemas. The number of events in this additional set of schemas ranges from 3 to 6, with 4 being the median number of events.

5 Schema Library Evaluation

In this section, we evaluate the match between proposed schemas and a corpus. We propose the **dataset coverage** measure to evaluate such a match, given no additional labels. When the documents are provided along with schemas that apply to them, we additionally compute the **ranking** measures, which treat the matching between a document and a schema as a retrieval task where both can participate either as a query, or as an indexed document.

To quantify what constitutes a match between documents and schemas, we define a similarity function $sim(d, s) = |d \cap s|/|d|$ which counts how many events in a document d are matched by a schema s. For the purposes of this metric, we treat documents as multisets and schemas as sets of events. Thus, if a document $d = \{\text{LIFE.INFECT} : 2, \text{MEDICAL.VACCINATE} : 1\}$ is matched with a schema $s = \{\text{LIFE.INFECT}, \text{LIFE.DIE}\}$, the similarity will be $sim(d, s) = 2/3$. We bucket the results by the number of identified events $\mathrm{N}_{\text{events}}$ in each document. We compute the final metrics using bootstrap over 100 samples and report the mean performance.

To extract events from text documents, we use the FrameNet parser from the LOME IE system (Xia et al., 2021), which identifies FrameNet events and their arguments. We map the events to KAIROS ontology using a rule-based mapping[6].

5.1 Datasets

LDC corpus Under the DARPA KAIROS program, the Linguistic Data Consortium (LDC) has annotated 924 multilingual multimedia documents (covering images, audio, video, and text in English and Spanish) with KAIROS event types and a com-

[6] The mapping rules can be accessed at this link: `https://bit.ly/3a7TMJl`

	Schema ranking			Document ranking		Dataset coverage		
N_{events}	Avg Rank	MRR	R@10	R@30	nDCG	Cov@0.5	Cov@0.7	Cov@0.9
$[1; 5)$	26.4	0.112	0.244	0.387	0.246	0.960	0.852	0.797
$[5; 10)$	23.8	0.147	0.340	0.472	0.276	0.937	0.785	0.614
$[10; \infty)$	20.8	0.194	0.410	0.545	0.269	0.925	0.759	0.533
$[1; \infty)$	21.1	0.191	0.404	0.442	0.272	0.925	0.761	0.542

Table 1: Summary of the evaluation of 82 schemas on the LDC corpus, using gold events.

	Schema ranking			Document ranking		Dataset coverage		
N_{events}	Avg Rank	MRR	R@10	R@30	nDCG	Cov@0.5	Cov@0.7	Cov@0.9
$[1; 5)$	35.4	0.072	0.199	0.293	0.162	0.895	0.576	0.491
$[5; 10)$.	32.1	0.088	0.193	0.347	0.170	0.833	0.502	0.334
$[10; \infty)$	30.6	0.105	0.229	0.411	0.247	0.759	0.417	0.242
$[1; \infty)$	30.2	0.109	0.239	0.351	0.240	0.745	0.400	0.223

Table 2: Summary of the evaluation of 82 schemas on the LDC corpus, using IE extracted events.

plex event (CE) label.[7] The CE label indicates the complex event (from LDC2020E25) that best applies to a document. Each CE label is covered by 11 documents on average, each document having one CE label. Out of 924 documents, 921 have partial event annotations (event type, link to a complex event step, and a provenance link to a span/offset in a document) and 36 have complete annotations (with identified and provenance linked entities and relations). Given the sparsity of the latter, we opted to only use event type annotations in order to compute ranking-based metrics.

Gigaword We pick (uniformly at random) a subset of 100K documents from the NYTimes portion of the Fifth Edition of the English Gigaword (Graff et al., 2003) corpus, spanning the New York Times news articles from years 1994–2010. This corpus is typically employed in the script induction literature. We use it to compute dataset coverage of our schemas over a large newswire corpus.

CC-News Additionally, we employ the CC-News corpus (Nagel, 2016) which provides a wide array of news articles over multiple languages. We pick a random subset of 100K English-language articles from years 2016–2017. To evaluate cross-lingual abilities of schemas, we also pick 100K news documents in Chinese and Russian from the same corpus, covering years 2016–2019. We use the cld3 library along with the "meta_lang" field from the news source for language ID. The collection is deduplicated in the following way: first we run the clustering over all documents in a particular language. If an article is in a cluster with more than 20 documents we remove all documents in that cluster, otherwise we select one representative.

Both Gigaword and CC-News are tokenized using spaCy v2.3.5 (Honnibal et al., 2020).

[7]At the time of writing, these annotations have been split into three collections: LDC2020E24, LDC2020E31, and LDC2020E35. While rarely freely released, historically such collections are eventually made available under a license to anyone, under some timeline established within a program.

5.2 Schema ranking

How well can we predict the true CE using the match between schema and document events as a ranking function? To answer this, we are using documents from the LDC corpus, where each document d has precisely one CE label in that dataset. For each document d, we rank schemas according to $sim(d, s)$ and report the average rank (lower is better), Mean Reciprocal Rank (MRR, higher is better), and Recall@10 (R@10, higher is better) of the gold CE label in Table 1. The performance tends to improve with longer documents, as the documents' descriptions (in terms of identified events) become richer and match a particular schema more precisely. Note however that perfect performance is not expected here: there are many contributing factors to a correct prediction of a schema other than matching the set of events. If this task was solved perfectly, this would render complex schema-based inference systems unnecessary, which we are not trying to show. Instead, we argue for event match to be a useful first step to narrow down a set of candidates. We also compare gold event annotations with IE extracted events in Table 2.

	82 schemas			82 + 150 schemas		
$\mathrm{N}_{\text{events}}$	Cov@0.5	Cov@0.7	Cov@0.9	Cov@0.5	Cov@0.7	Cov@0.9
$[1;5)$	0.887	0.531	0.425	0.975 (+10%)	0.637 (+20%)	0.509 (+20%)
$[5;10)$	0.791	0.391	0.233	0.892 (+13%)	0.496 (+27%)	0.278 (+19%)
$[10;\infty)$	0.695	0.313	0.164	0.807 (+16%)	0.379 (+21%)	0.195 (+19%)
$[1;\infty)$	0.684	0.303	0.154	0.798 (+17%)	0.367 (+21%)	0.183 (+19%)

Table 3: Gigaword dataset coverage, using IE extracted events.

	82 schemas			82 + 150 schemas		
$\mathrm{N}_{\text{events}}$	Cov@0.5	Cov@0.7	Cov@0.9	Cov@0.5	Cov@0.7	Cov@0.9
$[1;5)$	0.874	0.588	0.529	0.980 (+12%)	0.719 (+22%)	0.643 (+22%)
$[5;10)$	0.784	0.450	0.303	0.915 (+17%)	0.558 (+24%)	0.368 (+21%)
$[10;\infty)$	0.708	0.376	0.224	0.850 (+20%)	0.472 (+26%)	0.272 (+21%)
$[1;\infty)$	0.720	0.392	0.246	0.860 (+19%)	0.490 (+25%)	0.299 (+22%)

Table 4: CC-News (English) dataset coverage, using IE extracted events.

5.3 Document ranking

In a similar vein, we ask: how well do event types allow us to rank documents given a schema as a query? We report Recall@30 and normalized discounted cumulative gain (nDCG) of the gold annotated documents. Similar to schema ranking, the ranking of documents improves with longer documents as they become more descriptive of a true complex event.

5.4 Dataset coverage

How many documents are explained, or "covered" by at least one schema? We say that a document d is "t-covered" by a schema s if $sim(d,s) \geq t$ and define "coverage at t", or Cov@t, as the ratio of documents in the dataset t-covered by at least one schema. This measure does not require any CE label annotations, so we compute it for the LDC corpus (Tables 1 and 2), Gigaword (Table 3), and English CC-News (Table 4). One particular advantage of the KAIROS schema representation is that it is language-agnostic: given a schema automatically induced, say, over English documents, we can apply it to match with and rank documents in other languages. Thus, in addition to English, we also evaluate coverage for Russian (Table 5) and Chinese (Table 6) newswire documents.

We observe that the initial set of 82 schemas covers a substantial portion of the newswire corpora. Even in the most extreme case of long documents ($\mathrm{N}_{\text{events}} \geq 10$) and high required coverage (Cov@0.9, meaning that 90% of documents' events need to match the events of at least one schema), around 20-25% of each document set is covered by our 82 schemas. Extending the schema library and adding additional 150 schemas boosts this result to 27-31%. The results are similar across all languages considered. This suggests feasibility of expanding schema-labeled corpora (such as the LDC corpus) to a much larger newswire corpora and demonstrates cross-lingual potential of schemas.

6 Conclusions

In this work, we developed a semi-automatic approach to the creation of structured scripts/schemas. The automatic portion of our pipeline is rooted in a new method combining recent advances in script induction: an ARM based method which finds interesting subsequences, and a causal inference based scoring metric for filtering out and fusing together these interesting subsequences.

The manual portion of our pipeline is made possible through our newly developed annotation tool: SchemaBlocks, a block based interface developed to make annotation of schema (or schema-like) knowledge structures intuitive and easy, even for non-expert annotators.

We show how SchemaBlocks allows annotators to both create schemas starting from an automatically learned skeleton, and create full schemas completely from scratch. We release both these annotated schemas and the SchemaBlocks interface to the community to facilitate further efforts in what is traditionally an interminable pain for all looking to build robust AI systems: the annotation of robust commonsense knowledge structures.

	82 schemas			82 + 150 schemas		
N_{events}	Cov@0.5	Cov@0.7	Cov@0.9	Cov@0.5	Cov@0.7	Cov@0.9
$[1; 5)$	0.886	0.612	0.561	0.983 (+11%)	0.734 (+20%)	0.670 (+19%)
$[5; 10)$	0.778	0.465	0.335	0.921 (+18%)	0.586 (+26%)	0.408 (+22%)
$[10; \infty)$	0.688	0.387	0.250	0.839 (+22%)	0.492 (+27%)	0.306 (+22%)
$[1; \infty)$	0.713	0.414	0.287	0.858 (+20%)	0.523 (+26%)	0.349 (+21%)

Table 5: CC-News (Russian) dataset coverage, using IE extracted events.

	82 schemas			82 + 150 schemas		
N_{events}	Cov@0.5	Cov@0.7	Cov@0.9	Cov@0.5	Cov@0.7	Cov@0.9
$[1; 5)$	0.875	0.589	0.528	0.981 (+12%)	0.718 (+22%)	0.639 (+21%)
$[5; 10)$	0.776	0.460	0.314	0.924 (+19%)	0.582 (+27%)	0.387 (+23%)
$[10; \infty)$	0.699	0.408	0.251	0.877 (+25%)	0.531 (+30%)	0.314 (+25%)
$[1; \infty)$	0.713	0.422	0.271	0.885 (+24%)	0.545 (+29%)	0.337 (+24%)

Table 6: CC-News (Chinese) dataset coverage, using IE extracted events.

References

Valerie Abbott, John B Black, and Edward E Smith. 1985. The representation of scripts in memory. *Journal of memory and language*, 24(2):179–199.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90.

Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *Proceedings of EMNLP*.

Anton Belyy and Benjamin Van Durme. 2020. Script induction as association rule mining. In *Proceedings of the First Joint Workshop on Narrative Understanding, Storylines, and Events*, pages 55–62, Online. Association for Computational Linguistics.

Gordon H Bower, John B Black, and Terrence J Turner. 1979. Scripts in memory for text. *Cognitive psychology*, 11(2):177–220.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the Association for Computational Linguistics (ACL)*, Hawaii, USA.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Association for Computational Linguistics (ACL)*, Singapore.

Nathanael Chambers and Daniel Jurafsky. 2010. A database of narrative schemas. In *LREC*.

Gerald DeJong. 1983. Acquiring schemata through understanding and generalizing plans. In *IJCAI*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, pages 837–840. Lisbon.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34.

Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. *ACM sigmod record*, 29(2):1–12.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Marvin Minsky. 1974. A framework for representing knowledge. MIT Laboratory Memo 306.

Raymond Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 681–687.

Erik T Mueller. 1999. A database and lexicon of scripts for thoughttreasure.

Sebastian Nagel. 2016. Cc-news.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988. Association for Computational Linguistics.

Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67.

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

Roger C Schank and Robert P Abelson. 1977. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ere: annotation of entities, relations, and events. In *Proceedings of the the 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 89–98.

Lilian Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2017. Inducing script structure from crowdsourced event descriptions via semi-supervised clustering. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 1–11, Valencia, Spain. Association for Computational Linguistics.

Lilian D. A. Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2016. A crowdsourced database of event sequence descriptions for the acquisition of high-quality script knowledge. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3494–3501, Portorož, Slovenia. European Language Resources Association (ELRA).

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

Patrick Xia, Guanghui Qin, Siddharth Vashishtha, Yunmo Chen, Tongfei Chen, Chandler May, Craig Harman, Kyle Rawlins, Aaron Steven White, and Benjamin Van Durme. 2021. LOME: large ontology multilingual extraction. *CoRR*, abs/2101.12175.