Inducing Neural Models of Script Knowledge

Modi, A.; Titov, I.

DOI
10.3115/v1/W14-1606

Publication date
2014

Document Version
Final published version

Published in
CoNNL-2014 : Eighteenth Conference on Computational Natural Language Learning

License
CC BY-NC-SA

Citation for published version (APA):
Abstract

Induction of common sense knowledge about prototypical sequence of events has recently received much attention (e.g., Chambers and Jurafsky (2008); Regneri et al. (2010)). Instead of inducing this knowledge in the form of graphs, as in much of the previous work, in our method, distributed representations of event realizations are computed based on distributed representations of predicates and their arguments, and then these representations are used to predict prototypical event orderings. The parameters of the compositional process for computing the event representations and the ranking component of the model are jointly estimated. We show that this approach results in a substantial boost in performance on the event ordering task with respect to the previous approaches, both on natural and crowdsourced texts.

1 Introduction

It is generally believed that natural language understanding systems would benefit from incorporating common-sense knowledge about prototypical sequences of events and their participants. Early work focused on structured representations of this knowledge (called scripts (Schank and Abelson, 1977)) and manual construction of script knowledge bases. However, these approaches do not scale to complex domains (Mueller, 1998; Gordon, 2001). More recently, automatic induction of script knowledge from text have started to attract attention: these methods exploit either natural texts (Chambers and Jurafsky, 2008, 2009) or crowdsourced data (Regneri et al., 2010), and, consequently, do not require expensive expert annotation. Given a text corpus, they extract structured representations (i.e. graphs), for example chains (Chambers and Jurafsky, 2008) or more general directed acyclic graphs (Regneri et al., 2010). These graphs are scenario-specific, nodes in them correspond to events (and associated with sets of potential event mentions) and arcs encode the temporal precedence relation. These graphs can then be used to inform NLP applications (e.g., question answering) by providing information whether one event is likely to precede or succeed another. Note that these graphs encode common-sense knowledge about prototypical ordering of events rather than temporal order of events as described in a given text.

Though representing the script knowledge as graphs is attractive from the human interpretability perspective, it may not be optimal from the application point of view. More specifically, these representations (1) require a model designer to choose an appropriate granularity of event mentions (e.g., whether nodes in the graph should be associated with verbs, or also their arguments); (2) do not provide a mechanism for deciding which scenario applies in a given discourse context and (3) often do not associate confidence levels with information encoded in the graph (e.g., the precedence relation in Regneri et al. (2010)).

Instead of constructing a graph and using it to provide information (e.g., prototypical event ordering) to NLP applications, in this work we advocate for constructing a statistical model which is capable to “answer” at least some of the questions these graphs can be used to answer, but doing this without explicitly representing the knowledge as a graph. In our method, the distributed representations (i.e. vectors of real numbers) of event realizations are computed based on distributed representations of predicates and their arguments, and then the event representations are used in a ranker to predict the prototypical ordering of events. Both the parameters of the compositional process for computing the event representation and the rank-
ing component of the model are estimated from
texts (either relying on unambiguous discourse
clues or natural ordering in text). In this way we
build on recent research on compositional distri-
butional semantics (Baroni and Zamparelli, 2011;
Socher et al., 2012), though our approach specif-
ically focuses on embedding predicate-argument
structures rather than arbitrary phrases, and learn-
ing these representation to be especially informa-
tive for prototypical event ordering.

In order to get an intuition why the embedding
approach may be attractive, consider a situation
where a prototypical ordering of events the bus
disembarked passengers and the bus drove away
needs to be predicted. An approach based on fre-
quency of predicate pairs (Chambers and Jurafsky,
2008) (henceforth CJ08), is unlikely to make a
right prediction as driving usually precedes disem-
barking. Similarly, an approach which treats the
whole predicate-argument structure as an atomic
unit (Regneri et al., 2010) will probably fail as
well, as such a sparse model is unlikely to be ef-
fectively learnable even from large amounts of un-
labeled data. However, our embedding method
would be expected to capture relevant features of
the verb frames, namely, the transitive use for the
predicate disembark and the effect of the particle
away, and these features will then be used by the
ranking component to make the correct prediction.

In previous work on learning inference
rules (Berant et al., 2011), it has been shown
that enforcing transitivity constraints on the
inference rules results in significantly improved
performance. The same is likely to be true for
the event ordering task, as scripts have largely
linear structure, and observing that \( a \prec b \) and
\( b \prec c \) is likely to imply \( a \prec c \). Interestingly, in
our approach we learn the model which satisfies
transitivity constraints, without the need for any
explicit global optimization on a graph. This
results in a significant boost of performance when
using embeddings of just predicates (i.e. ignoring
arguments) with respect to using frequencies of
ordered verb pairs, as in CJ08 (76% vs. 61% on
the natural data).

Our model is solely focusing on the ordering
task, and admittedly does not represent all the in-
formation encoded by a script graph structure. For
example, it cannot be directly used to predict a
missing event given a set of events (the narrative
cloze task (Chambers and Jurafsky, 2009)). Nev-
ertheless, we believe that the framework (a proba-
bilistic model using event embeddings as its com-
ponent) can be extended to represent other aspects
of script knowledge by modifying the learning ob-
jective, but we leave this for future work. In this
paper, we show how our model can be used to pre-
dict if two event mentions are likely paraphrases
of the same event.

The approach is evaluated in two set-ups. First,
we consider the crowdsourced dataset of Regneri
et al. (2010) and demonstrate that using our model
results in the 13.5% absolute improvement in \( F_1 \)
on event ordering with respect to their graph in-
duction method (84.1% vs. 70.6%). Secondly,
we derive an event ordering dataset from the Gi-
gaword corpus, where we also show that the em-
bedding method beats the frequency-based base-
line (i.e. reimplementation of the scoring compo-
nent of CJ08) by 22.8% in accuracy (83.5% vs.
60.7%).

2 Model

In this section we describe the model we use for
computing event representations as well as the
ranking component of our model.

2.1 Event Representation

Learning and exploiting distributed word represen-
tations (i.e. vectors of real values, also known as
embeddings) have been shown to be benefi-
cial in many NLP applications (Bengio et al.,
2001; Turian et al., 2010; Collobert et al., 2011).
These representations encode semantic and syn-
tactic properties of a word, and are normally

---

**Figure 1**: Computation of an event representation for a predicate with two arguments (*the bus disembarked passengers*), an arbitrary number of arguments is supported by our approach.
learned in the language modeling setting (i.e.
learned to be predictive of local word context),
though they can also be specialized by learning
in the context of other NLP applications such as
PoS tagging or semantic role labeling (Collobert
et al., 2011). More recently, the area of dis-
tributional compositional semantics have started
to emerge (Baroni and Zamparelli, 2011; Socher
et al., 2012), they focus on inducing representa-
tions of phrases by learning a compositional
model. Such a model would compute a representa-
tion of a phrase by starting with embeddings of
individual words in the phrase, often this composi-
tion process is recursive and guided by some form
of syntactic structure.

In our work, we use a simple compositional
model for representing semantics of a verb frame
\(e\) (i.e. the predicate and its arguments). We will
refer to such verb frames as events. The model is
shown in Figure 1. Each word \(c_i\) in the vocab-
ulary is mapped to a real vector based on the cor-
responding lemma (the embedding function \(C\)).
The hidden layer is computed by summing linearly
transformed predicate and argument1 embeddings
and passing it through the logistic sigmoid func-
tion. We use different transformation matrices
for arguments and predicates, \(T\) and \(R\), respectively.
The event representation \(f(e)\) is then obtained by
applying another linear transform (matrix \(A\)) fol-
lowed by another application of the sigmoid func-
tion. Another point to note in here is that, as in
previous work on script induction, we use lemmas
for predicates and specifically filter out any tense
markers as our goal is to induce common-sense
knowledge about an event rather than properties
predictive of temporal order in a specific discourse
context.

We leave exploration of more complex and
linguistically-motivated models for future work.2
These event representations are learned in the con-
text of event ranking: the transformation parame-
ters as well as representations of words are forced
to be predictive of the temporal order of events.
In our experiments, we also consider initialization
of predicate and arguments with the SENNA word
embeddings (Collobert et al., 2011).

\footnote{Only syntactic heads of arguments are used in
this work. If an argument is a coffee maker, we will use only the
word maker.}

\footnote{In this study, we apply our model in two very different
settings, learning from crowdsourced and natural texts.
Crowdsourced collections are relatively small and require not
over-expressive models.}

2.2 Learning to Order

The task of learning stereotyped order of events
naturally corresponds to the standard ranking set-
ting. We assume that we are provided with se-
quences of events, and our goal is to capture this
order. We discuss how we obtain this learning ma-
terial in the next section. We learn a linear ranker
(characterized by a vector \(w\)) which takes an event
representation and returns a ranking score. Events
are then ordered according to the score to yield
the model prediction. Note that during the learn-
ing stage we estimate not only \(w\) but also the
event representation parameters, i.e. matrices \(T\),
\(R\) and \(A\), and the word embedding \(C\). Note that
by casting the event ordering task as a global rank-
ning problem we ensure that the model implicitly
exploits transitivity of the relation, the property
which is crucial for successful learning from finite
amount of data, as we argued in the introduction
and will confirm in our experiments.

At training time, we assume that each training
example \(k\) is a list of events \(e_1^{(k)}, \ldots, e_n^{(k)}\) pro-
vided in the stereotypical order (i.e. \(e_i^{(k)} \prec e_j^{(k)}\) if
\(i < j\)), \(n^{(k)}\) is the length of the list \(k\). We mini-
mize the \(L_2\)-regularized ranking hinge loss:

\[
\sum_k \sum_{i<j \leq n^{(k)}} \max(0, 1 - w^T f(e_i^{(k)}; \Theta) + w^T f(e_j^{(k)}; \Theta)) \\
+ \alpha (\|w\|_2 + \|\Theta\|_2),
\]

where \(f(e; \Theta)\) is the embedding computed
for event \(e\), \(\Theta\) are all embedding param-
ters corresponding to elements of the matrices
\(\{R, C, T, A\}\). We use stochastic gradient descent,
gradients w.r.t. \(\Theta\) are computed using back propa-
gation.

3 Experiments

We evaluate our approach in two different set-ups.
First, we induce the model from the crowdsourced
data specifically collected for script induction by
Regneri et al. (2010), secondly, we consider an
arguably more challenging set-up of learning the
model from news data (Gigaword (Parker et al.,
2011)), in the latter case we use a learning sce-
nario inspired by Chambers and Jurafsky (2008).3

\footnote{Details about downloading the data and models are at:
http://www.coli.uni-saarland.de/projects/smile/docs/nmReadme.txt}
Table 1: Results on the crowdsourced data for the verb-frequency baseline (BL), the verb-only embedding model (EEverb), Regneri et al. (2010) (MSA), Frermann et al. (2014) (BS) and the full model (EE).

<table>
<thead>
<tr>
<th>Activity</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus</td>
<td>70.1</td>
<td>81.9</td>
<td>80.0</td>
</tr>
<tr>
<td>Coffee</td>
<td>70.1</td>
<td>81.3</td>
<td>81.0</td>
</tr>
<tr>
<td>Fastfood</td>
<td>69.9</td>
<td>81.0</td>
<td>80.0</td>
</tr>
<tr>
<td>Return</td>
<td>74.0</td>
<td>94.1</td>
<td>89.2</td>
</tr>
<tr>
<td>Iron</td>
<td>73.4</td>
<td>80.1</td>
<td>86.9</td>
</tr>
<tr>
<td>Microw.</td>
<td>72.6</td>
<td>79.2</td>
<td>81.4</td>
</tr>
<tr>
<td>Eggs</td>
<td>72.7</td>
<td>71.4</td>
<td>75.9</td>
</tr>
<tr>
<td>Shower</td>
<td>62.2</td>
<td>76.2</td>
<td>74.9</td>
</tr>
<tr>
<td>Phone</td>
<td>67.6</td>
<td>87.8</td>
<td>81.7</td>
</tr>
<tr>
<td>Vending</td>
<td>66.4</td>
<td>87.3</td>
<td>84.2</td>
</tr>
<tr>
<td>Average</td>
<td>69.9</td>
<td>81.3</td>
<td>83.9</td>
</tr>
</tbody>
</table>

3.1 Learning from Crowdsourced Data

3.1.1 Data and task

Regneri et al. (2010) collected descriptions (called event sequence descriptions, ESDs) of various types of human activities (e.g., going to a restaurant, ironing clothes) using crowdsourcing (Amazon Mechanical Turk), this dataset was also complemented by descriptions provided in the OMICS corpus (Gupta and Kochenderfer, 2004). The datasets are fairly small, containing 30 ESDs per activity type in average (we will refer to different activities as scenarios), but in principle the collection can easily be extended given the low cost of crowdsourcing. The ESDs list events forming the scenario and are written in a bullet-point style. The annotators were asked to follow the prototypical event order in writing. As an example, consider a ESD for the scenario prepare coffee:

\[
\text{\{go to coffee maker\} } \rightarrow \text{\{fill water in coffee maker\} } \rightarrow \text{\{place the filter in holder\} } \rightarrow \text{\{place coffee in filter\} } \rightarrow \text{\{place holder in coffee maker\} } \rightarrow \text{\{turn on coffee maker\}}
\]

Regneri et al. also automatically extracted predicates and heads of arguments for each event, as needed for their MSA system and our compositional model.

Though individual ESDs may seem simple, the learning task is challenging because of the limited amount of training data, variability in the used vocabulary, optionality of events (e.g., going to the coffee machine may not be mentioned in a ESD), different granularity of events and variability in the ordering (e.g., coffee may be put in the filter before placing it in the coffee maker). Unlike our work, Regneri et al. (2010) relies on WordNet to provide extra signal when using the Multiple Sequence Alignment (MSA) algorithm. As in their work, each description was preprocessed to extract a predicate and heads of argument noun phrases to be used in the model.

The methods are evaluated on human annotated scenario-specific tests: the goal is to classify event pairs as appearing in a stereotypical order or not (Regneri et al., 2010). The model was estimated as explained in Section 2.2 with the order of events in ESDs treated as gold standard. We used 4 held-out scenarios to choose model parameters, no scenario-specific tuning was performed, and the 10 test scripts were not used to perform model selection. The selected model used the dimensionality of 10 for event and word embeddings. The initial learning rate and the regularization parameter were set to 0.005 and 1.0, respectively and both parameters were reduced by the factor of 1.2 every epoch the error function went up. We used 2000 epochs of stochastic gradient descent. Dropout (Hinton et al., 2012) with the rate of 20% was used for the hidden layers in all our experiments. When testing, we predicted that the event pair \((e_1, e_2)\) is in the stereotypical order \((e_1 \prec e_2)\) if the ranking score for \(e_1\) exceeded the ranking score for \(e_2\).

3.1.2 Results and discussion

We evaluated our event embedding model (EE) against baseline systems (BL, MSA and BS). MSA is the system of Regneri et al. (2010). BS is a hierarchical Bayesian model by Frermann et al. (2014). BL chooses the order of events based on the preferred order of the corresponding verbs in the training set: \((e_1, e_2)\) is predicted to be in the

The event pairs are not coming from the same ESDs making the task harder as the events may not be in any temporal relation.
stereotypical order if the number of times the corresponding verbs $v_1$ and $v_2$ appear in this order in the training ESDs exceeds the number of times they appear in the opposite order (not necessary at adjacent positions); a coin is tossed to break ties (or if $v_1$ and $v_2$ are the same verb). This frequency counting method was previously used in CJ08.\footnote{They scored permutations of several events by summing the logarithmed differences of the frequencies of ordered verb pairs. However, when applied to event pairs, their approach would yield exactly the same prediction rule as BL.}

We also compare to the version of our model which uses only verbs (EE$_{verb}$). Note that EE$_{verb}$ is conceptually very similar to BL, as it essentially induces an ordering over verbs. However, this ordering can benefit from the implicit transitivity assumption used in EE$_{verb}$ (and EE), as we discussed in the introduction. The results are presented in Table 1.

The first observation is that the full model improves substantially over the baseline and the previous method (MSA) in F1 (13.5\% improvement over MSA and 6.5\% improvement over BS). Note also that this improvement is consistent across scenarios: EE outperforms MSA and BS on 9 scenarios out of 10 and 8 out of 10 scenarios in case of BS. Unlike MSA and BS, no external knowledge (i.e. WordNet) was exploited in our method.

We also observe a substantial improvement in all metrics from using transitivity, as seen by comparing the results of BL and EE$_{verb}$ (11\% improvement in F1). This simple approach already substantially outperforms the pipelined MSA system. These results seem to support our hypothesis in the introduction that inducing graph representations from scripts may not be an optimal strategy from the practical perspective.

We performed additional experiments using the SENNA embeddings (Collobert et al., 2011). Instead of randomly initializing arguments and predicate embeddings (vectors), we initialized them with pre-trained SENNA embeddings. We have not observed any significant boost in performance from using the initialization (average F1 of 84.0\% for EE). We attribute the lack of significant improvement to the following three factors. First of all, the SENNA embeddings tend to place antonyms / opposites near each other (e.g., come and go, or end and start). However, ‘opposite’ predicates appear in very different positions in scripts. Additionally, the SENNA embeddings have dimensionality of 50 which appears to be too high for small crowd-sourced datasets, as it forces us to use larger matrices $T$ and $R$. Moreover, the SENNA embeddings are estimated from Wikipedia, and the activities in our crowdsourced domain are perhaps underrepresented there.

### 3.1.3 Paraphrasing

Regneri et al. (2010) additionally measure paraphrasing performance of the MSA system by comparing it to human annotation they obtained: a system needs to predict if a pair of event mentions are paraphrases or not. The dataset contains 527 event pairs for the 10 test scenarios. Each pair consists of events from the same scenario. The dataset is fairly balanced containing from 47 to 60 examples per scenario.

This task does not directly map to any statistical inference problem with our model. Instead we use an approach inspired by the interval algebra of Allen (1983).

Our ranking model maps event mentions to positions on the time line (see Figure 2). However, it would be more natural to assume that events are intervals rather than points. In principle, these intervals can be overlapping to encode a rich set of temporal relations (see (Allen, 1983)). However, we make a simplifying assumption that the intervals do not overlap and every real number belongs to an interval. In other words, our goal is to induce a segmentation of the line: event mentions corresponding to the same interval are then regarded as paraphrases.

One natural constraint on this segmentation is the following: if two event mentions are from the same training ESD, they cannot be assigned to the same interval (as events in ESD are not supposed to be paraphrases). In Figure 2 arcs link event mentions from the same ESD. We look for a segmentation which produces the minimal number of segments and satisfy the above constraint for event mentions appearing in training data.

Though inducing intervals given a set of temporal constraints is known to be NP-hard in general (see, e.g., (Golumbic and Shamir, 1993)), for our constraints a simple greedy algorithm finds an optimal solution. We trace the line from the left maintaining a set of event mentions in the current unfinished interval and create a boundary when the constraint is violated; we repeat the process until we processed all mentions. In Figure 2, we would create the first boundary between arrive in a restaurant and order beverages: order bev-
erages and enter a restaurant are from the same ESD and continuing the interval would violate the constraint. It is not hard to see that this results in an optimal segmentation. First, the segmentation satisfies the constraint by construction. Secondly, the number of segments is minimal as the arcs which caused boundary creation are non-overlapping, each of these arcs needs to be cut and our algorithm cuts each arc exactly once.

This algorithm prefers to introduce a boundary as late as possible. For example, it would introduce a boundary between browse a menu and review options in a menu even though the corresponding points are very close on the line. We modify the algorithm by moving the boundaries left as long as this move does not result in new constraint violations and increases margin at boundaries. In our example, the boundary would be moved to be between order beverages and browse a menu, as desired.

The resulting performance is reported in Table 2. We report results of our method, as well as results for MSA, BS and a simple all-paraphrase baseline which predict that all mention pairs in a test set are paraphrases (APBL). We can see that interval induction technique results in a lower F1 than that of MSA or BS. This might be partially due to not using external knowledge (WordNet) in our method.

We performed extra analyses on the development scenario doorbell. The analyses revealed that the interval induction approach is not very robust to noise: removing a single noisy ESD results in a dramatic change in the interval structure induced and in a significant increase of F1. Consequently, soft versions of the constraint would be beneficial. Alternatively, event embeddings (i.e. continuous vectors) can be clustered directly. We leave this investigation for future work.

### 3.2 Learning from Natural Text

In the second set of experiments we consider a more challenging problem, inducing knowledge about the stereotyped ordering of events from natural texts. In this work, we are largely inspired by the scenario of CJ08. The overall strategy is the following: we process the Gigaword corpus with a high precision rule-based temporal classifier relying on explicit clues (e.g., “then”, “after”) to get ordered pairs of events and then we train our model on these pairs (note that clues used by the classifier are removed from the examples, so the model has to rely on verbs and their arguments). Conceptually, the difference between our approach and CJ08 is in using a different temporal classifier, not enforcing that event pairs have the same protagonist, and learning an event embedding model instead of scoring event sequences based on verb-pair frequencies.

We also evaluate our system on examples extracted using the same temporal classifier (but validated manually) which allows us to use much larger tests set, and, consequently, provide more detailed and reliable error analysis.

### 3.2.1 Data and task

The Gigaword corpus consists of news data from different news agencies and newspapers. For testing and development we took the AFP (Agence France-Presse) section, as it appeared most different from the rest when comparing sets of extracted event pairs (other sections correspond mostly to US agencies). The AFP section was not used for
training. This selection strategy was chosen to create a negative bias for our model which is more expressive than the baseline methods and, consequently, better at memorizing examples.

As a rule-based temporal classifier, we used high precision “happens-before” rules from the VerbOcean system (Chklovski and Pantel, 2004). Consider “to (verb-X) and then (verb-Y)” as one example of such rule. We used predicted collapsed Stanford dependencies (de Marneffe et al., 2006) to extract arguments of the verbs, and used only a subset of dependents of a verb. This preprocessing ensured that (1) clues which form part of a pattern are not observable by our model both at train and test time; (2) there is no systematic difference between both events (e.g., for collapsed dependencies, the noun subject is attached to both verbs even if the verbs are conjoined); (3) no information about the order of events in text is available to the models. Applying these rules resulted in 22,446 event pairs for training, and we split additional 1,015 pairs from the AFP section into 812 for final testing and 203 for development. We manually validated random 50 examples and all 50 of them followed the correct temporal order, so we chose not to hand correct the test set.

We largely followed the same training and evaluation regime as for the crowdsourced data. We set the regularization parameter and the learning rate to 0.01 and 5, e.g. 4 respectively. The model was trained for 600 epochs. The embedding sizes were 30 and 50 dimensions for words and events, respectively.

### 3.2.2 Results and discussion

In our experiments, as before, we use BL as a baseline, and EEverb as a verb-only simplified version of our approach. We used another baseline consisting of the verb pair ordering counts provided by Chambers and Jurafsky (2008).8 We refer this baseline as CJ08. Note also that BL can be regarded as a reimplemention of CJ08 but with a different temporal classifier. We report results in Table 3.

The observations are largely the same as before: (1) the full model substantially outperforms all other approaches (p-level < 0.001 with the permutation test); (2) enforcing transitivity is very helpful (75.9 % for EEverb vs. 60.1% for BL). Surprisingly CJ08 rules produce as good results as BL, suggesting that maybe our learning set-ups are not that different.

However, an interesting question is in which situations using a more expressive model, EE, is beneficial. If these accuracy gains have to do with memorizing the data, it may not generalize well to other domains or datasets. In order to test this hypothesis we divided the test examples in three frequency bands according to the frequency of the corresponding verb pairs in the training set (total, in both orders). There are 513, 249 and 50 event pairs in the bands corresponding to unseen pairs of verbs, frequency ≤ 10 and frequency > 10, respectively. These counts emphasize that correct predictions on unseen pairs are crucial and these are exactly where BL would be equivalent to a random guess. Also, this suggest, even before looking into the results, that memorization is irrelevant. The results for BL, CJ08, EEverb and EE are shown in Figure 3.

One observation is that most gains for EE and EEverb are due to an improvement on unseen pairs. This is fairly natural, as both transitivity and information about arguments are the only sources of information available. In this context it is important to note that some of the verbs are light, in the sense that they have little semantic content of their own (e.g., take, get) and the event semantics can only be derived from analyzing their arguments (e.g., take an exam vs. take a detour). On the high frequency verb pairs all systems perform equally well, except for CJ08 as it was estimated from somewhat different data.

In order to understand how transitivity works, we considered a few unseen predicate pairs where the EEverb model was correctly predicting their order. For many of these pairs there were no infer-

---

8These verb pair frequency counts are available at www.usna.edu/Users/cs/nchamber/data/schemas/acl09/verb-pair-orders.gz

<table>
<thead>
<tr>
<th></th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL</td>
<td>60.7</td>
</tr>
<tr>
<td>CJ08</td>
<td>60.1</td>
</tr>
<tr>
<td>EEverb</td>
<td>75.9</td>
</tr>
<tr>
<td>EE</td>
<td>83.5</td>
</tr>
</tbody>
</table>

Table 3: Results on the Gigaword data for the verb-frequency baseline (BL), the verb-only embedding model (EEverb), the full model (EE) and CJ08 rules.
ence chains of length 2 (e.g., chain of length 2 was found for the pair accept ≺ carry: accept ≺ get and get ≺ carry but not many other pairs). This observation suggest that our model captures some non-trivial transitivity rules.

4 Related Work

Additionally to the work on script induction discussed above (Chambers and Jurafsky, 2008, 2009; Regneri et al., 2010), other methods for unsupervised learning of event semantics have been proposed. These methods include unsupervised frame induction techniques (O’Connor, 2012; Modi et al., 2012). Frames encode situations (or objects) along with their participants and properties (Fillmore, 1976). Events in these unsupervised approaches are represented with categorical latent variables, and they are induced relying primarily on the selectional preferences’ signal. The very recent work of Cheung et al. (2013) can be regarded as their extension but Cheung et al. also model transitions between events with Markov models. However, neither of these approaches considers (or directly optimizes) the discriminative objective of learning to order events, and neither of them uses distributed representations to encode semantic properties of events.

As we pointed out before, our embedding approach is similar (or, in fact, a simplification of) the phrase embedding methods studied in the recent work on distributional compositional semantics (Baroni and Zamparelli, 2011; Socher et al., 2012). However, they have not specifically looked into representing script information. Approaches which study embeddings of relations in knowledge bases (e.g., Riedel et al. (2013)) bear some similarity to the methods proposed in this work but they are mostly limited to binary relations and deal with predicting missing relations rather than with temporal reasoning of any kind.

Identification of temporal relations within a text is a challenging problem and an active area of research (see, e.g., the TempEval task (UzZaman et al., 2013)). Many rule-based and supervised approaches have been proposed in the past. However, integration of common sense knowledge induced from large-scale unannotated resources still remains a challenge. We believe that our approach will provide a powerful signal complementary to information exploited by most existing methods.

5 Conclusions

We have developed a statistical model for representing common sense knowledge about prototypical event orderings. Our model induces distributed representations of events by composing predicate and argument representations. These representations capture properties relevant to predicting stereotyped orderings of events. We learn these representations and the ordering component from unannotated data. We evaluated our model in two different settings: from crowdsourced data and natural news texts. In both set-ups our method outperformed baselines and previously proposed systems by a large margin. This boost in performance is primarily caused by exploiting transitivity of temporal relations and capturing information encoded by predicate arguments.

The primary area of future work is to exploit our method in applications such as question answering. Another obvious applications is discovery of temporal relations within documents (UzZaman et al., 2013) where common sense knowledge implicit in script information, induced from large unannotated corpora, should be highly beneficial. Our current model uses a fairly naive semantic composition component, we plan to extend it with more powerful recursive embedding methods which should be especially beneficial when considering very large text collections.

6 Acknowledgements

Thanks to Lea Frermann, Michaela Regneri and Manfred Pinkal for suggestions and help with the data. This work is partially supported by the MMCI Cluster of Excellence at the Saarland University.
References


Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin Marlin. 2013. Relation extraction with matrix factorization and universal schemas. TACL.


