SemEval 2018 Task 6: Parsing Time Normalizations

Egoitz Laparra
University of Arizona
Tucson, AZ 85721, USA
laparra@email.arizona.edu

Dongfang Xu
University of Arizona
Tucson, AZ 85721, USA
dongfangxu9@email.arizona.edu

Steven Bethard
University of Arizona
Tucson, AZ 85721, USA
bethard@email.arizona.edu

Ahmed S. Elsayed
University of Colorado Boulder
Boulder, CO 80309
ahmed.s.elsayed@colorado.edu

Martha Palmer
University of Colorado Boulder
Boulder, CO 80309
martha.palmer@colorado.edu

Abstract

This paper presents the outcomes of the Parsing Time Normalization shared task held within SemEval-2018. The aim of the task is to parse time expressions into the compositional semantic graphs of the Semantically Compositional Annotation of Time Expressions (SCATE) schema, which allows the representation of a wider variety of time expressions than previous approaches. Two tracks were included, one to evaluate the parsing of individual components of the produced graphs, in a classic information extraction way, and another one to evaluate the quality of the time intervals resulting from the interpretation of those graphs. Though 40 participants registered for the task, only one team submitted output, achieving 0.55 F1 in Track 1 (parsing) and 0.70 F1 in Track 2 (intervals).

1 Introduction

The task of extracting and normalizing time expressions (e.g., finding phrases like *two days ago* and converting them to a standardized form like 2017-07-17) is a fundamental component of any time-aware language processing system. TempEval 2010 and 2013 (Verhagen et al., 2010; UzZaman et al., 2013) included a restricted version of a time normalization task as part of their shared tasks. However, the annotation scheme used in these tasks (TimeML; (ISO, 2012)) has some significant limitations: it assumes times can be described as a prefix of YYYY-MM-DDTHH:MM:SS (so it can't represent, e.g., *the past three summers*), it is unable to represent times that are are relative to events (e.g., *three weeks postoperative*), and it fails to reflect the compositional nature of time expressions (e.g., that *following* represents a similar temporal operation in *the following day* and *the following year*). This latter issue especially has discouraged machine learning approaches to time normalization; the most accurate systems for normalizing times are still based on sets of complex, manually-constructed rules (Bethard, 2013; Lee et al., 2014; Strötgen and Gertz, 2015).

The Parsing Time Normalizations shared task is a new approach to time normalization based on the Semantically Compositional Annotation of Time Expressions (SCATE) schema (Bethard and Parker, 2016), in which times are annotated as compositional time entities. Such entities are more expressive, being able to represent many more time expressions, and are more machine-learnable, as they can naturally be viewed as a semantic parsing task. The top of Figure 1 shows an example. Each annotation in the example corresponds to a formally defined time entity. For instance, the annotation on top of *since* corresponds to a BETWEEN entity that identifies an interval starting at the most recent March 6 and ending at the document creation time. The bottom of Figure 1 shows how those time entities can be composed to identify appropriate intervals on the timeline. Here, the BETWEEN entity finds the interval on the timeline that is between the intervals of its two arguments: the LAST and the DOC-TIME. Formally, this BETWEEN operator is defined as:

$$\begin{aligned}\textsc{Between}([t_1, t_2)&\colon \textsc{Interval},\\ [t_3, t_4)&\colon \textsc{Interval})\colon \textsc{Interval}\\ &= [t_2, t_3)\end{aligned}$$

In the proposed task, systems need only to identify time entities in text and link them correctly to signal how they are to be composed (i.e., systems would only need to produce annotation structures like those at the top of Figure 1). The timeline intervals implied by such system output are inferred through a time entity interpreter provided to the participants by the workshop organizers.

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 88–96
New Orleans, Louisiana, June 5–6, 2018.

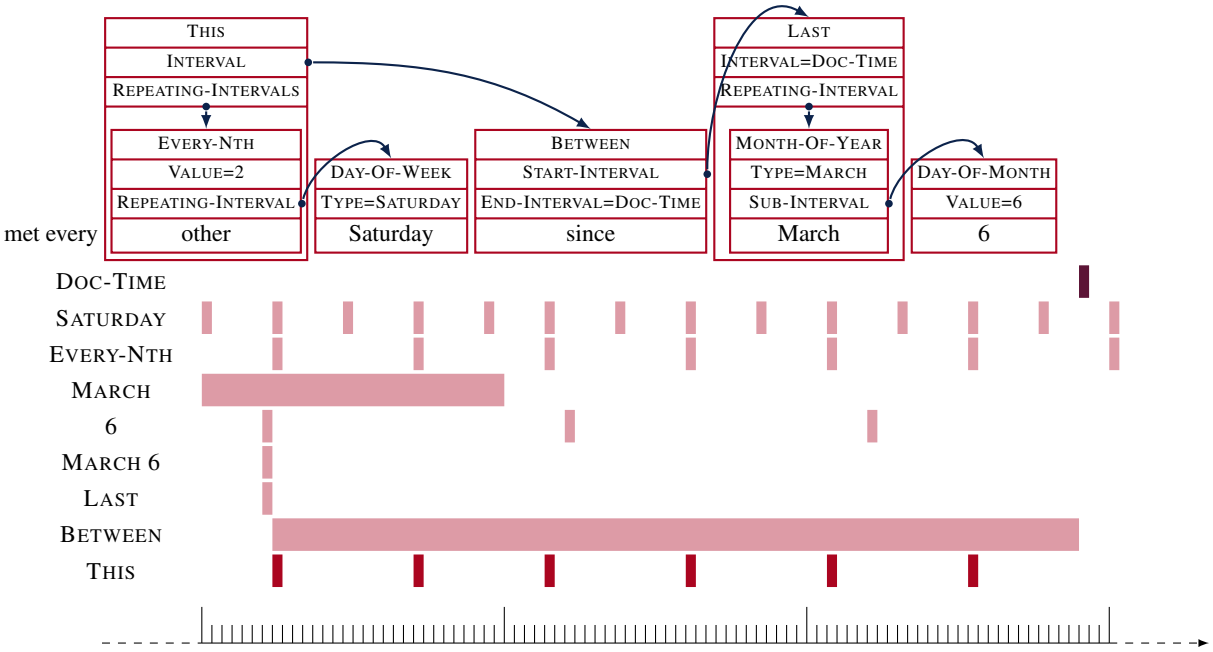

Figure 1: Example of semantically compositional time annotations and their interpretation.

The remainder of this paper is organized as follows. We describe the task goal and proposed tracks in Section 2. Section 3 contains the description of the data annotation schema and the statistics of our dataset. In Section 4, we explain the two evaluation metrics used in the task and in Section 5 the models used as baselines. We present the participant systems in Section 6 and the results obtained in Section 7. Finally, we discuss some conclusions learned in Section 8.

2 Tasks

The ultimate goal of the shared task is to interpret time expressions, identifying appropriate intervals that can be placed on a timeline. Given a document, a system must identify the **time entities** by detecting the spans of characters and labeling them with the proper SCATE type. Examples of time entities and their corresponding types in Figure 1 would be (*6*, DAY-OF-MONTH), (*Saturday*, DAY-OF-WEEK), (*March*, MONTH-OF-YEAR) or (*since*, BETWEEN). Besides the time entities explicitly expressed in the text, implicit occurrences must also be identified, like the THIS and LAST time entities in Figure 1 that do not have any explicit triggers in the text.

Once time entities have been identified, they should be linked together using the relations described in the SCATE schema. Following with the example in Figure 1, the time entity *6* should be linked as a SUB-INTERVAL of *March*, *Saturday* should be a REPEATING-INTERVAL of the time entity *other*, and so on. Finally, all the time entities must be completed with some additional properties needed for their interpretation. For example, the time entity *other* should have a VALUE of 2, the END-INTERVAL of *since* is the Document Creation Time, etc. Once again, the properties required by each time entity type are defined by the SCATE schema.[1]

Every resulting graph, composed of a set of linked time entities, represents a **time expression** that can be semantically interpreted. For this purpose, we provide a Scala library[2] that reads the graphs in Anafora XML format (Chen and Styler, 2013) and converts them into intervals on the timeline.

An example of interpreting the time entities corresponding to the expression *every Saturday since March 6* relative to an anchor time of April 21, 2017 is given in Figure 2. In this example, the values *today* and *result* store the entities that represent the time expressions April 21, 2017 and *every Saturday since March 6* respectively. The Scala command on the right side interprets the latter and produces the corresponding time intervals.

The task includes two evaluation methods, one for the parsing step, i.e. time entity identification

[1] https://github.com/clulab/anafora-annotations/blob/master/.schema/timenorm-schema.xml

[2] https://github.com/clulab/timenorm

```
scala> val today =
     |   ThisRI(
     |     ThisRI(
     |       Year(2017),
     |       RepeatingField(MONTH_OF_YEAR, 4)),
     |     RepeatingField(DAY_OF_MONTH, 21))
scala> val result =
     |   ThisRIs(
     |     Between(
     |       LastRI(
     |         today,
     |         Intersection(Set(
     |           RepeatingField(MONTH_OF_YEAR, 3),
     |           RepeatingField(DAY_OF_MONTH, 6)))),
     |       today),
     |       RepeatingField(DAY_OF_WEEK, 6))
```

```
scala> for (Interval(start, end) <- result.intervals)
     |   println(start, end)
(2017-03-11T00:00,2017-03-12T00:00)
(2017-03-18T00:00,2017-03-19T00:00)
(2017-03-25T00:00,2017-03-26T00:00)
(2017-04-01T00:00,2017-04-02T00:00)
(2017-04-08T00:00,2017-04-09T00:00)
(2017-04-15T00:00,2017-04-16T00:00)
(2017-04-22T00:00,2017-04-23T00:00)
...
```

Figure 2: Interpretation of *every Saturday since March 6*.

and linking, and one to score the resulting time intervals. For the later, we only consider time expressions that yield a finite set of bounded intervals, for example, *last Monday*. Time expressions that refer to an infinite set of intervals, like *every month*, are not considered in the interval-based part of the evaluation.

Participants only need to produce Anafora outputs with parsed time entities; the interpretation is carried out by the evaluation system. The evaluation system is also able to obtain the intervals from timestamps in TimeML format. Thus, systems can be evaluated by both methods or just by the interval-based one, depending on the output format.

In summary, the tasks offers two tracks:

Track 1: Parse text to time entities. Systems must identify time entities in text and link them correctly to signal how they have to be composed. The output must be given in Anafora format. In this track, all time entities and relations of every time expression are evaluated.

Track 2: Produce time intervals. Systems can participate through Track 1 or by providing TimeML annotations. In both cases, the intervals are inferred by our interpreter. In this track, only bounded time intervals are scored.

3 Data

The Parsing Time Normalization corpus[3] covers two different domains: newswire and clinical notes. For the former, we have annotated a subset of Tempeval-2013 corpus (UzZaman et al., 2013), which contains a collection of news articles from different sources, such as Wall Street Journal, New York Times, Cable News Network, Voices of America, etc. For the clinical domain, we have annotated a subset of the THYME corpus used in the Clinical TempEvals (Bethard et al., 2015, 2016, 2017), which includes a set of de-identified clinical notes and pathology reports from cancer patients at the Mayo Clinic.

The Newswire annotation was performed by linguistic students at the University of Alabama at Birmingham, and by linguistics students at the University of Arizona, funded as part of a university-sponsored undergraduate research opportunity. The clinical portion of the corpus was annotated by linguistics students at the University of Colorado, funded as part of the United States National Institutes of Health (NIH) award R01LM010090.

Documents have been annotated by two annotators and adjudicated by a third, and despite the complexity of the annotation scheme, high levels of inter-annotator agreement have been achieved: 0.917 F_1 on annotation spans and types, and 0.821 F_1 on the complete task of spans, types, and links. (We use F_1 since the κ coefficient (Cohen, 1960) converges to F_1 in cases where the number of non-annotations is much larger than the number of annotations (Hripcsak and Rothschild, 2005).)

Annotated data is stored in Anafora XML format (Chen and Styler, 2013), where, for example, the annotations from Figure 1 look like Figure 3. A more detailed explanation of the annotation guidelines can be found in Bethard and Parker (2016). Libraries for parsing this format are available to

[3] https://github.com/bethard/anafora-annotations/releases

```
<data>
 <annotations>
  <entity>
   <id>1@e@gold</id>
   <span>11,19</span><!-- "Saturday" -->
   <type>Day-Of-Week</type>
   <properties>
    <Type>Saturday</Type>
   </properties>
  </entity>
  <entity>
   <id>2@e@gold</id>
   <span>10,15</span><!-- "other" -->
   <type>This</type>
   <properties>
    <Interval>4@e@gold</Interval>
    <Repeating-Intervals>2@e@gold</Repeating-Intervals>
   </properties>
  </entity>
  <entity>
   <id>3@e@gold</id>
   <span>10,15</span><!-- "other" -->
   <type>Every-Nth</type>
   <properties>
    <Value>2</Value>
    <Repeating-Interval>1@e@gold</Repeating-Interval>
   </properties>
  </entity>

  ...

</annotations>
</data>
```

Figure 3: Snippet of the Anafora XML for Figure 1.

participants in both Python[4] and Scala[5].

Table 1 shows the statistics of the resulting annotation. The Newswire portion of the corpus contains 98 documents with 2,428 time entities annotated. These entities compose a total of 968 time expressions of which 564 correspond to bounded intervals. The Clinical portions of the corpus includes 408 documents. The annotation covers 27,362 time entities that compose 8,163 time expressions. From these, 4,204 yield bounded intervals.

4 Evaluation Metrics

We propose two types of scoring metrics for this task, one for the evaluation of each track. The first follows a more traditional information extraction evaluation: measure the precision and recall of finding and linking the various time entities. Specifically, we define:

$$P(S,H) = \frac{|S \cap H|}{|S|}$$

$$R(S,H) = \frac{|S \cap H|}{|H|}$$

$$F_1(S,H) = \frac{2 \cdot P(S,H) \cdot R(S,H)}{P(S,H) + R(S,H)}$$

where S is the set of items predicted by the system and H is the set of items produced by the humans. For these calculations, each item is an annotation, and one annotation is considered as equal to another if it has the same character span (offsets), type, and properties (with the definition applying recursively for properties that point to other annotations).

The second scoring method evaluates the accuracy of systems with respect to the timeline in a more direct way. First, annotations, in either TimeML or SCATE format, are converted into time intervals. TimeML TIMEX3 (time expression) annotations are translated into intervals following ISO 8601 semantics of their VALUE attribute. For example, 2010-02-25 is converted to the interval [2010-02-25T00:00:00, 2018-02-26T00:00:00), that is, the 24-hour period starting at the first second of the day on 2010-02-25 and ending just before the first second of the day on 2010-02-26. SCATE annotations are converted to intervals according to the formal semantics of each entity, using the Scala library provided by Bethard and Parker (2016). For example, Next(Year(2010), SimplePeriod(YEARS, 4)), is converted to [2011-01-01T00:00, 2015-01-01T00:00), i.e., the 4 years following 2010. Note that there may be more than one interval associated with a single annotation, as in the *every Saturday since March 6* example in Figure 2. Once all annotations have been converted into intervals along the timeline, we can calculate the overlap between the intervals of different annotations.

Given two sets of intervals, we define the interval precision, P_{int}, as the total length of the intervals in common between the two sets, divided by the total length of the intervals in the first set. Interval recall, R_{int} is defined as the total length of the intervals in common between the two sets,

[4] https://github.com/bethard/anaforatools

[5] https://github.com/bethard/timenorm

	Newswire			Clinical		
	Train	Dev	Test	Train	Dev	Test
Documents	64	14	20	232	35	141
SCATE entities	1,628	402	398	14,936	2,896	9,530
SCATE time exp.	636	146	186	4,469	879	2,815
SCATE bounded	391	80	93	2,303	430	1,471

Table 1: Number of documents and SCATE annotations for both sections of the corpus following the SCATE schema.

divided by the total length of the intervals in the second set. Formally:

$$I_S \bigcap I_H = \{i \cap j : i \in I_S \wedge j \in I_H\}$$

$$P_{\text{int}}(I_S, I_H) = \frac{\sum_{i \in \text{COMPACT}(I_S \bigcap I_H)} |i|}{\sum_{i \in I_S} |i|}$$

$$R_{\text{int}}(I_S, I_H) = \frac{\sum_{i \in \text{COMPACT}(I_S \bigcap I_H)} |i|}{\sum_{i \in \cup I_H} |i|}$$

where I_S and I_H are sets of intervals, $i \cap j$ is the possibly empty interval in common between the intervals i and j, $|i|$ is the length of the interval i, and COMPACT takes a set of intervals and merges any overlapping intervals.

Given two sets of annotations (e.g., one each from two time normalization systems), we define the overall precision, P, as the average of interval precisions where each annotation from the first set is paired with all annotations that textually overlap it in the second set. Overall recall is defined as the average of interval recalls where each annotation from the second set is paired with all annotations that textually overlap it in the first set. Formally:

$$\text{OI}_a(B) = \bigcup_{b \in B : \text{OVERLAPS}(a,b)} \text{INTERVALS}(b)$$

$$P(S, H) = \frac{1}{|S|} \sum_{s \in S} P_{int}(\text{INTERVALS}(s), \text{OI}_s(H))$$

$$R(S, H) = \frac{1}{|H|} \sum_{h \in H} R_{int}(\text{INTERVALS}(h), \text{OI}_h(S))$$

where S and H are sets of annotations, $\text{INTERVALS}(x)$ gives the time intervals associated with the annotation x, and $\text{OVERLAPS}(a, b)$ decides whether the annotations a and b share at least one character of text in common.

Note that as defined, P and R can be applied only to time expressions that yield a finite set of bounded intervals.

5 Baseline systems

Two systems were used as baselines to compare the participating systems against.

Character-based model (Laparra et al., 2018) is a novel supervised approach for time normalization that follows the SCATE schema. This model decomposes the normalization of time expressions into two modules:

time entity identification detects the spans of characters that belong to each time expression and labels them with their corresponding time entity type. This step is performed by character-based recurrent neural network with two stacked bidirectional Gated Recurrent Units.

time entity composition links relevant time entities together while respecting the entity type constraints imposed by the SCATE schema. This component is a rule-based algorithm that iterates over the time entities that are found by the previous step, linking them and filling in the required properties. The version used for the shared task includes some improvements to the sentence segmentation and Month-Of-Year normalization of Laparra et al. (2018).

These two tasks are run sequentially using the output of the former as input to the latter. Once identification and composition steps are completed, the final product, i.e. the semantic composition of the time entities, can fed to the SCATE interpreter to produce time intervals.

HEIDELTIME (Strötgen and Gertz, 2015)[6] is rule-based temporal tagger with multilingual support that includes English, German, Dutch, Vietnamese, Arabic, Spanish, Italian, French, Chinese and Russian. HeidelTime identifies temporal expressions based on language specific patterns and

[6] https://code.google.com/p/heideltime/

normalizes them according to TIMEX annotations (Sundheim, 1996). As the output of HeidelTime follows TimeML format, we use this system as a baseline only for Track 2.

6 Participating systems

Although 40 people registered to the the evaluation task, only 1 team submitted results. The team participated in Track 1 and, consequently, in Track 2. The team also submitted improved results just after the end of the evaluation phase. This improvement was obtained by solving a few bugs in the original system, and with no access to the test data, so we have included the fixed version in this paper as an additional run.

CHRONO (Olex et al., 2018) is a primarily rule-based system that performs time normalization by running the following three steps:

1) Temporal tokens are identified and flagged using regex expressions to identify formatted dates/times, and by parsing out specific temporal words and numeric tokens.

2) Temporal phrases are identified by searching for consecutive numeric/temporal tokens according to certain constraints.

3) Temporal phrases are parsed and normalized into the SCATE schema via detailed rule-based parsing, including the utilization of part-of-speech tags, to identify each component of an expression and link sub-intervals appropriately.

A machine learning approach is taken to disambiguate PERIODS and CALENDAR-INTERVALS after the rule-base parsing has determined it is one or the other (e.g. if it sees the word *week* it will pass it to the ML module for assignment to a PERIODS or CALENDAR-INTERVALS). The ML feature vector is a boolean vector composed of the target token's temporal status (1=temporal, 0=not temporal), the temporal context (1=at least one temporal token within a window of ± 5, 0=no temporal tokens within window), the numeric context (1=a numeric token exists immediately before of after the target, 0=no numeric tokens in context), and the lexical context of all words within a 5-word window of the target (1=word is present, 0=word is not present). The group explored different supervised models like naive Bayes, decision trees, support vector machines, and neural networks. They found that the best results were obtained by the neural network.

Domain	Model	F_1	P	R
Newswire	Character	0.51	0.57	0.46
Newswire	Chrono	0.44	0.46	0.42
Newswire	Chrono*	0.55	0.61	0.50
Clinical	Character	0.57	0.52	0.63

Table 2: Official results in Track 1 (parsing) for the Newswire and Clinical domains.

CHRONO* improves the previous version by solving three bugs in the model. First, the parsing method for various types of temporal components were supposed to be executed in a specific order. However, some of them were swapped and not analyzed in the expected order. Second, the system was supposed to assume there is only one year, one month, and one day mentioned per temporal phrase. This worked for the month and day, however, it was failing with 4-digit years. Finally, for most parsing methods the system loops through each token in the temporal phrase but it skipped the loop when identifying full numeric expressions, like "1953" or "08091998". Thus, phrases like "Last 1953" were not being counted as having any numeric values in them.

7 Evaluation Results

The official results are presented in Table 2 and Table 4. For each track we present the precision (P), recall (R) and F_1 score obtained by the metrics presented in Section 4. The only participant of the task submitted output just for the Newswire domain, thus, we only report the performance of this system in this domain. The results of the Character-based baseline have been obtained training the model with the training set of the corresponding domain (Newswire or Clinical) and a set of randomly generated dates, as explained in Laparra et al. (2018).

In Track 1 (Table 2) the original version of CHRONO do not reach the Character-based baseline, 0.44 F_1 vs 0.51 F_1. However, the fixed version of the system (CHRONO*) outperforms the baseline in terms of F_1 (0.55) as well as in terms of P (0.61) and R (0.50).

Table 3 shows a more detailed comparison between the Character-based baseline and CHRONO*. These figures represent the performances of both models for each SCATE temporal type. This includes the identification of the time entity, its prop-

SCATE-type	#	Char	Chrono*
AMPM-Of-Day	1	0.000	0.667
After	19	0.000	0.000
Before	20	0.105	0.000
Between	7	0.000	0.000
Calendar-Interval	27	0.698	0.526
Day-Of-Month	22	0.917	1.000
Day-Of-Week	16	0.812	0.903
Hour-Of-Day	2	0.000	0.667
Intersection	3	0.000	0.000
Last	40	0.500	0.333
Minute-Of-Hour	1	0.000	0.000
Month-Of-Year	36	0.824	0.917
Next	14	0.053	0.412
NthFromStart	4	0.000	0.000
Number	27	0.596	0.522
Part-Of-Day	3	0.000	1.000
Period	56	0.391	0.409
Season-Of-Year	10	0.000	0.182
Sum	1	0.000	0.000
This	37	0.429	0.552
Time-Zone	1	0.000	0.000
Two-Digit-Year	1	0.000	0.000
Year	50	0.822	0.826

Table 3: Results in Track 1 per SCATE type. Char stands for Character-based baseline. The number of gold cases per type is included (#).

Domain	Model	F_1	P	R
Newswire	HeidelTime	0.74	0.71	0.77
Newswire	Character	0.77	0.83	0.72
Newswire	Chrono	0.65	0.66	0.63
Newswire	Chrono*	0.70	0.65	0.75
Clinical	HeidelTime	0.70	0.60	0.82
Clinical	Character	0.72	0.70	0.75

Table 4: Official results in Track 2 (intervals) for the Newswire and Clinical domains.

erties and links. In general, CHRONO* performs better or similar for all the types. It is remarkable that, while the outcomes for the THIS and NEXT operators are much better, CHRONO* fails to extract properly the LAST operator.

In Track 2 (Table 4) the best system is the Character-based baseline with 0.76 F_1, followed by HEIDELTIME with 0.74 F_1. None of the versions of CHRONO performs better than the baselines, although the fixed version (CHRONO*) gets enhanced results, 0.65 F_1 vs 0.70 F_1 , following the improvement obtained in Track 1. It is remarkable that HEIDELTIME and CHRONO*, essentially rule-based systems, obtain better R than the Character-based baseline, that relies strongly on a supervised model. Specifically, HEIDELTIME obtains the best R (0.77), but CHRONO* also outperforms the Character-based baseline in terms of R, 0.75 vs 0.71. However, the Character-based baseline obtains a much higher P (0.83) than the 0.71 of HEIDELTIME and the 0.65 of CHRONO*.

As explained in Section 4, the metric for Track 1 evaluates the individual temporal components extracted by the systems, either time entities or links between time entity pairs. On the other hand, the intervals scored by the metric for Track 2 are produced by interpreting the whole graph. Moreover, not all the time expressions yield a finite set of bounded intervals, as can be seen in Table 1. Consequently, better performances in Track 1 do not necessarily yield better results in Track 2. In particular, although CHRONO* is better than the Character-based baseline in Track 1 it produces an excessive number of time expressions yielding bounded intervals (108), which affects the P in Track 2. In contrast, the Character-based baseline is more conservative and accurate in this respect (85).

Although we didn't receive any submission for the Clinical domain, in order to set a reference for future research, we present in Table 2 and Table 4 the performances of the baseline systems in this domain.

8 Conclusion

The Parsing Time Normalization task is the first effort to extend time normalization to richer and more complex time expressions. We have provided a complete annotation for two different domains, newswire and clinical notes, and introduced two different metrics for evaluation. In particular, the interval based evaluation for Track 2 is a novelty for these kind of tasks. The performance of the systems shows that there is still room for improvement, especially for Track 1.

Although, CHRONOS included a small supervised component in its architecture, we were expecting a higher number of machine learning based approaches. However, CHRONOS shows that rule-based models can obtain competitive results. Sadly, the scarcity of participating systems does not allow us to form a further judgment.

No submissions were received for the clinical

domain, despite a wider and more complete dataset for this domain. This was almost certainly the result of a change in management at the Mayo Clinic that put on hold the data use agreement process (which is required for access to the clinical data) for several months during the practice phase. Thus, though many people showed interest in the task (more than 40 people registered) and Mayo reported several data use agreement applications, this problem de-motivated the participation.

The CodaLab competition for the Parsing Time Normalizations shared task[7] will continue to accept submissions in its Post-Evaluation phase indefinitely, so as more researchers make it through the data use agreement process, we expect we will see future participation in this task.

Acknowledgements

The work is funded by the THYME project (R01LM010090) from the National Library Of Medicine. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Library Of Medicine or the National Institutes of Health.

References

Steven Bethard. 2013. A synchronous context free grammar for time normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 821–826, Seattle, Washington, USA. Association for Computational Linguistics.

Steven Bethard, Leon Derczynski, Guergana Savova, James Pustejovsky, and Marc Verhagen. 2015. Semeval-2015 task 6: Clinical tempeval. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 806–814, Denver, Colorado. Association for Computational Linguistics.

Steven Bethard and Jonathan Parker. 2016. A semantically compositional annotation scheme for time normalization. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA). [Acceptance rate 60%].

Steven Bethard, Guergana Savova, Wei-Te Chen, Leon Derczynski, James Pustejovsky, and Marc Verhagen. 2016. Semeval-2016 task 12: Clinical tempeval. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1052–1062, San Diego, California. Association for Computational Linguistics.

Steven Bethard, Guergana Savova, Martha Palmer, and James Pustejovsky. 2017. Semeval-2017 task 12: Clinical tempeval. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 565–572, Vancouver, Canada. Association for Computational Linguistics.

Wei-Te Chen and Will Styler. 2013. Anafora: A web-based general purpose annotation tool. In *Proceedings of the 2013 NAACL HLT Demonstration Session*, pages 14–19, Atlanta, Georgia. Association for Computational Linguistics.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

George Hripcsak and Adam S. Rothschild. 2005. Agreement, the f-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298.

ISO. 2012. Language resource management – semantic annotation framework (semaf) – part 1: Time and events (semaf-time, iso-timeml). Technical report. 24617-1:2012.

Egoitz Laparra, Dongfang Xu, and Steven Bethard. 2018. From characters to time intervals: New paradigms for evaluation and neural parsing of time normalizations. *Transactions of the Association for Computational Linguistics*.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447, Baltimore, Maryland. Association for Computational Linguistics.

Amy Olex, Luke Maffey, Nicholas Morgan, and Bridget McInnes. 2018. Chrono at semeval-2018 task 6: A system for normalizing temporal expressions. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, Luisiana. Association for Computational Linguistics.

Jannik Strötgen and Michael Gertz. 2015. A baseline temporal tagger for all languages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 541–547, Lisbon, Portugal. Association for Computational Linguistics.

Beth M. Sundheim. 1996. Overview of results of the muc-6 evaluation. In *Proceedings of a Workshop on Held at Vienna, Virginia: May 6-8, 1996*, TIPSTER '96, pages 423–442, Stroudsburg, PA, USA. Association for Computational Linguistics.

[7] https://competitions.codalab.org/competitions/17286

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Georgia, USA. Association for Computational Linguistics.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 57–62, Uppsala, Sweden. Association for Computational Linguistics.