Event Extraction and Coreferencing

Deep Learning Term Project

Avirup Saha, Gourab Kumar Patro, Soumyadeep Roy
Project Group id - 15

Problem Statement

Given a news document

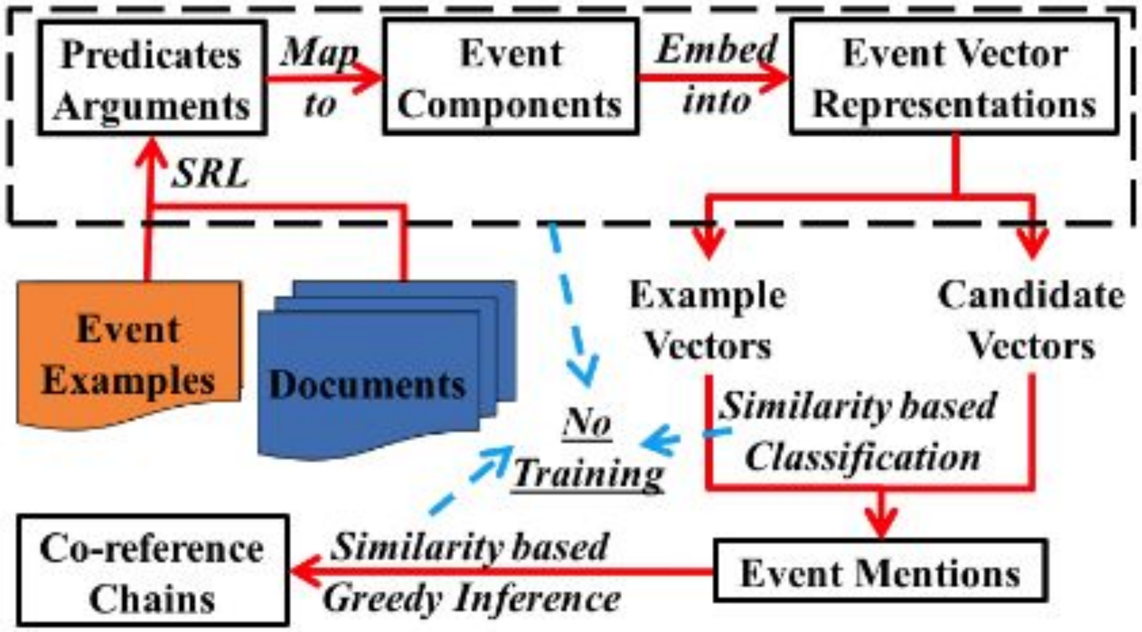
1. Extract the word or phrase that most clearly expresses the occurrence of an event (Event Trigger Detection)
2. Classify the type of the event (Event Classification)
3. Extract arguments with different roles (Argument identification).
4. Identify those event triggers which refer to the same event (event coreferencing)

Dataset - 85 Newswire articles from ACE2005 event extraction dataset

Roadmap

- Event Detection and Co-reference with Minimal Supervision. (EMNLP 2016) by Haoruo Peng, Yangqiu Song, and Dan Roth.
- Event Detection and Domain Adaptation with Convolutional Neural Networks (ACL-IJCNLP 2015) by Thien Huu Nguyen and Ralph Grishman
- Event Extraction via Dynamic Multi-Pooling CNN (ACL-IJCNLP 2015) by Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng and Jun Zhao

Model



An overview of the end-to-end MSEP system. “Event Examples” are the only supervision here, which produce “Example Vectors”. No training is needed for MSEP.

Methodology

1. We use the SRL component of the general purpose NLP tool Senna to pre-process the text.
2. We identify the five most important and abstract event semantic components: action, $agent_{sub}$, $agent_{obj}$, location and time.
3. We convert each event component to its corresponding vector representation.
4. If there are missing event arguments, we set the corresponding vector to be NIL, i.e. set each element of the corresponding vector to zero.

$$\textbf{[event] = [action | agent}_{\textbf{sub}}\textbf{ | agent}_{\textbf{obj}}\textbf{ | location | time]}$$

Methodology (contd.)

1. **Event mention detection** - We use the cosine similarity between an event candidate with the event type representation (avg. of all event vectors under that type) to determine whether the candidate belongs to an event type.
2. **Event co-reference** - We use cosine similarities computed from pairs of event mentions for event co-reference

$$\mathbf{S(e1, e2) = vec(e1).vec(e2) / (||\ vec(e1)\ ||\ .\ ||\ vec(e2)\ ||)}$$

Results

Embedding	Precision	Recall	F1
LexVec (Span)	0.762	0.859	0.808
LexVec (Span+Type)	0.603	0.677	0.638
Word2Vec (Span)	0.688	0.808	0.743
Word2Vec (Span+Type)	0.527	0.622	0.570
GloVe (Span)	0.813	0.904	0.856
GloVe (Span+Type)	0.647	0.733	0.687

Event detection (trigger identification) results

Embedding	MUC	B^3	$CEAF_e$	BLANC
LexVec	0.237	0.771	0.390	0.510
Word2Vec	0.236	0.775	0.399	0.510
GloVe	0.235	0.768	0.386	0.510

Event co-reference results

Roadmap

- Event Detection and Co-reference with Minimal Supervision. (EMNLP 2016) by Haoruo Peng, Yangqiu Song, and Dan Roth.
- Event Detection and Domain Adaptation with Convolutional Neural Networks (ACL-IJCNLP 2015) by Thien Huu Nguyen and Ralph Grishman
- Event Extraction via Dynamic Multi-Pooling CNN (ACL-IJCNLP 2015) by Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng and Jun Zhao

Work Highlights

- The authors formulates the event detection problem as a multi-class classification problem and used CNN in the current setting
    - Given a sentence, for every token we predict whether the current token is an event trigger or not and classify it into one of the event subtypes (Example - Life.Injure)
    - Previous feature-based models require complicated feature engineering and proves difficult when adapting to other domains
- Achieved an F1-score of 0.8623 based on the hyperparameters mentioned by Kim(2014)
    - Compared performance across different window sizes(11, 21, 31) to understand the extent of useful context

Model

- Each token is represented by a vector of size m_t formed by concatenation
  - Word embedding table - Initialized with word2vec of dimension 300 (Mikolov et al, 2013)
  - Position embedding table - Embeds the relative distance between x_i and x_0. Initialized randomly

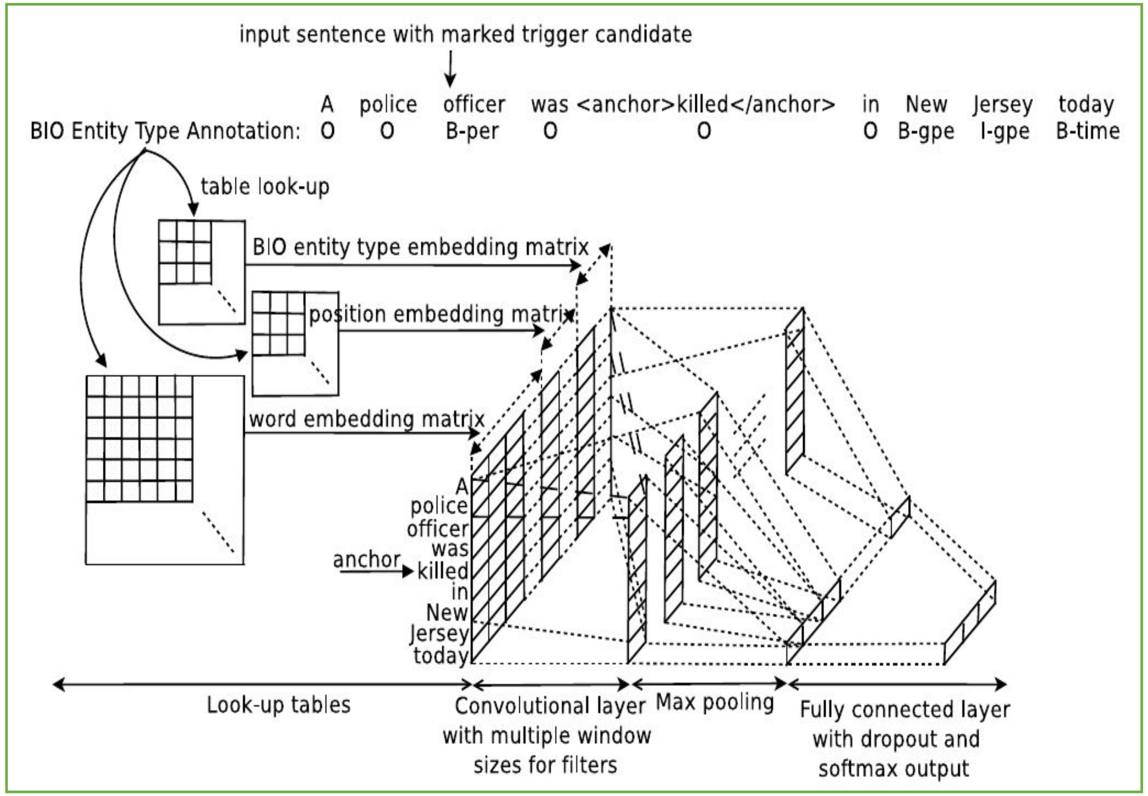


Fig 1 : Proposed CNN architecture for Event Detection

Evaluation and performance

- 43592 sentences were extracted with a vocabulary size of 8777
- For training used 30,514 sentences and 6538 for testing

- Code available on Github[1]
- Future work
  - Add Entity-type embedding
  - Add pos-tag embedding

window size	f1-score
11	0.9001
21	0.87385
31(as in paper)	0.86085

Table 1 : Performance across different window sizes

[1] https://github.com/roysoumya/event_detector

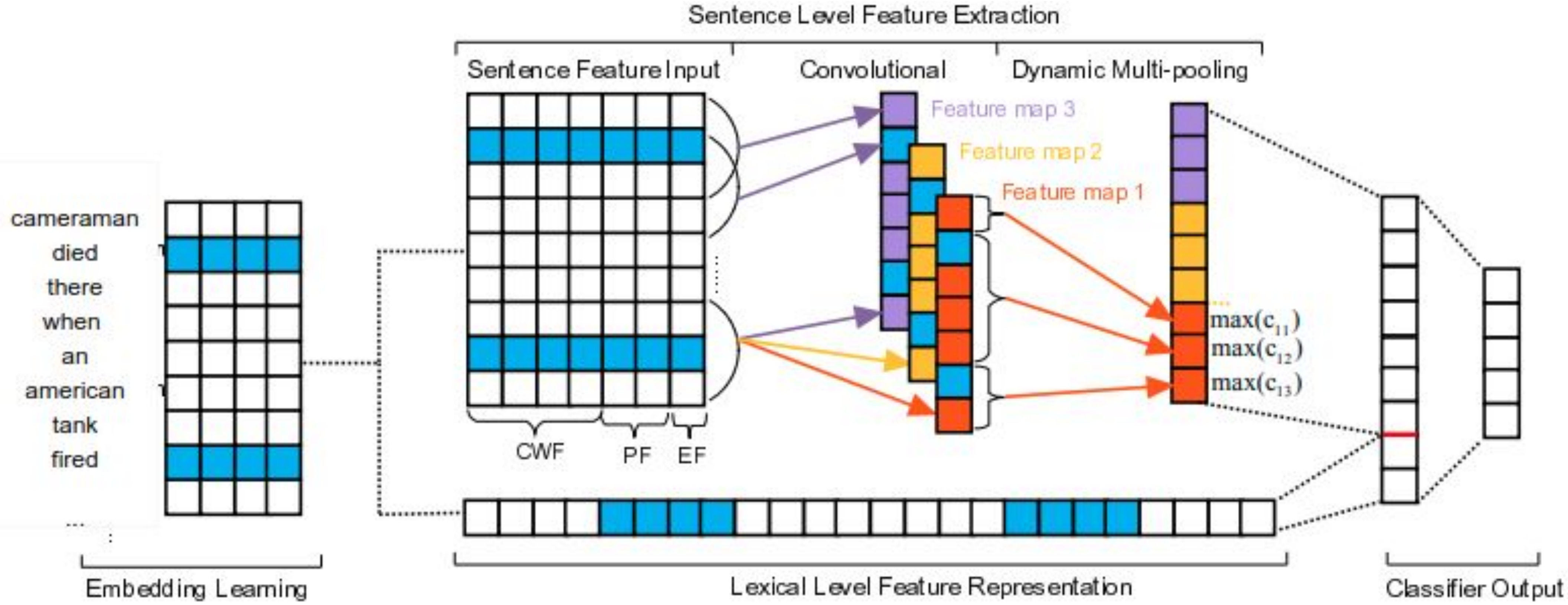
Roadmap

- Event Detection and Co-reference with Minimal Supervision. (EMNLP 2016) by Haoruo Peng, Yangqiu Song, and Dan Roth.
- Event Detection and Domain Adaptation with Convolutional Neural Networks (ACL-IJCNLP 2015) by Thien Huu Nguyen and Ralph Grishman
- Event Extraction via Dynamic Multi-Pooling Convolutional Neural Network (ACL-IJCNLP 2015) by Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng and Jun Zhao

Problems with Earlier Event Detection Approaches

- Using NLP tools : Elaborately designed features
  Lack generalization
  Data Sparsity problem
- Using CNN : CNN has been able to capture sentence level clues, but it can capture only most important information because of the max pooling layer
- CNNs may miss events in case of multiple events in the same sentence (27.3 % of data has multiple events)
- Proposed Approach : Dynamic Multi-pooling CNN

Model



Sentence Level Feature Extraction
Sentence Feature Input
Convolutional
Dynamic Multi-pooling
Feature map 3
Feature map 2
Feature map 1
cameraman
died
there
when
an
american
tank
fired
CWF
PF
EF
max(c11)
max(c12)
max(c13)
Embedding Learning
Lexical Level Feature Representation
Classifier Output

Methodology

- **Embedding Learning** : Skip-gram Word embeddings over the complete corpus
- Context Word feature (**CWF**) : vector with stacked word embeddings
- Position Feature (**PF**) (only used in event type classification not in event detection): position of predicted event trigger word
- Lexical Level Feature Representation : concatenation of word embeddings
- Multiple **Feature Maps** with certain window size (here 3)
- **Dynamic Multi-pooling** (For event detection : feature map into 3 equal parts, For event type classification : feature map into 2 parts before and after the event trigger, then max in all the parts were taken)

Evaluation

Settings: skipgram.w.=2+2,feature map window=3,#feature maps=300,dropout = 0.5

Methods	Trigger Identification(Percentage)	Trigger Identification and Classification
Li's baseline	P=76.2, R=60.5 ,F=67.4	P=74.5, R=59.1, F=65.9
Liao's cross-event	-	P=68.7, R=68.9, F=68.8
Hong's cross-entity	-	P=72.9, R=64.3, F=68.3
Li's structure	P=76.9, R=65.0, F=70.4	P=73.7, R=62.3 ,F=67.5
DMCNN (reported in paper)	P=80.4, R=67.7, F=73.5	P=75.6, R=63.6, F=69.1
DMCNN (implemented here)	P=78.2, R=68.6 ,F=73.1	P=74.4,R=63.1,F=68.2

Table 1: Performance Table

Out of all multiple event cases, 61.2 % were identified correctly.

Thank You !!!