Generating Coherent Event Schemas at Scale

Niranjan Balasubramanian, Stephen Soderland, Mausam, Oren Etzioni
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
{niranjan,ssoderlan, mausam, etzioni}@cs.washington.edu

Abstract

Chambers and Jurafsky (2009) demonstrated that event schemas can be automatically induced from text corpora. However, our analysis of their schemas identifies several weaknesses, e.g., some schemas lack a common topic and distinct roles are incorrectly mixed into a single actor. It is due in part to their pair-wise representation that treats subject-verb independently from verb-object. This often leads to subject-verb-object triples that are not meaningful in the real-world.

We present a novel approach to inducing open-domain event schemas that overcomes these limitations. Our approach uses co-occurrence statistics of semantically typed relational triples, which we call Rel-grams (relational n-grams). In a human evaluation, our schemas outperform Chambers's schemas by wide margins on several evaluation criteria. Both Rel-grams and event schemas are freely available to the research community.

1 Introduction

Event schemas (also known as templates or frames) have been widely used in information extraction. An event schema is a set of actors (also known as slots) that play different roles in an event, such as the perpetrator, victim, and instrument in a bombing event. They provide essential guidance in extracting information related to events from free text (Patwardhan and Riloff, 2009), and can also aid in other NLP tasks, such as coreference (Irwin et al., 2011), summarization (Owczarzak and Dang, 2010), and inference about temporal ordering and causality.

Actor	Rel	Actor
A1:<person>	failed	A2:test
A1:<person>	was suspended for	A3:<time period>
A1:<person>	used	A4:<substance, drug>
A1:<person>	was suspended for	A5:<game, activity>
A1:<person>	was in	A6:<location>
A1:<person>	was suspended by	A7:<org, person>
Actor Instances:		
A1: {Murray, Morgan, Governor Bush, Martin, Nelson}		
A2: {test}		
A3: {season, year, week, month, night}		
A4: {cocaine, drug, gasoline, vodka, sedative}		
A5: {violation, game, abuse, misfeasance, riding}		
A6: {desert, Simsbury, Albany, Damascus, Akron}		
A7: {Fitch, NBA, Bud Selig, NFL, Gov Jeb Bush}		

Table 1: An event schema produced by our system, represented as a set of $(Actor, Rel, Actor)$ triples, and a set of instances for each actor $A1$, $A2$, etc. For clarity we show unstemmed verbs.

Until recently, all event schemas in use in NLP were hand-engineered, *e.g.*, the MUC templates and ACE event relations (ARPA, 1991; ARPA, 1998; Doddington et al., 2004). This led to technology that could only focus on specific domains of interest and has not been applicable more broadly.

The seminal work of Chambers and Jurafsky (2009) showed that event schemas can also be induced automatically from text corpora. Instead of labeled roles these schemas have a set of relations and actors that serve as arguments.[1] Their system is fully automatic, domain-independent, and scales to large text corpora.

However, we identify several limitations in the schemas produced by their system.[2] Their schemas

[1] In the rest of this paper we use event schemas to refer to these automatically induced schemas with actors and relations.

[2] Available at http://www.usna.edu/Users/cs/nchamber/data/schemas/acl09

Actor	Rel	Actor
A1	caused	A2
A2	spread	A1
A2	burned	A1
-	extinguished	A1
A1	broke out	-
-	put out	A1
Actor Instances:		
A1: {fire, aids, infection, disease}		
A2: {virus, bacteria, disease, urushiol, drug}		

Table 2: An event schema from Chambers's system that mixes the events of fire spreading and disease spreading.

often lack coherence: mixing unrelated events and having actors whose entities do not play the same role in the schema. Table 2 shows an event schema from Chambers that mixes the events of fire spreading and disease spreading.

Much of the incoherence of Chambers's schemas can be traced to their representation that uses *pairs* of elements from an assertion, thus, treating subject-verb and verb-object separately. This often leads to subject-verb-object triples that do not make sense in the real world. For example, the assertions "fire caused virus" and "bacteria burned AIDS" are implicit in Table 2.

Another limitation in schemas Chambers released is that they restrict schemas to two actors, which can result in combining different actors. Table 4 shows an example of combining perpeterators and victims into a single actor.

1.1 Contributions

We present an event schema induction algorithm that overcomes these weaknesses. Our basic representation is *triples* of the form (Arg1, Relation, Arg2), extracted from a text corpus using Open Information Extraction (Mausam et al., 2012). The use of triples aids in agreement between subject and object of a relation. The use of Open IE leads to more expressive relation phrases (*e.g.*, with prepositions). We also assign semantic types to arguments, both to alleviate data sparsity and to produce coherent actors for our schemas.

Table 1 shows an event schema generated by our system. It has six relations and seven actors. The schema makes several related assertions about a person using a drug, failing a test, and getting suspended. The main actors in the schema include the person who failed the test, the drug used, and the agent that suspended the person.

Our first step in creating event schemas is to tabulate co-occurrence of tuples in a database that we call Rel-grams (relational n-grams) (Sections 3, 5.1). We then perform analysis on a graph induced from the Rel-grams database and use this to create event schemas (Section 4).

We compared our event schemas with those of Chambers on several metrics including whether the schema pertains to a coherent topic or event and whether the actors play a coherent role in that event (Section 5.2). Amazon Mechanical Turk workers judged that our schemas have significantly better coherence – 92% versus 82% have coherent topic and 81% versus 59% have coherent actors.

We release our open domain event schemas and the Rel-grams database[3] for further use by the NLP community.

2 System Overview

Our approach to schema generation is based on the idea that frequently co-occurring relations in text capture relatedness of assertions about real-world events. We begin by extracting a set of relational tuples from a large text corpus and tabulate occurrence of pairs of tuples in a database.

We then construct a graph from this database and identify high-connectivity nodes (relational tuples) in this graph as a starting point for constructing event schemas. We use graph analysis to rank the tuples and merge arguments to form the actors in the schema.

3 Modeling Relational Co-occurrence

In order to tabulate pairwise occurences of relational tuples we need a suitable relation-based representation. We now describe the extraction and representation of relations, a database for storing co-occurrence information, and our probabilistic model for the co-occurrence. We call this model Rel-grams, as it can be seen as a relational analog to the n-grams language model.

3.1 Relations Extraction and Representation

We extract relational triples from each sentence in a large corpus using the OLLIE Open IE system

[3] Available at http://relgrams.cs.washington.edu

Tuples Table

Id	Arg1	Rel	Arg2	Count
...	...	...	...	...
13	bomb	explode in	<loc>	547
14	bomb	explode in	Baghdad	22
15	bomb	explode in	market	7
...	...	...	...	...
87	bomb	kill	<per>	173
...	...	...	...	...
92	<loc>	be suburb of	<loc>	1023
...	...	...	...	...

BigramCounts Table

T1	T2	Dist.	Count	E11	E12	E21	E22
...	...	...	...	...	...	...	...
13	87	1	27	25	0	0	0
13	87	2	8	8	0	0	0
...	...	...	...	...	...	...	...
13	87	10	2	2	0	0	0
87	13	1	6	0	0	0	0
...	...	...	...	...	...	...	...
92	13	1	12	0	0	12	0
...	...	...	...	...	...	...	...

Figure 1: Tables in the Rel-grams Database: *Tuples* maps tuples to unique identifiers, *BigramCounts* provides the co-occurrence counts (Count) within various distances (Dist.), and four types of argument equality counts (E11-E22). E11 is the number of times when T1.Arg1 = T2.Arg1, E12 is when T1.Arg1 = T2.Arg2 and so on.

(Mausam et al., 2012).[4] This provides relational tuples in the format (Arg1, Relation, Arg2) where each tuple element is a phrase from the sentence. The sentence "He cited a new study that was released by UCLA in 2008." produces three tuples:

1. (He, cited, a new study)
2. (a new study, was released by, UCLA)
3. (a new study, was released in, 2008)

Relational triples provide a more specific representation which is less ambiguous when compared to (subj, verb) or (verb, obj) pairs. However, using relational triples also increases sparsity. To reduce sparsity and to improve generalization, we represent the relation phrase by its stemmed head verb plus any prepositions. The relation phrase may include embedded nouns, in which case these are stemmed as well. Moreover, tuple arguments are represented as stemmed head nouns, and we also record semantic types of the arguments.

We selected 29 semantic types from WordNet, examining the set of instances on a small development set to ensure that the types are useful, but not overly specific. The set of types are: person, organization, location, time_unit, number, amount, group, business, executive, leader, effect, activity, game, sport, device, equipment, structure, building, substance, nutrient, drug, illness, organ, animal, bird, fish, art, book, and publication.

To assign types to arguments, we apply Stanford Named Entity Recognizer (Finkel et al., 2005)[5], and also look up the argument in WordNet 2.1 and record the first three senses if they map to our target semantic types. We use regular expressions to recognize dates and numeric expressions, and map personal pronouns to <person>. We associate all types found by this mechanism with each argument. The tuples in the example above are normalized to the following:

1. (He, cite, study)
2. (He, cite, <activity>)
3. (<person>, cite, study)
4. (<person>, cite, <activity>)
5. (study, be release by, UCLA)
6. (study, be release by, <organization>)
7. (study, be release in, 2008)
8. (study, be release in, <time_unit>)
9. (<activity>, be release by, UCLA)

...

In our preliminary experiments, we found that using normalized relation strings and semantic classes for arguments results in a ten-fold increase in the number of Rel-grams with a minimum support.

3.2 Co-occurrence Tabulation

We construct a database to hold co-occurrence statistics for pairs of tuples found in each document as shown in Figure 1. The database consists of two tables: 1) *Tuples* – Maps each tuple to a unique identifier and tabulates tuple counts. 2) *BigramCounts* – Stores the directional co-occurrence frequency, a count for tuple T followed by T' at a distance of k, and tabulates the number of times the same argument was present in the pair of tuples.

Equality Constraints: Along with the co-occurrence counts, we record the equality of arguments in Rel-grams pairs. We assert an argument pair is equal if they are from the same token se-

[4] Available at: http://knowitall.github.io/ollie/

[5] We used the system downloaded from: http://nlp.stanford.edu/software/CRF-NER.shtml and used the seven class CRF model distributed with it.

Table 3: Given a source tuple, the Rel-grams language model estimates the probability of encountering other relational tuples in a document. For clarity, we show the unstemmed version.

Top tuples related to (<person>, convicted of, murder)
1. (<person>, convicted in, <time_unit>)
2. (<person>, sentenced to, death)
3. (<person>, sentenced to, year)
4. (<person>, convicted in, <location>)
5. (<person>, sentenced to, life)
6. (<person>, convicted in, <person>)
7. (<person>, convicted after, trial)
8. (<person>, sent to, prison)

quence in the source sentence or one argument is a co-referent mention of the other. We use the Stanford Co-reference system (Lee et al., 2013)[6] to detect co-referring mentions. There are four possible equalities depending on the specific pair of arguments in the tuples are the same, shown as E11, E12, E21 and E22 in Figure 1. For example, the E21 column has counts for the number of times the Arg2 of T1 was determined to be the same as the Arg1 of T2.

Implementation and Query Language: We populated the Rel-grams database using OLLIE extractions from a set of 1.8 Million New York Times articles drawn from the Gigaword corpus. The database consisted of approximately 320K tuples that have frequency ≥ 3 and 1.1M entries in the bigram table.

The Rel-grams database allows for powerful querying using SQL. For example, Table 3 shows the most frequent rel-grams associated with the query tuple (<person>, convicted of, murder).

3.3 Rel-grams Language Model

From the tabulated co-occurrence statistics, we estimate a bi-gram conditional probabilities of tuples that occur within a window of k tuples from each other. Formally, we use $P_k(T'|T)$ to denote the conditional probability that T' follows T within a window of k tuples. To discount estimates from low-frequency tuples, we use a δ-smoothed estimate:

$$P_k(T'|T) = \frac{\#(T, T', k) + \delta}{\sum_{T'' \in V} \#(T, T'', k) + \delta \cdot |V|} \quad (1)$$

[6]Available for download at: http://nlp.stanford.edu/software/dcoref.shtml

where, $\#(T, T', k)$ is the number of times T' follows T within a window of k tuples. $k = 1$ indicates adjacent tuples in the document. $|V|$ is the number of unique tuples in the corpus. For experiments in this paper, we set δ to 0.05.

Co-occurrence within a small window is usually more reliable but is also sparse, whereas co-occurrence within larger windows addresses sparsity but may lead to topic drift. To leverage the benefits of different window sizes, we also define a metric with a weighted average of window sizes from 1 to 10, where the weight decays as window size increases. For example, with α set to 0.5 in equation 2, a window of $k+1$ has half the weight of a window of k.

$$P(T'|T) = \frac{\sum_{k=1}^{10} \alpha^k P_k(T'|T)}{\sum_{k=1}^{10} \alpha^k} \quad (2)$$

We believe that Rel-grams is a valuable source of common-sense knowledge and may be useful for several downstream tasks such as improving information extractors, inference of implicit information, etc. We assess its usefulness in the context of generating event schemas.

4 Schema Generation

We now use Rel-grams to identify relations and actors pertaining to a particular event. Our schema generation consists of three steps. First, we build a relation graph of tuples (G) using connections identified by Rel-grams. Second, we identify a set of seed tuples as starting points for schemas. We use graph analysis to find the tuples most related to each seed. Finally, we merge the arguments in these tuples to create actors and output the final schema. Next we describe each of these steps in detail.

4.1 Rel-graph construction

We define a Rel-graph as an undirected weighted graph $G = (V, E)$, whose vertices (V) are relation tuples with edges (E), where an edge between vertices T and T' is weighted by the symmetric conditional probability $SCP(T, T')$ defined as

$$SCP(T, T') = P(T|T') \times P(T'|T) \quad (3)$$

Both conditional probabilities are computed in Equation 2. Figure 2 shows a portion of a Rel-graph

where the thickness of the edge indicates symmetric conditional probability.

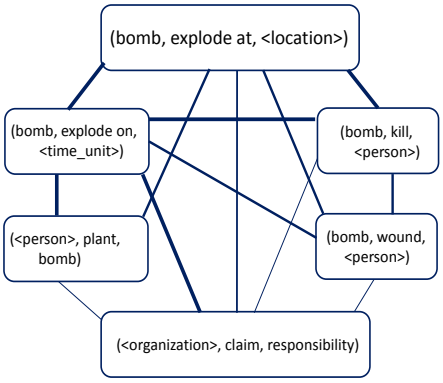

Figure 2: Part of a Rel-graph showing tuples strongly associated with *(bomb, explode at, <location>)*. Undirected edges are weighted by symmetric conditional probability with line thickness indicating weight.

4.2 Finding Related Tuples

Our goal is to find closely related tuples that pertain to an event or topic. First, we locate high-connectivity nodes in the Rel-graph to use as seeds. We sort nodes by the sum of their top 25 edge weights[7] and take the top portion of this list after filtering out redundant views of the same relation.

For each seed (Q), we find related tuples by extracting the sub-graph (G_Q) from Q's neighbors (within two hops from Q) in the Rel-graph. Graph analysis can detect the strongly connected nodes within this sub-graph, representing tuples that frequently co-occur in the context of the seed tuple.

Page rank is a well-known graph analysis algorithm that uses graph connectivity to identify important nodes within a graph (Brin and Page, 1998). We are interested in connectivity within a subgraph with respect to a designated query node (the seed). Connection to a query node can help minimize concept drift and ensure that the selected tuples are closely related to the main topic of the sub-graph.

In this work, we adapt the Personalized PageRank algorithm (Haveliwala, 2002). The personalized version of PageRank returns ranks of various nodes with respect to a given query node and hence is more appropriate for our task than the basic PageRank algorithm. Within the subgraph G_Q for a given seed Q, we compute a solution to the following set of PageRank Equations:

$$
\begin{aligned}
PR_Q(T) & \\
&= (1-d) + d\sum_{T'} SCP(T,T')PR_Q(T') \quad \text{if } T = Q \\
&= d\sum_{T'} SCP(T,T')PR_Q(T') \qquad\qquad \text{otherwise}
\end{aligned}
$$

Here $PR_Q(T)$ denotes the page rank of a tuple T personalized for the query tuple Q. It is a sum of all its neighbors' page ranks, weighted by the edge weights; d is the damping probability, which we set to be 0.85 in our implementation.

The solution is computed iteratively by initializing the page rank of Q to 1 and all others to 0, then recomputing page rank values until they converge to within a small ϵ. This computation remains scalable, since we restrict it to subgraphs a small number of hops away from the query node. This is a standard practice to handle large graphs (Agirre and Soroa, 2009; Mausam et al., 2010).

4.3 Creating Actors and Relations

We take the top n tuples from G_Q according to their Page rank scores. From each tuple T : $(Arg1, Rel, Arg2)$ in G_Q, we record two actors (A_1, A_2) corresponding to $Arg1$ and $Arg2$, and add Rel to the list of relations that they participate in.

Then, we merge actors in two steps. First, we collect the equality constraints for the tuples in G_Q. If the arguments corresponding to any pair of actors have a non-zero equality constraint then we merge them. Second, we merge actors that perform similar actions. A_1 and A_2 are merged if they are connected to the same actor A_3 through the same relation. For example, A1 and A2 in (A1:lawsuit, file by, A3:company) and (A2:suit, file by, A3:company), will be merged into a single actor. To avoid merging distinct actors, we use a small list of rules that specify the semantic type pairs that cannot be merged (*e.g.*, location-date). Also, we do not merge two actors, if it can result in a relation where the same actor is both the Arg1 and Arg2.

Finally, we generate an ordered list of tuples using the final set of actors and their relations. The output tuples are sorted by the average page rank of the original tuples, thereby reflecting their importance within the sub-graph G_Q.

[7] This avoids overly general tuples that occur in many topics. These general tuples tend to have many weak edges.

System	A_1	Rel	A_2
Relgrams	{bomb, missile, grenade, device}	explode in	{city, Bubqua, neighborhood}
	{bomb, missile, grenade, device}	explode kill	{people, civilian, lawmaker, owner, soldier}
	{bomb, missile, grenade, device}	explode on	{Feb., Fri., Tues., Sun., Sept.}
	{bomb, missile, grenade, device}	explode wound	{civilian, person, people, soldier, officer}
	{bomb, missile, grenade, device}	explode in	{Feb., Beirut Monday, Sept., Aug.}
	{bomb, missile, grenade, device}	explode injure	{woman, people, immigrant, policeman}
Chambers	{bomb, explosion, blast, **bomber**, mine}	explode	{soldier, child, civilian, bomber, palestinian}
	{soldier, **child**, **civilian**, bomber, palestinian}	set off	{bomb, explosion, blast, **bomber**, mine, bombing}
	{bomb, explosion, blast, bomber, mine}	kill	{soldier, child, civilian, bomber, palestinian}
	{soldier, **child**, **civilian**, bomber, palestinian}	detonate	{bomb, explosion, blast, **bomber**, mine, bombing}
	{bomb, explosion, blast, bomber, mine}	injure	{soldier, child, civilian, bomber, palestinian}
	{soldier, **child**, **civilian**, bomber, palestinian}	plant	{bomb, explosion, blast, **bomber**, mine, **bombing**}
Relgrams	{Carey, John Anthony Volpe, Chavez, She }	veto	{legislation, bill, law, measure, version}
	{legislation, bill, law, measure, version}	be sign by	{Carey, John Anthony Volpe, Chavez, She }
	{legislation, bill, law, measure, version}	be pass by	{State Senate, State Assembly, House, Senate, Parliament}
	{Carey, John Anthony Volpe, Chavez, She }	sign into	{law}
	{Carey, John Anthony Volpe, Chavez, She }	to sign	{bill}
	{Carey, John Anthony Volpe, Chavez, She }	be governor of	{Massachusetts, state, South Carolina, Texas, California}
Chambers	{clinton, bush, **bill**, president, house}	oppose	{bill, measure, legislation, plan, law}
	{clinton, bush, **bill**, president, **house**}	sign	{bill, measure, legislation, plan, law}
	{clinton, bush, **bill**, president, house}	approve	{bill, measure, legislation, plan, law}
	{clinton, bush, **bill**, president, **house**	veto	{bill, measure, legislation, plan, law}
	{clinton, bush, **bill**, president, house}	support	{bill, measure, legislation, plan, law}
	{**clinton**, **bush**, **bill**, **president**, house}	pass	{bill, measure, legislation, plan, law}

Table 4: "Bombing" and "legislation" schema examples from Rel-grams and Chambers represented as a set of (A_1, Rel, A_2) tuples, where the schema provides a set of instances for each actor A_1 and A_2. Relations and arguments are in the stemmed form, e.g., 'explode kill' refers to 'exploded killing'. Instances in bold produce tuples that are not valid in the real world.

5 Evaluation

We present experiments to explore two main questions: How well do Rel-grams capture real world knowledge, and what is the quality of event schemas built using Rel-grams.

5.1 Evaluating Rel-grams

What sort of common-sense knowledge is encapsulated in Rel-grams? How often does it indicate an implication between a pair of statements, and how often does it indicate a common real-world event or topic? To answer these questions, we conducted an experiment to identify a subset of our Rel-grams database with high precision for two forms of common-sense knowledge:

- **Implication:** The Rel-grams express an implication from T to T' or from T' to T, a bi-directional form of the Recognizing Textual Entailment (RTE) guidelines (Dagan et al., 2005).
- **Common Topic:** Is there an underlying common topic or event to which both T and T' are relevant?

We also evaluated whether both T and T' are **valid tuples** that are well-formed and make sense in the real world, a necessary pre-condition for either implication or common topic.

We are particularly interested in the highest precision portion of our Rel-grams database. The database has 1.1M entries with support of at least three instances for each tuple. To find the highest precision subset of these, we identified tuples that have at least 25 Rel-grams, giving us 12,600 seed tuples with a total of over 280K Rel-grams. Finally, we sorted this subset by the total symmetrical conditional probability of the top 25 Rel-grams for each seed tuple.

We tagged a sample of this 280K set of Rel-grams for valid tuples, implication between T and T', and common topic. We found that in the top 10% of this set, 87% of the seed tuples were valid and 74% of the Rel-grams had both tuples valid. Of the Rel-grams with both tuples valid, 83% expressed an implication between the tuples, and 90% had a common topic.

There were several reasons for invalid tuples – parsing errors; binary projections of inherently n-ary relations, for example (<person>, put, <person>);

System	Id	A_1	Rel	A_2
Relgrams	R1	bomb bomb bomb ...	explode in explode kill explode on ...	city people Fri. ...
Chambers	C1	blast child bomber ...	explode detonate plant ...	child bomber bomb ...

Table 5: A grounded instantiation of the schemas from Table 4, where each actor is represented as a randomly selected instance.

System	Id	A_1	Rel	A_2
Relgrams	R11	bomb missile grenade ...	explode in explode in explode in ...	city city city ...
Relgrams	R21	missile missile missile ...	explode in explode in explode in ...	city neighborhood front ...

Table 6: A schema instantiation used to test for actor coherence. Each of the top instances for A_1 or A_2 is presented, holding the relation and the other actor fixed.

head-noun only representation omitting essential information; and incorrect semantic types, primarily due to NER tagging errors.

While the Rel-grams suffer from noise in the tuple validity, there is clearly strong signal in the data about common topic and implication between tuples in the Rel-grams. As we demonstrate in the following section, an end task can use graph analysis techniques to amplify this strong signal, producing high-quality relational schemas.

5.2 Schemas Evaluation

In our schema evaluation, we are interested in assessing how well the schemas correspond to common-sense knowledge about real world events. To this end, we focus on three measures, *topical coherence*, *tuple validity*, and *actor coherence*.

A good schema must be topically coherent, i.e., the relations and actors should relate to some real world topic or event. The tuples that comprise a schema should be valid assertions that make sense in the real world. Finally, each actor in the schema should belong to a cohesive set that plays a consistent role in the relations. Since there are no good automated ways to make such judgments, we perform a human evaluation using workers from Amazon's Mechanical Turk (AMT).

We compare Rel-grams schemas against the state-of-the-art narrative schemas released by Chambers (Chambers and Jurafsky, 2009).[8] Chambers's schemas are less expressive than ours – they do not associate types with actors and each schema has a constant pre-specified number of relations. For a fair comparison we use a similarly expressive version of our schemas that strips off argument types and has the same number of relations per schema (six) as their highest quality output set.

[8]Available at `http://www.usna.edu/Users/cs/nchamber/data/schemas/acl09`

5.2.1 Evaluation Design

We created two tasks for AMT annotators. The first task tests the coherence and validity of relations in a schema and the second does the same for the schema actors. In order to make the tasks understandable to unskilled AMT workers, we followed the accepted practice of presenting them with *grounded* instances of the schemas (Wang et al., 2013), e.g., instantiating a schema with a specific argument instead of showing the various possibilities for an actor.

First, we collect the information in schemas as a set of tuples: $S = \{T_1, T_2, \cdots, T_n\}$, where each tuple is of the form $T : (X, Rel, Y)$, which conveys a relationship *Rel* between actors X and Y. Each actor is represented by its highest frequency examples (instances). Table 4 shows examples of schemas from Chambers and Rel-grams represented in this format. Then, we create grounded tuples by randomly sampling from top instances for each actor.

Task I: Topical Coherence To test whether the relations in a schema form a coherent topic or event, we presented the AMT annotators with a schema as a set of grounded tuples, showing each relation in the schema, but randomly selecting one of the top 5 instances from each actor. We generated five such instantiations for each schema. An example instantiation is shown in Table 5.

We ask three kinds of questions on each grounded schema: (1) is each of the grounded tuples valid (i.e. meaningful in the real world); (2) do the majority of relations form a coherent topic; and (3) does each tuple belong to the common topic. Similar to pre-

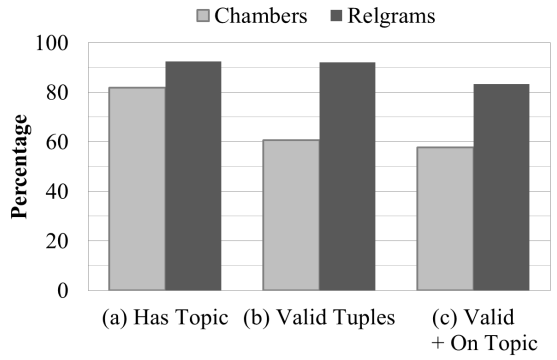

Figure 3: (a) Has Topic: Percentage of schema instantiations with a coherent topic. (b) Valid Tuples: Percentage of grounded statements that assert valid real-world relations. (c) Valid + On Topic: Percentage of grounded statements where 1) the instantiation has a coherent topic, 2) the tuple is valid and 3) the relation belongs to the common topic. All differences are statistically significant with a *p-value* < 0.01.

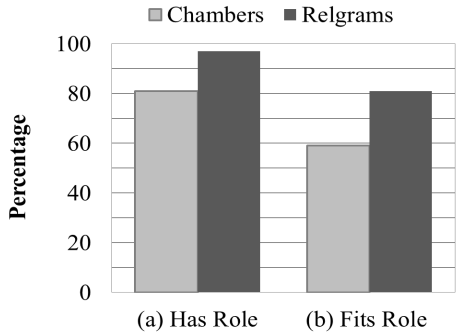

Figure 4: Actor Coherence: *Has Role* bars compare the percentage of tuples where the tested actors have a coherent role. *Fits Role* compares the percentage of top instances that fit the specified role for the tested actors. All differences are statistically significant with a *p-value* < 0.01.

vious AMT studies we get judgments from multiple (five) annotators on each task and use the majority labels (Snow et al., 2008).

Our instructions specified that the annotators should ignore grammar and focus on whether a tuple may be interpreted as a real world statement. For example, the first tuple in R1 in Table 5 is a valid statement – "a bomb exploded in a city", but the first tuple in C1 "a blast exploded a child" and second tuple "a child detonated a bomber" don't make sense.

Task II: Actor Coherence To test whether the instances of an actor form a coherent set, we held the relation and one actor fixed and presented the AMT annotators with the top 5 instances for the other actor. The first example R11 in Table 6 holds the relation "explode in" fixed, and A2 is grounded to the randomly selected instance "city". We present grounded tuples by varying A1 and ask annotators to judge whether these instances form a coherent topic and whether each instance belongs to that common topic. As with Task I, we create five random instantiations for each schema.

5.2.2 Results

We obtained a test set of 100 schemas per system by randomly sampling from the top 500 schemas from each system. We evaluate this test set using Task I and II as described above. For both tasks we used the majority labels as the final annotation.[9]

Does the schema belong to a single topic? The *Has Topic* bars in Figure 3 show results for schema coherence. Rel-grams has a higher proportion of schemas with a coherent topic, 91% compared to 82% for Chambers's. This is a 53% reduction in incoherent schemas.

Do tuples assert valid real-world relations? The *Valid Tuples* bars in Figure 3 compare the percentage of valid grounded tuples in the schema instantiations. A tuple was labeled *valid* if a majority of the annotators labeled it to be meaningful in the real world. Here we see a dramatic difference – Rel-grams have 92% valid tuples, compared with Chambers's 61%.

What proportion of tuples belong? The *Valid + On Topic* bars in Figure 3 compare the percentage of tuples that are both *valid* and *on topic*, i.e., fits the main topic of the schema. Tuples from schema instantiations that did not have a coherent topic were labeled incorrect.

Rel-grams have a higher proportion of valid tuples belonging to a common topic, 82% compared to 58% for Chambers's schemas, a 56% error reduction. This is the strictest of the experiments de-

[9] We used an expert annotator (one of the authors) to obtain ratings for the 'Has Topic' question in Task I. The agreement between the expert and the majority turker ratings was 73%, primarily because the Turkers were more lenient in assessing tuple validity.

scribed thus far – 1) the schema must have a topic, 2) the tuple must be valid, and 3) the tuple must belong to the topic.

Do actors represent a coherent set of arguments? We evaluated schema actors from the top 25 schemas in Chambers's and Rel-grams schemas, using grounded instances such as those in Table 6. Figure 4 compares the percentage of tuples where the actors play a coherent role (Has Role), and the percentage of instances that fit that role for the actor (Fits Role). Rel-grams has much higher actor coherence than Chambers's, with 97% judged to have a topic compared to 81%, and 81% of instances fitting the common role compared with Chambers's 59%.

5.2.3 Error Analysis

The errors in both our schemas and those of Chambers are primarily due to mismatched actors and from extraction errors, although Chambers's schemas have a larger number of actor mismatch errors and the cause of the errors is different for each system.

Examining the data published by Chambers, the main source of invalid tuples are mismatch of subject and object for a given relation, which accounts for 80% of the invalid tuples. We hypothesize that this is due to the pair-wise representation that treats subject-verb and verb-object separately, causing inconsistent s-v-o tuples. An example is (boiler, light, candle) where (boiler, light) and (light, candle) are well-formed, yet the entire tuple is not. In addition, 43% of the invalid tuples seem to be from errors by the dependency parser.

Our schemas also suffer from mismatched actors, despite the semantic typing of the actors – we found a mismatch in 56% of the invalid tuples (5% of all tuples). A general type such as <person> or <organization> may still have an instance that does not play the same role as other instances. For example a relation (A1, graduated from, A2) has A2 that is mostly school names, but also includes "church" which leads to an invalid tuple.

Extraction errors account for 47% of the invalid tuples in our schemas, primarily errors that truncate an n-ary relation as a binary tuple. For example, the sentence "Mr. Diehl spends more time ... than the commissioner" is misanalysed by the Open IE extractor as (Mr. Diehl, spend than, commissioner).

6 Related Work

Prior work by Chambers and Jurafsky (2008; 2009; 2010) showed that event sequences (narrative chains) mined from text can be used to induce event schemas in a domain-independent fashion. However, our manual evaluation of their output showed key limitations which may limit applicability.

As pointed out earlier, a major weakness in Chambers's approach is the pair-wise representation of subject-verb and verb-object. Also, their released a set of schemas are limited to two actors, although this number can be increased by setting a chain splitting parameter.

Chambers and Jurafsky (2011) extended schema generation to learn domain-specific event templates and associated extractors. In work parallel to ours, Cheung et al. (2013), developed a probabilistic solution for template generation. However, their approach requires performing joint probability estimation using EM, which can limit scaling to large corpora.

In this work we developed an Open IE based solution to generate schemas. Following prior work (Balasubramanian et al., 2012), we use Open IE triples for modeling relation co-occurrence. We extend the triple representation with semantic types for arguments to alleviate sparisty and to improve coherence. We developed a page rank based schema induction algorithm which results in more coherent schemas with several actors. Unlike Chambers's approach this method does not require explicit parameter tuning for controlling the number of actors.

While our event schemas are close to being templates (because of associated types, and actor clustering), they do not have associated extractors. Our future work will focus on building extractors for these. It will also be interesting to compare with Cheung's system on smaller focused corpora.

Defining representations for events is a topic of active interest (Fokkens et al., 2013). In this work, we use a simpler representation, defining event schemas as a set of actors with associated types and a set of roles they play.

7 Conclusions

We present a system for inducing event schemas from text corpora based on Rel-grams, a database of

co-occurrence statistics of relational triples (Arg1, Relation, Arg2) extracted by a state-of-the-art Open IE system. By using triples rather than a pair-wise representation of subject-verb and verb-object, we achieve more coherent schemas than Chambers and Jurafsky (2009). In particular, our schemas have higher topic coherence (92% compared to Chambers's 82%; make a higher percentage of valid assertions (94% compared with 61%); and have greater actor coherence (81% compared with 59%).

Our schemas are also more expressive than those published by Chambers – we have semantic typing for the actors, we are not limited to two actors per schema, and our relation phrases include prepositions and are thus more precise and have higher coverage of actors involved in the event.

Our future plans are to build upon our event schemas to create an open-domain event extractor. This will extend each induced schema to have associated extractors. These extractors will operate on a document and instantiate an instance of the schema.

We have created a Rel-grams database with 1.1M entries and a set of over 2K event schemas from a corpus of 1.8M New York Times articles. Both are freely available to the research community[10] and may prove useful for a wide range of NLP applications.

Acknowledgments

We thank the anonymous reviewers, Tony Fader, and Janara Christensen for their valuable feedback. This paper was supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory (AFRL) contract number FA8650-10-C-7058 and in part by the Defense Advanced Research Projects Agency (DARPA) via Air Force Research Laboratory (AFRL) contract number AFRL FA8750-13-2-0019. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

[10] available at *http://relgrams.cs.washington.edu*

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 33–41.

ARPA. 1991. *Proc. 3rd Message Understanding Conf.* Morgan Kaufmann.

ARPA. 1998. *Proc. 7th Message Understanding Conf.* Morgan Kaufmann.

Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni, et al. 2012. Rel-grams: a probabilistic model of relations in text. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 101–105. Association for Computational Linguistics.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117.

N. Chambers and D. Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*.

N. Chambers and D. Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of ACL*.

N. Chambers and D. Jurafsky. 2010. A database of narrative schemas. In *Proceedings of LREC*.

N. Chambers and D. Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of ACL*.

J. Cheung, H. Poon, and L. Vandervende. 2013. Probabilistic frame induction. In *Proceedings of NAACL HLT*.

I. Dagan, O. Glickman, and B. Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 1–8.

G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, , and R. Weischedel. 2004. The automatic content extraction (ACE) program-tasks, data, and evaluation. In *Procs. of LREC*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Antske Fokkens, Marieke van Erp, Piek Vossen, Sara Tonelli, Willem Robert van Hage, BV SynerScope, Luciano Serafini, Rachele Sprugnoli, and Jesper Hoeksema. 2013. Gaf: A grounded annotation framework for events. *NAACL HLT 2013*, page 11.

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *WWW*, pages 517–526.

Joseph Irwin, Mamoru Komachi, and Yuji Matsumoto. 2011. Narrative schema as world knowledge for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 86–92. Association for Computational Linguistics.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, (Just Accepted):1–54.

Mausam, Stephen Soderland, Oren Etzioni, Daniel Weld, Kobi Reiter, Michael Skinner, Marcus Sammer, and Jeff Bilmes. 2010. Panlingual lexical translation via probabilistic inference. *Artificial Intelligence Journal (AIJ)*.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of EMNLP*.

K. Owczarzak and H.T. Dang. 2010. Overview of the tac 2010 summarization track.

S. Patwardhan and E. Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of EMNLP 2009*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics.

A. Wang, C.D.V. Hoang, and M-Y. Kan. 2013. Perspectives on crowdsourcing annotations for Natural Language Processing. *Language Resources and Evaluation*, 47:9–31.