**Event Mention Detection Scoring**

**August 25, 2014**

**Language Technology Institute**

**Carnegie Mellon University**

# Overall workflow

We show an overall workflow of evaluation for event mention detection in Figure 1.

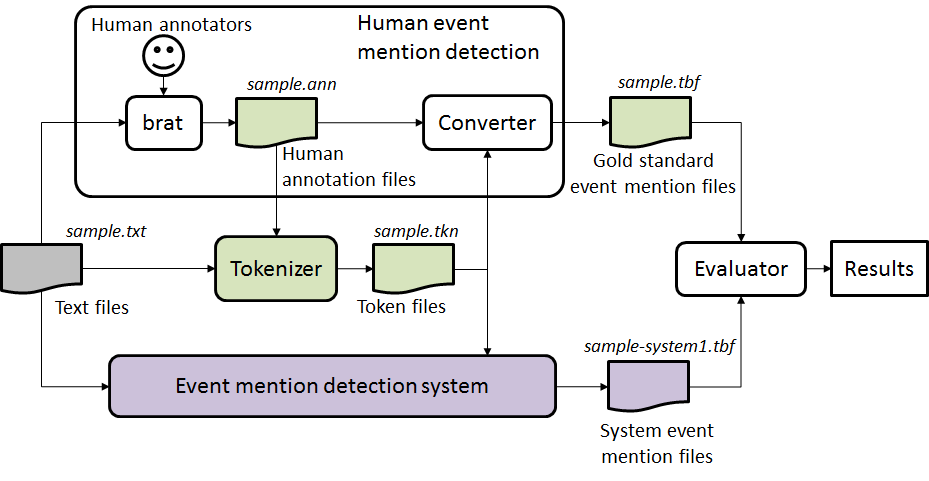


Figure 1: An overall workflow of evaluation for event mention detection.

For each text file, human annotators use the **brat** rapid annotation tool[[1]](#footnote-1) to create a gold standard annotation file. We convert the brat annotation file to our evaluation file format. Let us refer to the output of an event mention detection system as a system event mention file. We require a system event mention file to be given in the same file format as the gold standard file. The evaluator (scorer) takes the gold standard file and a system event mention file as input, and compares them to give a score for the system.

# Pre-tokenization

As shown in Figure 1, we carry out tokenization in our evaluation mechanism. We call it **pre-tokenization** since it is done before evaluation. The reason for pre-tokenization is two-fold.

First, modern language technologies on English are normally based on tokens. Mostly, a token is the smallest unit to be used by a complex natural language processing system. Specifically in the case of event mention detection, if we do not incorporate tokens in evaluation at all, then a system’s gaining or missing one correct token ends up with different scores according to the length of the token (i.e., the number of characters in the token). This is unfair from the perspective of evaluation. Therefore, it is necessary to evaluate the performance of event mention detection in terms of tokens.

Second, a modern English tokenizer is basically reliable enough to help human annotators correct (or rethink at least) their event mention annotation, particularly in the relatively unstable process of their creating the gold standard together. This is not only about trivial annotation errors (e.g., misselection of a span with extra whitespace in their web browser) but also about linguistically questionable event mentions. We will give an example to show the second point later.

The input and output of our tokenizer are defined as follows.

**Input of the tokenizer**:

1. Original text files
2. Gold standard annotation files in the brat standoff format[[2]](#footnote-2).

**Output of the tokenizer**: Token files (.tkn)

The token file is a tab-separated file with each line containing the following fields:

<token ID><TAB><token string><TAB><begin offset><TAB><end offset>. The first character of the document has an offset of 0. The <begin offset> is inclusive and the end offset can be calculated by <begin offset> + length of span (which make is exclusive).

Below is an example of the token file. Special characters such as “tab” or “new line” are replaced into space for easy manipulation.

48 It 386 388

49 has 389 392

50 been 393 397

51 suggested 398 407

52 that 408 412

Our tokenizer implementation is based on the tokenizer in the Stanford CoreNLP tool[[3]](#footnote-3). It is implemented in Java, and the jar file is evm-eval-1.0.2-jar-with-dependencies.jar . The requirements of the software are as follows:

1. Java 1.7
2. The same number of text files and brat annotation files (\*.ann) with the same file base name

You can run the tokenizer by the following command.

$ java -cp ./evm-eval-1.0.2-jar-with-dependencies.jar evmeval.TokenFileMaker

If you run the tokenizer without any options, you should see the usage of the software as follows.

usage: java evmeval.TokenFileMaker -a <annotation> -e <extension> [-h]

-o <output> [-s <separator>] -t <text>

-a <annotation> annotation directory

-e <extension> text file extension

-h print this message

-o <output> output directory

-s <separator> separator chars for tokenization

-t <text> text directory

As seen in the usage, the tokenizer takes a text file directory path and an annotation file directory path as input, instead of individual text files and annotation files. The tokenizer outputs the same number of output files as that of input files in the text (annotation) file directory. You can run the tokenizer by specifying options as follows, for example.

$ java -cp ./evm-eval-1.0.2-jar-with-dependencies.jar evmeval.TokenFileMaker -t ./data/LDC2014R19/ERE/ -a ./data/LDC2014R19/ERE/ -e txt -o ./output/ERE

If the software runs successfully, you might get the following log message.

[WARN] Boundary mismatch found in d21dc2cb6e6435da7f9d9b0e5759e214: Token [1716,1735] [buffet/music/buying] vs. EventMentionSpan [1729,1735] [buying]

[WARN] Boundary mismatch found in d21dc2cb6e6435da7f9d9b0e5759e214: Token [1747,1754] [in/hire] vs. EventMentionSpan [1750,1754] [hire]

The log message shows two boundary mismatches between a system token and an event mention span in the document d21dc2cb6e6435da7f9d9b0e5759e214. The first one says that the tokenizer has found a token “buffet/music/buying” whereas a human annotator annotated “buying” as an event mention. This example indicates that human annotators can take advantage of the information to go over linguistically questionable event mentions as well as trivial annotation errors.

Note that an event mention span is not necessarily the whole of an event mention; the brat annotation tool can deal with discontinuous annotation, and thus an event mention span corresponds to either an event mention itself or a part of an event mention annotated discontinuously.

The final note on the tokenization is a command option to set the additional separator characters for tokenization. The idea of the additional separator characters is to let users control a more fine-grained level of tokenization beyond the Stanford tokenization. Our tokenizer is exactly the same as the Stanford one by default. We observe that sometimes we might want to split the Stanford tokens further (e.g., “buffet/music/buying” into “buffet”, “music” and “buying”). To make evaluation more flexible, we provide an additional command option -s, which defines a set of additional separators for splitting tokens on top of the Stanford tokenizer. For instance, if you use the option in the command above, you will get the following result:

$ java -cp ./evm-eval-1.0.2-jar-with-dependencies.jar evmeval.TokenFileMaker -t ./data/LDC2014R19/ERE/ -a ./data/LDC2014R19/ERE/ -e txt -o ./output/ERE –s /

[INFO] Successfully completed.

This means that no boundary mismatch is found, since the additional separator character / enables the boundaries of all tokens to be aligned with those of all event mention spans. You can set multiple characters as a string with option -s, if necessary.

# Evaluation

**Input of Scorer:**

1. Gold standard annotation for documents, in format (one line per mention), all annotations are contained in one file only.

2. System output annotation for documents submitted by participants, in format (one line per mention), all annotations are contained in one file only.

3. Tokenization files associated with each document, one file per document.

**Output of Scorer:**

1. System output annotation as item 2 in Input, with addition of a mention detection score, realis status detection and mention type detection score for each mention appended to each line.

2. Overall performance report for system, as described in “Scoring” section.

## Formats

### System and gold standard annotation file format:

1. All event mention annotations for all documents in the corpus are written into one single file
2. A header will indicate the start of a new document
   1. Header := #BeginOfDocument<s><doc ID>
3. A footer will indicate the end of a document
   1. Footer := #EndOfDocument
4. Different event mentions should not include the same token

The event mentions file is a tab-separated file. To be specific, for each mention line, we follow the following format.

### Definition of event mention format (one per line):

event-mention := <system ID><TAB><doc ID><TAB><mention ID><TAB><token ID list><TAB><mention><TAB><event-type><TAB><realis status><TAB><score1><TAB> <score2><TAB><score3>

**Explanation:**

<system ID> := the name of the system

<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

<token ID list> := list of IDs for the token(s) of the current mention, in ascending order, separated by commas (,)

<mention> := the actual character string of the mention

<event-type> := the ACE hierarchy type

<realis status> := the REALIS label

<score1> := any score (confidence, etc.) the system wants to assign (ignored)

<score2> := score assigned in the evaluation

<score3> := additional possible score assigned by human

<TAB> := tab character

**Example:**

system1 0f03cc5a508d630c6c8c8c61396e31a9 E6 t96 talking Contact\_Meet Other 1

# Scoring

## Scoring for one document

We denote a gold standard mention with G, and a system mention with S. Overlap(G,S) is a token-based F1-score function of G and S that returns a score between 0 and 1 (see the OVERLAP subroutine in the Pseudo-code (Appendix 1) for details). All invisible words are already removed from G and S (see **Note 1**).

### Create a mapping

To perform scoring for a document, system mentions are mapped to gold standard mentions based on the Overlap score. A system mention is always mapped to one and only one gold standard mention with which the system mention will have the highest overlap score. However, one gold standard mention can be mapped to multiple system mentions.

### 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 can be mapped to multiple system mentions, we only choose one system mention that maximize , and denote . Let be a set that contains all such system mentions across the whole document.
2. True Positive =
3. False Positive = #System Mention - | |
4. Precision = True Positive / (True Positive + False Positive)
5. Recall = True Positive / #Gold Mention
6. F1 = H (Precision, Recall), where H is the harmonic average function

### Scoring realis status and mention type detection

To score realis status and mention type detection, we use the same mapping:

1. For each gold standard mention , we count the number of system mentions that are mapped to it as ,
2. We use \_realis and \_mention to define the realis status and mention type of mention respectively.
3. Initialize with realis\_score = 0; mention\_score = 0
4. If \_realis = realis , realis\_score = realis\_score + 1/;
5. Similarly, if \_mention = mention , mention\_score = mention\_score + 1/;
6. realis\_detection\_accuracy = realis\_score / #GoldStandardMentions
7. type\_detection\_accuracy = mention\_score / #GoldStandardMentions

**Note 1**: Invisible words are ignored in scoring. They include: determiners {the, a, an}, pronouns {I, you, he, she, we, my, your, her, our}, relative pronouns {who, what, where, when}.

Note that “it” and “that” and pronouns including {his, ours, mine, yours, ours, they} are not included in the invisibles list because they can occasionally be resolved as nominal event mentions.

Examples:

Rule 1: do not accept prepositions but include particles

* [look] up a chimney *vs.* [look up] a dictionary
* [climb] up the ladder
* [take responsibility for]
* sing [all the way] to school
* [go] to school

Rule 2: consider the maximum extent of an event mention, but don't worry about determiners (they are invisible)

* [takes a shower] ==> it is okay for annotators to include "a" in their annotation; we ignore "a" for evaluation
* [make a quick decision] ==> it is okay for annotators to annotate the whole phrase; we ignore "a" and include "quick" in the evaluation

## 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, as listed below:

**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)

Type\_detection\_accuracy\_macro = sum of all type\_detection\_accuracy / #document

Realis\_detection\_accuracy\_macro = sum of all realis\_detection\_accuracy / #document

**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 )/ (sum of TP on all docs + sum of FP on 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)

Type\_detection\_accuracy\_micro = sum of num\_type\_correct / (total number of gold standard mention in all docs)

Realis\_detection\_accuracy\_micro = sum of realis\_detection\_score / (total number of gold standard mention in all docs)

# Appendix 1: Pseudo-code for scoring one document:

Initialize mappingScores as an empty list.

#STEP 1: Compute overlap scores for each pair of Gold/System Mention

FOR each system mention S := {S\_mid, S\_tokens, S\_realis, S\_type} (one per line)

Let S\_mid := mention id of S

Let S\_tokens := token IDs associated with S

Let S\_tokens := S\_tokens – {token IDs of invisible words} **#See NOTE 1**

Let S\_realis := realis status of S

Let S\_type := mention type of S

FOR each gold mention G:= {G\_mid, G\_tokens, G\_realis, G\_type}

Let G\_mid := mention id of G

Let G\_tokens := token IDs associated with G

Let G\_tokens := G\_tokens – {token IDs of invisible words}

Let G\_realis := realis status of G

Let G\_type := mention type of G

Let overlap := OVERLAP(S\_tokens, G\_tokens)

IF overlap > 0

mappingScores := mappingScores + (G, S, overlap)

END IF

END FOR

END FOR

#STEP2: After calculating all pairs, we find the best mapping between System

#Mentions and Gold Standard Mentions

Sort mappingScores based on overlap

Initialize Mapping as an empty list to hold mapping records

WHILE mappingScores != {}:

(G, S, overlap) = mappingScores.pop() #get the item with the highest overlap

#if G and S have not been mapped, it means there are no better overlaps

IF G has not been mapped and S has not been mapped

THEN Mapping := Mapping + {G,S, overlap}

ELSE IF G has been mapped but S has not been mapped

THEN Find the record in Mapping that contains G, append S to record

END IF

END WHILE

#Append system score to the gold standard file

FOR each gold mention G:

Score := Mapping[G].overlap

append Score to the end of the line of G\_mid in Gold Standard, in position <score2>

END FOR

#STEP3.1: Compute document level errors and corrects on mention detection

TP := 0

FOR EACH System Mention S

IF S is contained in Mapping

TP := TP + Mapping[S].overlap

ELSE

FP := FP + 1

END IF

END FOR

#STEP3.2: Compute document level precision, recall for mention detection:

Precision := TP / (TP+FP)

Recall := TP / #GoldStandardMentions

F1\_Score := 2\*Precision\*Recall/(Precision+Recall)

#STEP3.3: Compute mention and realis type detection score:

type\_correct\_score := 0

realis\_correct\_score := 0

FOR EACH LINE (G,{S}, overlap) in Mapping

Mapping\_num:= |{S}|

Single\_score := 1/ Mapping\_num

FOR EACH LINE S in {S}

IF G\_type == S\_type

type\_correct\_score := type\_correct\_score + Single\_score

END IF

IF G\_realis == S\_realis

realis\_correct\_score := realis\_correct\_score + Single\_score

END IF

END FOR

END FOR

Type\_detection\_accuracy := type\_correct\_score / #GoldStandardMentions

Realis\_detection\_accuracy:= realis\_correct\_score / #GoldStandardMentions

# Return and report the following measures for this document:

Measures for this doc = {TP, FP, type\_correct\_score, realis\_correct\_score , Precision, Recall, F1\_Score, Type\_detection\_accuracy, Realis\_detection\_accuracy }

**Subroutine OVERLAP(G,S):**

IF G == S, THEN score := 1.0

IF G∧S == {}, THEN score := 0.0

ELSE

precision\_m := (|S∧G|)/|S|

recall\_m := (|S∧G|)/|G|

score := 2\*precision\_m\*recall\_m / (precision\_m + recall\_m)

RETURN score

End Subroutine

# Appendix 2: Example of scoring computation:

Sample System output:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| System Id | Doc Id | Event Mention Id | Token Id List | Mention Text | Event Type | Realis Status | System Confidence |
| sue | sample | E1 | t17 | advice | Communicate | Other | 1 |
| sue | sample | E2 | t19 | reassurance | Communicate | Other | 1 |
| sue | sample | E3 | t33 | came | Transport-Person | Actual | 1 |
| sue | sample | E4 | t52 | going | Transport-Person | Actual | 1 |

Gold annotations:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| System Id | Doc Id | Event Mention Id | Token Id List | Mention Text | Event Type | Realis Status | System Confidence |
| gold | sample | E1 | t52 | going | Transport-Person | Actual | 1 |
| gold | sample | E2 | t33 | came | Transport-Person | Actual | 1 |
| gold | sample | E3 | t87 | got | Transport-Person | Actual | 1 |
| gold | sample | E4 | t14,t17,t18,t19 | offer advice or reassurance | Communicate | Other | 1 |

In the following tables, the “Event Type” and “Realis Status” are omitted for clarity

## STEP 1 : Compute overlap scores for each pair of Gold/System Mention

There are no invisible words, so no removal is done

Compute the “mappingScore” table as followed:

|  |  |  |
| --- | --- | --- |
| Gold Mention | System Mention | Overlap |
| (E1, [52]) | (E4, [52]) | 1 |
| (E2, [33]) | (E3, [33]) | 1 |
| (E4, [14,17,18,19]) | (E1, [17]) | 2/5 (See #) |
| (E4, [14,17,18,19]) | (E2, [19]) | 2/5 (Same as above) |

# Example calculation of overlap:

Prec(G\_E4,S\_E1) = (|E1 ^ E4|) / |E1| = 1/1 = 1;

Recall(G\_E4,S\_E1) = (|E1 ^ E4|) / |E4| = ¼ = ¼;

Overlap(G\_E4,S\_E1) = 2 \* Prec(G\_E4,S\_E1) \* Recall(G\_E4,S\_E1) / (Prec(G\_E4,S\_E1) + Recall(G\_E4,S\_E1) ) = 2 \* 1 \* ¼ / (1 + ¼ ) = 2/5

## STEP2: After the calculation of all pairs, we can find the best mapping between System Mention and Gold Standard Mentions

Sort the “mappingScore” table based on overlap (ties are currently broken on their appearance in the data):

|  |  |  |
| --- | --- | --- |
| Gold Mention | System Mention | Overlap |
| (E1, [52]) | (E4, [52]) | 1 |
| (E2, [33]) | (E3, [33]) | 1 |
| (E4, [14,17,18,19]) | (E1, [17]) | 2/5 |
| (E4, [14,17,18,19]) | (E2, [19]) | 2/5 |

We select mappings from the table above from top to bottom:

1. In row1, Select Gold, E1 to map to System, E4, we also record the overlap score = 1
2. In row2, Select Gold, E2 to map to System, E3, we also record the overlap score = 1
3. In row3, Select Gold, E4 to map to System, E1, we record the overlap score = 2/5
4. In row4, Select Gold, E4 to map to System, E2, since Gold E4 has already been mapped to a mention System E1, we do not record an overlap score, but we record the system mention here so we know that E4 is mapped to 2 system mention

We have the following mapping table (mappingScore table):

|  |  |  |
| --- | --- | --- |
| Gold Mention | System Mention | Overlap |
| (E1, [52]) | (E4, [52]) | 1 |
| (E2, [33]) | (E3, [33]) | 1 |
| (E4, [14,17,18,19]) | (E1, [17]) , (E2,[19]) | 2/5 |

## STEP3.1: Compute document level errors and corrects

TP is the sum of the overlap in the mappingScore table:

TP = 1 + 1 + 2/5 = 2.4

S{E2} is not contained in the mappingScore table, so

FP = 1

## STEP3.2: Compute document level precision, recall:

Precision := TP / (TP+FP) = 2.4 / (2.4+1) = 0. 7059

Recall := TP / #GoldStandardMentions = 2.25/4 = 0.6

F1 := 2\*Precision\*Recall/ (Precision+Recall) = 2\*0. 7059\*0.6/ (0. 7059+0. 6) = 0.6487

## STEP3.3: Compute mention type and realis status detection score:

For each row in the mapping table, we check whether the system mention(s) has/have the same realis status and mention type as the gold mention.

G\_E1 – S\_E4 and G\_E2 – S\_E3 are both one-to-one mappings, so N = 1. Both mention types and realis status are correct, giving type\_score = 2, realis\_score = 2.

G\_E4 is mapped to 2 mentions {S\_E1, S\_E2}, so N = 2. Both mention types and realis status are correct, giving type\_score = ½ + ½ = 1, realis\_score = ½ + ½ = 1.

The sum of type score is 2 + 1 = 3, and the total realis score 2+1 = 3. This gives the accuracy as:

Type\_detection\_accuracy := 3 / #GoldStandardMentions = 0.75

Realis\_detection\_accuracy:= 3 / #GoldStandardMentions = 0.75

## Final Output:

### Output1: The score appended gold standard file will be like the following

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| System Id | Doc Id | Event Mention Id | Token Id List | Mention Text | Event Type | Realis Status | System Confidence | Sue Mention score |
| gold | sample | E1 | t52 | going | Transport-Person | Actual | 1 | 1 |
| gold | sample | E2 | t33 | came | Transport-Person | Actual | 1 | 1 |
| gold | sample | E3 | t87 | got | Transport-Person | Actual | 1 | - |
| gold | sample | E4 | t14,t17,t18,t19 | offer advice or reassurance | Communicate | Other | 1 | 0.4 |

### Output2: Individual document performance and averaged performance

We only take one document as example, which make the micro and macro measures to be the same.



1. http://brat.nlplab.org/ [↑](#footnote-ref-1)
2. http://brat.nlplab.org/standoff.html [↑](#footnote-ref-2)
3. http://nlp.stanford.edu/software/corenlp.shtml [↑](#footnote-ref-3)