**Event Nugget Detection and Coreference Scoring**

**July 28th, 2016**

**Language Technologies Institute**

**Carnegie Mellon University**

# Overall workflow

This document describes the event nugget and coreference evaluation.

We first give an overview of the whole scoring process. For detailed format description please refer to the corresponding sections.

For each text file, LDC annotators provide human annotations as gold standard files. System submissions will be evaluated against certain gold standard based on the task it participates. To work with 3 different languages, we change our evaluation methods to character based scoring instead. In our experiments on systems submitted in 2015, the averaged score difference between character based evaluation and token-based evaluation is less than 0.1%. As a result, we will not provide tokenization table to the participants for the test data.

Each participant event nugget detection system will be given the source text as input.

Let us refer to the output of the system as a system event nugget file. We require a system event nugget file to be given in the same file format as the gold standard file. The evaluator (scorer) takes the gold standard file, the system event nugget file, and the token mapping file as input, and compares them to give a score for the system. The scoring is currently hosted at: https://github.com/hunterhector/EvmEval

# Evaluation Process and Formats

The submission format described below will be used for both Event nugget Detection task and Event Coreference task.

**Input of Scorer:**

1. Gold standard annotation for documents, in Tab Based Format (tbf).

2. System output annotation for documents submitted by participants, in Tab Based Format (tbf)

**Output of Scorer:**

1. Overall performance report for system, as described in “Scoring” section.
2. Comparison of the system and gold standard in text form if the corresponding arguments are specified. (This is not the core functionality for scoring, please refer to the README file of the scorer for details)

**Evaluation Script Options:**

Please follow the README and help information in the script distribution.

## Possible Formats You Might See

Since there are changes in annotation scheme and annotation tool over the years, participants may come across different formats of data. Here we roughly give an overview of these formats.

### Brat Annotation Format

Brat standoff annotation formats[[1]](#footnote-1) are ended with extension “.ann”. Some human annotators may use these formats to create annotated data. Note that Brat formats provide nugget offsets in character base.

### LDC’s rich ERE XML annotation

Annotators from LDC normally create annotation in rich ERE format. The details of these formats can be found within the data set.

### The submission format (TBF)

There are 2 possible tasks for evaluation this year, the submission format of these tasks are the same. The content of the submission may vary:

1. Event Nugget Evaluation: Only contain the event nugget annotations by the systems
2. Event Nugget Detection and Coreference: Contain both event nugget annotations and coreference by the systems

The TBF format is almost identical to previous years. The only change is that we replace token ids with character spans. The following sections describe the format in details.

### System and gold standard annotation file format:

1. All event nugget annotations (and/or coreference annotation) for all documents in the corpus are written into one single file
2. A header will indicate the start of a new document (<s> is the space character)
   1. Header := #BeginOfDocument<s><doc ID>
3. A footer will indicate the end of a document
   1. Footer := #EndOfDocument

### Definition of event nugget format:

Event nuggets are represented as tab-separated lines. To be specific, for each mention line, we have the following columns:

* <system ID> := the name of the system in order to distinguish participants
* <doc ID> := the ID of the input document
* <mention ID> := the ID of the mention, which should uniquely identify the mention within the current document
* <character span> := begin and end character offset of the current nugget, separated by commas (,) . “begin” is the character offset of the first character of the nugget. “end” is “begin” + “nugget length”
* <mention> := the actual character string of the nugget
* <event-type> := the ACE hierarchy type
* <realis status> := the REALIS label

It is optional to append the following confidence columns (which can be helpful for follow up study and performance diagnostics):

* <span confidence> := a score (confidence, etc.) the system wants to assign for the mention span detection. This score will not affect evaluation results
* <type confidence> := a score (confidence, etc.) the system wants to assign for the mention type detection. This score will not affect evaluation results
* <realis confidence>:= a score (confidence, etc.) the system wants to assign for the mention realis detection. This score will not affect evaluation results

### Definition of event coreference format:

All coreference clusters annotations are **appended after all event nuggets lines of this file, before the #EndOfDocument footer**. System submissions should make sure transitive closure of coreference are resolved so that each coreference line indicate the whole cluster.

Each coreference cluster should also be represented as a tab-separated line, with the following columns:

* <relation name> := The relation name with a special indicator character (@) as prefix , **in this task, it is always “@Coreference”**
* <relation id> := A relation id. The relation id used in the gold standard files will be in form of “R<id>” (e.g. R3). This is for bookkeeping purposes, which will not be read by the scorer. However, system should **always** give a non-empty string without whitespace here.
* <event mentions> := A list of event mention ids in this coreference cluster, separated by comma (,). In terms of coreference, the ordering of event mentions does not matter. Each event mention id should have already presented in the list of nugget definition above.

**Example:**

#BeginOfDocument sample

system1 sample E2 t1069 married Life\_Marry Actual 1 1 1

system1 sample E4 t1096 divorced Life\_Divorce Actual 1 1 1

system1 sample E5 t1109 married Life\_Marry Actual 1 1 1

system1 sample E6 t1157 married Life\_Marry Actual 1 1 1

@Coreference R1 E6,E2

#EndOfDocument

# Selected Event Types and Subtypes

For purposes of this evaluation, an event must fall under one of the event types and subtypes. 18 event types and subtypes are selected according to the **Rich ERE Annotation Guidelines: Events** (listed in the following table). When during submission, please note that system submissions should only contain the following types.

|  |  |  |
| --- | --- | --- |
| Type | Subtype | 2016 eval |
| conflict | attack | yes |
| conflict | demonstrate | yes |
| contact | broadcast | yes |
| contact | contact | yes |
| contact | correspondence | yes |
| contact | meet | yes |
| justice | arrestJail | yes |
| life | die | yes |
| life | injure | yes |
| manufacture | artifact | yes |
| movement | transportartifact | yes |
| movement | transportperson | yes |
| personnel | elect | yes |
| personnel | endPosition | yes |
| personnel | startPosition | yes |
| transaction | transaction | yes |
| transaction | transferMoney | yes |
| transaction | transferOwnership | yes |

# Event Nugget Detection Scoring

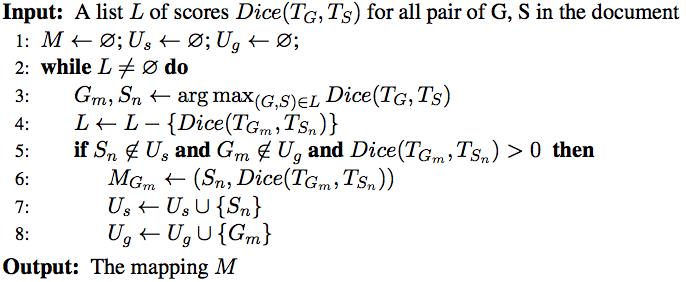
In this section we describe the algorithm for scoring event nugget detection.

## Scoring for one document

We denote a gold standard mention with **G**, and a system mention with **S**. We use **TS**to represent the span of the mention. **Dice(TG,TS)** is the dice coefficient function that returns a score between 0 and 1. **Dice(TG,TS)** is calculated as , where represent the set of characters that overlaps between the mentions.

### Create mappings

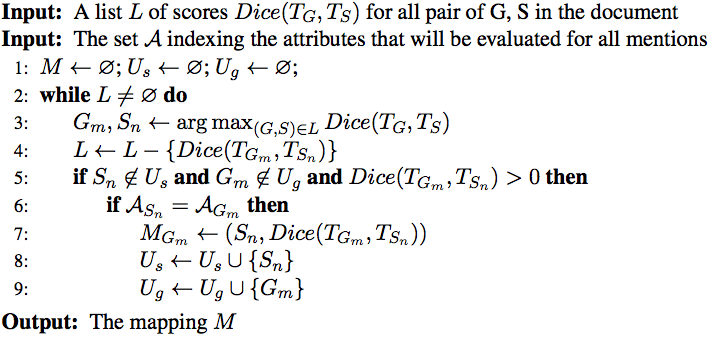
To perform scoring for a document, system mentions are mapped to gold standard mentions based on the Overlap score (computed as token level Dice Coefficient). We use a greedy algorithm for comparing such the score:



Algorithm 1 Span only Greedy Mapping

The algorithm above iteratively pops the highest Dice score in all the remaining mention pairs. Line 5 ensures that one system mention can be only mapped to one gold mention and vice versa.

Similarly, we create a mapping for attribute-based evaluations. We create all attribute combinations and create the mapping as followed:



Algorithm 2 Attribute Augmented Greedy Mapping

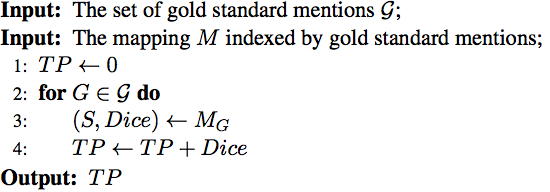
The attribute-based mapping algorithm add an additional check in line 6 to greedily search for the current best Gold-System mapping that also matches their attributes.

### Scoring mention detection

To score mention detection, a mention-based F1 score is computed in the following way:

1. For each gold standard mention , recall that is greedily mapped to a system mention , we denote .
2. True Positive =
3. Precision = True Positive / #System Mention
4. Recall = True Positive / #Gold Mention
5. F1 = H (Precision, Recall), where H is the harmonic average function

Given a mapping from Algorithm 1, we can simply get the true positive by aggregate the matching score of each gold standard mention:



We can then simply compute F1 score with the following:

We will then iterate through all possible mappings, taking the highest score as the score for the system.

### Scoring realis status and mention type detection

To score realis status and mention type detection, we augment the Span-based F-1 score slightly. The only difference is that we will use the mapping computed using the attributes under evaluation. For example, to evaluate “Realis” and “Type” jointly, we compute the mapping using Algorithm 2, where Input Set **A** will be the set that indexes both “Realis” and “Type” for all mentions.

We can then calculate the attribute augmented TP and F1 the same as above. Our evaluation script will produce scores for all possible attribute combination. In current case, these combinations will be Realis only, Type only and Realis &Type.

## Summarization score

After all documents are scored, we also report scores that give a summary of performance over the whole corpus by taking the average across documents. We use the standard Micro and Macro average definition. Given that the number of mentions in a document is usually small, Macro average scores are subject to high variance. We will use the Micro Average scores for final ranking of systems.

### Macro Average Scores (numerical average over the document scores):

Precision\_macro = sum of all Precision / #document

Recall\_macro = sum of all Recall / #document

F1\_macro = 2\* Precision\_macro \* Recall\_macro / (Precision\_macro + Recall\_macro)

### Micro Average Scores (sum of the individual true positives, false positives, and false negatives of each mention to calculate the overall F-Score)

Precision\_micro = (sum of TP on all docs)/ (total number of system mention in all docs)

Recall\_micro = (sum of TP on all docs) / (total number of gold standard mention in all docs)

F1\_micro = 2\* Precision\_ micro \* Recall\_ micro / (Precision\_ micro + Recall\_ micro)

# Event Coreference Scoring

Our evaluation script will simply call the Reference Implementation of Coreference Scoring algorithms used in CoNLL shared tasks. The script will convert both gold standard and system results to the required format of the CoNLL scorer. We will use the latest version of the scorer[[2]](#footnote-2). At the time of this writing, the scorer version is v8.01.

## Mapping to CoNLL conversion

The CoNLL conversion (built in the scorer) will read the mapping between gold and system to produce coreference scores. We take the mention type augmented mapping, which ensures each mapped pair share the same mention type.

We then create a CoNLL mapping from the mention type mapping. We only create a mapping in CoNLL format when the Dice score for the pair of gold and system mention reach a predefined threshold. Following the convention in previous task, we set this threshold to be 1, which means that the mapping can only be valid for exact match cases. However, the scorer can be configured to test the impact of different thresholds.

## Final Score

Systems will be ranked using the unweighted average of the following 4 metrics produced by CoNLL scorer:

1. MUC
2. BCUBED
3. CEAFE (entity based CEAF)
4. BLANC

Note that following the CoNLL evaluation convention, we choose to use CEAFE, and do not include CEAFM (mention based CEAF) in our final scores.

## Problem of Double Annotation

In the current annotation guideline, it is allowed to have double annotations (e.g. the exact span is marked with multiple different event mention triggers). The CoNLL scorer cannot handle such case properly because it distinguishes mentions solely based on the mention span.

Let’s consider the following example:

#BeginOfDocument sample

gold sample E1 2,8 murder Life\_Die Actual

gold sample E2 2,8 murder Conflict\_Attack Actual

gold sample E3 12,16 kill Conflict\_Attack Actual

@Coreference R1 E2,E3

#EndOfDocument

#BeginOfDocument sample

sys sample E1 2,8 murder Conflict\_Attack Actual

sys sample E2 12,16 kill Conflict\_Attack Actual

@Coreference R1 E1,E2

#EndOfDocument

The gold standard is annotated with 3 event mentions, E1 and E2 shares the same span (t1). The system response only detects 2 mentions, missing one of the double annotated “murder”.

When scoring for coreference, the CoNLL scorer will first align the mentions from gold and system with the exact span. System E1 can be aligned to gold E1 or E2, and the resulting coreference score will be different. This is an inherited ambiguity in span representation.

To find the best possible CoNLL score, one need to enumerate all possible alignments and score them all. We find this computational infeasible because exhaustive enumeration is exponential to the number of such cases (a document might contain 50 such cases). Hence we currently adopt a greedy method to approximate it.

## Coreference Cluster Validation

We introduced several pre-evaluation checks. These checking are built in the validator and scorer provided. A system response has to pass all the checks in order to get a score; otherwise, the scorer will refuse to do the scoring.

### With-in clustering duplication check

This checks that no mentions in the same coreference chain can have the same span. Although two mentions can share the same span, they cannot refer to the same event, otherwise they should be the same mention.

### Cluster transitive closure check

The transitive closure of coreference should be solved before submission: if multiple clusters can be merged as a bigger cluster, the submitted response should have written them all in one line. For example:

@Coreference R1 E1,E2

@Coreference R1 E2,E3

is not allowed and should be submitted as:

@Coreference R1 E1,E2,E3

1. http://brat.nlplab.org/standoff.html [↑](#footnote-ref-1)
2. http://conll.github.io/reference-coreference-scorers/ [↑](#footnote-ref-2)