

FEEL: Featured Event Embedding Learning

I-Ta Lee and Dan Goldwasser

Purdue University

lee2226@purude.edu, dgoldwas@purdue.edu

Abstract

Statistical script learning is an effective way to acquire world knowledge which can be used for commonsense reasoning. Statistical script learning induces this knowledge by observing event sequences generated from texts. The learned model thus can predict subsequence events, given earlier events. Recent approaches rely on learning event embeddings which capture script knowledge.

In this work, we suggest a framework for injecting event embedding with fine grained information. We suggest a general learning model—Featured Event Embedding Learning (FEEL). In addition to capturing the dependencies between subsequent events, our model can take into account **higher level abstractions** of the input event which help the model **generalize better** and account for the global context in which the event appears.

We evaluated our model over three narrative cloze tasks, and showed that our model can outperform the current state-of-the-art. We also show that our resulting embedding can be used as a strong representation for advanced semantic tasks such as discourse parsing.

Introduction

Many natural language understanding tasks rely on world knowledge. Such knowledge can help support common sense reasoning and provide the context needed for disambiguating text. *Scripts*, introduced by (Schank and Abelson 1977), are structured knowledge representations capturing the **relationships** between prototypical event sequences and their participants. Scripts model our expectations about the relevant causal relationships between events, and as a result can be used to infer how events will unfold in a given scenario. For example, given the event *John shot Jim with a gun*, we can infer that *he got arrested by police* is more probable than *he fell asleep*. Scripts provide the foundation for automatically making such inferences, supporting semantic tasks such as co-reference resolution, discourse parsing and question answering.

As the example above suggests, predicting “*what happens next?*”, also known as the narrative cloze task (Chambers and Jurafsky 2008; Granroth-Wilding and Clark 2016), is the preferred way of evaluating such models. In this paper, we

propose a generalization of this task highlighting the importance of evaluating inferences over chains of future events. We look into two variants of this task, **the first predicts future events (one or more)**, and the second predicts an **explanation, connecting the beginning and end of a longer narrative chain**. The following example describes a simplified version of these tasks.

Narrative Cloze

Jenny went to a restaurant and ordered a lasagna plate. Jenny liked the food and felt satisfied.

Which of the following events could happen *next*?

- a) *She scolded the server.*
- b) *She fell asleep.*
- c) *She left a big tip.*
- d) *She ran out of battery.*

Narrative Explanation

Jenny went to a restaurant and left a big tip.

Which of the following events chains *explain* what happened?

- a) *She ordered her food and liked it.*
- b) *She hated the food and left angry.*
- c) *She walked to a bus station and got on a bus.*

Humans can easily identify that *c)* and *a)* would be the correct answers. However, for machines, making such inferences requires understanding the events, their properties and their implications.

While early works focused on manual construction of script knowledge, the difficulty of scaling these methods to realistic domains, has ignited considerable interest in statistical script learning methods (Chambers and Jurafsky 2008; Modi and Titov 2014; Rudinger et al. 2015; Pichotta and Mooney 2016b; Peng and Roth 2016; Granroth-Wilding and Clark 2016; Modi et al. 2017).

We build on the work of (Chambers and Jurafsky 2008) which used **pairs of event predicates and dependency** in-

formation (corresponding to the *subject/object* dependency links) to represent events and formed event chains based on **co-reference relationships** between these pairs. Their model assumed a discrete event representation, and computed the **Pairwise Mutual Information (PMI)** between event pairs, in order to support inference tasks. Following the surge of interest in distributed representations of discrete objects (Mikolov et al. 2013), most recent approaches represent events using dense continuous vectors, known as **event embedding**. For example, (Pichotta and Mooney 2016a; Granroth-Wilding and Clark 2016) both proposed a neural network model that composes event embeddings with their predicate, dependency, and argument information (subject, object, and prepositional object), either using a feed-forward architecture defined over pairs of events, or using a Recurrent architecture (in this case an LSTM) **to capture the dependencies between longer sequences of events**.

In this paper we contribute to this body of work, and introduce **FEEL - Featured Event Embedding Learning**. Our model is designed to **capture fine-grained event properties**, that can be exploited to **reduce ambiguity** when inferring future events. For example, the **sentiment polarity** of a given event (e.g., “*Jenny liked the food*” implies positive sentiment), can impact the probability of options for future events (e.g., the probability of negative-sentiment events, such as *a*) should decrease). The **animacy** of the event’s arguments can also provide valuable information, as some actions can only be performed by living entities, and some events change meaning when taking inanimate objects as arguments (e.g., “*this song is sick!*” vs. “*this person is sick!*”). In our example above, option *d*) can be ruled out based on this information.

We focused on these **two types of features**, however there are many other event properties which can be useful for language understanding. Our goal is to provide a general framework for including such information. Specifically, our model makes three contributions.

(1) *Novel Neural Architecture for Event Learning*. We suggest to set up learning for event-embedding as multi-task representation learning. The joint objective function combines both **intra-event learning objectives** (for example, representing prototypical connections between arguments and predicates, such as *policeman* and *arrest*), and **an inter-event learning objective which captures prototypical connections between event chains**.

(2) *Features Enriched Event Embedding* Our architecture provides a highly **flexible framework** for injecting world knowledge and relevant contextual information needed to accurately represent events. **This information is injected into the embedding function learning step, resulting in a richer event representation**. We specifically looked into **higher level abstractions of events** - the overall sentiment of the event and animacy information of the event arguments.

(3) *Structured Event Chains Evaluation* We evaluated our model in several settings, using both intrinsic and extrinsic tasks. We followed the evaluation settings of (Granroth-Wilding and Clark 2016), and showed that using the same resources, our architecture leads to improved performance. Our feature-enriched model resulted in further improve-

ment. Since looking at single event transitions can fall short of evaluating full scripts, we also defined two additional narrative inference tasks over event sequences. Finally, we evaluated our model by using the generated representation as features for two semantic prediction tasks.

Related Work

Early works conducted in the 1970s defined a script as a sequence of structured events organized in temporal order (Schank and Abelson 1977). While these early works provided us with the core concepts of script knowledge, the manual methods employed were difficult to scale to complex domains. Interest in script learning was revived by (Chambers and Jurafsky 2008)’s work, which introduced a statistical approach to obtaining script knowledge. This work proposed an unsupervised framework to model event sequences. Using existing NLP tools, such as a Coreference resolution system and a dependency parser, narrative event chains following mentions of an entity in narrative text were automatically extracted, and **relationship between events** was computed using the **Point-wise Mutual Information (PMI) model**. The model’s ability to capture commonsense knowledge was evaluated using the Narrative cloze task— trying to recover a missing event given a sequence of preceding events. More recent works (Chambers 2017; Rudinger et al. 2015) extend the definition of events to include more information.

The narrative cloze task was refined by (Granroth-Wilding and Clark 2016) to include a closed set of options for replacing the missing event. In this paper, we **recreate their settings, but also introduce to additional intrinsic evaluation tasks—MCNS and MCNE**. These tasks were designed to evaluate the model’s ability to **infer longer events sequences**, which better account for narrative structures. This evaluation approach joins other recent attempts to include reasoning over narrative structures as part of script knowledge evaluation (Modi et al. 2017).

Multiple neural-network-based models were proposed to improve the quality of modeling commonsense event patterns. These works suggest replacing the symbolic event representation used by (Chambers and Jurafsky 2008) with a dense vector representation. (Granroth-Wilding and Clark 2016) applied Word2Vec (Mikolov et al. 2013) and a compositional neural network to learn event embeddings on narrative event chains, where **each event token in the chain is either a predicate word or an argument word**. (Pichotta and Mooney 2016a) proposed a Long Short Term Memory Recurrent Neural Networks (LSTM-RNN), coupled with Beam Search algorithm, which conditions the event representation on longer sequences of previous events.

The fact that argument information is very useful to learn better event embeddings was implicitly agreed in these works. Other works such as (Ahrendt and Demberg 2016) explicitly identified this fact (the term “participants” is used in their work instead of “arguments”). This inspires us to find **richer event properties, such as arguments, sentiment, and animacy, or event event time and location information**, can potentially improve event representations. These can act

as event modifiers and should be considered in script learning models. Our multi-task learning approach to embedding has been previously explored when constructing social embedding (Li, Ritter, and Jurafsky 2015), where the authors learned embeddings for users co-located in a network graph and their properties. FEEL follows this direction and develops such extensions for general statistical script learning, which can take rich event properties into consideration.

Model

Model Overview

From a high level perspective, learning for narrative event models can be broken down into two phases.

First, large amounts of narrative text are **preprocessed** and **event chains are extracted**. **Early systems** (Chambers and Jurafsky 2008) used a **dependency parser** (for connecting verbs and their typed arguments, resulting in a *(predicate, dependency.type)* event representation) and a **co-reference resolution system** (for forming chains with the same protagonist). For example, “*Jessie killed a man. She was arrested.*” has an event chain *(kill, subj), (arrest, obj)* for the protagonist *Jessie*. **Later systems** (Granroth-Wilding and Clark 2016; Pichotta and Mooney 2016a) also included the **argument words, as well as prepositional phrases**.

Second, these chains are used for training statistical script models. Initially by computing the PMI between events (Chambers and Jurafsky 2008) capturing event co-occurrence statistics. Later systems constructed event embedding, connecting event tokens with their argument information to form the event representation.

FEEL follows this setup, but also **adds an event property extraction step** in between, which helps inform the training process. In the following subsections we describe the three phases, (1) Narrative Event Chain Extraction, (2) Event Property Extraction, and (3) Model Training, in FEEL.

Narrative Event Chain Extraction

We first preprocess the text using Stanford CoreNLP (Manning et al. 2014), extract **dependency parses and co-reference chains**. We follow the co-reference chain to form the event chains, by associating each entity mention in the chain with an event defined as a tuple *(tok(e), subj(e), obj(e), prep(e))*, where *tok(e) = (predicate, dependency.type)* is a token generated by concatenating the predicate and its dependency relation to the protagonist of the event *e*; *subj(e)*, *obj(e)*, *prep(e)* are the subject word, object word, prepositional object word respectively of the event *e*.

All the words are in lower-case and lemmatized, and we represent multi-word noun phrases, using their head word. For example the chain associated with *Jenny* in the sentence “*Jenny went to a restaurant and ordered a lasagna plate*”, will be *(go, subj, jenny, NONE, restaurant), (order, subj, jenny, plate, NONE)*

Several additional detailed processes are listed below:

- Predicates are not limited to verbs, and include predicative adjectives, that can provide important causal information. For example, *Jame was hungry. He ate a burger.*

- Verbs such as *go*, *have* and *get*, are too *weak* to express nuanced event semantics. We address this issue by including their particles and clausal complements (xcomp) in the predicate representation. For example, *go to sleep* will be represented as one token *go_to_sleep*.
- We include negation in the predicate representation. For instance, *didn’t enjoy hiking* is represented as *not_enjoy_hike*.
- The possible dependencies *d(e)* are limited to *subject, object, and indirect object*.
- To avoid that frequent events dominate the distribution, we remove the 10 most frequent predicates and their events from the extracted event chains.

Event Property Extraction

FEEL provides a general framework for including event properties into its representation. Our first step is to include the argument information as a type of event properties, however the true strength of the model is in modeling higher level abstractions of events. In this work we focused on two abstractive properties, sentiment which captures the overall tone surrounding the event, and argument animacy information which can help identify nuanced language use, such as idiomatic expressions. Our results show that the contribution of the different properties depends on the specific task. We designed our framework to incorporate additional types allowing users to adapt their embedding to their specific task.

To **incorporate sentence level sentiment information**, we use Vader sentiment analyzer (Hutto and Gilbert 2014) from NLTK (Bird, Klein, and Loper 2009). The raw sentiment scores range from -1 (negative) to 1 (positive). We discretize the scores into sentiment labels—**Negative, Neutral, and Positive**—by setting up two thresholds on -0.5 and 0.5. Animacy information is added by **observing the animacy of the argument associated with the event token**. There are three possible animacy types: **animate, inanimate, or unknown**.

When adding the two event properties, the event 4-tuple is re-written as a 6-tuple *(tok(e), subj(e), obj(e), prep(e), f₁(e), f₂(e))*, where *f₁(.)* refers to the event sentiment and *f₂(.)* refers to the event animacy.

Model Training

As illustrated in Fig. 1 FEEL uses a hierarchical multi-task model for constructing the event representation. FEEL jointly learns for an inter-event (contextual) objective and several intra-event (local) objectives.

The inter-event objective, defined over two events *e₁* and *e₂*, captures the dependencies between subsequence events in a given narrative. **This objective can capture the relationship between the object of *stab* and the subject of *injured*.**

The intra-event objective defined over the event properties, namely *tok(.)*, *subj(.)*, *obj(.)*, *prep(.)*, *f₁(.)*, and *f₂(.)* are the extractors for event tokens, subjects, objects, prepositional objects, sentiments, and animacy respectively. Each will learn an embedding, represented in the bottom embedding layer in Fig. 1. This formulation allows the different properties and the event token to share information.

The inter-event and intra-event objectives are combined in a global objective function.

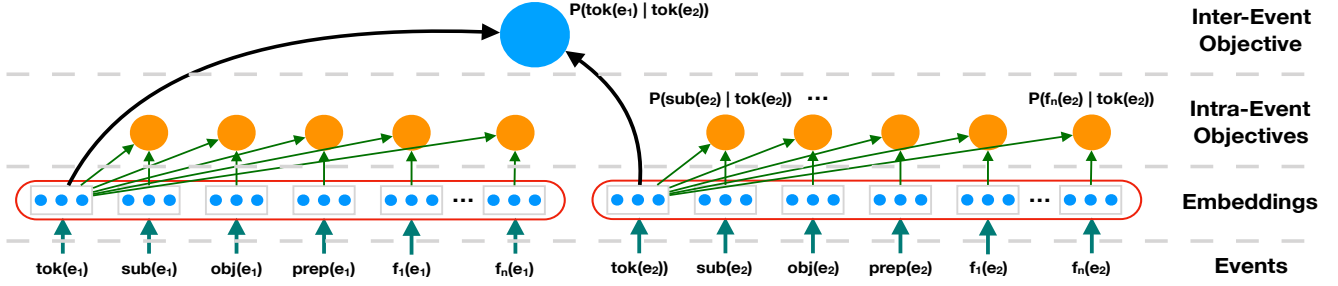


Figure 1: FEEL Model Objectives

We use the **Skip-gram objective function** (Mikolov et al. 2013) to learn the inter-event objective, defined as follows:

$$\begin{aligned} p(C(e)|e) &= \prod_{e' \in C(e)} p(e'|e) \\ &= \prod_{e' \in C(e)} \frac{\exp(v_{e'} \cdot v_e)}{\sum_{e^* \in E} \exp(v_{e^*} \cdot v_e)}, \end{aligned} \quad (1)$$

where e is the current event, $C(e)$ is the context events in a pre-defined window size k , E is all event vocabularies, and v_e is the vector representation of the event e . This probability can be learned by minimizing the margin-based ranking loss:

$$L_C(e) = \sum_{e' \in C(e)} \sum_{e^* \notin C(e)} \max(0, \delta - v_e \cdot v_{e'} + v_e \cdot v_{e^*}), \quad (2)$$

where δ is the margin. The (e, e') pair can be regarded as a positive example pair and (e, e^*) can be regarded as negative example pair. The amount-of, and proportion-between, positive and negative pairs is a hyperparameter defined by the window size k and the negative sampling ratio r respectively¹.

The intra-event objectives model local information for each event independently. Each property is trained with the base event token $tok(e)$, which bias the learned embeddings to become more similar if the property tends to occur together with the event.

The loss function for that objective is similar to Equation (2), but now the e' refers to a positive event property while the e^* refers to an sampled negative property.

FEEL jointly learns for all the objectives by taking a weighted summary:

$$\mathcal{L}(e) = \sum_{i \in \{C, S, O, P, T, A\}} \lambda_i L_i(e) + \lambda_r \|p\|_2, \quad (3)$$

where $L_C(e)$ means the inter-event (context) objective of the event e . L_S , L_O and L_P refer to the intra-event objectives between the e and its subject, object and prepositional object, respectively. L_T refers to the local objective between e and sentence-level sentiment and finally, L_A refers to the local objective between the e and the entity animacy. The

¹In our experiment, $k = 5$ and $r = 10$

hyperparameter λ_r is the weight for the $\|\cdot\|$ regularization and p refers to all the trainable parameters².

In addition to embedding the event token (i.e., (predicate, dependency) pair), our model constructs an embeddings for each event property. Users can determine how to aggregate these embeddings. For simplicity, to show the model's effectiveness, we simply concatenate all of the embeddings generated by an event tuple.

Experiments

We train our event embedding model over the New York Times (NYT) section of English Gigaword³. It contains about 2M documents of newswire text data and about 1.4M words. We replicate the experimental set up described in (Granroth-Wilding and Clark 2016), and split the data into training/dev/testing accordingly⁴. For FEEL, we use a 300 dimension space to embed each event property. Our full model which includes subject, direct object, indirect object, sentiment, and animacy information, is 1800 dimensional.

The FEEL embeddings are evaluated over three intrinsic tasks: (1) Multi-Choice Narrative Cloze (MCNC), (2) Multi-Choice Narrative Sequences (MCNS), and (3) Multi-Choice Narrative Explanation (MCNE); and two extrinsic tasks: (2) Semantic Relatedness on Sentences Involving Compositional Knowledge (SICK), and (3) Implicit Discourse Sense Classification.

Multi-Choice Narrative Event Cloze (MCNC)

MCNC task is a multiple-choice variant of the Narrative Cloze task proposed by (Chambers and Jurafsky 2008). We follow the evaluation settings proposed in (Granroth-Wilding and Clark 2016), which randomly sampled four extra choices from the vocabulary (the random guess baseline will have a 20% accuracy). The question sets they used are not explicitly released, instead we sampled question sets using the same data splits as theirs. This might lead to differences in the resulting scores, to mitigate it, we run each experiment for 10 times and report the average score.

²For simplicity, λ_i and λ_r are fixed to 1 in this paper.

³English Gigaword <https://catalog.ldc.upenn.edu/ldc2011t07>

⁴The data splits can be downloaded from their website <http://stackoverflow.com/>

The event chains and question sets can be downloaded from <http://anonymized-url>

	Accuracy	MRR
Granroth-Wilding et al., 2016	0.4957	-
PredDep	0.4232	0.6271
PredDep+Args	0.5135	0.6827
PredDep+Args+S	0.5166	0.6844
PredDep+Args+A	0.5503	0.7096
PredDep+Args+S+A	0.5418	0.7031

Table 1: The results of multi-choice narrative cloze test. Accuracy and Mean Reciprocal Rank (MRR) score are both reported for measuring the quality of ranking.

Table 1 shows the accuracy and Mean Reciprocal Rank (MRR) scores of the test. The first row lists the best score reported in (Granroth-Wilding and Clark 2016), and the rest describe the results obtained by the variants of our model. *PredDep* is the model with the context (inter-event) objective only; *PredDep+Args* is the model that includes the arguments (subject, object, prepositional objects) into the objective; *PredDep+Args+S* and *PredDep+Args+A* further include the sentence-level sentiment information and protagonist’s animacy respectively; and *PredDep+Args+S+A* contains all the information mentioned. The results show that *PredDep* performs worse than *Granroth-Wilding et al., 2016*, this is not surprising as it does not model the event argument information. When this information is used (*PredDep+Args*), our model outperforms Granroth-Wilding et al.’s model. Moreover, when additional properties are used, our model’s performance is improved, most significantly when using the animacy information.

Interestingly, the results show that including too many properties might hurt performance. As illustrated in the last row. We hypothesize that this is because different properties bias the predictions but some of them have conflicts in making the decisions. Therefore, carefully choosing the appropriate properties for the target application is important. We leave advanced learning protocols that can help resolve such conflicts to future work.

Multi-Choice Narrative Sequences (MCNS)

MCNC evaluates the model’s ability to infer an event form its context. A natural generalization of this task is to consider inferences over longer sequences of events, as these can better account for narrative structure, rather than pairwise event relationships. We propose a new evaluation task script learning, Multi-Choice Narrative Sequences (MCNS).

We set up this task by following these steps –

1. We subsampled n questions of length l from the test set of MCNC.
2. We generated an extra x choices for each event, except the first event. Note the difference from the MCNC setting, as a multiple-choice question is associated with each time stamp.
3. We modeled each event chain as a Markov Chain with l time stamps and $x + 1$ states at each time stamp, where the

first time stamp only contains the starting state. We used an inference algorithm (Viterbi (Viterbi 1967)) to identify the highest scoring event chain.

This task evaluates the model’s ability to make longer commonsense inference, instead of just predicting one event. In this paper we used a simple inference algorithm (a sequence model), but we consider incorporating advanced reasoning algorithms as a very promising direction for future work. In order to evaluate this approach, we used this algorithm over all of our script models. We also replaced the inference algorithm with a simple greedy baseline and perfect skyline.

- **Skyline:** Break down a sequence of predictions into individual decisions, and give the correct previous state for each decision.
- **Baseline:** No inference. Instead of Viterbi, for each time stamp, greedily pick the best transition and move to the next time stamp.

We also included a strong baseline text similarity model. The popular word embedding model—GloVe (Pennington, Socher, and Manning 2014)—is used to score the transitions between events by computing the similarity between the averaged vectors of the event words.

We measure the contribution of the embedding generated by FEEL by observing the performance difference when these embedding are concatenated with the base Glove event representation.

In this experiment, 1000 ($n = 1000$) length-5 questions with $4(x = 4)$ extra choices at each time stamp are sampled. Accuracy is used as the evaluation metric. The three columns of Table. 2 show the results, which show that in all cases FEEL embeddings offer performance gains when added to GloVe, even when only the event token information (*PredDep*) is provided. Both the sentiment and animacy property provide helps in this task.

The best model (*GloVe+PredDep+Args+S*) brings the performance up to 0.411 from 0.331 using Viterbi.

Similar to the MCNC results, using all event properties jointly (*GloVe+PredDep+Args+S+A*) does not improve performance.

Multi-Choice Narrative Explanation (MCNE)

We suggest an additional extension to the MCNC task. The *Multi-Choice Narrative Explanation* (MCNE) task is also designed to evaluate reasoning over longer event sequences, similarly to MCNS. However, instead of just providing the initial event as input, in the MCNE task both the starting and the ending events are provided, and the prediction task is to infer what happened in between. Human commonsense can build an explanation that connects the two points. For example, given a beginning “Jenny went to a restaurant” and an ending “She felt satisfied”, can you figure out what might happen in the restaurant? We might guess that “she liked the food” is more likely than “she waited for an hour”. MCNE provides a platform for script models to demonstrate such deeper understanding of world knowledge. The setup is exactly the same as MCNS, except leaving for the prediction at the final time stamp that is given as an input.

	MCNS-Viterbi	Baseline	Skyline	MCNE-Viterbi
GloVe	0.353	0.297	0.356	0.385
GloVe+PredDep	0.359	0.302	0.362	0.389
GloVe+PredDep+Args	0.332	0.366	0.434	0.37
GloVe+PredDep+Args+S	0.416	0.385	0.460	0.448
GloVe+PredDep+Args+A	0.399	0.396	0.465	0.429
GloVe+PredDep+Args+S+A	0.365	0.383	0.452	0.403

Table 2: Results of MCNS (left) and MCNE (right) using Viterbi, the skyline, and the baseline as inference models on FEEL and GloVe embeddings. *PredDep*, *Args*, *S*, *A* respectively mean that the event token, argument, sentiment, and animacy information are included in the training. The + operators means vector concatenations.

We use the same models as in MCNS. Note that when calculating the accuracies we did not include the ending state in both MCNS and MCNE, so the same baseline and skyline used in MCNS are applicable for this task. The right-most column of Table 2 summarizes the results.

We observe a very similar trend in the results to MCNS’s, but with higher overall accuracies, since additional information is provided to the model during inference. The results also show our models’ ability to improve commonsense reasoning using inference. Similar to MCNS, both the sentiment and animacy information help inferences, while the all-featured model result in a small performance drop. We hypothesize the reason is the same as before.

Semantic Relatedness on Sentences Involving Compositional Knowledge (SICK)

We also evaluate our embedding model as a feature representation for the SICK task.

This dataset was used as popular shared task in SemEval-2014⁵. It measures the semantic relatedness of a given sentence pair. The gold relatedness score is averaged across ten human-annotated scores for each sentence pair, ranging from 1.0 to 5.0, where 1.0 means completely unrelated and 5.0 means very related. The training/dev/testing splits are available on the task website.

Our goal in these experiments is to evaluate whether the event embedding can help capture the structural properties of sentence. To evaluate this property we augment a baseline system, which uses the GloVe word embedding, with our event embedding, and compare the performance over different variants of our embedding model.

Obtained a performance improvement over GloVe is not trivial. Glove leverages global matrix factorization and local context windows methods to build general-purpose word embeddings, which have been shown to have better performance than the other popular word embedding model—word2vec (Mikolov et al. 2013)—in word similarity tasks. We use the 300-dimension version, pre-trained on Gigaword and Wikipedia, available on their website⁶.

To construct the input sentence representation for both GloVe and FEEL, first, all the available embeddings in the input sentences are extracted and summed (word and

event embedding separately). Second, these representations are fed into a neural-network-based regression model (Tai, Socher, and Manning 2015) for predicting the related scores. The network architecture is designed for text similarity tasks and is shown below:

$$\begin{aligned}
h_* &= v_{s1} \otimes v_{s2} \\
h_{\Delta} &= |v_{s1} - v_{s2}| \\
h &= h_* \oplus h_{\Delta} \\
p &= \text{softmax}(W \cdot h),
\end{aligned}$$

where $v_{s1} \in R^k$ and $v_{s2} \in R^k$ are vector representations of the first and second inputs respectively; \otimes means element-wise multiplication; \oplus is the vector concatenation operator; $W \in R^{5 \times 2k}$ is the weight matrix to be trained; the 5 softmax outputs corresponds to scores 1-5. The cross-entropy loss function and mini-batch Adam (Kingma and Ba 2014) are used to optimize the model. The final score is calculated by taking the expectation value of the 5 softmax outputs.

Table 3 shows Pearson Correlation Scores between the gold and predicted scores. The first row denotes the classification quality with using Glove-only. It turns out that this simple representation is sufficient to provides a reasonably good result (Pearson Correlation of 0.71). Using FEEL alone does not lead to the optimal results (*Predicate + Args + S* has the best result among the variants, which is 0.68). We suspect that this is because when constructing FEEL event embeddings, some input components that are not considered. This simplifies model training by modeling only high-level event concepts while incurs losing some details. However, this does help FEEL captures more "structured" event semantics. The third row of the table indicates the result of using both Glove and FEEL together, by concatenating them to form the input representations. The results of the combined model are far better than the previous two representations and go up to 0.77 if the animacy information is included.

We also observe that while animacy information was very useful, the sentiment information generally does not provide additional information for this task. This might be due to the word embedding already capturing this information. Also, since many examples in this dataset have neutral sentiment this contribution of this property is likely to be small. This is not always the case, as we can see in our next task.

⁵<http://alt.qcri.org/semeval2014/task1/>

⁶Glove <https://nlp.stanford.edu/projects/glove/>

Pearson	PredDep	PredDep+Args	PredDep+Args+S	PredDep+Args+A	PredDep+Args+S+A
GloVe	0.7102				
FEEL	0.4452	0.6574	0.6791	0.6714	0.6714
GloVe+FEEL	0.7382	0.7572	0.7518	0.7676	0.7604

Table 3: Pearson scores of SICK task using a well-known word embeddings–Glove–and the event embeddings–FEEL

Implicit Discourse Sense Classification (IDSC)

Our final evaluation task is the CONLL 2016 Shared Task (Xue et al. 2016) on discourse parsing. The original goal is twofold: (1) locate discourse connectives and their arguments and (2) classify discourse senses. In this evaluation, we consider focus on the highly challenging Implicit Discourse Sense Classifications (IDSC) task. Unlike explicit discourse senses, where discourse connectives, e.g., *because*, *but*, and *however*, provide strong cues that help identify the correct sense, implicit discourse senses can only be inferred from the two given argument spans, and as a result relies very heavily on modeling the semantic relationships between them.

The dataset used is based on the Penn Discourse Tree Bank (PDTB) (Prasad et al. 2007), where all the arguments, connectives, and senses are annotated. We used the same settings as the CoNLL shared task: the data splits include training, development, test, and blind test sets; four relation types (non-explicit) and fifteen valid sense classes are used. More detailed information can be found in (Prasad et al. 2007).

Similar to the SICK task, GloVe and FEEL are evaluated together. We use the same method for building the sentence representations. For GloVe and FEEL, the embeddings of the words and events in each argument are summed up respectively. A two-hidden-layer neural network is then used to trained a multi-class classifier. The network structured is formulated as below:

$$\begin{aligned}
v &= v_{a1} \oplus v_{a2} \\
h_1 &= g(W_1 \cdot v) \\
h_2 &= g(W_2 \cdot h_1) \\
p &= \text{softmax}(W_3 \cdot h_2),
\end{aligned}$$

where $v_{a1} \in R^k$ and $v_{a2} \in R^k$ are the representations of the first and second arguments; k is the dimension of input representations; \oplus is the vector concatenation operator; $W_1 \in R^{n_1 \times 2k}$, $W_2 \in R^{n_2 \times n_1}$, and $W_3 \in R^{15 \times n_2}$; n_1 and n_2 are the sizes of the first and second hidden layers; $g(\cdot)$ is the rectified linear unit activation function. The cross-entropy loss function and mini-batch Adam (Kingma and Ba 2014) are used for optimizing the parameters.

Table 4 shows the micro average F1 scores across all the senses. Like SICK task, GloVe performs reasonably well here, as it captures general semantics at the word level.

We can observe the performance improvement obtained by adding event properties introduced by FEEL to GloVe. Specifically the the most complex event representation *GloVe+PredDep+Args+S+A*, performs the best over both the test and blind test sets. This suggests that the benefit of using specific properties to enrich the event representation

Micro Average F1	Test	Blind Test
GloVe	0.2982	0.2815
GloVe+PredDep	0.2921	0.2886
GloVe+PredDep+Args	0.2983	0.2862
GloVe+PredDep+Args+S	0.2996	0.3102
GloVe+PredDep+Args+A	0.3063	0.3111
GloVe+PredDep+Args+S+A	0.3174	0.3111

Table 4: Micro average F1 scores across all the discourse senses under the setting of CONLL 2016 shared task.

is task specific and when used appropriately it can capture more fine-grained semantics.

Conclusion and Future Work

In this paper, we present a feature-enriched script learning model, FEEL. Our model learns robust event embeddings by **jointly conditioning the event representation on neighboring events and the inner properties of the event**. This highly flexible approach can easily be adapted to the specific needs of different end applications.

The trained event embeddings are evaluated on three intrinsic tasks, including two newly proposed tasks highlighting the importance of evaluating narrative inferences. We also evaluate our model over two extrinsic tasks, by using it as a learning representation. Our results show that FEEL can indeed utilize event properties and better account of the structured event semantics.

We currently consider several directions for future work. First, we plan to incorporate advanced learning architectures such as the **Attention mechanism**, to improve our script learning model. This mechanism has shown promise in improving sequence model learning and reasoning. The attention model can also be used to **find out soft alignments between objects** (for example when used for Neural Machine Translation). These alignments can help support event reasoning tasks which we highlighted in this paper.

References

- Ahrendt, S., and Demberg, V. 2016. Improving event prediction by representing script participants. In *HLT-NAACL*, 546–551.
- Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”.
- Chambers, N., and Jurafsky, D. 2008. Unsupervised learning of narrative event chains. In *ACL*, volume 94305, 789–797.
- Chambers, N. 2017. Behind the scenes of an evolving event cloze test. *LSDSem 2017* 41.

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

Hutto, C. J., and Gilbert, E. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, J.; Ritter, A.; and Jurafsky, D. 2015. Learning multi-faceted representations of individuals from heterogeneous evidence using neural networks. *arXiv preprint arXiv:1510.05198*.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J. R.; Bethard, S.; and McClosky, D. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, 55–60.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Modi, A., and Titov, I. 2014. Inducing neural models of script knowledge. In *CoNLL*, volume 14, 49–57.

Modi, A.; Titov, I.; Demberg, V.; Sayeed, A.; and Pinkal, M. 2017. Modeling semantic expectation: Using script knowledge for referent prediction. *TACL*.

Peng, H., and Roth, D. 2016. Two discourse driven language models for semantics. In *ACL*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Pichotta, K., and Mooney, R. J. 2016a. Learning statistical scripts with lstm recurrent neural networks. In *AAAI*, 2800–2806.

Pichotta, K., and Mooney, R. J. 2016b. Statistical script learning with recurrent neural networks. *EMNLP 2016* 11.

Prasad, R.; Miltsakaki, E.; Dinesh, N.; Lee, A.; Joshi, A.; Robaldo, L.; and Webber, B. L. 2007. The penn discourse treebank 2.0 annotation manual.

Rudinger, R.; Rastogi, P.; Ferraro, F.; and Van Durme, B. 2015. Script induction as language modeling. In *EMNLP*, 1681–1686.

Schank, R. C., and Abelson, R. P. 1977. Scripts, plans, goals and understanding: An inquiry into human knowledge structures.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Viterbi, A. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory* 13(2):260–269.

Xue, N.; Ng, H. T.; Pradhan, S.; Rutherford, A.; Webber, B. L.; Wang, C.; and Wang, H. 2016. Conll 2016 shared task on multilingual shallow discourse parsing. In *CoNLL Shared Task*, 1–19.