Learning Rich Event Representations and Interactions for Temporal Relation Classification

Onkar Pandit[1], Pascal Denis[1] and Liva Ralaivola[2]

1- MAGNET, Inria Lille - Nord Europe, Villeneuve d'Ascq, France
onkar.pandit@inria.fr, pascal.denis@inria.fr

2- QARMA, IUF, LIS, Aix-Marseille University, CNRS, Marseille, France,
Criteo AI Labs, Paris, France. liva.ralaivola@lif.univ-mrs.fr

Abstract. Most existing systems for identifying temporal relations between events heavily rely on hand-crafted features derived from event words and explicit temporal markers. Besides, less attention has been given to automatically learning contextualized event representations or to finding complex interactions between events. This paper fills this gap in showing that a combination of rich event representations and interaction learning is essential to more accurate temporal relation classification. Specifically, we propose a neural architecture, in which i) Recurrent Neural Network (RNN) is used to extract contextual information for pairs of events, and ii) a deep Convolutional Neural Network (CNN) architecture is used to find out intricate interactions between events. We show that the proposed approach outperforms most existing systems on commonly used datasets, while using fully automatic feature extraction and simple local inference.

1 Introduction

Recovering temporal information from texts is an essential part of language understanding, and it has applications such as question answering, text summarization, etc.

Temporal relation identification is divided into two main tasks, as identified by TempEval campaigns [1]: i) the identification of EVENTs and other time expressions (the so-called TIMEX's), and ii) the classification of temporal relations (or TLINKs) among and across events and time expressions.

In this work, we concentrate on temporal relation classification, specifically EVENT-EVENT relations, the most frequent type of TLINKs and arguably the most challenging task. What makes this problem difficult is that, in the absence of explicit temporal connectives (e.g., *before*, *during*), determining temporal relations depends on numerous factors, ranging from tense and aspect, to lexical semantics and even world knowledge. To address this issue, most state-of-the-art systems for EVENT-EVENT classification [2, 3, 4] rely on manually-crafted feature sets directly extracted from annotations, complemented with syntactic features, and semantic features extracted from static knowledge bases like WordNet or VerbOcean. Such an approach is tedious, error-prone and the semantics of events is poorly modelled due to lack of coverage of existing lexical resources and blindness to event contexts.

We here propose a radically different approach where we altogether dispense with hand-designed features, and instead learn task-specific event representations. These representations include information both from the event words *and* its surrounding

context, thus giving access to the events' arguments and modifiers. Plus, we also attempt to learn the potentially rich interactions between events. Concretely, our learning framework, as depicted in Fig.1, is based on a neural nework architecture, wherein: i) Recurrent Neural Network (RNN) is used to learn contextualized event representations, and ii) a deep Convolutional Neural Network (CNN) architecture is then used to acquire complex, non-linear interactions between these representations.

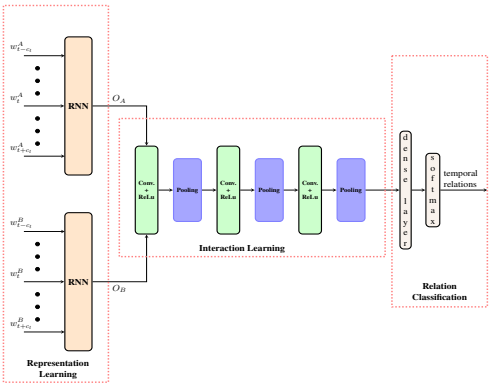

Fig. 1: Architecture of our proposed model.

This is one important step up from the recent work of Mirza and Tonelli [5], which simply use pre-trained word embeddings for event words and still have to resort to additional hand-engineered features to achieve good temporal classification accuracy. We show our system based on fully automatic feature extraction and interaction learning outperforms other local classifier systems.

2 Related Work

Recent temporal classification systems use machine learning techniques due to the availability of annotated datasets. Earlier work[2] studied *local* models (i.e., making pairwise decisions on pairs of events) and used gold-standard features extracted from TimeML annotations. State-of-the-art local models such as ClearTK [6] relied on an enlarged set of predicted features, relying on a cascade of classifiers. A downside of these local models is that they often generate globally incoherent temporal relations, in the sense that the symmetry and transitivity holding between relations are not explicitly enforced at the document level. This problem has led to the development of various *global* models, wherein temporal relation prediction is modeled as a constrained optimization problem and using Integer Linear Programming [7, 8]. While inference is global, model learning remains local. More recently, the CAVEO system [3] proposes a multi-sieve approach in which several hand-coded rules and locally-trained classifiers are applied in sequence, enforcing global coherence at each step. The state-of-the-art method of [4] proposes a structured prediction approach, in which global inference is also performed during training.

These methods all rely on manually engineered features, which fail to model the semantics of events. To address this issue, [5] have evaluated the effectiveness of pre-trained word embeddings of event head-word. They also demonstrated the potency of basic vector combination schemes. However, representing events with word embeddings of only its head-word is not effective and important contextual information is lost. Recently, [9] proposed an LSTM-based neural network architecture to learn event representation. However, in that work, they lacked in finding complicated interaction between events with only concatenation of event features. Also, they used syntactically parsed trees as inputs to the LSTM which adds burden of pre-processing.

3 Method

Our proposed neural architecture (Fig.1), consists of three main components: Representation Learning, Interaction Learning and Temporal Classification. In the Representation Learning part, a bag-of-words in a fixed size window centred on each event word is fetched and fed into a RNN to get more expressive and compact representation of events. As output of the RNN for each event, we get a fixed dimensional vector representation. This vector representation is then used at the Interaction Learning stage: the vector representation of each event is fed to a convolution layer and the final pooling layer outputs an interaction vector between the events. A dense layer is used to combine the interaction vector before obtaining a probability score for each temporal relation at the Relation Classification section.

3.1 Context-based Event Representation

Each word is encoded in the event-word window with word embeddings [10]. As a result, each word is assigned a fixed d-dimensional vector representation. Let c_l be the context length for each event head word. Thus we consider a window of $2c_l + 1$ words as input to the RNN. We represent this as matrix $W = [\mathbf{w_{t-c_l}} \cdots \mathbf{w_t} \cdots \mathbf{w_{t+c_l}}] \in \mathcal{R}^{(2c_l+1)\times d}$. Note that while considering event context we stop at sentence boundary, also special symbols are padded if context is less than c_l. The relation between input and output of RNN at each time t is given as follows,

$$h_t = \sigma_h(Q_h w_t + U_h h_{t-1} + b_h) \tag{1}$$

$$o_t = \sigma_o(Q_o h_t + b_o) \tag{2}$$

where, w_t is the word embedding vector provided at each time step, h_t is hidden layer vector and o_t is output vector. Q, U, and b are weight matrices and vector; σ_h and σ_o are activation (ReLU) functions. For a given event, the o_{t+c_l} output vector captures a complete information about the whole sequence. The outputs of the RNN networks give compact representations O_A and O_B of the events.

3.2 Interaction Learning

A deep Convolution Neural Network (CNN) is employed to learn nonlinear interactions from O_A and O_B. It is comprised of three convolution and pooling layers placed alternatively. We feed concatenated learned event representations

$$O_{AB} = O_A \oplus O_B \tag{3}$$

to the first convolution layer, where $\oplus$ is the concatenation operation. Each convolution layer i use filters f_i, after what we compute a feature map

$$m_i^k = \sigma(f_i.O_{AB} + b_i) \qquad \forall k \in \{1, 2, 3\}, \tag{4}$$

where f_i, b_i are filters and bias matrices respectively and σ is the ReLU activation. The output is down-sampled with a max-pooling layer to keep prominent features intact. The output of the last layer gives the interaction between A and B (ρ is max-pooling).

$$O_{comb} = \rho(m_i^3) \tag{5}$$

Systems	Pair Classification			Temporal Awareness		
	P	R	F1	P	R	F1
$w_A \oplus w_B$	39.3	34.2	35.5	27.1	45.8	34.1
$O_A \oplus O_B$	35.7	38.9	37.2	36.5	35.9	36.2
$DCNN(w_A, w_B)$	39.3	36.8	38.1	42.6	35.2	38.5
$MLP(O_A, O_B)$	40.7	38.9	39.7	39.6	38.7	39.1
$CNN(O_A, O_B)$	39.4	41.9	40.6	41.2	38.3	39.7
$DCNN(O_A, O_B)$	42.4	41.3	41.8	46.9	41.5	44.1
ClearTK	-	-	-	33.1	35.0	34.1
LSTM	38.7	43.1	40.5	34.6	51.7	41.4
SP	-	-	-	69.1	65.5	67.2

Table 1: Results of baseline and state-of-the-art systems

3.3 Classification

The combined O_{comb} vector is fed to a fully connected dense layer, followed by a softmax function to get a probability score for each temporal relation class. The temporal relation class is determined according to the maximum probability as

$$\arg \max_{i \in \{1 \cdots n\}} \frac{\exp(h_i^\top O_{comb})}{\sum_{j=1}^{n} \exp(h_j^\top O_{comb})} \tag{6}$$

where n is the number of temporal relations.

4 Experiments

4.1 Datasets and Evaluation

Relations Following recent work [4], reduced set of temporal relations :*after, before, includes, is_included, equal, vague* are considered for classification.

Evaluation Complying with common practice, system's performance is measured over gold event pairs (pairs for which relation is known). Our main evaluation measure is the Temporal Awareness metric [11], adopted in recent TempEval campaigns. We also used standard precision, recall, and F1-score to allow direct comparison with [5].

Datasets We used TimeBank(TB) and AQUAINT (AQ) dataset for training, TimeBank-Dense(TD) for development and Platinum (TE3-PT) dataset for testing. These are the most popular datasets used for the task which have been provided at TempEval3[1].

4.2 Training Details

We used pre-trained Word2Vec vectors from Google[1]. Each word in the context window of event is represented with this 300-dimension vector. Hyperparameters were tuned on the development set using a simple grid search. Considered values are: window size (c_l

[1] `https://code.google.com/archive/p/word2vec/`

:3,4,5), number of neurons at RNN (#RNN:64,128,256,512), number of filters for CNN (#filters: 32,64,128,256), dropout at input (0.1,0.2,0.3,0.4). We also explored a number of optimization algorithms such as AdaDelta, Adam, RMSProp and Stochastic Gradient Descent(SGD). Optimal hyper-parameter values are $c_l = 4$,#RNN =256, #filters = 64, dropout = 0.4 and Adam optimizer.[2] Once we got the optimal parameter values from the validation set, we re-trained multiple models with 50 different seed values on the combined training and development data and report the averaged test performances.

4.3 Comparison to Baseline Systems

We first compare our RNN-Deep CNN approach to various baseline systems to assess the effectiveness of the learned event representations. We also want to disentangle the respective role of the representation learning and interaction learning components.

Baseline systems First, we re-implemented the system of Mirza and Tonelli [5] (noted $w_A \oplus w_B$). Specifically, we used scikit-learn Logistic regression module, using l_2 regularization. Word embeddings w_A and w_B of events A and B, obtained from Word2Vec, were simply concatenated (as this is their best performing system). As additional baselines, we used our Representation Model to learn O_A and O_B, but combined these vectors with simple concatenation ($O_A \oplus O_B$). We did representation learning over pre-trained word embeddings of events to get $CNN(w_A, w_B)$. In another setting learned representation is combined with simple multi-layer perceptron (MLP) and with single-layer convolution (CNN).

Results Table 1 summarizes the performance of these different systems in terms of pairwise classification accuracy and temporal awareness scores. Looking at the first two rows of the table, we see that, as hypothesized, contextually rich features outperform pre-trained event head word embeddings when combined with simple concatenation, both in pairwise classification and in temporal awareness. The gains are more substantial in the latter metric, with a 2.1 absolute F1 increase. Comparing $CNN(w_A, w_B)$ to $w_A \oplus w_B$, we also see that allowing for richer, non-linear interactions between event representations also results in important performance gains in pairwise F1 and temporal awareness F1. Leveraging both contextualized event representations learning and interaction learning yield the best scores overall, cf. $CNN(O_A, O_B)$ and $DCNN(O_A, O_B)$, which shows their complementarity. There, the Deep CNN outperforms the single-layer CNN, with F1 scores of 44.1 and 39.7, respectively.

4.4 Comparison with State-of-the-art

Finally, we now compare the performance results of our best system, $DCNN(O_A, O_B)$, with recently proposed systems : ClearTK [6], which was the winner of the TempEval 2013 campaign, the structured prediction (SP) approach[4], which is the best system to date and recently proposed LSTM based system [9]. Our system(Table 1) delivers substantial improvements over ClearTK and performs very well compared to LSTM based system. However our system lags in comparison with SP as it relies only on sim-

[2]We tried different unidirectional and bidirectional variations of RNN-LSTM and GRU, but RNN gave the best development results.

ple local inference opposed to global inference at learning step in SP. We intend to do global inference with our system as well for more appropriate comparison.

5 Conclusion and Future Work

In this work, we proposed RNN based neural architecture to learn event representation and CNN model to get interaction between events. A new perspective towards combination of events proved to be effective in getting compound interactions. We compared result of our system with multiple baselines and state-of-the art systems and shown effectiveness. We now plan to learn features as well as interaction while considering global consistency in the relations of event pairs.

Acknowledgement

This work was supported by ANR Grant GRASP No. ANR-16-CE33-0011-01, as well as by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020.

References

[1] N. UzZaman, H. Llorens, L. Derczynski, J. Allen, M. Verhagen, and J. Pustejovsky. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. ACL, 2013.
[2] I. Mani, M. Verhagen, B. Wellner, C. M. Lee, and J. Pustejovsky. Machine learning of temporal relations. ACL, 2006.
[3] N. Chambers, T. Cassidy, B. McDowell, and S. Bethard. Dense event ordering with a multi-pass architecture. *TACL*, 2014.
[4] Q. Ning, Z. Feng, and D. Roth. A structured learning approach to temporal relation extraction. EMNLP, 2017.
[5] P. Mirza and S. Tonelli. On the contribution of word embeddings to temporal relation classification. COLING, 2016.
[6] S. Bethard. ClearTK-TimeML: A minimalist approach to tempeval 2013. ACL, 2013.
[7] P. Bramsen, P. Deshpande, Y. K. Lee, and R. Barzilay. Inducing temporal graphs. EMNLP, 2006.
[8] P. Denis and P. Muller. Predicting globally-coherent temporal structures from texts via endpoint inference and graph decomposition. IJCAI, 2011.
[9] Y. Meng, A. Rumshisky, and A. Romanov. Temporal information extraction for question answering using syntactic dependencies in an lstm-based architecture. *CoRR*, abs/1703.05851, 2017.
[10] T. Mikolov, I. Sutskever, K. Chen, G. S Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. NIPS, 2013.
[11] N. UzZaman and J. F. Allen. Temporal evaluation. ACL, 2011.