Inducing Event Types and Roles in Reverse: Using Function to Discover Theme

Natalie Ahn
University of California, Berkeley
natalieahn@berkeley.edu

Abstract

With growing interest in automated event extraction, there is an increasing need to overcome the labor costs of hand-written event templates, entity lists, and annotated corpora. In the last few years, more inductive approaches have emerged, seeking to discover unknown event types and roles in raw text. The main recent efforts use probabilistic generative models, as in topic modeling, which are formally concise but do not always yield stable or easily interpretable results. We argue that event schema induction can benefit from greater structure in the process and in linguistic features that distinguish words' functions and themes. To maximize our use of limited data, we reverse the typical schema induction steps and introduce new similarity measures, building an intuitive process for inducing the structure of unknown events.

1 Introduction

Automated event extraction is mainly used in a few areas of high interest and resource investment, especially conflict and biomedical research. Yet there is growing interest in applying event extraction to new languages and substantive domains. Identifying meaningful representations of who did what to whom can enable us not only to study how known topics are described in pre-categorized texts, but to use unlabeled records to discover what has happened in the world that we don't yet know how to label, or disagree about how to define.

Event extraction is a complex task, combining multiple subtasks that continue to be studied in their own right. To determine that an election occurred and who voted, won, or lost, we must identify segments of text that mention the topic of electing public officials, and determine which entities are attributed certain roles. Finding that a document is about elections is not enough to determine who attained power and which citizens they represent. Finding only that someone won a vote, without thematic context, is not enough to know whether they won political power, a corporate board decision, or figurative social approval.

There is growing interest in finding new ways to induce event frames and patterns linking entities to event roles. This paper builds on that emerging body of work, while introducing new ideas about event narratives and their components. Our contributions involve reversing the typical schema induction process and combining multiple measures of word similarity, to dissect words' functional relatedness and incorporate hierarchical information from public WordNet and Wikipedia resources.

This paper proceeds as follows. Section 2 discusses prior work and defines the terms we use. Section 3 explains our methodology, including our re-ordered process and steps for inducing event roles and event types. Section 4 presents evaluations using the MUC-4 data set, with comparison to other work, and Section 5 offers discussion.

2 Related Work

Early automation of event extraction relied on rule-based pattern matching, using hand-written templates (Chinchor et al., 1993; Schrodt et al., 1994). Modern efforts have focused on supervised machine learning, using annotated corpora for training data, again with pre-defined event types and roles (Miyao et al., 2008; Bjorne and Salakoski, 2011; Bunescu and Mooney, 2004).

Semi-supervised approaches have been used to identify relations between pairs of entities, using seed pairs with known relations (Brin, 1998; Culotta and Sorensen, 2004; Mintz et al., 2009).

Open IE systems (Banko et al., 2007; Angeli et al., 2015) extract general relational patterns between entity pairs, based on domain-independent patterns or heuristics. Similar efforts have emerged to extract more complex event frames by bootstrapping from seed event patterns (Huang and Riloff, 2012; Surdeanu et al., 2006; Yangarber et al., 2000; Patwardhan and Riloff, 2007).

There has been growing work over the past decade on purely unsupervised role induction. Most of these efforts begin with a set of documents known to cover a type of event or domain, then cluster verb arguments to determine each verb's role slots within that domain (Filatova et al., 2006; Sekine, 2006). These approaches typically learn verb-specific roles, rather than multi-verb event schemas. Other recent work models multiple verb roles in combination, in various forms of subject-verb-object relational triples (O'Connor et al., 2013; Balasubramanian et al., 2013).

In the last few years, several important efforts have broken new ground with more comprehensive event schema induction. These efforts discover new event types in unfiltered text, and identify verb argument positions associated with overall event roles. Chambers and Jurafsky (2011) used a pipeline approach, first discovering related event patterns, then clustering arguments into event slots. For the first step, they tested both LDA and agglomerative clustering, based on event terms' co-occurrence. They used the MUC-4 data set, but relied on an additional external corpus for role induction, due to data limitations when clustering roles separately in each event category.

Chambers (2013), Cheung et al (2013), Nguyen et al (2015), and Sha et al (2016) all use probabilistic generative models that jointly model the assignment of predicates to event schemas and arguments to event roles. Chambers uses an entity-driven model, linking coreferring arguments to the same event role. Cheung et al focus on event clauses and model transitions between them, using a pair of HMMs. Nguyen et al (2015) add phrase modifiers to argument similarity scores, and Sha et al (2016) add a normalized cut approach to maximize intra-class similarity within slots.

2.1 Problem Setup and Terminology

Our goal is to learn a set of meaningful events and participant roles from a body of text. For instance, given a collection of news reports, we may want to identify that some of them are about elections, others are about crime, etc. We also want to learn that an election involves voters, candidates, polling sites, and the office to be won.

As we identify meaningful roles, we also want to learn how to extract particular instances, by identifying textual positions that refer to each role. The subjects of the verbs *vote* and *elect* are likely to be voters, while the direct object of *elect* or the subject of *campaign* is likely to be a candidate.

We use the term "event type" to refer to a thematic event category (e.g. *election*), which may be described using a variety of related verbs. We use "role" to refer to the semantic role of an event participant (e.g. *candidate*), and "event schema" to refer to the set of an event type's roles. We use "entity" to refer to a specific actor or object, which might be described by multiple coreferences.

To distinguish specific words, we use "predicate" to refer to a verbal or nominal event predicate (e.g. *campaign*), and "argument" to refer to a syntactic argument of a predicate; "argument term" refers to the argument's head word. We use "argument position" to refer to the combination of a predicate and a dependency relation in which an argument might appear (e.g. *elect:dobj*). We cluster these positions into "slots" which map to event roles. Figure 1 shows an example.

Event type: *election*
Schema slots: {*voter*, *candidate*, ...}
Slot positions: *voter*:{*elect:subj, vote:subj*}
candidate:{*elect:dobj, win:subj*}

Mention 1		Mention 2	
predicate: *elect*		predicates: *vote, win*	
position:	arg term:	position:	arg term:
elect:subj	*Berliners*	*vote:subj*	*Canadians*
elect:dobj	*Merkel*	*win:subj*	*Trudeau*

Figure 1: Example of an event schema.

3 Process Overview

The traditional event extraction process generally involves two parts: 1) identifying segments of text with action terms or phrases that relate to a particular event type, and 2) identifying entities in relevant argument positions that fill the event type's roles. The first task represents a text classification or topic identification problem at the level of the document. The second task (semantic role labeling) is more difficult, since it involves complex word-level assignments and relationships, requiring a lot of data and features to capture all possible patterns that link an argument to its correct role.

If we attempt to learn semantic roles only after separating documents by event type, we have much less data to work with in identifying event-specific roles. To overcome this limitation, Chambers and Jurafsky (2011) augmented their role induction stage with a larger external corpus, but our goal is to induce roles using only the documents contained in the MUC-4 dataset. Many event types share similar roles, and some argument positions will signal the same role in multiple event types (e.g. *kill:dobj*, signaling the victim of a bombing, murder, or other attack).

We can make much greater use of limited data by learning general semantic roles from the whole corpus. Learning general roles first also helps us identify event types, by segmenting the text into narrative sequences with coherent argument roles, then identifying trigger words that represent these event narratives. Finally, we construct event schemas by refining the general roles based on argument frequencies in specific event contexts.

3.1 Inducing General Argument Roles

We begin with dependency parsing and coreference resolution, using the Stanford CoreNLP toolkit (Manning et al., 2014), then identify all predicates' arguments as candidates for semantic roles. Predicates are any verb or any noun under the WordNet synset for "event"; arguments are any of their syntactic dependents. We collect arguments by their argument position, defined as the argument's predicate head paired with its dependency relation, e.g. *kill:dobj*. All arguments with the dependency relation *dobj* to the verb *kill* are assigned to the same slot. Similar to Chambers and Jurafsky (2011) and Cheung et al (2013), we separate slots by high-level entity type: 1) Person or Organization, 2) Location, 3) Physical Object, or 4) Other. The position *take:dobj:[person]* is clustered separately from *take:dobj:[object]*.

We cluster argument positions using two similarity scores: one for the functional position itself (i.e. the predicate dependency relation), and one for the argument terms that appear in that position throughout the text. We begin with Chambers and Jurafsky's (2011) measures of argument similarity: the cosine similarity between vectors of argument terms, and the cosine similarity between vectors of other positions that share coreferring arguments. To build on Chambers and Jurafsky's work and show a meaningful comparison, we use their method for combining these two scores, taking the maximum if either score is above 0.7 (which they optimized on the MUC-4 training set) and backing off to the average between the two otherwise. We also add noun phrase modifiers to the argument term vector, following Nguyen et al (2015).

3.1.1 Adding Argument Hypernyms

Our contributions to this stage of the process are to add two major sources of information about argument positions, that shed greater light on the similarity between their semantic roles. First, we add argument hypernyms. Many entity terms appear infrequently in the corpus, such as names of specific people, locations, or precise objects. Yet categorical groupings that fall between the word itself and its high level entity type may be important.

For instance, the two positions *attack:iobj-on:[object]* and *attack:iobj-with:[object]* have the same high-level entity type. But the first is more likely to contain buildings, while the second is more likely to contain weapons. Even some person types share more hypernyms than others, such as the terms "attacker" and "kidnapper," which share the hypernym "wrongdoer". Using the full hypernym chain enables us to avoid making arbitrary decisions about how much granularity to use in subdividing entity types. Figure 2 shows an example of top hypernym counts from the data set:

pos_1 = *kill:dobj*	pos_2 = *die:subj*
H_{pos_1}={*person*: 387, *group*: 155, *worker*: 50, *leader*: 45...}	H_{pos_2}={*person: 70*, *group*: 35, *unit*: 11, *worker*: 10...}

Figure 2: Top hypernyms in two similar positions.

To label argument hypernyms, we look up the argument head word in WordNet. If more than one synset is given, we select the synset whose other lemmas have the most similar word embeddings to the target word, using Word2Vec cosine similarity. This is a simple approach to select one synset for all mentions of the same term throughout the corpus, rather than performing word sense disambiguation on each mention, since existing methods for WSD still rarely beat selecting the first WordNet synset for all mentions (Raganato et al., 2017).

If a term does not appear in WordNet, we look it up in Wikipedia. We use Wikipedia's API to query for a page with a title exactly matching the argument phrase or head word, and if not found, use the search query for a page with a partially matching title. For each page returned, we retrieve the description, label, alias, and page categories, and look up the head word of the first noun phrase in each, until we find a match in WordNet. To make sure we should use the synset, we again compare the word embeddings of the synset's lemmas and the target term in our corpus. We keep the synset from Wikipedia if its lemmas have a higher average Word2Vec similarity to the target term than the average for all other nouns in the corpus.

This process works well at finding person and place names that don't appear in WordNet. It works less well for common nouns, which don't usually have their own Wikipedia page, but those terms are overwhelmingly found in WordNet already. In the MUC-4 training set, we found about 95% of noun phrase head words in WordNet. Our Wikipedia search found a suitable synset that met the word embedding check for close to 30% of remaining noun phrases, which slightly improved our evaluation scores over using WordNet alone.

We then construct a vector of hypernym counts H_{pos_i} for each argument position pos_i, as in Figure 2 above. For all arguments in pos_i throughout the corpus, we take their assigned WordNet synsets and count all hypernyms in their full hypernym chains, except the three most general categories of "entity", "physical entity", or "abstraction". We take the cosine similarity between these hypernym vectors and multiply it by the Chambers and Jurafsky (2011) similarity score for argument terms and coreferring positions.

$$\begin{aligned} hyp\_sim(pos_i, pos_j) &= cosine(H_{pos_i}, H_{pos_j}) \\ arg\_sim(pos_i, pos_j) &= CJ\_sim(pos_i, pos_j) \\ &\quad \times hyp\_sim(pos_i, pos_j) \end{aligned}$$

3.1.2 Adding Predicate Functionality

Second, we add a new measure of functional similarity between two predicate dependency relations (i.e. the syntactic base of the argument position, as opposed to the terms that fill the position). Again, an argument position is defined by a predicate and a dependency relation, e.g. *kill:dobj*. There are two parts to our functional similarity measure: the similarity between the predicates themselves, and whether the positions share the same dependency relation to their respective predicates.

Consider a victim-type role, which might appear in the positions *kill:dobj*, *murder:dobj*, or *die:subj*. The verbs *kill* and *murder* are functionally similar; they both have human subjects and direct objects (and often instruments after "with"). But *die*, while thematically related, is functionally different: it is intransitive and has no direct object. The *victim* role appears in the same dependency relation (*dobj*) to *kill* and *murder*, but in a different relation (*subj*) to *die*. If two positions represent the same semantic role, they should either fill the same dependency relation to functionally similar predicates, or have different relations to predicates that tend to have different argument structures.

For each predicate, we assemble a count vector of all of its arguments' dependency relations in the corpus, and take the cosine similarity between two predicates' dependency relation count vectors. We multiply this dependent similarity score by the cosine similarity of the predicates' word embeddings, to confirm that the two verbs are used in similar ways throughout the corpus.

Then for each pair of argument positions pos_i = $pred_i$:dep_i and pos_j = $pred_j$:dep_j, we look at whether they have the same dep. If they do, we use the functional similarity score for their two predicates $pred\_sim(pred_i, pred_j)$ as the similarity score for the two argument positions. If the positions have different $deps$, we use 1 - $pred\_sim(pred_i, pred_j)$, so that positions with different *deps* will only be merged if they're dependent on functionally different predicates.

We're more confident that this is a meaningful comparison of positions with the same *dep* than with different *deps*. So far, we would give *kill:dobj* and *die:subj* the same similarity score as any other non-matching dependents of *kill* and *die*. Instead, we'd like to infer which non-matching dependents of two functionally different predicates might fill similar roles. To do so, we weight the second case by the cosine similarity of the two positions' hypernym vectors. (This means we use hypernym similarity twice for positions with different $deps$, effectively squaring it in our final slot similarity score, which we consider reasonable given the greater uncertainty that they fill the same role.)

$$funct\_sim(pos_i, pos_j) = \begin{cases} pred\_sim(pred_i, pred_j) & \text{if } dep_i = dep_j \\ (1 - pred\_sim(pred_i, pred_j)) & \text{if } dep_i \neq dep_j \\ \quad \times hyp\_sim(pos_i, pos_j) & \end{cases}$$

3.1.3 Clustering Combined Scores

For two positions' overall similarity, we multiply the argument similarity and functional similarity scores, to ensure we merge positions that are related on both dimensions. We use agglomerative clustering with average linkage scores, and apply constraints against merging two positions that meet any of the following conditions. Versions of the first two were also used by Sha et al (2016) and Chambers and Jurafsky (2011), respectively:

1. ***Sentence co-occurrence*:** The positions appear in the same sentence for more than a minimal percentage of occurrences.

2. ***Functional incompatibility*:** The positions share the same predicate but different base dependency relations (e.g. *subj* vs. *dobj* or *dobj* vs. *iobj*). These pairs already have a functional similarity score of 0, but we allow indirect objects to merge if they have different prepositions, since *iobj-at* and *iobj-in* may both refer to a verb's location.

3. ***Non-overlapping hypernyms*:** The positions have a hypernym similarity score equal to 0, which only applies to functional positions with the high-level entity type "Other".

These constraints prevent highly dissimilar argument positions from being merged even as average similarities between clusters grow. We merge up to a maximum distance close to 1 (0.999), to merge as many compatible slots as possible. The resulting clusters have reasonable sizes (the largest usually had about 50 argument positions).

3.2 Segmenting Event Narratives

To leverage information from our first step to identify thematic event types, we add an intermediate step: chunking the text into potential event narratives. This relates to Cheung et al's (2013) modeling of event frame transitions between clauses.

The motivation for segmenting narrative sequences is to help us determine which verbs might be part of the same event descriptions, to cluster event triggers in our final stage. Other papers have clustered event predicates based on nearness in the text, using different sentence windows (Chambers and Jurafsky, 2011; Jiang et al., 2014). Chunking event narratives allows us to relate predicates in nearby sentences, when the text between them appears to be part of a continuous event report, without selecting an arbitrary window of how many words or sentences apart they can be.

Cheung et al used a stickiness parameter to encourage neighboring clauses to remain in the same event frame. We approach event segmentation from the other direction, assuming that neighboring text is part of the same event until it no longer can be, because it contains elements that have internally incoherent semantic roles. We segment by paragraph, since the MUC-4 corpus contains news reports, which have short paragraphs of one or two sentences usually referring to the same event.

Consider the following two segments, each of which contain the same number of sentences, predicates and arguments:

1. "***Insurgents*** *attacked a* ***village****.*
   Four people *were killed.*"

2. "***Insurgents*** *attacked a* ***village****.*
   An airport *was bombed.*"

In the first example, the insurgents are the only perpetrators, the village is the physical target of *attacked* and the four people the victims of *killed*. In the second example, the village is again the target of *attacked*, but there is a second physical target – the airport – of *bombed*. This suggests that the second example might contain two different events.

Our narrative segmentation is simple. If two neighboring paragraphs have non-coreferring arguments (i.e. different entities) in the same general semantic role, we assume that they are part of different event narratives, and split the document between those paragraphs. We consider the resulting sequences likely to be coherent narratives with internally consistent themes, and use them to cluster thematically related event predicates next.

3.3 Inducing Trigger Verbs for Event Types

In supervised or rule-based document classification, a common approach to identifying events is to search for "trigger" words, i.e. action terms that are highly representative of a specific type of event. For instance, verbs like *choose* or *win* might signal their arguments' roles in an election context, but those same verbs appear in other thematic contexts as well. The verbs *vote* or *elect*, or the nominal predicate *election*, are better indications that a document is actually about an election. Trigger words are often hand selected, which does not enable the discovery of new event types.

When inducing event types, other researchers have sought to assign all predicates in a corpus to event clusters, often using probabilistic distributions to allow more general verbs to appear in more than one event type. However, very general terms can still have much in common with thematically specific terms, so that including all of them can result in loosely associated clusters that may shift considerably with different algorithm parameters. Cheung et al (2013) included a "background" frame with a binary switching variable in their event sequences, so that some clauses may contain terms used in any context.

We focus instead on identifying only a limited number of highly eventful verbs that are likely to represent a particular type of event. We inspected mentions of events in the corpus, comparing more event-specific terms like "election" to more thematically general verbs like "take" or "see". We observed that event trigger words tend to appear in prominent syntactic positions like the root of a sentence, in both verb and nominal form (e.g. "attacked" and later "the attack"), and to often have theme-specific objects, while general verbs have a wider variety of argument terms.

Based on this review, we chose two criteria for triggers that also roughly parallel our approach to semantic roles, combining aspects of functional positions and argument terms. We did not test other ideas, so there may be room to add other features related to event-specific verbs in the future. Our criteria for event triggers are as follows:

1. ***Major functional positions***: We count the number of times that a form of the verb appears in the following positions: a) in a sentence's "root" dependency relation; b) as an object of a reporting verb in the position *report:iobj-that*; c) in nominal form with definite article as the subject of another verb; and d) in nominal form with definite article as the direct object of an auxiliary or control verb. We use lists of reporting verbs, auxiliary verbs, and control verbs from Wiktionary (a Wikimedia dictionary resource) to identify these major action positions, and exclude the enabling terms from being event triggers themselves, as well as the Wiktionary category for copulative verbs and WordNet synonyms of "occur" and "happen".

2. ***Argument concentration***: We calculate a verb's argument concentration using a type of ratio used in economics for industry firms. For each verb, we count how many of its arguments contain one of the verb's top 50% most frequent argument terms, and divide by the verb's total arguments. This gives us the percentage of the verb's arguments that are covered by its most repeated argument terms.

We apply these criteria to verb infinitives, including all mentions of the verb in conjugated or nominal form. For each verb in the corpus that appears in at least three of the major functional positions, we multiple the log number of mentions in major positions with the argument concentration ratio to get our potential event trigger score. We select all predicates with a score above a threshold (0.2, chosen by inspection to ensure enough meaningful candidate terms in the training set).

We cluster these trigger words to get event types, based on their proximity in the text and similarity of arguments. As discussed in section 3.2, we use the narrative sequences from the previous stage to identify term co-occurrence. For each pair of trigger verbs, we calculate how many times they appear in the same narrative sequence, as a percentage of their total mentions in the corpus. We multiple this co-occurrence score by the percentage overlap in the two verbs' argument term count vectors, and by the cosine similarity of the triggers' word embedding vectors.

As in the first stage, incorporating multiple criteria enables us to focus on words that perform prominent eventful functions in the text, in similar ways and in meaningful proximity to each other. We again cluster using average linkage scores, applying constraints so that two predicates will not be merged if they have no co-occurring mentions and no overlapping argument terms.

3.4 Event Role Extraction

After inducing general roles and event triggers, we are ready to extract specific mentions of events. We classify a narrative sequence as a mention of a specific event type if it contains at least one of the event type's trigger words. For event schema slots, we look for argument positions from the general roles that appear in event-specific narratives, and calculate the probability of argument terms falling into each slot within each event context. This allows us to refine the general roles into thematically relevant versions for each event type, without having to recluster argument positions within a much more limited set of event-related documents.

Our extraction rules follow those used by Chambers and Jurafsky (2011). First, if an argument in an event narrative has a predicate and dependency relation assigned to an event slot, and has the correct high-level entity type for that slot, we assign it to the corresponding event role. If an argument's functional position was not assigned to any learned slot, we see if the argument term has a high probability of falling into one of the learned slots in the given event context, and assign the argument to the corresponding role if it does.

Our argument hypernym vectors enable us to add a similar rule for the probability of certain hypernyms appearing in specific event slots. If an argument has a hypernym with a high probability of appearing in a learned slot, we assign the argument to the corresponding role. For instance, if we come across the name of a rebel group we haven't seen before, but Wikipedia identifies it as an insurgent group, we can assign it to the same slot that other insurgent groups were clustered into, when the group is mentioned in an event context in which insurgents usually fill one particular role.

We get our best results if we only cluster argument positions that appear in the corpus at least 10 times, because less frequent positions are unlikely to have enough data to end up in the right cluster. This restriction also makes the time complexity and memory usage more manageable, given that we're calculating multiple pairwise similarity scores between argument positions. Then during extraction, for arguments in rare unclustered positions, we use the term or hypernym slot probabilities to assign them to their most likely event role.

4 Evaluation

We used the same information extraction task and sought to match our evaluation settings to those used in the other event schema induction papers. The evaluation data set is from the Fourth Message Understanding Conference (MUC-4) (Sundheim, 1992), which contains 1300 documents for training, plus 200 documents for development and 200 documents for testing.

The documents contain English newswire articles about conflict events in Latin America, annotated with four types of events: Attack, Bombing, Kidnapping, and Arson. As in the other papers, we tested entity extraction for the four main template roles: *perpetrator* (combining both individuals and organizations), *human target* (i.e. victim), *physical target*, and *instrument*, and ignored entries marked "optional". For the final tests, we induced event schemas from all 1700 documents in training, development, and test sets, and report scores for the 200 documents in the test set.

4.1 Experiments: General Role Extraction

To evaluate our first stage, we present results for the best mapping of our general roles to the four MUC-4 template roles, combining like roles (e.g. all *perpetrator* roles) across the four event types. To isolate the analysis of our new predicate structure and hypernym similarity measurements, we use this stage to compare our contributions to previous measures of argument similarity. We apply our general roles to test documents labeled with at least one of the four MUC-4 event templates, and show our results alongside Chambers' (2013) and Cheung et al's (2013) scores assuming perfect document classification. Since we induced corpus-wide roles using only the documents in the MUC-4 data set, the most relevant comparison is among the versions of our own implementation, in which we've sought to replicate argument similarity metrics used by others, then added our own.

Evaluation: General Roles, Gold Documents

Role	P	R	F1
Comparison scores (as reported)			
Chambers 2013	41	44	43
Cheung et al 2013	49	43	46
Component measures (our implementation)			
Arg terms+corefs (C&J 2011)	26	41	32
w/ mods (Nguyen et al 2015)	38	31	34
+ hypernym similarity	47	30	37
+ pred-dep functional sim	51	39	45
+ hyper sim + pred-dep sim	**53**	**42**	**47**

Table 1: MUC-4 role extraction, mapping general slots to documents with at least one labeled event.

Our first stage performs well, applying general learned roles to gold documents. Adding each of our contributions individually improved upon the scores we obtained using others' argument term similarity scores alone. Adding both of our contributions of hypernyms and argument position functional similarity performed best overall.

Again, we were able to do so using only the documents in the MUC-4 dataset, because we reversed the order of the process and only induced general semantic roles at this stage. One concern might be that our general roles could be overly broad when induced from corpora with more varied topics, since the MUC-4 data is overwhelmingly dominated by conflict events that share the same set of roles. We discuss the need for more varied evaluation datasets in the final section.

4.2 Full Process Evaluation: Event Roles

For our second stage, we used our induced trigger words to assign narrative sequences to MUC-4 event types, then extracted entities in event slots as described in 3.4. To evaluate the full process, we need to map our event-specific slots to MUC-4 template roles. Since we now have both event types and component slots, there are two ways to do the mapping: 1) map any learned slot to any template role (called "slot-only mapping"), or 2) map learned schemas to MUC-4 templates, then only map slots from one schema to roles in its matching template (called "template mapping"). Most of the recent papers reported slot-only mapping scores for the MUC-4 dataset, while fewer reported stricter template mapping scores as well. However, as Chambers (2013) discussed, the latter is the more comprehensive (and ideal) method for evaluating induced event schemas as a whole.

We first report slot-only mapping scores in comparison to the scores from the previous papers, in Table 2. We then discuss the greater difficulty but more precise evaluation using template mapping, in Table 3, along with possible ways to improve.

Evaluation: Learned Events, Slot Mapping

	P	R	F1
C&J 2011	**48**	25	33
Chambers 2013	41	41	41
Cheung et al 2013	32	37	34
Nguyen et al 2015	36	54	43
Sha et al 2016	39	**70**	**50**
Our results, all template roles	33	39	36

Table 2: MUC-4 role extraction on narratives with event triggers, mapping slots to any template role.

The results in Table 2 are comparable to some of the earlier work on this task, but do not reach the level achieved by the most recent efforts. We believe there is room for improvement in our document classification stage, since we only induced event trigger words. We explored more complex approaches to clustering all event predicates while allowing some to appear in multiple events. However, the more promising options became too complicated to fully develop in this paper. We opted instead for a focused approach to event triggers that highlights our intuition about functional relationships between eventful words.

For slot-only mapping, we still restricted candidate slots to the four schemas that mapped to the MUC-4 templates. Since the slots in each of our schemas are derived from the same general semantic roles, the difference between the two mappings is that the stricter template mapping tests whether we were able to correctly distinguish an entity as the perpetrator of a bombing, rather than the perpetrator of another form of attack. In other words, for schema slots that share a general role structure, the stricter template mapping places greater emphasis on our ability to distinguish between specific event types in document classification.

The relatively homogenous nature of the MUC-4 corpus makes it easier to identify documents that contain any of its main event types, but more difficult to distinguish between them. Bombings, kidnappings, arson, and (other) attacks often use similar argument terms. For the stricter template mapping evaluation, we found that we needed to stop clustering event trigger words at a slightly smaller maximum distance score (0.99 rather than 0.999, chosen on the training set), to keep some trigger words for each MUC-4 event type in separate clusters. This resulted in very few triggers for each schema, but reasonable template mapping scores for at least some events, both shown in Table 3.

Evaluation: Strict Template Mapping

	Arson	Bomb	Attack	Kidnap
triggers	*burn*	*explode, damage*	*attack, kill*	*kidnap, release*
F1	40	36	25	29

Table 3: Event trigger words and MUC-4 role extraction F1 scores, mapping slots to roles separately for each mapped schema-template pair.

The difference in performance across event types seems to relate to the number of triggers needed to capture each event concept. (Note that schema slots still use more predicates, the trigger words only classify the event narratives.) If we stop merging even sooner and retain a schema with only the trigger word "kidnap", the F1 score for Kidnapping goes up to 40, but the score for Attack (the largest category in the dataset) goes down. Chambers and Jurafsky (2011) achieved their best results when mapping several schemas to the Attack template, including subtypes for shootings, murders, and coups. Our trigger learning approach does not enable us to learn a larger cluster of Attack trigger words without merging in the trigger words for the other MUC-4 event types as well.

This suggests that a challenge for correct one-to-one mapping of event schemas to gold templates is achieving the right level of aggregation for all event types in a given corpus. Whether to label very fine-grained events like shootings and murders, or higher-level categories like attacks, crimes, or conflicts, is a subjective judgment often driven by the substantive motivation of the research. To induce event types that can be mapped to labeled events with the right level of granularity between related concepts, we may need to learn hierarchies of actions or events. Emerging efforts to identify event-event relations and event coreference offer promising avenues (Hong et al., 2016).

We might also do better at distinguishing similar event types if we combine our structural and functional contributions with a more probabilistic approach to schema induction. By breaking apart the process as we've done, we've been able to explore and test various new components, that could be incorporated into more concise models for better overall task performance in the future.

5 Discussion

In this paper, we have offered a novel approach to event schema induction, reversing the typical pipeline process to maximize the use of limited training data, and inducing general semantic roles that help distinguish coherent event narratives. Our approach differs from the dominant use of generative probabilistic models that jointly model event schemas and role slots. In keeping the steps separate, while leveraging rich information throughout, we've constructed a process that can be manipulated intuitively at different stages, incorporating structure and distinguishing word features related to function and theme.

While joint models may be mathematically cleaner, our process yields meaningful components along the way, that might be useful to researchers in their own right. These include the mapping of event-specific roles to common general semantic roles, the segmentation of coherent event narratives, and the induction of prominent eventful trigger words. Separating out the steps in a pipeline process also allows us to explore different types of intuition at each stage, since event topics are qualitatively different types of concepts from semantic roles.

In general, we've sought to induce event components intuitively, and to aid those tasks by incorporating knowledge from public, general-domain resources. WordNet and Wikipedia don't contain event frames, but they add general information about word functions and themes beyond what can be observed in relatively small corpora. We include word embeddings to confirm the relevance of certain elements to our corpus, in order to construct domain-specific event schemas when the only domain resource is raw text. WordNet and Wikipedia are available in multiple languages and are easy to use, reducing the burden on other researchers seeking to apply similar methods. In the future, we would also be interested in inductive approaches to learning word taxonomies, to ensure that the hierarchical structures used to induce semantic roles accurately reflect the senses and relationships of words as used in the relevant domain.

As a final note about data, we sought to make our evaluation directly comparable to previous work, and the MUC-4 dataset has been the standard for evaluating event extraction in the past. But the dataset is now over two decades old, and we struggled with some of its shortcomings. It was designed to evaluate rule-based pattern matching and supervised extraction algorithms, and there are coding nuances that may not be inferable from the raw text alone. In addition to the narrow focus on four somewhat overlapping types of violent attacks, our inspection of incorrect extractions in the training set revealed some entities that are clearly attack perpetrators or targets, but are not labeled as such in the key. We are encouraged by current efforts to develop new annotated corpora that might be more useful for evaluating the emerging research on inductive event extraction, and that cover a wider variety of real-world events.

Acknowledgments

Many thanks to the reviewers for thoughtful feedback, to David Bamman and Christopher Hench for helpful draft comments and discussion, to Brendan O'Connor for general advice, and to Sarah Anzia for encouragement and support.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. pages 344–354. https://doi.org/10.3115/v1/P15-1034.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. volume 1721–1731. http://aclweb.org/anthology/D13-1178.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*. pages 2670–2676. https://doi.org/10.978.157735/2983.

Jari Bjorne and Tapio Salakoski. 2011. Generalizing biomedical event extraction. In *Proceedings of BioNLP Shared Task 2011 Workshop of the Association for Computational Linguistics (ACL)*. pages 183–191. http://aclweb.org/anthology/W11-1828.

Sergei Brin. 1998. Extracting patterns and relations from the world wide web. In *Proceedings of the International Workshop on the World Wide Web and Databases*. https://doi.org/10.1007/10704656_11.

Razvan Bunescu and Raymond J. Mooney. 2004. Collective information extraction with relational markov networks. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*. http://aclweb.org/anthology/P04-1056.

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. http://aclweb.org/anthology/D13-1185.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. http://aclweb.org/anthology/P11-1098.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 837–846. http://aclweb.org/anthology/N13-1104.

Nancy Chinchor, David D. Lewis, and Lynette Hirschman. 1993. Evaluating message understanding systems: an analysis of the third message understanding conference (muc-3). *Journal of Computational Linguistics* 19(3):409–449. http://aclweb.org/anthology/J93-3001.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*. http://aclweb.org/anthology/P04-1054.

Elena Filatova, Vasileios Hatzivassiloglou, and Kathleen McKeown. 2006. Automatic creation of domain templates. In *Proceedings of the Conference on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL)*. pages 207–214. http://aclweb.org/anthology/P06-2027.

Yu Hong, Tongtao Zhang, Tim O'Gorman, Sharone Horowit-Hendler, Heng Ji, and Martha Palmer. 2016. Building a cross-document event-event relation corpus. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*. pages 1–6. https://doi.org/10.18653/v1/W16-1701.

Ruihong Huang and Ellen Riloff. 2012. Bootstrapped training of event extraction classifiers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. pages 286–295. http://aclweb.org/anthology/E12-1029.

Tingsong Jiang, Lei Sha, and Zhifang Sui. 2014. Event schema induction based on relational cooccurrence over multiple documents. *Communications in Computer and Information Science, Third International Conference on Natural Language Processing and Chinese Computing (NLPCC)* https://doi.org/10.1007/978-3-662-45924-9_3.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. pages 55–60. https://doi.org/10.3115/v1/P14-5010.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. 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 1003–1011. http://aclweb.org/anthology/P09-1113.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of the Association for Computational Linguistics: Human*

Language Technologies (ACL-HLT). pages 46–54. http://aclweb.org/anthology/P08-1006.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besancon. 2015. Generative event schema induction with entity disambiguation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. pages 188–197. https://doi.org/10.3115/v1/P15-1019.

Brendan O'Connor, Brandon M. Stewart, and Noah A. Smith. 2013. Learning to extract international relations from political context. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. pages 1094–1104. http://aclweb.org/anthology/P13-1108.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective information extraction with semantic affinity patterns and relevant regions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. http://aclweb.org/anthology/D07-1075.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. pages 99–110. http://aclweb.org/anthology/E17-1010.

Philip A. Schrodt, Shannon G. Davis, and Judith L. Weddle. 1994. Political science: Keds - a program for the machine coding of event data. *Social Science Computer Review* 12(4). https://doi.org/10.1177/089443939401200408.

Satoshi Sekine. 2006. On-demand information extraction. In *In Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL)*. pages 731–738. http://aclweb.org/anthology/P06-2094.

Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Joint learning templates and slots for event schema induction. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 428–434. https://doi.org/10.18653/v1/N16-1049.

Beth M. Sundheim. 1992. Overview of the fourth message understanding evaluation and conference. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*. http://aclweb.org/anthology/M92-1001.

Mihai Surdeanu, Jordi Turmo, and Alicia Ageno. 2006. A hybrid approach for the acquisition of information extraction patterns. In *Proceedings of the Workshop on Adaptive Text Extraction and Mining (ATEM 2006)*. http://aclweb.org/anthology/W06-2207.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th Conference on Computational Linguistics (COLING)*. http://aclweb.org/anthology/C00-2136.