AIPHES-HD system at TAC KBP 2016: Neural Event Trigger Detection and Event Type and Realis Disambiguation with Word Embeddings

Todor Mihaylov
Research Training Group AIPHES
Institute for Computational Linguistics
Heidelberg University
mihaylov@cl.uni-heidelberg.de

Anette Frank
Research Training Group AIPHES
Institute for Computational Linguistics
Heidelberg University
frank@cl.uni-heidelberg.de

Abstract

With this work we participate in the TAC-KBP 2016 Event Nugget Track for Event Nugget Detection. We implement a simple but effective system which uses a Bi-Directional LSTM with embeddings short-cuts to the output for Event Trigger detection and simple Logistic Regression Classifiers with event trigger context features based on word embeddings for Event Type and Realis detection. Our post-submission models yield state-of-the-art results on the official task dataset. We train and evaluate our system for English and it can easily be adapted for Chinese, Spanish or other languages as our best model uses only word embeddings as external resources.

1 Introduction

With this work we participate in the TAC-KBP 2016 Event Nugget Track for Event Nugget Detection. Event Nugget Detection is a task that performs detection of an event trigger and predicts its attributes: the event type (current tasks distinguish up to 33 semantic types) and a *REALIS* feature (for instance, Generic, Actual, Other). In the given example (1) the event triggers are marked in bold and the attributes are shown in brackets.

(1) The **attack**$_{e1}$ [Conflict.Attack, ACTUAL] **killed**$_{e2}$ [Life.Die, ACTUAL] seven and **injured**$_{e3}$ [Life.Injure, ACTUAL] twenty.

This year's task is multilingual and includes evaluation data for English, Chinese and Spanish. We train and evaluate our system for Event Nugget Detection for English only. Our system uses only Stanford CoreNLP parses and word embeddings as resources for training a neural network system, so it can easily be adapted for Chinese and Spanish.

This rest of the paper is organized as follows: in Section 2 we present and overview of our system for the Event Nugget Detection task. In Section 3 we describe the used datasets, the used evaluation and scoring and the pre-processing step that we apply to the data. In Section 4 we describe the training and tuning of the system as well as the achieved results. In Section 5 we present an overview of recent work in the field of event Detection. In Section 6 we present a brief overview of our work and our best model.

2 System Overview

Our Event Nugget Detection system consists of 3 modules which can be run independently: Event Trigger Detection, Event Type classification and Event Realis Classification.

2.1 Event Trigger Detection

We tackle the Event Trigger Detection task as a sequence labeling task. For our official submissions to the task we initially used the BIO scheme to encode the Event Trigger in the sentence tokens, where *B-EVENT* marks the beginning token of a trigger event, *I-EVENT* a token that is part of the event span in the golden annotation and is not the event beginning token and *O* marks a token that does not participate as an event trigger in the current sentence. However we later dropped the multi-token events from training since our post-submission experiments have shown that using multi-word event triggers in the training yields worse results on the evaluation datasets, as they contain only single word event triggers. For the sequence labeling task we use a bi-directional Long Short-Term Memory (Bi-LSTM) (Schuster and Paliwal, 1997) based neural architecture as shown in Figure 1. Each input example consists of a sentence as extracted from the input document in the pre-processing step. In our base system we implement the first layer as a word embedding lookup, which retrieves the embedding vector for each input token of the sentence. The output of the first layer is fed into a single Bi-Directional LSTM layer. The output of every left-to-right LSTM cell and right-to-left LSTM cell is concatenated and fed to a SoftMax layer which outputs one of the BIO classes mentioned above. We use word2vec (Mikolov et al., 2013) word embeddings trained on the Google News 1B corpus and optimize them during training. We also make experiments with adding Part of Speech (POS) tag embeddings as well as the sum of Dependency tag embeddings '(DEP) for each token. The POS and DEP type embeddings are randomly initialized and optimized during training. We experiment with different combinations of using these three types of word embeddings. If we use more than one embedding type (Word embeddings, PoS embeddings or Dependency sum embeddings), we concatenate them. We also experiment with shortcutting, by concatenating the retrieved embedding representations with the output layer of the Bi-LSTMs (see the dashed arcs in Figure 1).

2.2 Event Type and Realis Detection

For Event Type and Realis classification we train Logistic Regression classifiers with features based on word embeddings. We use the following features:

- Averaged word embedding vectors over all tokens in the event trigger.
- Averaged word embedding vectors over all tokens in the sentence.
- Averaged word embedding vectors for the 2 tokens preceding the event trigger and 2 tokens following the event trigger, if any.
- Averaged word embedding vectors for all tokens with a direct connection to the event trigger token in the dependency tree, where the event trigger is the head in the dependency triple.
- Averaged word embedding vectors for all tokens with a direct connection to the event trigger token in the dependency tree, where the event trigger is the not the head in the dependency triple.
- Averaged word embedding vectors for all tokens with a direct connection to the event trigger token in the dependency tree. This includes all tokens from the previous two features.

The above centroid word vectors are used as features with dimensionality of the chosen embedding size. In our model we chose word embeddings of size 300. A similar approach including semantic features based on word embeddings yielded very good results in other tasks such as Discourse Relation Sense classification (Mihaylov and Frank, 2016). Community Question Answering (Mihaylov and Nakov, 2016; Mihaylova et al., 2016) and story understanding (Mihaylov and Frank, 2017).

3 Data and Evaluation

In this section we present the used datasets, data preporcessing that we apply and the scoring and evaluation for the task.

3.1 Datasets

We use the training data provided by the task organizers - *LDC2016E36*, which contains the data originally released for the DEFT 2014 Event Nugget Evaluation (LDC2014E121, LDC2015E03, LDC2015E69) and data for the TAC KBP 2015 Event Nugget Tracks (LDC2015E73, LDC2015E94, LDC2015R26). For the official evaluation and post-submission experiments we use the provided evaluation data (LDC2016E72) which contains 44 documents in English. According to the structure of the *data* directories in the *LDC2016E36* and *LDC2016E72*, we split the provided data into 5 parts, as shown in Table 1, specified in the *Data set* column. In Table 1, column *Docs* shows the number of documents in every subset, column *Name* contains the name of the subset or a combined name of combination of subsets used in the text below for

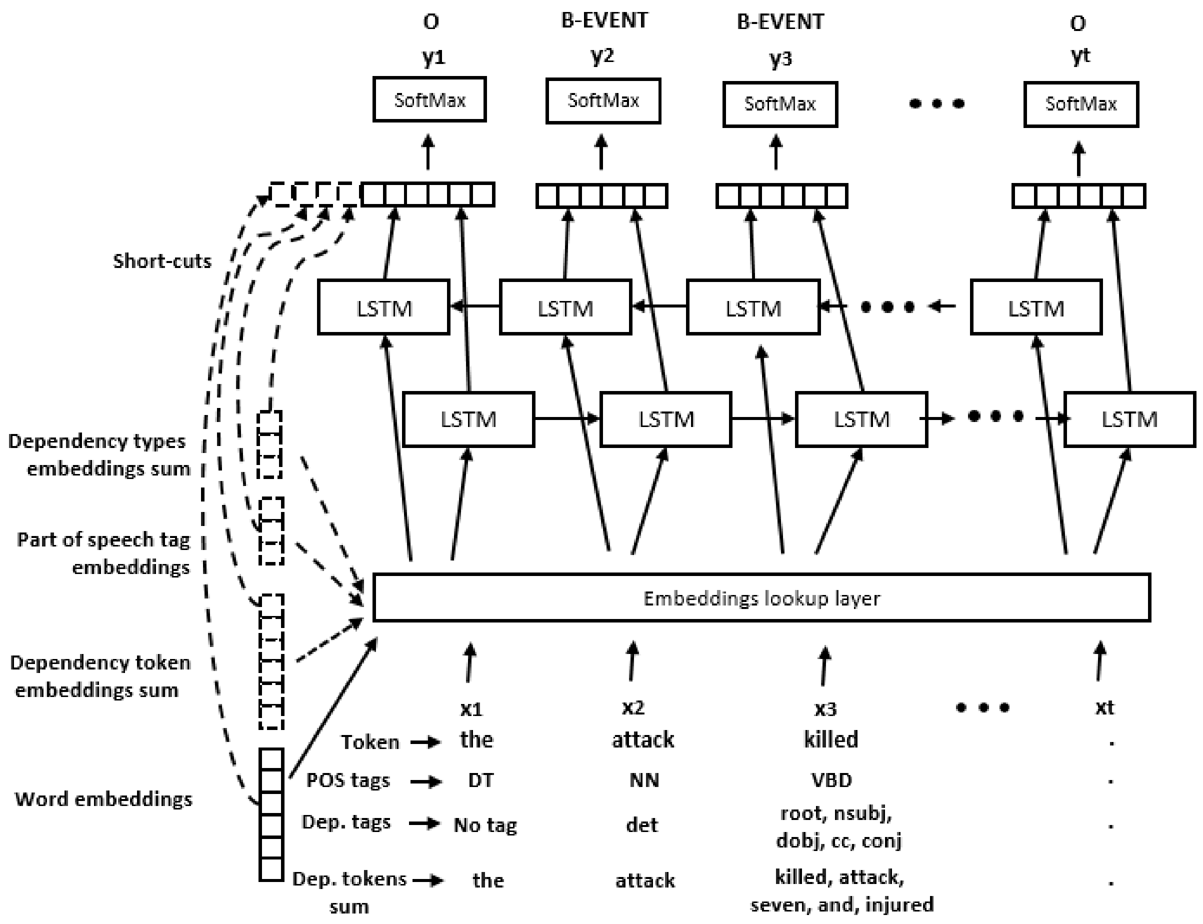

Figure 1: Event Trigger labeling module neural architecture. Dotted lines show optional modifications using additional dependency type embeddings and part of speech embeddings as well as short-cutting the embeddings from the input to the Bi-LSTM output layer.

convenience. Column *Tokens* shows the number of tokens after the pre-processing step described below and *Events* show the number of events (tokens with non-zero (non-*O*) label) among all tokens in the set. We can see that *train*, *dev* and *eval* have different distributions of non-zero event labels.

For most of our experiments we train our system on *train* (combined *train 2014*, *eval 2014* and *train 2015*), tune the model on *dev* (represented by *eval 2015*) and evaluate on *eval* (represented by *eval 2016*) for our official submission.

3.2 Pre-processing

Before we feed our system with the input data, we pre-process the raw HTML files to extract raw text and text tokens. For each given HTML document we perform the following text clearing operations:

Data sub-set	Docs	Name	Tokens	Events
train 2014	151			
eval 2014	200	train	320k	16k
train 2015	158			
eval 2015	202	dev	180k	9k
eval 2016	44	eval	105k	3.5k

Table 1: Data used for training and evaluation. The provided dataset parts are combined/renamed for convenience to *train, dev, eval*. The number of tokens and events for *train* includes the sum of all three sub-sets *train 2014, eval 2014* and *train 2015*

- Replace *img* HTML tags with the *IMG* token.
- Replace opening *a* (hyperlink) tags with the *HYPERLINK* token.

- Replace all other tags with a sequence of empty space symbols corresponding to their character length.

We preserve character offsets by inserting additional empty space characters when necessary. We then process the clear text of the document with the Stanford CoreNLP (Manning et al., 2014) parser in order to obtain sentence and token splitting, as well as Part of Speech and dependency tags for every token. For every token in the parsed data we check if its offset overlaps with tokens in the golden annotation (extracted from the provided annotations in Brat format) and generate and assign them labels either *B-EVENT* or *I-EVENT*[1] or *O* if they do not match. We use the output of the Stanford CoreNLP parser as an input data format for our system.

Event Types In the TAC 2016 Event Nugget Detection task, only 18 event types are selected for evaluation, in contrast to 38 on the TAC 2015 Event Nugget Detection task. The selected types are shown in Table 2.

Type	**Subtype**
Conflict	Attack
Conflict	Demonstrate
Contact	Broadcast
Contact	Contact
Contact	Correspondence
Contact	Meet
Justice	Arrest-Jail
Life	Die
Life	Injure
Manufacture	Artifact
Movement	Transport-Artifact
Movement	Transport Person
Personnel	Elect
Personnel	End-Position
Personnel	Start-Position
Transaction	Transaction
Transaction	Transfer-Money
Transaction	Transfer-Ownership

Table 2: Event types in TAC 2016 Event Nugget Detection task.

3.3

Evaluation and Scoring The official scoring[2] on the Event Nugget Detection tasks is performed by the official scorer provided by the organizers[3]. This evaluation provides scores in terms of macro- and micro-average precision, recall and f-scores for Event Trigger Detection *(plain)*, Event Type disambiguation *(mention_type)* and Event Realis Disambiguation *(realis_status)* and an overall summary score *(mention_type+realis_status)*. These evaluations are used for the official submissions. Note that the score for Event Type and Realis detection is calculated on the detected event triggers, which propagates the error from event trigger detection. Also some event tokens in text trigger two event types (this applies to around 9% of all event triggers). In Ex.2, for instance, the trigger 'kill' triggers *Conflict.Attack* and *Life.Die* event. Such cases are also captured by the official scorer in the reported Event Trigger detection *(plain)*. That means that if there are 100 event triggers in the evaluation document, it is likely that 9 of them trigger 2 event types and the scorer will expect the system to output 109 event triggers.

(2) The terrorists **killed**$_{e1}$ [(Conflict.Attack, ACTUAL);(Life.Die, ACTUAL)] seven and **injured**$_{e2}$ [Life.Injure, ACTUAL] twenty.

4 Experiments and Results

In this section we describe implementation details and experimental setups that we have examined as well as our evaluation results.

4.1 Event Trigger Detection

For Event Trigger Detection we used the neural network architecture based on Bi-Directional LSTM networks as shown in Figure 1. For model implementation we use the Tensorflow package (Abadi et al., 2015). We examine different configurations and

[1]Our experiments have shown that dropping multi-word event triggers from the *train* data improves the results on the evaluation dataset *eval* since it does not contain multi-word event triggers.

[2]http://cairo.lti.cs.cmu.edu/kbp/2016/event/Event-Mention-Detection-scoring-2016-v29 Official Event Nugget Detection and Coreference Scoring for TAC KBP 2016.

[3]https://github.com/hunterhector/EvmEval/releases/tag/v1.7.3 - Official evaluation scorer for the Event Nugget Detection task.

hyper-parameters for training and evaluation of the model.

Embeddings. For the input to the neural model we use word embeddings retrieved for every token in the sentence, part of speech embeddings, retrieved for the POS tag of the token, the sum over the dependency tag embeddings of all dependency triples where the current token is the head, as well as the sum over the token word embeddings of all tokens which are in direct dependence of the current token. We experiment with different combinations of using these three types of word embeddings. If we use more than one embedding type (Word embeddings, PoS embeddings or Dependency sum embeddings), we concatenate them. We also experiment with shortcutting, by concatenating the retrieved embedding representations with the output layer of the Bi-LSTMs (see the dashed arcs in Figure 1).

Word embeddings (WE). We initialize the word embeddings with word2vec vectors with 300 dimensions (Mikolov et al., 2013) pretrained on the Google News 1B corpus and optimize them during training. We also performed experiments with randomly initialized word embeddings with various sizes but they performed worse than the word2vec embeddings. We initialize the words vocabulary with all words that occur at least 5 times in the training set. We initialize the embeddings with the word2vec embeddings for words which are found in word2vec or if their lower-cased version is found in the word2vec vocabulary. All other words in the vocabulary are initialized randomly. We also add an *UNKNOWN* token which we initialize with the centroid of all words in the word2vec vocabulary. During training and evaluation we replace out-of-vocabulary tokens with the *UNKNOWN* token.

Dependency tokens word embeddings sum (DT). Here, every token in the input sequence we retrieve all dependency tokens from the dependency triples where the current token is the head and sum them. In this sum we also include the current token (see Figure 1). Our assumption is that this way we can present the dependency context which in the case of event detection might be event arguments.

Part of Speech embeddings (POS). We retrieve a vocabulary of all POS tags in the training data (these are usually all POS tags produced by the Stanford CoreNLP parser). We initialize the embeddings randomly and optimize them during training. For every token in the tagged input sequence we retrieve its POS tag embedding. We use embedding vectors with size 30.

Dependency type embeddings sum (DEP). We retrieve a vocabulary of all dependency type labels from the parse trees in the training data (these are usually all basic dependencies produced by the Stanford CoreNLP parser). We use dependency type embeddings with size 30 and we initialize them randomly and optimize them during training. For every token in the input sequence we retrieve all dependency types from the dependency triples where the token is the head and sum them.

In our model we use one Bi-Directional LSTM (Schuster and Paliwal, 1997) layer. It consists of left-to-right and right-to-left LSTM cells whose output is concatenated for every token. In our primary implementation the left-to-right and right-to-left LSTM cells share the same parameters. We experimented with LSTM cells with non-shared parameters but they performed worse.

Training and evaluation.

Hyperparameters. We perform experiments with configurations of the model using various hyperparameters:

- Embeddings initialization. Random and initialized from word2vec model. Embeddings initialized with word2vec vectors perform much better so below we report results with this initialization type.
- Embeddings optimization. We experimented with static word2vec embeddings and optimizing them during training. Optimization of the embeddings performed better than using static ones and results in much faster convergence of the model. A reasonable explanation is that around 10 percent of the words in the training data vocabulary are not covered in the word2vec vocabulary and we initialize these randomly.
- LSTM cell output size. We experiment with different sizes of the LSTM cell such as 256,

300, 512, 600, 1024, 2048, 4096. On most performed experiments LSTM with output layer size of 512 performed well so we use this setting in most of our experiments.[4]

- Number of training epochs. We experiment with training our models with up to 100 epochs. However after 10 epochs the models start to overfit the training data, which lowers the results on the given evaluation datasets. We therefore train the model in our further experiments for up to 10 epochs. Our post-submission experiments with evaluation on the *dev* dataset has shown that best results for all configurations are obtained after the second epoch.

- Computing the gradients. We use *AdamOptimizer* for computing the gradients during training with initial learning rate of 0.001. We experimented with different initial learning rates including (0.1, 0.01, 0.001, 0.0001).

For model tuning we keep the learned parameters from the epoch (referred to as *best_epoch* below) on which the evaluation of the model performs best on the *dev* dataset. We evaluate the model on the *eval* dataset using the saved learned parameters. We then train a new model on *train+dev* datasets for 2 epochs (optimal performance on *dev*) and evaluate on *eval*. We re-train the model 3 times for every configuration and report the result that performed best. The results are given in Table 3. Note that in this table we present results for event trigger detection without taking into account multi-type evaluation. This means that results in this table are higher (by around 9%) than the *plain* results calculated by the official score.

In Table 3 we can see that the best performing configuration (in terms of F-score) uses only WE (word embeddings) as input and short-cut with word embeddings (WE). This configuration also yields the best precision. The second best result model (-0.02) uses word embeddings (WE) and direct token dependency sum (DT) in the input layer without short-cut on to the output. The third best F-score is achieved by a model that uses WE on the input layer and WE and DT on the output layer. The overall F-score results show that POS embeddings and DEP (dependency embeddings sum) does not improve the model. However, using all representations (WE, POS, DEP and DT) on both input and short-cut on output layer we achieve the best recall.

[4]Note that for the Bi-LSTM layer that we use, we have 2 LSTM cells and thus the concatenated output layer size is twice the LSTM cell size: 512, 600, 1024, 1200 etc.

4.2 Event Type Classification

For event type classification we build a simple model containing only features based on word embeddings as described in Section 2.2. For each event trigger we extract the features mentioned above. The number of features sums up to 1800 (6 feature groups with 300 features each). We scale the features to the range -1 to 1. We train and evaluate a L2-regularized Logistic Regression classifier with the LIBLINEAR (Fan et al., 2008) solver as implemented in *scikit-learn* (Pedregosa et al., 2011). For our experiments in event type classification, we set the C (cost) parameter to 0.1. We perform experiments for event type detection for 38 types and 18 types. In Table 5 we report the results on the **gold labels** only (this does not propagate the error of event trigger detection, unlike the official scorer).

Experiments with TAC 2015 data setup. We train the model on all event triggers with event types from the TAC 2015 Event Nugget Detection Task in *train 2014, eval 2014, train 2015* (14948 instances) and evaluate on all event triggers in *eval 2015* (5689 instances). Results are presented in Table 5, row 1.

Experiments with TAC 2016 data setup. We train the model on all event triggers which belong to event type in the TAC 2016 Event Nugget Detection task selected 18 event types in *train 2014, eval 2014, train 2015, eval 2015* (14899 instances) and evaluate on the event triggers in *eval 2016* (3595 instances). Results are presented in Table 5, row 2.

4.3 Event Realis Classification

For the Realis classification we use the same feature set and model as the one explained above (Section 4.2). For our experiments in realis classification, we set the C (cost) parameter to 0.2 as it yielded best results in 5 fold cross-validation on the train set.

In Layer	Out Layer + Short-cut	Precision	Recall	F1
Word	-	57.72	59.06	58.38
Word	Word	**58.19**$_1$	58.94	**58.56**$_1$
Word	DEP	57.31	59.28	58.28
Word	DT	57.28	59.74	58.48
Word	POS	56.79	57.80	57.29
Word	Word DEP	58.11$_2$	58.66	58.39
Word	Word DT	57.77	59.31	58.53$_3$
Word	Word POS	57.04	57.39	57.21
Word DEP	-	57.08	59.55	58.29
Word DT	-	57.84$_3$	59.25	58.54$_2$
Word POS	-	56.53	59.34	57.90
Word DEP	Word	57.24	59.67	58.43
Word DT	Word	56.41	60.29$_2$	58.29
Word POS	Word	55.48	59.92$_3$	57.61
Word POS DEP DT	Word POS DEP DT	54.63	**60.56**$_1$	57.45
Word	Word POS DEP DT	57.22	58.79	57.99
Word POS DEP	Word	56.04	59.71	57.82
Word POS DEP DT	Word	56.42	59.03	57.70

Table 3: Experiments with different feature representations on input layer and short-cut on the output layer. Rankings of top 3 scores for each measure are shown with subscript. (Word - Word embeddings; DEP - Dependency tag embeddings sum; POS - Part of speech tag embeddings; DT - Direct token dependencies word embeddings sum). (This is not the same score as in the official evaluation as it does not take into account multi-label event triggers which is the case in the official scorer.)

	Micro average			Macro average		
Attributes	Prec	Rec	F1	Prec	Rec	F1
plain	58.41	51.60	54.80	55.71	49.68	52.52
mention_type	48.45	42.81	45.45	45.78	40.96	43.23
realis_status	46.17	40.79	43.31	43.64	39.06	41.22
mention_type+realis_status	38.07	33.64	35.72	35.65	31.97	33.71

Table 4: Post-submission. This table shows results for our best post-submission system, using WE for the input and WE short-cut on the output (official scorer results).

Setup	Prec	Rec	F1
TAC 2015, 38 types	76.11	69.33	69.60
TAC 2016, 18 types	70.21	67.51	66.91

Table 5: Event Type classification on data setup from TAC 2015 and TAC 2016. Macro average on instances extracted from the gold datasets. (This is not the same value as the official evaluation as it does not propagates the error of event trigger detection which is calculated in the official scorer.).

Experiments with TAC 2015 data setup. We train the model on all event triggers with event types from TAC 2015 Event Nugget Detection Task in *train 2014, eval 2014, train 2015* (15777 instances) and evaluate on all event triggers in *eval 2015* (5689 instances). Results are presented in Table 6, row 1.

Experiments with TAC 2016 data setup. We train the model on all event triggers in *train 2014, eval 2014, train 2015, eval 2015* (21466 instances) and evaluate on the event triggers in *eval 2016* (3595 instances). Results are presented in Table 6, row 2.

Setup	Prec	Rec	F1
TAC 2015	69.46	61.41	64.25
TAC 2016	66.00	66.41	64.80

Table 6: Realis classification on data from TAC 2015 and TAC 2016. Macro average on instances extracted from the gold datasets. (This is not the same value as the official evaluation as it does not propagates the error of event trigger detection which is calculated in the official scorer)

4.4 Comparison to other systems

In Table 7 we present comparison between our best system and other top systems participating in the TAC KBP 2016 Event Nugget Detection Task. The in column *Task*, *plain* represents the Event Trigger detection task, *type* shows results for the Event Type disambiguation task, *realis* represents the Event Realis disambiguation task and *overall* shows the result for all attributes of event nugget detection. The results for every sub-task are ranked F1 score.

4.5 Official submissions

Above we described our experiments with various post-submission setups which improved our results. In this Section we discuss our official results and experiments. Initially we trained our Event Trigger detection model on a data setup for TAC 2015 with event nuggets on all 38 types. In our official submissions we evaluate on *eval 2016 (English)* with this model. The drawback here is that this model tries to recognize all event triggers for **38 event types** instead of the selected **18 event types** from TAC 2016. We then perform the Event Type and Realis classifications on the extracted event triggers, trained on event triggers over 38 event types. For the official submissions we keep the output only for detected events in the selected 18 types for TAC 2016. This way we propagate the error from the Event Type disambiguation task on the Event Trigger detection task which explains our low official scores. We fixed this and after the official submission we trained the model with only 18 event types.

For the Event Type and Realis detection, for all runs, we use all feature groups described in Section 2.2 except for the last one (averaged word embedding vectors for all tokens with a direct connection to the event trigger tokens in the dependency tree) which we introduced later.

Task	System name	Prec	Rec	F1
plain	**Our system**	58.41	51.60	54.80
	UTD1	55.36	53.85	54.59
	NYU3	51.02	57.52	54.07
	SoochowNLP3	58.11	45.17	50.83
	LTI-CMU1	69.82	39.54	50.49
type	UTD1	47.66	46.35	46.99
	Our system	48.45	42.81	45.45
	LTI-CMU1	61.69	34.94	44.61
	wip1	51.76	38.98	44.47
	NYU3	41.88	47.21	44.38
realis	**Our system**	46.17	40.79	43.31
	NYU2	40.53	45.07	42.68
	UTD1	40.34	39.23	39.78
	SoochowNLP3	43.84	34.08	38.35
	wip3	42.86	32.49	36.96
overall	**Our system**	38.07	33.64	35.72
	NYU1	33.47	37.21	35.24
	UTD1	34.05	33.12	33.58
	wip3	38.38	29.10	33.10
	SoochowNLP3	37.26	28.97	32.59

Table 7: Comparison of our refined post-submission system and other systems from the official evaluation. Reported results are Macro-Average precision, recall and F1 score, produced by the official scorer.

The approach and training datasets described above are valid for all three runs that we have submitted. Below we describe the run specific settings for each submission and report the official submissions scores.

Submission 1. We perform Event Nugget detection with our base 1-layer Bi-LSTM model only with word embeddings on the input layer. We use word2vec embeddings with size 300 and LSTM with output size 600. We trained the model on *train 2014, eval 2014, train 2015* and took the best performing parameters snapshot for the *eval 2015* dataset. We then extract Event Triggers and assign them Event Type and Realis classes. For the Event Type classification we use C=0.1 and for Realis Classification we use C=0.2. This is our best official submission.

	Micro			Macro		
Attributes	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
plain	52.89	42.06	46.85	49.49	40.50	44.55
mention_type	36.83	29.28	32.62	33.58	27.96	30.51
realis_status	41.57	33.06	36.83	38.35	31.42	34.54
mention_type+realis_status	29.94	23.81	26.53	26.98	22.33	24.44

Table 8: Official submission 1. Model trained in TAC 2015 data setup with 38 event types instead of 18. Results are filtered in 18 types during submission export.

Submission 2. We perform Event Nugget detection with our base 1-layer Bi-LSTM model with word embeddings and DEP embeddings with size 50 on the input layer. We use LSTM with output size 300. We trained the model on *train 2014, eval 2014, train 2015* and took the best performing parameter snapshot for the *eval 2015* dataset. For the Event Type classification we use C=1 and for Realis Classification we use C=2.

Submission 3. We perform Event Nugget detection with our base 1-layer Bi-LSTM model with word embeddings and POS embeddings with size 50 on the input layer. We use LSTM with output size 300. We trained the model on *train 2014, eval 2014, train 2015* and took the best performing parameter snapshot for the *eval 2015* dataset. For the Event Type classification we use C=0.1 and for Realis Classification C=0.2.

5 Related Work

Most recent work in event detection includes deep learning systems. Recent state of the art systems on event datasets such as ACE2005 use architectures based on Convolutional Neural Networks (Chen et al., 2015; Nguyen and Grishman, 2015) and Recurrent Neural Network variations (Nguyen et al., 2016a; Jagannatha and Yu, 2016). Some approaches use multi-stage approaches for extending event detection to new types via extended neural networks (Nguyen et al., 2016b).

Earlier state-of-the-art feature-based event detection approaches include joint event extraction via structured prediction employing global document features (Li et al., 2013) and minimally supervised approaches (Bronstein et al., 2015) use event type descriptions to extract actual event triggers.

In the TAC KBP 2015 Event Nugget track best results in event trigger detection were achieved by a system based on deep neural networks and gradient boosted decision trees (Reimers and Gurevych, 2015) followed by robust, feature-rich pipelined systems (Zhengzhong Liu and Hovy, 2015; Yu Hong and Ji, 2015).

6 Conclusion

With this work we participate in the TAC-KBP 2016 Event Nugget Track for Event Nugget Detection. We implement a system which uses Bi-Directional LSTM for Event Trigger detection and simple Logistic Regression Classifiers with event trigger context features based on word embeddings for Event Type and Realis detection.

We experiment with various configurations of our models including using Part-of-speech and Dependency tag embeddings as additional information for our Event Trigger detection system. We perform experiments with short-cuts to the output layer and we show that they improve results. Our post-submission model for event detection with word embeddings short-cut on the output layer yields state-of-the-art results on the official task dataset.

We train and evaluate our system for English and it can easily be adapted for Chinese, Spanish or other languages as our best model uses only word embeddings as external resources.

Acknowledgments. This work is supported by the German Research Foundation as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ofer Bronstein, Ido Dagan, Qi Li, Heng Ji, and Anette Frank. 2015. Seed-Based Event Trigger Labeling: How far can event descriptions get us? *Proc. 53rd Annu. Meet. Assoc. Comput. Linguist. 7th Int. Jt. Conf. Nat. Lang. Process. (Volume 2 Short Pap.*, pages 372–376.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.

Abhyuday N. Jagannatha and Hong Yu. 2016. Bidirectional rnn for medical event detection in electronic health records. In *HLT-NAACL*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint Event Extraction via Structured Prediction with Global Features. *Proc. 51st Annu. Meet. Assoc. Comput. Linguist. (Volume 1 Long Pap.*, pages 73–82.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June. Association for Computational Linguistics.

Todor Mihaylov and Anette Frank. 2016. Discourse Relation Sense Classification Using Cross-argument Semantic Similarity Based on Word Embeddings. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task*, pages 100–107, Berlin, Germany. Association for Computational Linguistics.

Todor Mihaylov and Anette Frank. 2017. Story cloze ending selection baselines and data examination. In *Proceedings of the Second Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics Shared Task*, Valencia, Spain. Association for Computational Linguistics.

Todor Mihaylov and Preslav Nakov. 2016. SemanticZ at SemEval-2016 Task 3: Ranking relevant answers in community question answering using semantic similarity based on fine-tuned word embeddings. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, USA.

Tsvetomila Mihaylova, Pepa Gencheva, Martin Boyanov, Ivana Yovcheva, Todor Mihaylov, Momchil Hardalov, Yasen Kiprov, Daniel Balchev, Ivan Koychev, Preslav Nakov, Ivelina Nikolova, and Galia Angelova. 2016. SUper Team at SemEval-2016 Task 3: Building a feature-rich system for community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, USA.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '13, pages 746–751, Atlanta, Georgia, USA.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *ACL*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016a. Joint event extraction via recurrent neural networks. In *HLT-NAACL*.

Thien Huu Nguyen, Lisheng Fu, Kyunghyun Cho, and Ralph Grishman. 2016b. A two-stage approach for extending event detection to new types via neural networks.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Nils Reimers and Iryna Gurevych. 2015. Event nugget detection, classification and coreference resolution using deep neural networks and gradient boosted decision trees. In *Proceedings of the Eighth Text Analysis Conference (TAC 2015)*.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November.

Dian Yu Xiaoman Pan Lifu Huang Yu Hong, Di Lu and Heng Ji. 2015. Rpi blender tac-kbp2015 system description. In *Proceedings of the Eighth Text Analysis Conference (TAC 2015)*.

Dheeru Dua Teruko Mitamura Zhengzhong Liu, Jun Araki and Eduard Hovy. 2015. Cmu-lti at kbp 2015 event track. In *Proceedings of the Eighth Text Analysis Conference (TAC 2015)*.