A Database of Narrative Schemas

Nathanael Chambers, Dan Jurafsky

Department of Computer Science
Stanford University
natec@stanford.edu, jurafsky@stanford.edu

Abstract

This paper describes a new language resource of events and semantic roles that characterize real-world situations. *Narrative schemas* contain sets of related events (*edit* and *publish*), a temporal ordering of the events (edit *before* publish), and the semantic roles of the participants (*authors* publish *books*). This type of world knowledge was central to early research in natural language understanding. *Scripts* were one of the main formalisms, representing common sequences of events that occur in the world. Unfortunately, most of this knowledge was hand-coded and time consuming to create. Current machine learning techniques, as well as a new approach to learning through coreference chains, has allowed us to automatically extract rich event structure from open domain text in the form of narrative schemas. The narrative schema resource described in this paper contains approximately 5000 unique events combined into schemas of varying sizes. We describe the resource, how it is learned, and a new evaluation of the coverage of these schemas over unseen documents.

1. Introduction

Natural Language Understanding research in the 1970s and early 1980s heavily depended on hand-coded knowledge structures for interpretation. Scripts (Schank and Abelson, 1977) were one of the main formalisms, representing common sequences of events that occur in the world. The most famous example, the restaurant script, contained events such as entering a restaurant, sitting down, ordering food, eating, etc. These sequences were important for reasoning tasks such as inference (someone who is eating must have ordered) and coreference (the person eating is the one who sat down). Unfortunately, scripts were too expensive to create and too brittle to be used across domains.

A new representation, *narrative schemas*, has recently been proposed for characterizing script-like knowledge (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009). One brief example is shown here:

Events	Roles
A write B A edit B A publish B C distribute B C sell B	A = *Author* B = *Book* C = *Company*

This schema characterizes a book publishing domain, yet the algorithm to learn this schema does not use topic-sorted documents or labeled text. It learns domain-specific narrative relations by following coreferring entity chains within documents. By observing events that predicate the same entity, the algorithm can learn narrative structure.

There are two steps to learning our new database. The first learns sets of related events and their participants (as in the diagram above), and the second learns the temporal order of the events. The former uses unsupervised learning and the latter requires temporally labeled data. We describe both processes briefly in section 3.

The narrative schema resource described in this paper contains approximately 5000 unique events combined into schemas of varying sizes. Each schema fully specifies the arguments seen filling the event roles, and a separate temporal ordering of the events is provided. Section 4. presents several schema examples from this database, illustrating their application to diverse domains. In addition, we describe past comparisons against current knowledgebases like FrameNet (Baker et al., 1998), as well as a new evaluation of the schema database on unseen documents. Finally, we provide the database for public access[1].

2. Application of Schemas

Learning event *structure* is novel to most current learning research. While there is significant work in fact-based, relational, and ontology learning, most algorithms learn atomic relations. Narrative schemas are unique because they describe structures that often cross sentence boundaries.

Schemas thus address an important task in NLP: document-level understanding. Several document-level tasks may be assisted by this type of structured knowledge:

1. **Information Extraction**: Most current work in information extraction is relational, extracting single events or arguments associated with a short query. Understanding related events to a user's query may help extract arguments not available to current techniques.

2. **Coreference**: Bean and Riloff (2004) showed how *caseframes* can improve coreference. A caseframe is a pair of predicates seen during training that share the same argument string (X stated, X added). Schemas build upon this idea by learning semantic roles over a *set* of predicates. The shared connections between arguments may be able to assist coreference decisions in a similar fashion. For example, the subject of *writes* is defined as the subject of *edits* and *publish* in our above schema example. Reference ambiguities involving these verbs may be partly solved just by knowing the correct schema.

[1] http://cs.stanford.edu/people/nc/schemas

3. **Summarization**: Recent work on summarization has shown how event-focused approaches can outperform token-based methods (Azzam et al., 1999; Filatova and Hatzivassiloglou, 2004). These approaches model individual event mentions in texts and score how important they are to the desired topic. Having *structured* event knowledge may help by providing extra knowledge to further connect a document's predicates.

4. **Inference**: Logical inference and deeper reasoning require temporal and causal relations to identify semantic connections between a document's sentences. Schemas are not the complete solution, but are a step toward learning these important relations. Reasoning over their temporal order may also be useful in learning causality. Paul et. al (2009) has recently used coreference patterns to learn reciprocal relationships. We believe structured schemas can help inform similar approaches.

3. Learning Narrative Schemas

Narrative schemas contain events and participants, but also a temporal ordering. We first describe how the schemas themselves are learned, and then describe our temporal ordering approach.

3.1. Events and their Participants

The key insight to learning schemas (not just sets of synonyms) is that entities help to identify discourse-level semantic connections between predicates. Chambers and Jurafsky (2008) called this the *Narrative Coherence Assumption*: predicates sharing coreferring arguments are related by virtue of narrative discourse structure. A system may not have a semantic representation of two verbs, but if *John* is the subject of both, they are likely related.

Using this connection between entities and predicates, the learning algorithm builds a graph of verb/grammatical-dependency nodes whose edges are functions of how often the nodes had coreferring arguments. Let this sentence serve as an example:

John threw the ball until it hit William.

The ball is coreferent with *it*, so we increase the count for the pair: (⟨throw,object⟩ , ⟨hit,subject⟩). All such pairs in each document are counted using the following procedure:

1. Parse all sentences into dependency graphs.
2. Run coreference over the document[2].
3. Count all pairs of verbs and dependencies (e.g. subject, object) that are filled by coreferring entities.
4. Record with each pair the head word of the shared argument.

After counting all pairs, we create a verb/dependency graph whose edge weights are the pointwise mutual information (PMI) scores between the verb/dependency nodes. Each edge also records the counts of all argument heads that were observed coreferring with the two particular verbs' dependencies.

Given this graph, we cluster verbs using the PMI edge weights and the edges' argument head counts. The similarity function between any two verbs aligns their subjects and objects so that argument overlap and PMI scores are maximized. By aligning arguments in the similarity score itself, the clustered schemas automatically learn constraints for their semantic roles (e.g. the object of *push* and the subject of *fall* tend to be *people*). Details about the similarity function and the full clustering algorithm are described in Chambers and Jurafsky (2009).

3.2. Temporal Relations

The Timebank Corpus (Pustejovsky et al., 2003) sparked a renewed interest in temporal classification by providing a labeled dataset on which machine learning algorithms could be evaluated. The corpus labels events (verbs, nominals, etc.) and binary relations between events representing temporal order (e.g. before, after, includes, simultaneous, etc.). Most work builds classifiers for these relations with standard feature-based machine learning approaches (Mani et al., 2006; Chambers et al., 2007), and an official TempEval contest has been held over the past two years (Verhagen et al., 2007; Verhagen et al., 2009).

While the TempEval and Timebank tasks are document-specific decisions (e.g. is the *arrest* event before the *convict* event in this document?), the order of a schema's events needs to be a generalized ordering (e.g. does *arrest* tend to occur before *convict* in most documents?). Our approach aggregates document-specific ordering decisions across the corpus to estimate a generalized order.

Our algorithm counts the number of times two events are locally classified as *before* or *after*. If the number of *before* relations classified in the corpus is significantly greater than *after*, a schema can order those two events. The classifier used for the local decisions is described in detail in Chambers and Jurafsky (2007). It is a three-way classifier (before, after and other) trained on Timebank's labels using a variety of features based on lexical items, tense, aspect, part of speech tags, and syntactic constraints.

We ran this classifier over all pairs of verbs in the New York Times section of the Gigaword Corpus (Graff, 2002). Each *before* and *after* classification is counted across all parsed documents. The resulting counts create a database of verb pairs representing how often the first verb occurs *before* the second. For instance, the verbs *arrest* and *convict* appear twice in the database as follows:

arrest	convict	684
convict	arrest	22

This indicates that arrest was classified 684 times as *before* convict, and convict was classified only 22 times as *before* arrest. Given pairs with an obvious preference to one ordering, the order can be superimposed onto a schema that contains both verbs. Our language resource with these ordering counts is described next.

[2]We use OpenNLP for coreference.

4. Format of the Database

The database[3] consists of two parts: unordered narrative schemas, and temporal orderings between events.

4.1. Schema Database

Figure 1 shows seven narrative schema examples, hand selected from our database to represent a variety of topics. As can be seen, a schema contains a set of events (unordered) and a set of semantic roles that define which syntactic positions are filled by each role. A semantic role contains two lines, the first lists the syntactic position the role fills for each verb, and the second lists the argument head words. For example, 'raise-s' indicates the subject of the verb raise and 'slash-o' is the object of slash. Only the top arguments for each role are listed to conserve space.

The newspaper data from which the schema database is learned is not domain specific, but contains a variety of topics within the corpus. These example schemas illustrate the diversity of situations that are captured by the entity-based learning algorithm. One of the main advantages of the algorithm is that it learns these diverse schemas without presorted documents.

We released four databases of varying schema sizes: 6, 8, 10 and 12 events. The clustering algorithm stops growing the clusters at each size, and we have made available the different sizes for download.

4.2. Temporal Database

The temporal database is a set of lemmatized verb pairs with the number of times the pair was classified as *before*, as described in section 3.2. All verb pairs that are seen together in a Gigaword document are included if they were classified as before or after. Below are a few more examples of counts with the verb *arrest*:

arrest	sentence	303
sentence	arrest	22
arrest	acquit	24
acquit	arrest	5
arrest	plead	132
plead	arrest	1

Users can create various confidence scores depending on their applications. From these examples, most applications can conclude that *plead*, *sentence* and *acquit* all occur after *arrest*.

However, more progress is needed in the field of temporal classication as there are also spurious orderings that are repeatedly classified with high confidence. A few examples are shown here for completeness:

arrest	murder	54
murder	arrest	0
publish	write	69
write	publish	22

A publishing event typically follows the writing, but here we observe the opposite in our data, publish occurring before write 69 times. Some of these counts are correct in that writers who publish tend to write again (and those who murder have often been arrested in the past for a separate incident). Thus, publishing before writing is correct, albeit the later writing event concerns a new paper or book. It is difficult to distinguish these cases from raw counts alone, and these issues remain unsolved for future work.

[3] http://cs.stanford.edu/people/nc/schemas

5. Comparison to FrameNet

FrameNet (Baker et al., 1998) is a collection of *frames*, hand-built structures of events and the *frame elements* of the arguments that the events predicate. Frames describe particular situations in the world and thereby encode a similar type of knowledge as narrative schemas.

Previous work on narrative schemas studied the level of agreement with FrameNet (Chambers and Jurafsky, 2008). The top 20 learned schemas were hand aligned with FrameNet frames, and the level of verb overlap was evaluated. We found that 13 of the 20 schemas were encoded in FrameNet, and 7 were novel situations. Of the 13 aligned schemas, 67% of their verbs were labeled in FrameNet.

These results also found that the learned schema arguments filled the correct frame elements in the aligned frames with 72% precision. Other metrics were presented, but we repeat these particular results to illustrate the extent to which the top schemas in our learned database agree with a human-created database. The results show areas of overlap, but also novel schemas that have not yet been encoded in FrameNet. While these numbers largely address schema precision, the next section describes a new evaluation to measure recall of real-world newspaper articles.

6. Evaluation

The previous FrameNet evaluation showed agreement between our learned schemas and a human created database. However, a concern for any automatically acquired knowledgebase is its relevance to and coverage over new data. We want to measure the extent to which this paper's narrative schema database can explain the events described in newspaper articles.

6.1. Event Coverage

This evaluation measures the amount of overlap between the schema database and the naturally occurring sets of events in documents. Newspaper articles describe sequences of events and often contain several narrative schemas. Chambers and Jurafsky (2008) defined a document's central entity as the *protagonist* and manually labeled a set of documents for the main narrative chain involving only that actor. While that work evaluated these narrative instances for an event prediction task, we make use of the same data to measure how much of the chain is covered by the narrative schemas in our database. The database contains generalized schemas, and so we do not expect all chains to be covered as documents describe very specific narrative instances, however, we would like to know the extent of overlap with the database.

6.1.1. Data

We use the narrative chain test set as presented in Chambers and Jurafsky (2008). The test set includes 69 randomly

Medical (Viral)
Events: infect transmit cause spread contract carry kill detect
Role 1: { transmit-o kill-s infect-s contract-o carry-o cause-s detect-o spread-s / virus disease bacteria cancer toxoplasma strain fire parasite bacterium }
Role 2: { detect-s kill-o spread-o carry-s transmit-s infect-o contract-s cause-o / mosquito aids virus tick catastrophe disease aboard mite others bacteria }

Financial
Verbs: cut raise reduce lower increase boost trim slash
Role 1: { raise-s cut-s increase-s reduce-s slash-s trim-s boost-s lower-s / company fed bank government rates bundesbank plan bill }
Role 2: { slash-o trim-o boost-o lower-o raise-o reduce-o cut-o increase-o / rates rate price tax risk dividend stake estimate rating }

Legal
Events: prohibit violate require allow bar forbid ban permit
Role 1: { violate-o forbid-s ban-s bar-s require-s allow-s prohibit-s permit-s / law bill rule amendment act treaty constitution laws policy government }
Role 2: { ban-o bar-o require-o permit-o forbid-o allow-o violate-s prohibit-o / company microsoft government iraq state use group banks student member }

Criminal
Events: arrest raid search detain found charge seize identify
Role 1: { detain-s found-s seize-s raid-s search-s charge-s identify-s arrest-s / police agent authorities officer official investigator fbi troops soldier }
Role 2: { identify-o charge-o arrest-o raid-o seize-o detain-o found-o search-o / suspect police padilla officer yates driver government member citizen }

Authorship
Events: publish sell write translate distribute edit produce read
Role 1: { translate-s produce-s sell-s write-s distribute-s publish-s read-s edit-s / company author group year microsoft magazine my time firm writer government }
Role 2: { produce-o edit-o sell-o translate-o publish-o read-o write-o distribute-o / book report novel article story letter magazine film letters movie show }

Sports
Events: outscore outshot outrebounded beat score outplay trail tie
Role 1: { beat-s tie-s outplay-s score-s outrebounded-s outscore-s outshot-s trail-s / king maverick sonics ranger lakers bruin angel dodger mets yankee }
Role 2: { beat-o tie-o score-o outrebounded-o outscore-o outshot-o outplay-o trail-o / knicks king net maverick lakers state point patriot yankee jet celtic }

Legislative
Events: veto pass oppose approve sign support require sponsor
Role 1: { sign-s oppose-s approve-s require-o veto-s sponsor-s support-s pass-s / clinton bill house bush president state congress voter governor group senate }
Role 2: { sponsor-o require-s veto-o pass-o approve-o oppose-o support-o sign-o / bill legislation measure law amendment plan treaty agreement resolution act proposal }

Figure 1: Narrative schemas: examples were hand selected from the database to illustrate the diversity of learned narratives.

selected documents from the 2001 NYT portion of the Gigaword Corpus (Graff, 2002). The most repeated entity in each document is labeled as the *protagonist*, and all verbs of which he/she is the subject, object or preposition phrase are hand extracted to represent the narrative chain (note that this is not a full schema since it extracts a single entity and only the grammatical positions that it fills). Verbs with low IDF scores[4] are ignored. This data is also available online[5].

[4] 0.9 threshold, removing any below it
[5] http://cs.stanford.edu/people/nc/data/chains

6.1.2. Overlap Metric

A narrative chain is a set of connected events, so we want to measure the connectivity between these same events in our schema database. An event in a narrative chain is a predicate p (e.g. arrest) and a syntactic position d (e.g. subject or object). In a test document, there are edges between all events involving the protagonist, so the events are *fully connected* by definition.

In the schema database, an edge exists between two events if there exists some narrative schema such that one of its roles contains both events. We define coverage as a graph

connectivity problem. Given a set of events from a document, how connected is the set in our database?

There are several options for measuring connectivity in graph theory. We adopt the *largest connected component* approach for this analysis. A connected component is a subset of vertices such that a path exists between every vertex in the subset. For each test document's connected events (the narrative chain), we compute the largest connected component in our database's edges and return the number of vertices in the component. For a test chain of length n, returning n indicates full coverage by the database (there exists a schema such that all n events are members). Returning zero means that none of the events in the test chain appear together in a single schema.

As a concrete example, consider a newspaper article describing someone who has written and published a book, using the verbs shown on page 1. Considering entity *B* as the protagonist of this article, the test set will include the narrative chain as follows: *write-obj, edit-obj, publish-obj, distribute-obj, sell-obj*. These object positions are the nodes of the graph, and we consider it fully connected. This evaluation finds the single narrative schema with the largest connected component including these verbs. If there is a schema with four of the five verbs connected by their objects, the score is $4/5 = 80\%$.

6.2. Results

For each test document, the size of the largest connected component is computed and the percentage of events covered by that component is returned. For instance, if a document has 10 events in its narrative chain and the database contains a narrative schema that includes 7 of the events, 70% is our coverage over that document. We macro-average these percents across all 69 documents' 740 events. The final percent coverage for our schema database using the database with schemas of size 12 is 34.0%.

7. Discussion

The database of narrative schemas connect approximately one out of every three events in newspaper articles. In other words, one third of a document's events are part of a single self-contained narrative schema. Since schemas characterize general sequences of events, it is expected that a significant portion of an article's events would occur outside a single schema, or the story would not be newsworthy for conveying new information. Chambers and Jurafsky (2008) found that only 3.5% of the events were completely unconnected to other events in the space of seen event pairs. This suggests that the two thirds of news events are connected, but not prevelant enough to draw generalizations in the form of a narrative schema and may be situation-specific. How many of these can still be learned through further advances remains for future work.

We hope this new database will assist and encourage work on event semantics and document-level understanding tasks, as well as movitave new approaches to learning structured knowledge.

8. Acknowledgements

This work was supported by the National Science Foundation IIS-0811974, and the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the AFRL. Thanks also to the anonymous reviewers for very helpful suggestions.

9. References

Saliha Azzam, Kevin Humphreys, and Robert Gaizauskas. 1999. Using coreference chains for text summarization. In *ACL Workshop on Coreference and its Applications*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In Christian Boitet and Pete Whitelock, editors, *ACL-98*, pages 86–90, San Francisco, California. Morgan Kaufmann Publishers.

David Bean and Ellen Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *North American Chapter of the Association for Computational Linguistics*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08*, Hawaii, USA.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of ACL-09*, Singapore.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of ACL-07*, Prague, Czech Republic.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

David Graff. 2002. English Gigaword. *Linguistic Data Consortium*.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of ACL-06*, July.

Michael Paul, Roxana Girju, and Chen Li. 2009. Mining the web for reciprocal relationships. In *Conference on Computational Natural Language Learning*.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, David Day, Lisa Ferro, Robert Gaizauskas, Marcia Lazo, Andrea Setzer, and Beth Sundheim. 2003. The timebank corpus. *Corpus Linguistics*, pages 647–656.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, plans, goals and understanding*. Lawrence Erlbaum.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of SemEval-2007*, pages 75–80.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Jessica Moszkowicz, and James Pustejovsky. 2009. The tempeval challenge: identifying temporal relations in text. In *Language Resources and Evaluation*.