Event Nugget Detection using Thresholding and Classification

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

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

Abstract

This paper described our Event Nugget Detection system that we submitted to the TAC KBP 2015. We considered the problem as document classification problem. We used confidence scores from classifier to detect the event by thresholding. Our feature vectors consist of event nugget context, part of speech tags of event nugget context and semantic similarity score between event nugget and event subtypes. Our performance is quite low; we got F1 measure of 0.33 for both event nugget detection task and coreference task.

1 Introduction

Text Analysis Conference Knowledge Base Population (TAC KBP) 2015 is an evaluation workshops in Natural Language Processing. They provided collection of raw data such as newswires, forums, including their annotation that suitable for using as training corpus. Besides, they have evaluation procedure to evaluate all of the participant system output. The Event track is one of three tracks that are provided in this year. In the Event track, we participated in two subtasks; Event Nugget Detection, and Event Nugget Detection and Coreference (Mitamura, 2014; Liu et.al., 2015). Event Nugget Detection goal is 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 Rich ERE Annotation guidelines. In addition, systems must identify Realis attribute (ACTUAL, GENERIC, OTHER), which are also described in the Rich ERE guidelines. For Event Nugget Detection and Coreference task, in addition to the system has to identify and classify the event mentions, it has to identify event coreference links at the same time.

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.

For our system, we considered the event nugget detection as the classification problem. Our idea is any words or phrases are considered as event have some similar characteristics. So we can categorized them into event and non-event. We will described our system description about features, classification model parameters in section 2. Event Detection and Classification are 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, we have to detect the event mentioned in the document and classify the event to its type. For this task, annotation guidelines (Mitamura, 2014) classify events into eight event types: Life, Movement, Business, Conflict, Contact, Personnel, Transaction, and Justice events. These eight types have a total of 33 subtypes. We partitioned the event nugget detection problem into two parts; event detection and event classification to its subtype. For the event detection part, we used simple threshold and brute force algorithm to find the event in the document. We will explain details in the section 3. For the classification part, we built feature vectors to train the classification model using Stanford Classifier tool.

Another task that we participate is the Event Nugget Detection and Coreference task. This task we have to link all mentions that are relevant to the same event altogether. The Event Nugget Detection and Coreference task, we find the coreference chain by using the Stanford CoreNLP tool, and then use matching rules to produce the coreference link of all mentions in an event. Details of our features and other tools that were used are described in the following.

2.1 Training data

When we detected event mentions in any document, they have to be classified to an event subtypes. The total event subtypes is 33 subtypes. Training data have to be prepared for using Stanford Classifier. After we formed the feature vector, we will use Stanford Classifier to build a classification model. Each training data instance consists of:

- event subtype name as the classname of data
- a set of context words
- part of speech tags of context words
- event nugget (mention)
- semantic similarity score between mention and event subtypes

We extracted the context of the event nugget from the source document. Our context is set of words around the event nugget that have the same part of speech as the event nugget. We used OpenNLP (Baldridge, 2005) to find part of speech of any words. These training data will be regenerate as useful natural language features later by Stanford Classifier. The semantic similarity score is measured by the UMBC semantic similarity system (Han, 2013). Each line in the training file is a features vector of an event nugget.

Training corpus are combination of the LDC2015E69 and LDC2014E121. The LDC2015E69 (DEFT 2014 Event Nugget Evaluation Annotation Data) consists of 200 files. The LDC2014E121 (DEFT 2014 Event Nugget Evaluation Training Data) contains 151 files. These two corpus come with source document and their annotations data. The total is 351 files. We divided them into training corpus (80%) and development data (20%). We have total of 8571 set of training instances.

2.1.1 Semantic Similarity

We used the UMBC semantic similarity system to compare similarity between event nugget and its event type. The UMBC semantic similarity (Han, 2013) is based on LSA word similarity model. LSA relies on the fact that semantically similar words are more likely to occur near one another in text. Thus evidence for word similarity can be computed from a statistical analysis of a large text corpus. They built the raw word co-occurrence statistics from a portion of a 2007 Stanford WebBase dataset (Stanford, 2001). The UMBC semantic similarity system has been developed a few versions. The version that we used is online as the web service at (http://swoogle.umbc.edu/StsService/GetStsSim). This version showed its high performance in the Semantic Evaluation (SemEval) 2013 and 2014 semantic similarity task (Han, 2013; Abhay, et. al., 2014).

2.2 Classification Model

We used Stanford Classifier to build the classification model. Stanford Classifier (Manning and Klein., 2003) is a probabilistic classifier. It can give a probability distribution over the predicted class for an input. It is a maximum entropy classifier. Maximum entropy models are comparable to multiclass logistic regression models. The advantage of using this tool is they can generate simple natural language features for text classification such as n-gram, word shape, etc. To use the tool we have to prepare the training file, prop file, and test file. Training file is example of data and their classes. Test file contains test data that need to be classified. Prop file is used to specify the features that the tool will generate for each training data and classifiers parameters.

We did some experiment to find the best features that can represent the characteristics of each event subtypes. Our final feature vectors were produced follows the configuration in the prop file. Each feature preprocessing is defined as follows:

1. **Context data** It is a set of words around the event nugget. We defined the preprocessor to do character n-grams, and word n-gram with possible n-gram length from 1 to 4. It means that the tool will generate unigram, bigram, trigram, and quadgram for the context data.

2. **Part of speech tags** They are part of speech tags of context data. This feature we did word n-gram with possible n-gram length from 1 to 4. The tool will produce unigram to quadgram to this data as well.

3. **Event nugget** It is the word that we brought from annotation file. We defined that it will be use as a whole string. Besides that the prefix and suffix will be generated, character n-gram with length from 4 to 9, and we check the word shape with *'dan1'*.

4. **Semantic similarity score** Because of the score is the numeric data from 0.0 to 1.0. We have to define this feature as real value.

Note that these parameter values are from experiment. In addition to the feature vector parameters, we ran grid search to find the optimal parameters for the classifier model. In addition, because in the testing step we cannot know the event subtype and event nugget before classification (they are what we are predicting), we cannot find the semantic similarity of event nugget and event subtype. So we built two classifiers, they are almost the same except that one classifier the semantic similarity score can be used and another classifier was not used semantic similarity score.

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 task annotation guidelines [].The feature vector of the event was built from these rules. To predict the Realis attribute, we trained a classifier model. The features consist of:

- Verb tense
- Has name entity
- Contain regular occurring words i.e. every, often, always.
- Contain conditional words i.e. whether, if.
- Negate event

2.4 Coreference Resolution

We find the coreference chain by using Stanford CoreNLP (Manning, et.al., 2014) . The Stanford CoreNLP is the collection of Natural Language Processing such as part of speech tagger, name entity recognizer, etc. Coreference resolution is another function that is provided in the package. We got the coreference set for each document from Stanford CoreNLP. Then we match the set with the detected event mention. The event mentions will be linked together when the mention has the same event type with other mentions in the same coreference chain.

3 Event Detection and Classification

As we mentioned above that to accomplish the event nugget detection task we need the event nugget detection step and event nugget classification step. For the event detection part, we used a simple brute force method and experimented event subtypes' thresholds. Detection and classification procedure is as followed.

1. **Chunking** The whole document was split into chunks follows part of speech tag. Only chunk of noun and chunk of verb are kept. For example, the sentence is *'He has been married 3 times'*, part of speech tag of this sentence is *B-NP B-VP I-VP I-VP B-NP I-NP*, This sentence is generated three chunks as *'He', 'has*

been married', '3 times'. The original sentence will be used as Context data. Chunks will be use as Event nugget. And their part of speech tags will be used as Part of speech tag data. Details in Section 2.2

2. **Pre-classify** Every chunk was pre-classified using our non-semantic similarity classifier model. The classifier model produced the confidence score for each chunk.

3. **Thresholding** The confidence score for each chunk was compared with its threshold of the predicted event subtypes. Each event subtype has a threshold that we got from experiment in the training step. The chunk that has higher confidence score than its threshold of the predicted event subtype will be kept as our event nugget. We got a collection of event nuggets and its predicted event subtype.

4. **Semantic similarity measuring** The event nugget and its predicted event subtypes were measured the semantic similarity using UMBC STS system.

5. **Classifying** The testing data in number 1 plus semantic similarity score from number 3 will be classified again with the classifier model that was trained with semantic similarity data. Finally, the predicted event type from the classifier model is our detected event nugget and its event subtype.

4 Experimental Results

Testing data consists of 202 documents. They consists of 98 files of newswire data and 104 files of multi-post discussion forum data. We sent two runs for the Event Nugget Detection task and the Event Nugget Coreference task. Two runs are different in the threshold for selecting the chunk as the event nuggets. The best result that we got is F1 measure of 0.33. It is quite low when compare with the best system of the event nugget task. The details of our best system performance are in Table 1.

The system performance of mention detection (plain) in Table 1 showed that our system can detect correctly 40.76 percent of retrieved events. Also our system can retrieve event only 28.88 percent of total event that should be detected. About the mention type performance of our system is quite low, our system can classify the event correctly about 32.46 percent of overall predicted event. And our system can classify the event correctly about 23.00 percent of all of event. For the event nugget coreference task, other performances are the same with additional of average CONLL score is 17.80. We will discuss what reasons are affected our system to has low performance in the next section.

Attributes	Precision	Recall	F1
plain	40.76	28.88	33.81
mention_type	32.46	23.00	26.93
realis_status	21.81	15.45	18.09
mention+realis	16.74	11.86	13.89

Table 1: The system performance

5 Discussion

After we looked through the details of each document result, we observed some remarkable points. We will analyze the system performance in two parts; mention detection and mention classification. About mention (event) detection, our system always produced high precision but low recall. For example, the highest precision that we have is 91.67 percent, but the recall is 33.33 percent. The system always got the low recall because we defined too high threshold. Most of detected mentions are correct but we left a lot of event out. If we lower the threshold it may produce higher recall lower precision, the F1 measure can be higher. Another problem is the mention span that affected the performance. Because we form the mention span from chunk of part of speech tag, the mention will not exactly match to the gold standard. In summary, the brute force and threshold method is simple but we need more experiment to select the better threshold.

For mention classification, both the precision and recall are the same range. It means that our features cannot reflect the characteristics of event subtypes. In this part, we can consider the dependency of happening of event and used as heuristic rules after classify the event. Some event mentions are less likely to be in the same sentence such as *Nominate* and

Charge. Some events often appeared together, for example, *Attack* and *Injure*. We can infer it from the statistical of training data. It will improve the accuracy of the event classification.

About the Realis attribute, it has to include other information of event. Only the *'yes'* or *'no'* features that we have cannot discover the right answer. We may use statistical about event types such as it is the definitely happened type, or it is inconsistency type, to predict the Realis attribute.

6 Conclusion

Our Event Nugget Detection system is simple but low performance, due to its low efficiency in identifying event mention, and event mention span. This error has been propagate to the event classification. To improve the performance, we plan to use characteristics of events, applying commonsense knowledge in the future.

References

Jason Baldridge. 2005. *The opennlp project*. URL: http://opennlp.apache.org/index.html

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

Lushan Han, Abhay L. Kashyap, Tim Finin, James Mayfield, and Johnathan Weese. 2013. *UMBC EBIQUITY-CORE: Semantic Textual Similarity Systems*,. In Second Joint Conf. on Lexical and Computational Semantics. Association for Computational Linguistics.

Tan Li Im, Phang Wai San, Chin Kim On, Rayner Alfred, and Philip Anthony. 2013. *Analysing market sentiment in financial news using lexical approach*. Open Systems (ICOS), 2013 IEEE Conference on, pp. 145-149. IEEE.

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

Abhay L. Kashyap, Lushan Han, Roberto Yus, Jennifer Sleeman, Taneeya Satyapanich, Sunil Gandhi, and Tim Finin, 2014. *Meerkat mafia: Multilingual and cross-level semantic textual similarity systems*,. Proceedings of the 8th International Workshop on Semantic Evaluation.

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

Zhengzhong Liu, Teruko Mitamura, and Eduard Hovy. 2015. *Evaluation Algorithms for Event Nugget Detection: A Pilot Study*. In Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT, pp. 53-57.

Yue Ma and Alifah Syamsiyah. 2014. *A hybrid approach to learn description logic based biomedical ontology from texts*. ISWC.

Christopher D. Manning and Dan Klein. 2003. *Optimization, Maxent Models, and Conditional Estimation without Magic*. Tutorial at HLT-NAACL 2003 and ACL 2003.

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.

Teruko Mitamura. 2014. *TAC KBP event detection annotation guidelines, v1.7*. Technical report, Carnegie Mellon University, September.

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.

Richard Wicentowski, and Matthew R. Sydes. 2012. *Emotion Detection in Suicide Notes Using Maximum Entropy Classification*. Biomedical Informatics Insights 5.Suppl 1 (2012): 5160. PMC.