Event Representations with Tensor-based Compositions

Noah Weber
Stony Brook University
Stony Brook, New York, USA
nwweber@cs.stonybrook.edu

Niranjan Balasubramanian
Stony Brook University
Stony Brook, New York, USA
niranjan@cs.stonybrook.edu

Nathanael Chambers
United States Naval Academy
Annapolis, Maryland, USA
nchamber@usna.edu

Abstract

Robust and flexible event representations are important to many core areas in language understanding. Scripts were proposed early on as a way of representing sequences of events for such understanding (Schank and Abelson 1977), and has recently attracted renewed attention. However, obtaining effective representations for modeling script-like event sequences is challenging. It requires representations that can capture event-level and scenario-level semantics. We propose a new tensor-based composition method for creating event representations. The method captures more subtle semantic interactions between an event and its entities and yields representations that are effective at multiple event-related tasks. With the continuous representations, we also devise a simple schema generation method which produces better schemas compared to a prior discrete representation based method. Our analysis shows that the tensors capture distinct usages of a predicate even when there are only subtle differences in their surface realizations.

Introduction

Understanding the events described in text is central to applications in artificial intelligence such as question answering, discourse understanding, and information extraction. Research in event understanding ranges from relation extraction of individual events to full document understanding of all its events. Inspired by the concept of *scripts*, proposed in the seminal work by Schank and Abelson (1977), much work has looked at modeling stereotypical sequences of events in order to drive discourse understanding. Early rule-based methods for this task were characteristically brittle and domain-specific. Later work proposed computational models for script learning and understanding (Mooney and DeJong 1985; Chambers and Jurafsky 2009; Balasubramanian et al. 2013), but they use shallow event representations dependent on their specific surface words. Others have focused on training neural networks for robust event representations, using them to predict which events are likely to occur next (Modi 2016; Pichotta and Mooney 2016).

To be broadly useful, a good representation should capture both event-level semantics (e.g., synonymy) and broader scenario-level semantics. Event-level semantics are accessible to simple composition methods. However, modeling scenario-level semantics is challenging. For example, consider the following events: (i) *she threw a football* and (ii) *she threw a bomb*. Even though the subject and the verb are the same, these two events denote entirely different scenarios – sports and terrorism. The interaction of *football* or *bomb* with the predicate *threw* is what determines the precise semantics of the event and the broader scenario in which it is embedded. A change to a single argument requires a large shift in the event's representation.

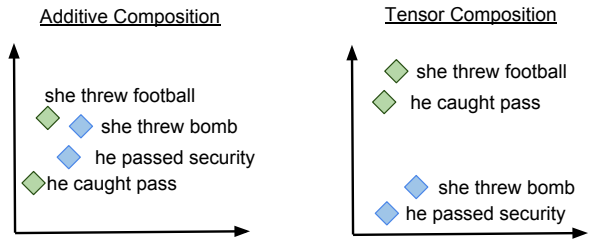

Figure 1: Additive compositions of word embeddings do not distinguish distinct events. Tensor compositions can tease out distinct events even with high lexical overlap, and recognize related events even with low lexical overlap.

As illustrated in Figure 1, due to the overlap of words, parameterized additive models (Granroth-Wilding and Clark 2016; Modi 2016) and RNN-based models (Pichotta and Mooney 2016; Hu et al. 2017) are limited in their transformations. Additive models combine the words in these phrases by the passing the concatenation or addition of their word embeddings to a parameterized function (usually a feed forward neural network) that maps the summed vector into event embedding space. The additive nature of these models makes it difficult to model subtle differences in an event's surface form. Instead of additive models, we propose tensor-based composition models, which combine the subject, predicate, and object to produce the final event representation. The models capture multiplicative interactions between these elements and are thus able to make large shifts in event semantics with only small changes to the arguments.

This paper puts forth three main contributions:

- A scalable tensor-based composition model for event representations, which can implicitly recognize different

event contexts based on predicate argument interactions.

- A broad set of evaluations on event-related tasks: we find that the tensor-based event representation outperforms additive compositions on (i) a sentence similarity task, (ii) a new *hard* similarity task, and (iii) an event prediction task (two variants of the narrative cloze), suggesting broad utility in event-related tasks.
- A simple yet effective method for generating event schemas: to our knowledge, ours is the first proposal for using continuous event representations in schema generation. We show that event tensors produce superior schemas compared to a prior distributional counting model (Balasubramanian et al. 2013).

Models: Tensor-based Event Composition

Event composition models produce vector representations of an event given its predicate and argument representations. To be broadly useful, a representation should have two properties: (i) Events that are usually part of the same underlying scenario should get similar representations – that is they should be embedded close together in the event space. (ii) Events that occur in different scenarios should get different representations, even if they have similar lexical expressions. In our earlier example, we want *she threw football* and *she threw bomb* to be farther apart in the embedding space, even though they share the subject and the verb.

An effective event composition model must be able to account for the numerous usage contexts for a given predicate and should be able to invoke the appropriate usage context based on the arguments. This problem is compounded even further when considering predicates that take on many different meanings depending on their arguments. This requires that the composition models be sensitive to small changes in the predicate argument interactions. Simple averaging or additive transformations are not enough.

One way to capture this rich interaction is through tensor composition. Tensor-based composition has been used for composing over tree based structures (Socher et al. 2013b) and in general for modeling words with operator like semantics (Grefenstette et al. 2013; Fried, Polajnar, and Clark 2015). A key benefit of tensor composition that is relevant to our setting is that they capture multiplicative interactions of all the elements involved in the composition, which allows for the composition to be sensitive to even small changes in predicate argument interaction.

Formally, we can define a tensor-based event composition model as follows. Given a predicate-specific tensor P, and the embeddings of the subject s and the object o, the representation for the event e can be composed through tensor contraction denoted as $e = P(s, o)$. Each element in the d'-dimensional event vector is obtained by the multiplicative interaction of all elements of the subject and object vectors, weighted by a term that depends on the predicate. The ith component of e is computed as:

$$e_i = \sum_{j,k} P_{ijk} s_j o_k \tag{1}$$

The key question with predicate-specific tensors is how to learn and reason with such a huge number of parameters.

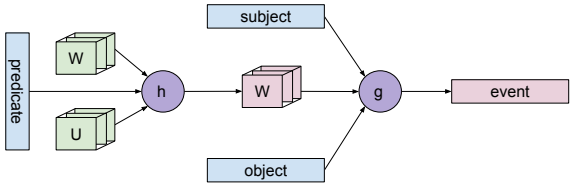

Figure 2: Predicate Tensor Model: Event representations are computed using a predicate tensor that combines the arguments. Elements in blue are inputs, green are model parameters, and pink are outputs generated by the model. Function h produces the predicate tensor using tensors W and U as shown in Equation 2. Function g produces the final event representation through tensor contraction as shown in Equation 3.

Prior work that models words as tensors are not scalable because the methods for learning such word-specific tensors require a large number of parameters, as well as a large number of training instances for each word. Relaxing the need for predicate specific training data and reducing the number of parameters motivate the two models we propose here.

Predicate Tensor Model

Rather than learn a predicate-specific tensor, we instead learn two general tensors that can *generate* a predicate-specific tensor on the fly. The predicate tensor model defines a function from a basic word embedding for the predicate to a tensor P. In this model, the predicate tensors are derived from a shared base tensor $W \in \mathbb{R}^{d \times d \times d}$ (where d is the input embedding dimension). To allow the predicate's word embedding to influence its resulting tensor, we allow each element of W (each one dimensional 'row' of W) to be scaled by a value that depends on a linear function of the predicate embedding p:

$$P_{ijk} = W_{ijk} \sum_a p_a U_{ajk} \tag{2}$$

Here U is also a tensor in $\mathbb{R}^{d \times d \times d}$ which defines linear functions for each one dimensional row of W, determining how p should scale that dimension. Now, given the predicate vector p, the subject vector s, and the object vector o, the original tensor contraction $P(s, o)$ we seek above is realized as follows. Each element in the resulting event vector e_i is computed as:

$$e_i = \sum_{a,i,j,k} p_a s_j o_k W_{ijk} U_{ajk} \tag{3}$$

Thus this model captures multiplicative interactions across all three: subject, verb, and object.

Role Factored Tensor Model

The representation power of the predicate tensor comes at the cost of model complexity and sensitivity. The ability to capture multiplicative interactions across all its arguments allows the predicate tensor to model complex relations between the predicate and its arguments. While this is useful from a representation stand point, it also makes the model

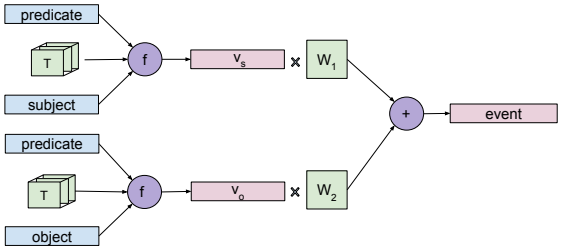

Figure 3: Role Factored Tensor Model: Uses two tensor contractions of the predicate with the subject and object, which are then combined. Elements in blue are inputs, green are model parameters, and pink are outputs generated by the model. Function f is the tensor contraction operation defined in Equation 6.

behavior more complex and highly sensitive – changes in the components of the event can drastically change the value of the event embedding.

In many cases, however, it may not be necessary to jointly model the interaction of a predicate with *all* of its arguments. Often times, the predicate's interaction with just one of its arguments is enough for recognizing its usage context. For example, knowing that *football* is an argument to *throw* immediately places the event in the context of sports.

Rather than model interactions between all components, we thus introduce a factored composition strategy: capture interactions between the predicate and its arguments separately to then combine these interactions into the final embedding. In this model, a single tensor $T \in \mathbb{R}^{h \times d \times d}$ (where h is the output dimension and d is defined as before) is used to capture these interactions. The resulting interactions are then combined using role-specific transformations[1].

Formally, let s, o, and p denote the word vectors for the subject, object, and the predicate respectively. The argument specific interactions are captured using two compositions of the predicate, one with the subject and one with the object:

$$v_s = T(s, p) \tag{4}$$

$$v_o = T(o, p) \tag{5}$$

As before, this composition is defined via tensor contraction. Given arbitrary arguments a and b, the element v_i of the output vector $T(a, b) = v \in \mathbb{R}^h$ is given by:

$$v_i = \sum_{j,k} T_{ljk} a_j b_k \tag{6}$$

The subject and object interactions v_s and v_o are then transformed via their respective role specific matrices, $W_s, W_v \in \mathbb{R}^{d' \times h}$ and summed together to obtain the final event embedding $e \in \mathbb{R}^{d'}$:

$$e = W_s v_s + W_o v_o \tag{7}$$

[1]This approach is similar to the application of recursive tensor composition (Socher et al. 2013b) but is not fully recursive in that the partial interactions are not combined using recursive tensor application. We found this additional recursion to be detrimental to performance.

Figure 3 displays the structure of the model. This factored model has fewer parameters compared to the Predicate Tensor Model and can also readily generalize to higher-arity events with any arbitrary number of arguments.

Training Tasks

In order to learn the parameters of the tensor composition models we employ two different training tasks, which correspond to predicting two different types of contexts.

Predict Events

One natural way to learn script knowledge is to directly learn how to predict what other events are likely to occur given a set of events (Pichotta and Mooney 2016; Modi 2016). We use an alternate learning formulation defined over a pair of events rather than over sequences of events. One goal is to learn representations that maximize some similarity measure between co-occurring events similar to the task defined by Granroth-Wilding and Clark (2016)[2].

Given some input event e_i, a target event e_t which occurs within in a window of w in front of e_i is randomly chosen. A negative event e_n is also randomly sampled from the entire corpus. The regularized learning objective is to minimize the following quantity:

$$\frac{1}{N} \sum_{i=1}^{N} max(0, m + sim(e_i, e_n) - sim(e_i, e_t)) + \lambda L(\theta)$$

where $sim(a, b)$ is cosine similarity, m is the margin, and $L(\theta)$ is $l2$ regularization on all model parameters θ.

Predict Words

A second training approach is to predict the nearby words of an event's sentence context, rather than just the event words. This is similar to most other word representation learning objectives. Formally the objective function to minimize is:

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{n_{e_i}} -log(P(w_{ij}|e_i)) + \lambda L(\theta)$$

where n_{e_i} is the number of words in the sentence that e_i appears in, and w_{ij} is the j^{th} word in the sentence for e_i. $P(w_{ij}|e_i)$ is computed via a softmax layer. The use of this type of inter sentence word prediction for finding embeddings for phrases has been shown to be useful (Le and Mikolov 2014).

Training Details

We use the New York Times Gigaword Corpus for training data. Event triples are extracted using the Open Information Extraction system Ollie (Mausam et al. 2012). We initialize the word embedding layer with 100 dimensional pretrained GloVe vectors (Pennington, Socher, and Manning

[2]The objective function in (Granroth-Wilding and Clark 2016) differs from ours in that they learn a coherence function which is used to measure similarity between events, rather than directly using cosine similarity.

2014). Like previous work (Modi 2016; Granroth-Wilding and Clark 2016), we allow the word embeddings to be further updated during training for all models.

We hold out 4000 articles from the corpus to construct dev sets for hyperparameter tuning, and 6000 articles for test purposes. We make all code and data publicly available.[3] Hyperparameter tuning was done using dev sets constructed for the CMCNC and MCNC tasks (see below). Training was done using Adagrad (Duchi, Hazan, and Singer 2011) with a learning rate of 0.01 and a minibatch size of 128.

Evaluation

We evaluate our proposed tensor models on a variety of event related tasks, comparing against a compositional neural network model, a simple multiplicative model, and an averaging baseline. Since our goal is to produce a event representation given a single event, and not a sequence as in Pichotta and Mooney (2016), RNN based models are not suitable for this task. We report results on both training tasks: predicting events (EV) and predicting words (WP).

Baselines

Compositional Neural Network (NN) Our first baseline is an neural network model used previously in recent work. The event representation in this model is computed by feeding the concatenation of the subject, predicate, and object embedding into a two layer neural network[4].

$$e = W * tanh(H[s; p; o])$$

where W and H are matrices, the main model parameters.

This basic architecture was used to generate event representations for narrative cloze tasks (Modi and Titov 2013; Modi 2016; Granroth-Wilding and Clark 2016). We adapt this architecture for our task – given the input event, whose representation is to be composed, predict the neighboring context (other events or words in the sentence). This deviates from the model used in (Modi 2016) mainly in training objective, and that the embedding of the protagonist's dependency is fed in as input (see the Related Works section for details). This architecture was also used in (Granroth-Wilding and Clark 2016) to obtain event representations, which are then fed to another network that measures coherence of the event representations. We only use the event representation part of this model and train it to maximize our training objectives.

Elementwise Multiplicative Composition The second baseline extends the additive composition model by simply concatenating the elementwise multiplications between the verb and its subject/object. This models some (though not all) of the multiplicative interactions. The embedding in this model is computed as:

$$e = W * tanh(H[s; p; o; p \odot s; p \odot o])$$

where $\odot$ denotes elementwise multiplication.

[3] github.com/stonybrooknlp/event-tensors

[4] We additionally tried a three layer network, however this slightly decreased performance on the dev set

System	ρ	
	WP	EV
Role Factor Tensor	**0.71**	**0.64**
Predicate Tensor	**0.71**	0.63
Comp. Neural Network	0.68	0.63
Elementwise Multiplicative	0.65	0.57
Averaging	0.67	

Table 1: Performance on the Transitive Sentence Similarity dataset, as indicated by Spearman's ρ

Averaging Baseline This represents each event as the average of the constituent word vectors using pretrained GloVe embeddings (Pennington, Socher, and Manning 2014).

Similarity Evaluations

Transitive Sentence Similarity Similarity tasks are a common way to measure the quality of vector embeddings. The transitive sentence similarity dataset (Kartsaklis and Sadrzadeh 2014a) contains 108 pairs of transitive sentences: short phrases containing a single subject, object, and verb (e.g., *agent sell property*). Every pair is annotated by a human with a similarity score from 1 to 7. For example, pairs such as *(design, reduce, amount)* and *(company, cut, cost)* are annotated with a high similarity score, while pairs such as *(wife, pour, tea)* and *(worker, join, party)* are given low similarity scores. Since each pair has several annotations, we use the average annotator score as the gold score. We evaluate using the Spearman's correlation of the cosine similarity given by each model and the annotated similarity score.

Hard Similarity Task The main requirement we laid out for event composition is that similar events should be embedded close to each other, and dissimilar events or those from distinct scenarios should be farther from each other. We create a hard similarity task to explicitly measure how well the representations satisfy this requirement. To this end, we create two types of event pairs, one with events that should be close to each other but have very little lexical overlap (*e.g., police catch robber / authorities apprehend suspect*), and another with events that should be farther apart but have high overlap (*e.g., police catch robber / police catch disease*.

A good composition model should have higher cosine similarity for the similar pair than for the dissimilar pair. To evaluate this directly we created 230 pairs (115 pairs each of similar and dissimilar types). To create the set, we have one annotator create similar/dissimilar pairs from a set of ambiguous verbs, while three different annotators give the similarity/dissimilarity rankings. We kept pairs where the annotators agreed completely. For each composition method, we obtain the cosine similarity of the pairs under that representation, and report the fraction of cases where the similar pair receives a higher cosine than the dissimilar pair.

Results Table 1 shows the Spearman's ρ scores of the various models on the transitive similarity task. Consistent with prior work (Milajevs et al. 2014), we find that simple averaging is a competitive baseline for the task. The neural network baseline is only slightly better than averaging. When

System	Accuracy	
	WP	EV
Role Factor Tensor	23.5	**43.5**†
Predicate Tensor	**35.7**†	41.0 †
Comp. Neural Network	20.9	33.0
Elementwise Mult.	11.3	33.9
Averaging	5.2	

Table 2: Hard Similarity Results: Accuracy is the percentage of cases where the similar pair had higher cosine similarity than the dissimilar pair.

System	CMCNC		MCNC	
	WP	EV	WP	EV
Role Factor Tensor	**70.1**†	**72.1** †	**42.2**	**46.5** †
Predicate Tensor	64.5	66.1	32.3	41.1
Comp. Neural Network	65.7	68.5	38.4	45.3
Elementwise Mult.	67.3	67.7	41.7	45.4
Averaging	26.7		14.3	

Table 3: CMCNC Results: Predict the held out event, given the observed events from the same document. The dataset is curated manually for coherence by removing noisy instances and context events. † denotes a statistically significant difference($\alpha < 0.05$) over the best competing baseline under a paired t-test.

training for word prediction, both the predicate and the role factored tensor models have a higher correlation (+3 points in ρ). Across all models, training for the word objective is better than training for the event objective. This is to be expected since the word prediction task focuses on a narrower context, requiring that representation to be consistent with information within the sentence in which it appears. This goal aligns well with the sentence similarity task where the events that are roughly exchangeable are deemed similar.

Table 2 compares the percentage of cases where the similar pairs had a higher cosine similarity than dissimilar pairs under each event composition method. Both tensor methods outperform the baselines showing that tensors capture a richer and broader set of semantics about the events. Interestingly, the event objective is better than word prediction for all models. We believe the broader context used by the event prediction objective helps capture broader event semantics necessary to generalize to the difficult pairs.

Coherent Multiple Choice Narrative Cloze

The above experiments judge event similarity, but our broader goal is to model real-world knowledge. The *narrative cloze task* was proposed to evaluate script knowledge and knowledge about events that occur together in the world (Chambers and Jurafsky 2008). The task starts with a series of events that are mentioned in a document, but hides one of the events. A reasoning system should predict what the held out event is, given only the other events. Event predication is typically done by choosing an event such that it maximizes some similarity score with the context events.[5] This formulation is difficult since the output space of possible events is rather large. We evaluate on variant called the multi-choice narrative cloze (MCNC) (Granroth-Wilding and Clark 2016), where the system should distinguish the held out event from a small set of randomly drawn events.

However, using automatic cloze evaluation has multiple issues (Chambers 2017), one of which is that it is overly sensitive to frequency cutoffs of common events (e.g. *said*, *was*, *did*, etc.) and errors in the preprocessing tools. To address these, we manually curated an evaluation set for the MCNC task, and call it the Coherent Multiple Choice Narrative Cloze (CMCNC). We generated an MCNC dataset and made three modifications.

[5]We use cosine similarity as our measure

First, we manually removed events that are either: (1) frequent events types that are in our stop event list , or (2) nonsensical events coming from obvious extraction errors .

Second, we discard heldout events that don't fit the following criteria: (i) One of its entities must appear in the given context events. (ii) Given the context events, the held out event should seem plausible to a human evaluator.
Third, when the heldout event is included with 5 randomly selected negative events, we replace the entities appearing in the negative events with similar (under word embedding distance) entities that appear in the context events.

Enforcing these constraints allows us to better evaluate how well the system learned commonsense script knowledge, rather than how well the system learned to emulate noise in the data. For comparison, we also report on an automatically generated MCNC task. Since we do not restrict the evaluation to narrative chains with a single protagonist, the numbers for the automatic MCNC task are lower than those reported in Granroth-Wilding and Clark (2016).

Results Table 3 show the results for the manually filtered CMCNC and the automatic MCNC tasks. Whereas the word-based objective excelled on similarity tasks, the event-based objective does better on cloze tasks. The Role Factor model shows significant improvements in accuracy compared to the neural network model (+4.4 points with word prediction and +3.6 points with event prediction). The Predicate tensor model, however, performs worse than the neural network model. The human-curated evaluation shows vastly different results than the automatic-MCNC, perhaps bolstering the idea that MCNC is not ideal (Chambers 2017). Our Role Factor model excels in both.

Generating Event Schemas

An event schema is a form of script like knowledge about a scenario (e.g., a *bank heist*). Chambers and Jurafsky (2009) introduce schemas as a set of the main entities in the scenario (the *robber, bank teller, etc.*) and the main events they participate in (*robber steals money, teller calls police etc.*).

Prior work on event schema generation use discrete representations of events and build count-based conditional models over these representations c.f., (Chambers and Jurafsky 2009; Balasubramanian et al. 2013). The basic idea behind

Role Factored

Arg 1	Predicate	Arg 2
X	stabbed	Y
Z	identify	Y
Z	recovered	machete
Z	offered	reward
X	arrested in	night

Relgrams

Arg 1	Predicate	Arg 2
X	stabbed	Y
X	entered	study
Z	recovered	machete
X	arrested in	night
X	ate	dinner

Figure 4: Example schemas from nearest neighbors and relgrams approach. X, Y, and Z are variables that denote distinct entities. Each row indicates the main event that the entities participate in.

these methods is to start with a seed event and locate other events that are also likely to occur in the scenario. These methods suffer fragmented counts because of synonymy (different tuples can denote the same event) and mixing events from different contexts because of polysemy (similar looking tuples denote distinct events). Continuous event representation provides an opportunity to address these issues.

The tensor-based representations yields a simpler and more direct approach to generating event schemas. Recall that with the predict events objective, events that tend to co-occur with each other end up with a similar representation and get embedded closer to each other in the event space. To create a schema seeded by an event, we can simply find its nearest neighbors in the event embedding space and use them as candidates. This method is simple, scalable, and does not require maintaining large tables of co-occurrence information (c.f. Balasubramanian et al. (2013)).

Nearest Neighbor Schema Generation Like previous systems, our system takes as input a single seed event and produces the schema based on this event. Given a seed event s the algorithm proceeds as follows:

- From a corpus of events, compute their representations using one of the composition methods described above
- Find the k nearest neighbors to s in the text corpus with respect to the cosine distance of their representations.
- Go through this list of nearest neighbors, add a neighbor x to the schema if all the following conditions are true:
  - The cosine distance between the GloVe embedding of x's predicate and all other predicates currently in the schema is greater than α
  - The cosine distance between the GloVe embedding of at least one of x's arguments and some entity e in the schema is less than β. The argument is then replaced with e to create a new event x'
  - The average cosine distance between the representation of x' (computed using the same composition function used to compute the original representations) with all other events in the schema is less than γ

The end result is a series of events representing a schema. For values of α, β and γ, we use $\alpha = 0.5, \beta = 0.25, \gamma = 0.2$ (for Role Factor), and $\gamma = 0.3$ (for Neural Network). Values were selected using a develoment set of 20 seeds.

System	Average Score	
	With 0's	Without 0's
Role Factor (EV)	**2.45**	**2.62**
Comp. Neural Network (EV)	2.26	2.47
(Balasubramanian et al. 2013)	1.51	1.78

Table 4: Average Annotator Scores for generated schemas (scale from 0-4). Higher scores indicate a more coherent schema.

Schema Evaluation

We compare against previous work that produces schemas using relation co-occurrence graphs (Balasubramanian et al. 2013). It also uses OpenIE triples mined from a text corpus. We refer to this as the *Relgrams approach*. For a fair comparison, we choose 20 seed tuples at random from the list of top seeds in (Balasubramanian et al. 2013). For each seed, we generate a schema of 10 grounded events using our best models. We report results using the Role Factor (EV) representations, the Compositional Neural Network (EV) representations, and the Relgrams approach.

Our evaluation is human-driven. We present annotators with 3 schemas (from the 3 models) using the same seed. For each event in each schema, annotators rated the event on a 0-4 scale with regards to its relevance to the seed and the rest of the schema. A score of 0 is reserved for events that are completely nonsensical (either caused by an extraction error, or bad entity replacement). A score of 1 indicates that the event was not relevant with no obvious relation to the scenario, while a score of 4 indicates the event is highly relevant and would be a core part of any description of the scenario. Table 4 shows the average scores given for each method. We report average ratings both with and without nonsensical events[6].

Both nearest neighbor schemas consistently out-ranked the Relgrams schemas by a large margin. Figure 4 shows the schemas produced by both systems using the same seed. The biggest problem with the schemas produced by the Relgrams method was their tendency to include overly specific triples in the schemas, simply because they co-occur with one of the events in the schema. The schemas based on continuous representations can avoid this problem. Here is a typical example: the event *(police, found, machete)* co-occurs a couple times in an article with *(he, likes to eat, dinner)*. Since *(police, found, machete)* is rare in the corpus, the count based system takes into consideration the co-occurrence. Although tuples similar to *(police, found, machete)* such as *(authorities, recovered, murder weapon)* may appear in the corpus, the counts based on discrete representations cannot share evidence across these similar tuples. The method based on continuous representations can take advantage of similarity of words in the two tuples. The tensor model further aids by helping the system better differentiate between events with many similar words, but different meanings.

While the Compositional Neural Network and Role Factor representations both perform well compared to the Rel-

[6] All differences in means are significant under t-test w/ $\alpha < .05$

context 1	context 2	context 3	context 4	context 5
hitting	police	shouted	farve	lodged
inning	marchers	chanted	elway	complaint
walked	chechens	chanting	yards	filed
hit	stoned	yelled	rookie	remand
fielder	protesters	shouting	broncos	lawsuit

Table 5: Nearest Neighbors for the implicitly learned contexts in which 'throw' may be used.

grams approach, the Role Factor factor does better, suggesting that the model's ability to tease out contextual usages of predicates can be useful in end tasks.

Discussion

This is the first paper to propose and evaluate event models on both event similarity and event schema prediction/generation. We believe there is significant overlap in goals (any event representation needs to learn a nearness between similar events), but there are also differences (script learning is not at all about synonymy).

Interpreting the Role Factor Model

The Role Factor model consistently outperforms on all tasks, in particular over additive/concatenation models from recent work in event learning. One question to ask is what exactly is the Role Factor model learning?

Besides simply allowing multiplicative interactions between the predicate and its arguments, the Role Factored model can be interpreted as capturing different scenarios or contexts in which a predicate is used. First, observe that performing the contraction to compute $v = T(a, p)$ can be done by first partially applying T to p, producing a matrix P, whose columns P_i are the result of multiplying each slice T_i in T by the predicate vector p. Thus, each slice T_i moves the predicate vector to a new point p' in the original word embedding space. If there are distinct usages for a predicate p, we hope that the slices of T provide a map from p to those usages in word embedding space.

Table 5 shows the nearest neighbors for several different P_i vectors for the verb *throw*. Each of these nearest neighbor clusters indicate what type of arguments (either subject or object) should 'activate' the i^{th} dimension, reflecting the type of event contexts that tensor T captures.

Script Generation

This is the first attempt to generate event scripts (or schemas) from continuous representations. Recent work focuses on event language models that can perform narrative cloze, but to our knowledge, the goal of the learning explicit scripts has been left behind. Our human evaluation of generated scripts shows that nearest neighbor selection with coherent entities builds far superior scripts than previous work. Figure 4 gives an example of a generated schema from both systems.

Related Work

Neural Event Representations Neural event embedding approaches learn representations by training to predict the next event. Granroth-Wilding and Clark (2016) concatenate predicate and argument embeddings and feed them to a neural network to generate an event embedding. Event embeddings are further concatenated and fed through another neural network to predict the coherence between the events. Modi (2016) encode a set of events in a similar way and use that to incrementally predict the next event – first the argument, then the predicate and then next argument. Pichotta and Mooney (2016) treat event prediction as a sequence to sequence problem and use RNN based models conditioned on event sequences in order to predict the next event. These three works all model *narrative chains*, that is, event sequences in which a single entity (the *protagonist*) participates in every event. Hu et al. (2017) also apply a RNN approach, applying a new hierarchical LSTM model in order predict events by generating discriptive word sequences.

Multiplicative and Tensor composition models Tensor based composition models have been shown to be useful for other NLP tasks such as sentiment analysis (Socher et al. 2011; 2013b), knowledge base completion (Socher et al. 2013a), and in general for demonstrating compositional semantics in measuring sentence similarity (Grefenstette et al. 2013; Kartsaklis and Sadrzadeh 2014b; Fried, Polajnar, and Clark 2015; Kim, de Marneffe, and Fosler-Lussier 2015).

Polajnar, Rimell, and Clark (2015) learn verb specific tensors by setting up a regression task where the learned tensor for a verb is expected to yield a representation for the sentential contexts where the verb is found. Huang et al. (2016) use tensor based autoencoders over AMR representations of events in order to induce event level (rather than scenerio level), ACE[7] style templates for use in event extraction. Similarly, Huang et al. (2017) use tensor compositions to build representations that facilitate zero shot relation extraction. Related to our hard similarity task, Tilk et al. (2016) use (factorized) tensor based methods to model the thematic fit between event arguments.

Schema/script learning Unsupervised learning of script knowledge can be traced back to (Chambers and Jurafsky 2008), which introduced a count based technique for inducing narrative chains. Chambers and Jurafsky (2009) extends this idea to creating full on narrative schemas by merging together narrative chains with argument overlap. Other unsupervised induction approaches include a relation n-gram based method (Balasubramanian et al. 2013), and generative latent variable models (Nguyen et al. 2015; Chambers 2013; Cheung, Poon, and Vanderwende 2013). All of these models worked on discrete representations for capturing co-occurrence statistics. In this work, we show that higher quality scripts can be produced using continuous representations instead.

Conclusions

Understanding events requires effective representations that contain both information that is specific to the event and in-

[7] www.itl.nist.gov/iad/mig/tests/ace/

formation that relates to the underlying context in which the event occurs. We propose tensor-based composition models which are able to capture the distinct event contexts in which a predicate gets used. This improved modeling of these event contexts allows the resulting continuous representations to be more effective in multiple event related tasks. Last, we also show that the continuous representations yield a simple schema generation method which produces schemas superior to a prior count based schema generation method.

Acknowledgements

This work is supported in part by the National Science Foundation under Grant IIS-1617969.

References

Balasubramanian, N.; Soderland, S.; Mausam, O. E.; and Etzioni, O. 2013. Generating coherent event schemas at scale. In *EMNLP*, 1721–1731.

Chambers, N., and Jurafsky, D. 2008. Unsupervised learning of narrative event chains. In *ACL*, 789–797.

Chambers, N., and Jurafsky, D. 2009. Unsupervised learning of narrative schemas and their participants. In *ACL*, 602–610.

Chambers, N. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*, 1797–1807.

Chambers, N. 2017. Behind the scenes of an evolving event cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, 41–45.

Cheung, J. C. K.; Poon, H.; and Vanderwende, L. 2013. Probabilistic frame induction. *arXiv preprint arXiv:1302.4813*.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 2121–2159.

Fried, D.; Polajnar, T.; and Clark, S. 2015. Low-rank tensors for verbs in compositional distributional semantics. In *ACL*, 731–736.

Granroth-Wilding, M., and Clark, S. 2016. What happens next? event prediction using a compositional neural network model. In *AAAI*, 2727–2733.

Grefenstette, E.; Dinu, G.; Zhang, Y.-Z.; Sadrzadeh, M.; and Baroni, M. 2013. Multi-step regression learning for compositional distributional semantics. In *IWCS*.

Hu, L.; Li, J.; Nie, L.; Li, X.-L.; and Shao, C. 2017. What happens next? future subevent prediction using contextual hierarchical lstm. In *AAAI*, 3450–3456.

Huang, L.; Cassidy, T.; Feng, X.; Ji, H.; Voss, C. R.; Han, J.; and Sil, A. 2016. Liberal event extraction and event schema induction. In *ACL*.

Huang, L.; Ji, H.; Cho, K.; and Voss, C. R. 2017. Zero-shot transfer learning for event extraction. *arXiv preprint arXiv:1707.01066*.

Kartsaklis, D., and Sadrzadeh, M. 2014a. A study of entanglement in a categorical framework of natural language. In *Proceedings of the 11th workshop on Quantum Physics and Logic, QPL*, 249–261.

Kartsaklis, D., and Sadrzadeh, M. 2014b. A study of entanglement in a categorical framework of natural language. *arXiv preprint arXiv:1405.2874*.

Kim, J.-K.; de Marneffe, M.-C.; and Fosler-Lussier, E. 2015. Neural word embeddings with multiplicative feature interactions for tensor-based compositions. In *VS@ HLT-NAACL*, 143–150.

Le, Q. V., and Mikolov, T. 2014. Distributed representations of sentences and documents. *CoRR* abs/1405.4053.

Mausam; Schmitz, M.; Bart, R.; Soderland, S.; and Etzioni, O. 2012. Open language learning for information extraction. In *EMNLP-CoNLL*, 523–534.

Milajevs, D.; Kartsaklis, D.; Sadrzadeh, M.; and Purver, M. 2014. Evaluating neural word representations in tensor-based compositional settings. *CoRR* abs/1408.6179.

Modi, A., and Titov, I. 2013. Learning semantic script knowledge with event embeddings. *CoRR* abs/1312.5198.

Modi, A. 2016. Event embeddings for semantic script modeling. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

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

Nguyen, K.-H.; Tannier, X.; Ferret, O.; and Besançon, R. 2015. Generative event schema induction with entity disambiguation. In *ACL*, 188–197.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*, 1532–1543.

Pichotta, K., and Mooney, R. J. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *AAAI*.

Polajnar, T.; Rimell, L.; and Clark, S. 2015. An exploration of discourse-based sentence spaces for compositional distributional semantics. In *Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem)*, 1.

Schank, R. C., and Abelson, R. P. 1977. *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. Hillsdale, NJ: L. Erlbaum.

Socher, R.; Pennington, J.; Huang, E. H.; Ng, A. Y.; and Manning, C. D. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, 151–161.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013a. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, 926–934.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 1631–1642.

Tilk, O.; Demberg, V.; Sayeed, A.; Klakow, D.; and Thater, S. 2016. Event participant modelling with neural networks. In *EMNLP*, 171–182.