Event Nugget Detection Task : UMBC systems

Taneeya Satyapanich and Tim Finin
University of Maryland, Baltimore County
Baltimore, MD, 21250, USA
taneeya1@umbc.edu, finin@umbc.edu

Abstract

This paper described our Event Nugget Detection system that we submitted to the TAC KBP 2016 Event Track. We sent out two runs; UMBC1 and UMBC2. UMBC1 is a sentence-level classification system based on Convolution Neural Network and applied the probability to select a word as an event nugget. UMBC2 is the classification model trained from our features using Weka and filtered out low confidence prediction output using threshold. Our performance is quite low; we got F1 measure of 34.14 for UMBC1 and 35.24 for UMBC2.

1 Introduction

Text Analysis Conference Knowledge Base Population (TAC KBP) 2016 is an evaluation workshop in Natural Language Processing. Several tasks had been launched such as Entity Discovery, Event Extraction, Sentiment analysis. They provided collections of annotated data such as newswires, discussion forums, including their source text that can be used as training corpus. Besides, they have evaluation procedure to evaluate all of the participant system output. The Event Nugget track (Mitamura and Hovy, 2016) is one of the two tasks in the Event track this year. They provided three languages test data in the Event Nugget Detection task, our system focused only the English documents. Event Nugget Detection goal is to identify all of the event mentions in document, also classify the event into an event type/subtype. The event types/subtypes are defined in the (DEFT Rich ERE Annotation Guidelines: Events v.2.5.1, 2015). In addition, systems must identify Realis attribute (ACTUAL, GENERIC, OTHER), which are also described in the Rich ERE guidelines.

Event means something that happens or may not happen at a particular place and time but it is mentioned in the document. There have been multiple research that studied the event mention detection. Supervised machine learning such as (Sammons, et.al., 2014; Grishman, et.al., 2005); (Sammons, et.al., 2014) used supervised machine learning method plus measuring overlapping of the event argument and the world knowledge corpus, (Grishman, et.al., 2005) used Maximum Entropy based classifiers to detect trigger word to distinguish event mentions from non-event-mentions. Unsupervised machine learning has been used in (Ji and Grishman, 2008). They created some inference rules on statistics value of detected triggers word that are associated with particular types of events to improve the cross-document event extraction. (Li et.al., 2013) proposed structured prediction method and global features to predict the event mention and its argument that occur in the same sentence simultaneously. (Liu, et.al., 2015) used a discriminatively trained Conditional Random Field (CRF) model to detect event mention span and event type. Recently, deep neural network had been used in (Reimersy and Gurevychyz, 2015). (Reimersy and Gurevychyz, 2015) trained deep feed forward network with features such as subject, object, lemma, part of speech, and word embeddings to detect event nuggets.

For our systems, we considered the event nugget detection as the classification problem. Our systems

was built based on an idea that any words or phrases that mentioned the same event type always has high semantic similarity. We will described both systems description about features, classification model parameters in section 2. The training data and development data we used are described in section 3. The official experimental results are in section 4. Discussions are in section 5. And conclusion is in section 6.

2 System Description

For the Event Nugget Detection task (Mitamura and Hovy, 2016), we have to detect the event mentioned in the document and classify the event to its type. There are eighteen event types; Conflict.Attack, Conflict.Demonstrate, Contact.Broadcast, Contact.Contact, Contact.Correspondence, Contact.Meet, Justice.Arrestjail, Life.Die, Life.Injure, Manufacture.Artifact, Movement.TransportArtifact, Movement.TransportPerson, Personnel.Elect, Personnel.Endposition, Personnel.Startposition, Transaction.Transaction, Transaction.Transfermoney, Transaction.TransferOwnership. Nuggets of the eighteen event types have to be identify with mention span and event type if they appeared in the document. Besides, the Realis values are ACTUAL, OTHER, GENERIC has to be chosen for each detected nuggets. We built two classification systems called UMBC1 and UMBC2. They have difference procedures and features. Both system used the same Realis procedure. Details of UMBC1, UMBC2 and Realis procedures are described in the following sections.

2.1 UMBC1

This system consists of two main procedures: Classification and Selection. The classification procedures was developed from Sentence Classification using Convolution Neural Network (Kim, 2014). Convolution Neural Network was introduced to do sentiment analysis for movie review data. Their system can predict the sentiment of sentences. They used word vector to represent each word in the sentence. The word matrix is built for each sentence by concatenating word vector. The Neural Network was trained with these word matrix. We applied their system to classify sentence to one of nineteen possible events (18 events plus non event). The pre-trained word vectors trained on part of Google News dataset (about 100 billion words) called GoogleNews-vectors-negative300.bin.gz (Mikolov, et.al., 2013) will be used in any word embeddings task for UMBC1 system. The model contains 300-dimensional vectors for 3 million words and phrases.

To train the Convolution Neural Network, the training data has to be prepared. Each training sample is a sentence annotated with an event type. The preprocessing is as followed. First, stopwords were filtered out. Then, every sentence in training documents were cut using window of size 10. Each window was overlapped three words within the sentence. If the sentence contains a trigger word, this sentence will be annotated using the annotation event. If the sentence has no trigger, this sentence will be annotated as non-event. The classification model is trained on this collections of processed sentences.

When testing, every sentence in the test files were preprocessed with the same step as the training data, before input to the classification model. The sentence will be classify to be either an event or non event. If it is classified as an event sentence, the sentence will be continued on Selection procedure.

Selection procedure, the sentence will be tokenized. Every single word in the sentence were used to compute a metric. The metric was used to select the final trigger word in the next step. The metric comprised of a similarity value, a probability to be an event $P(Event_i|W)$, a probability to be non-event $P(Nonevent|W)$. Details of three values are:

- The similarity value is the maximum semantic similarity value from word embeddings between the target word and every trigger in the training corpus of that event type. The statistical data of the trigger word that has highest semantic similarity will be used for computing the following probabilities.

- The probability to be an event of the trigger word $P(Event_i|W)$ is the conditional probability of trigger word is an event given the trigger word. It was computed from number of the trigger word that appeared in the corpus as the trigger word of this event divided by frequency of the trigger word.

- The probability to be non-event of the trigger word $P(Nonevent|W)$ is the conditional probability of trigger word is not an event given the trigger word. It was computed from number of the trigger word that appeared in the corpus as non-event divided by frequency of the trigger word.

Note that all statistical values of trigger words were collected from the training examples.

Every single word has a metric that contains three values. Only one word will be select to be the trigger word of the sentence. The word will be selected if the word has highest probability to be an event and the probability to be an event is higher than the probability to be non-event. If the word that has the highest probability to be an event is lower than the probability to be non-event, the second, the third, and so on will be checked until the answer is found.

2.2 UMBC2

This system is another classification model built from features that extracted from training data. We used Weka(Hall, et.al., 2009) to train the model. This classification model is also built to classify 19 types of events (18 subtypes plus a non-event). Event samples are collected from annotation data in the training corpus. Any words that are not appeared in the same sentence as the trigger words were collected and annotated as non-event samples. The pretrained word embeddings from Global Vector for word representation (GloVe), Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors)(Pennington, et.al., 2014) are used for any word embeddings task for UMBC2. Features consist of:

- similarity values
- part of speech tag of the target word
- frequencies of each name entity types found in the same sentence.
- dependency function of the target word

Similarity values are series of semantic similarity that are measured from word embeddings between representatives of each event type and the target/trigger word. Representatives of each event types are set of trigger words from training data. They were selected by K-mean clustering algorithm. We used K equals to 20. The K-mean clustering algorithm is used to cluster all triggers in the corpus. We used similarity value from word embeddings as the distance between each sample in each cluster. The twenty highest frequency trigger words are used as the initial centroids. Finally, we have twenty triggers as representatives per event types. Part of speech tag, name entity detection, and dependency parser were processed using Stanford CoreNLP tool(Manning, et.al., 2014).

We ran Auto-WEKA(Thornton, et.al., 2013) to find the best configurations and best training function. The best training model from our experiment is RandomForest with attribute selection using CfsSubsetEval and Best First Search algorithm.

When testing, every sentence is tokenized into words. Each word was used to compute feature vectors and put into classification model. This method we got the large number of prediction output from the classification model. To reduce the number of output,the low confidence score outputs were filtered out by using threshold 0.5. Threshold is from experiment.

2.3 Realis

Realis attribute indicates whether or not the event is occurred. It has three possible values; ACTUAL, GENERAL, OTHER. They provided some rules in the (DEFT Rich ERE Annotation Guidelines: Events v.2.5.1, 2015). To predict the Realis attribute, we trained a classifier model using Weka. The features consist of:

- Verb tense
- four name entity types surrounding the verb of the sentence
- number of habitual occurrence word in the sentence
- number of negation and conditional words i.e. whether, if in the sentence
- number of modality words in the sentence

When the nuggets are detected by UMBC1 or UMBC2, these nuggets will be use to compute Realis feature vectors and put into Realis classification

Attributes	Precision	Recall	F1
plain	46.38	27.01	34.14
mention_type	40.08	23.34	29.50
realis_status	32.35	18.84	23.82
mention+realis	27.94	16.27	20.57

Table 1: The performance of UMBC1

Attributes	Precision	Recall	F1
plain	41.70	30.51	35.24
mention_type	37.36	27.33	31.57
realis_status	28.45	20.81	24.04
mention+realis	25.64	18.76	21.67

Table 2: The performance of UMBC2

model. The model will predict one of three Realis values for each nuggets.

3 Corpus

All of training data we selected consist of ACE 2005 Multilingual Training Corpus (LDC2006T06), DEFT Rich ERE English Training Annotation V2 (LDC2015E29), DEFT Rich ERE English Training Annotation R2 V2 (LDC2015E68), and DEFT Rich ERE Chinese and English Parallel Annotation V2 (LDC2015E78 only English examples). We removed any redundant file off and got total of 1,159 files of source/annotation documents. Total number of 18 event types used as training data is 12,484 samples. Development data is DEFT Event Nugget Evaluation Training Data (LDC2014E121). Total number of 18 event types used as development data is 2,642 samples.

4 Experimental Results

We processed only English documents of the test data. English testing data consists of 169 documents. They can be categorized into 85 newswire documents and 84 discussion forum threads. We sent two runs for the Event Nugget Detection task. One run was from UMBC1 and another run from UMBC2. The performance of both systems was in the same range. The best result that we got is F1 measure of 35.24 from UMBC2. It is quite low when compare with the best system of the event nugget detection task. The details of our system performance are in Table 1 and Table 2.

The system performance of mention detection (plain) in Table 1 and Table 2 showed that UMBC1 can produced higher precision but lower recall when compared to UMBC2. The highest precision we got from UMBC1 is 46.38. The highest recall is from UMBC2 equals to 30.51. When measuring mention type, realis, and mention type+realis, the system got performance as 31.57, 24.04, and 21.67, respectively. We will discuss what reasons are affected our system to has low performance in the next section.

5 Discussion

To give more details for analyze the system performance, the statistical data of our train and development data were shown in Table 3. Table 3 showed our system performance and number of training data and development data we used to train the classification model by each event types. It can be obviously seen that there are imbalanced of train data, development data, and test (Gold) data. Only a few event types has high number of training samples. In the development phase, we tried to train the classification model with balanced training data 200 samples and 500 samples per event types. The training samples were balanced by replicating the small size data and random sampling the big size data. But they performed worse than the model that was trained with imbalance data. It may due to the sample size was too small. Finally, we chose to not balance training data. This may be a reason that we got low performance. If we can collect more training data, we have balanced big number of samples for each event types, we may get higher performance. In addition to low quality of classification model, small size of some event types in the training data also yielded to the performance of selection procedure of UMBC1. The selection procedure rely on statistical information collected from training data. We have small size of training data, it has high chance that the unseen and dissimilar sample from testing data will be encountered. In this case, the selection procedure will give the wrong output.

Another cause of poor performance is because of our development data cannot reflect the system performance in a good way. We observed that development samples of some event types are miss-

Event types	Train	Dev	Gold	S1	S2
attack	2541	538	499	0.22	0.27
demonstrate	184	88	189	0.38	0.27
broadcast	388	0	581	0.16	0.2
contact	613	0	446	0.07	0.1
correspond	216	0	96	0.07	0.03
meet	542	190	147	0.27	0.29
arrestjail	247	148	92	0.51	0.45
die	1259	394	233	0.41	0.46
injure	254	58	37	0.59	0.5
artifact	120	0	112	0.07	0.11
trnsprtartif	82	0	123	0.07	0.1
trnsprtperson	853	346	444	0.19	0.18
elect	217	25	86	0.28	0.09
endposition	447	108	186	0.31	0.36
strtposition	249	53	72	0.08	0.06
transaction	111	0	36	0	0
transfmoney	1056	247	477	0.12	0.12
transfownship	640	219	299	0.15	0.21

Table 3: Number of train/development data and Performance(F1) of UMBC1(S1), UMBC2(S2) by mention type

ing such as Contact.Broadcast, Contact.Contact, Movement.Transport-Artifact. We have no development data in these event types. That means the model is developed in the way that did not concern any performance on these event types. It yielded to these event types has very low performance on test data.

The last observation is about characteristics of the discussion forum data. It may not relevant only to the real performance of our system, but it effects the official result of other systems as well. The discussion forum data always has the redundancy phrases or sentences in the same document. If the system detected one of these redundant nuggets, the rest of the redundant nuggets always were annotated with the same event types. If the one was found, the rest will be found. This can exaggerate the performance in both way, much better performance if the nugget was detected or much worse performance if the nugget was neglected. So in our opinion, the discussion forum data is not the good example to measure performance of the event nugget detection system.

6 Conclusion

We developed two systems; UMBC1 and UMBC2, to participate in event nugget detection task of TAC2016. We considered the event nugget detection task as a classification problem. UMBC1 developed from Convolution Neural Network. UMBC2 extracted features and trained the classification model on Weka. Both systems got low performances. A lot of flaws were found such as imbalance training data, incomplete samples of development data, small size of training data. These observations will be used to improve our system in the future.

References

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. *Nyus english ace 2005 system description*. In Proceedings of ACE 2005 Evaluation Workshop.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten. 2009. *The WEKA Data Mining Software: An Update*. SIGKDD Explorations, Volume 11, Issue 1.

Heng Ji, and Ralph Grishman. 2008. *Refining Event Extraction through Cross-Document Inference*. In ACL, pp. 254-262.

Yoon Kim. 2014. *Convolutional Neural Networks for Sentence Classification*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 17461751, Doha, Qatar. Association for Computational Linguistics.

Qi Li, Heng Ji, and Liang Huang. 2013. *Joint Event Extraction via Structured Prediction with Global Features*. ACL

Zhengzhong Liu, Jun Araki, Dheeru Dua, Teruko Mitamura, Eduard Hovy. 2015. *CMU-LTI at KBP 2015 Event Track*. Eighth Text Analysis Conference.

Christopher D. Manning, Mihai Surdeanu, , John Bauer, Jenny Finkel, Steven J. 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, pp. 55-60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. *Distributed Representations of Words and Phrases and their Compositionality*. In Proceedings of NIPS.

Teruko Mitamura, and Eduard Hovy. 2016. *2016 TAC KBP Event Detection and Coreference Tasks, v2.1*. Technical report, Carnegie Mellon University.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. *GloVe: Global Vectors for Word Representation*.

Nils Reimersy and Iryna Gurevychyz. 2015. *Event Nugget Detection, Classification and Coreference Resolution using Deep Neural Networks and Grandient Boosted Decision Trees*. Eighth Text Analysis Conference.

Mark Sammons, Yangqiu Song, Ruichen Wang, Gourab Kundu, Chen-Tse Tsai, Shyam Upadhyay, Siddarth Ancha, Stephen Mayhew, and Dan Roth. 2014. *Overview of UI-CCG Systems for Event Argument Extraction, Entity Discovery and Linking, and Slot Filler Validation*. Urbana 51 (2014): 61801.

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. *Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms*. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 847-855.

DEFT Rich ERE Annotation Guidelines: Events v.2.5.1, Linguistic Data Consortium. February 24, 2015