SUTIME: A Library for Recognizing and Normalizing Time Expressions

Angel X. Chang, Christopher D. Manning

Computer Science Department, Stanford University, Stanford, CA, 94305
{angelx, manning}@cs.stanford.edu

Abstract

We describe SUTIME, a temporal tagger for recognizing and normalizing temporal expressions in English text. SUTIME is available as part of the Stanford CoreNLP pipeline and can be used to annotate documents with temporal information. It is a deterministic rule-based system designed for extensibility. Testing on the TempEval-2 evaluation corpus shows that this system outperforms state-of-the-art techniques.

Keywords: temporal tagger, time normalization, pattern matching

1. Introduction

The importance of modeling temporal information is increasingly apparent in natural language applications, such as information extraction and question answering. For instance, in relation extraction, it is not sufficient to just extract simple relations like *President(U.S.A, George Walker Bush)*. Rather, one would also need to extract fluents that capture the temporal range over which such relations hold. Given the sentence

> *George Walker Bush (born July 6, 1946) is an American politician who served as the 43rd President of the United States from 2001 to 2009.*

we can extract the following temporal information about when George Bush was born and when he was president:

	Expression	*Type*	*Normalized*
Birth	July 6, 1946	DATE	1946-07-06
Pres.	from 2001 to 2009	RANGE	2001/2009

Extracting such temporal information requires the ability to recognize temporal expressions, and to convert them from text to a normalized form that is easy to process. Systems that extract temporal expressions are known as *temporal taggers*. In this paper, we present SUTIME, a system for extracting and normalizing temporal expressions.

2. System Description

SUTIME is a rule-based temporal tagger built on regular expression patterns. Temporal expressions are bounded in their complexity, so many of them can be captured using finite automata. As shown by systems such as FASTUS (Hobbs et al., 1997), a cascade of finite automata can be very effective at extracting information from text. With SUTIME, we follow a similar staged strategy of (i) building up patterns over individual words to find numerical expressions; then (ii) using patterns over words and numerical expressions to find simple temporal expressions; and finally (iii) forming composite patterns over the discovered temporal expressions.

In NLP applications, text is usually first tokenized and annotated, making it convenient to specify regular expressions over the tokens. The resulting regular expressions can be more concise and easier to understand, manipulate and modify than traditional regular expressions over strings. Regular expressions over tokens facilitate access to additional features, such as parts-of-speech and named entity tags. With this approach, adding new rules to SUTIME is simple, by design.

We provide SUTIME as a Java library, implemented as an annotator in the Stanford CoreNLP pipeline;[1] its main features are described below:

Extraction of temporal expressions from text: Given tokenized English text, SUTIME finds temporal expressions and outputs annotations for further manipulation and interpretation. Its output includes annotations in the form of TIMEX3 tags. TIMEX3 is part of the TimeML annotation language (Pustejovsky et al., 2003) for markup of events, times and their temporal relations in documents. To better capture temporal information found in natural language, the value attribute of TIMEX3 tags extends the ISO 8601[2] standard. For temporal expressions not covered by this specification, we introduce additional attributes.

Representation of temporal objects as Java classes: Natural language uses many kinds of temporal expressions. SUTIME provides tools to map them to logical representations and data structures that are easier to handle programmatically. For interoperability, our representations are convertible to Joda-Time[3] classes.

Resolution of temporal expressions with respect to a reference date: When processing natural language text, one often has to work with expressions that refer to a relative time (e.g., *last Friday*). Determining the actual date to which such expressions refer requires a reference date, on which the statement was made. SUTIME uses document dates as references. For example, for a document from **2011-09-19**, SUTIME would resolve the date referred to by *last Friday* as **2011-09-16**. But there could be confusion about the time point to which expressions refer: e.g., from a date like

[1] nlp.stanford.edu/software/corenlp.shtml

[2] www.iso.org/iso/date_and_time_format

[3] Joda-Time — joda-time.sourceforge.net — is a comprehensive Java API for working with dates and times.

2011-09-19, a Monday, it is unclear whether *Friday* should refer to **2011-09-16** or **2011-09-23**. In such cases, the verb tense of the clause could help resolve the ambiguity.

2.1. Types of Temporal Expressions

SUTIME supports four basic types of temporal objects: **Time**, **Duration**, **Interval**, and **Set**.

Time: A time point indicating a particular instance on a time scale. SUTIME recognizes both relative times, such as *next Monday*, as well as absolute times, such as *January 12, 1999*. SUTIME also handles partially specified times, such as *the nineties*. Below, we give examples of some expressions that it can recognize, using the TIMEX3 type and normalized value (with **2011-09-19** as the reference date):

Expression	*Type*	*Value*
October of 1963	DATE	1963-10
October	DATE	2011-10
last Friday	DATE	2011-09-16
next weekend	DATE	2011-W39-WE
the day after tomorrow	DATE	2011-09-21
the nineties	DATE	199X
winter of 2000	DATE	2000-WI
5th century B.C.	DATE	-05XX
now	DATE	PRESENT_REF
Saturday morning	TIME	2011-09-24TMO
4 p.m. Tuesday	TIME	2011-09-20T16:00

Duration: The amount of intervening time between the two end-points of a time interval. Durations can be specified as a combination of a unit (e.g., day, month, year, etc.) and a numeric value (the quantity associated with the unit). SUTIME recognizes three types of durations:
- **exact** durations: both value and unit fully specified;
- **inexact** durations: unit known, but not its value; and
- duration **ranges**: the duration is bounded between a minimum and a maximum duration. Note that duration ranges are not part of the TIMEX3 standard.

Examples of temporal expressions corresponding to the three different types of durations are given below:

	Expression	*Type*	*Value*
Exact	3 days	DURATION	P3D
Inexact	a few years	DURATION	PXY
Range	2 to 3 months	DURATION	P2M/P3M

Interval: A range of time defined by a start and end time points. Recognizing an interval expression, such as *from July to August*, involves identifying the individual endpoints (*July* and *August*). While it is useful to recognize the entire expression as a time interval, it can also be helpful to identify the individual nested time expressions. SUTIME optionally tries to recognize ranges and includes such nested expressions. A time interval is not a separate type in TIMEX3, but can be represented as a **DURATION** with begin and end times. For example, SUTIME can provide three separate TIMEX tags for the expression *17 August 1656 - 21 January 1669*:

```
<TIMEX3 tid="t1" value="1656-08-17"
  type="DATE">17 August 1656</TIMEX3>
<TIMEX3 tid="t2" value="1669-01-21"
  type="DATE">21 January 1669</TIMEX3>
<TIMEX3 tid="t3" value="PT108960H" type="DURATION"
  beginPoint="t1" endPoint="t2">
  17 August 1656 - 21 January 1669</TIMEX3>
```

SUTIME also provides functions for extracting time intervals from temporal objects. For instance, the date **2011-09-19** can be converted to the interval from **2011-09-19T00:00:00** to **2011-09-19T24:00:00**.

Set: A set of temporals. SUTIME supports periodic temporal sets representing times that occur with some frequency. For example, it will provide the following TIMEX3 type and value for the expression *Every third Sunday*:

Expression	*Type*	*Value*
Every third Sunday	SET	XXXX-WXX-7

SUTIME will provide additional attributes with more information about the temporal set:

```
<TIMEX3 tid="t1" value="XXXX-WXX-7" type="SET"
  quant="every third" periodicity="P3W">
  Every third Sunday</TIMEX3>
```

2.2. Recognizing and Normalizing Expressions

To recognize temporal expressions, SUTIME applies three types of rules, in the following order:

1. **text regex** rules: mappings from simple regular expressions over characters or tokens to temporal representations;

2. **compositional** rules: mappings from regular expressions over chunks (both tokens and temporal objects) to temporal representations; compositional rules are iteratively applied after the text regex rules. At each stage, nested time expressions are removed,[4] and these rules are applied until the final list of time expressions stabilizes;

3. **filtering** rules: the final stage, in which ambiguous expressions that are likely to not be temporal expressions are removed from the list of candidates. For instance, we can specify a rule indicating that if a potential temporal expression is a single word *fall* and the part of speech tag is not a noun, then it is likely that *fall* refers to the act of falling and not the season autumn, and so SUTIME will refrain from marking it as a temporal expression.

After all the temporal expressions have been recognized, each temporal expression is associated with a temporal object. Each temporal object is resolved with respect to the reference date using heuristic rules. At this time, relative times are converted to an absolute time, and composite time objects are simplified as much as possible. Finally, SUTIME will take the internal time representation and produce a TIMEX3 annotation for each temporal expression.

2.3. Temporal Pattern Language

SUTIME uses a regular expression language for expressing how text should be mapped to temporal objects. SUTIME is built on top of TOKENSREGEX[5], a generic framework included in Stanford CoreNLP for defininig patterns over text and mapping to semantic objects. Using this framework, rules for how to map text to the temporal objects provided by the SUTime Java library are specified.

[4] SUTIME can optionally keep nested time expressions.

[5] nlp.stanford.edu/software/tokensregex.shtml

Below we give a short description of the temporal pattern language provided by SUTIME. The full specification for SUTIME's temporal pattern language can be found at `nlp.stanford.edu/software/sutime.shtml`.

SUTIME supports three types of patterns for matching text:

1. **token patterns**: patterns over tokens using the token regular expression language specified by TOKENSREGEX, which provides functionality similar to that of the Java regular expression library but over tokens. Since the patterns are specified over tokens, annotations on the tokens can be specified for matching.

   For example, here is a simple rule specifying a mapping from a token that matches the regular expression `/years?/` to the predefined duration type **YEAR**.

   ```
   { ruleType: "tokens",
     pattern: ( /years?/ ),
     result: YEAR }
   ```

   We can also specify more complex regular expression patterns to recognize durations such as *4 to 5 years*, mapping each to a **Duration** with the appropriate unit and numerical value.

   ```
   { ruleType: "tokens",
     pattern: ( ($NUM) /to|-/ ($NUM) [ "-" ]? ($TEUNITS_NODE) ),
     result: Duration( $1, $2, $3) }
   ```

   In this example, `Duration($1, $2, $3)` creates a range **Duration** object that uses the first matched group `$1` as the start of the range, the second matched group `$2` as the end of the range, and third matched group `$3` as the time unit of the duration. `$NUM` and `$TEUNITS_NODE` are macros (also defined using the temporal language) that are expanded during the token pattern matching to match numbers and duration unit tokens.

2. **string patterns**: patterns over text using Java regular expressions. Instead of targeting tokens, text can also be matched directly, independently of tokenization.

   For example, here is a rule specifying regular expression patterns that recognize durations such as *3-years*, mapping each to a **Duration** with the appropriate unit and numerical value.

   ```
   { ruleType: "text",
     pattern:  /(\d+)[-\s]($TEUnits)(s)?([-\s]old)?/ ,
     result:   Duration($1, $2) }
   ```

   Here, `Duration($1, $2)` creates a **Duration** object that uses the first matched group `$1` as the numerical value of the duration and the second matched group `$2` as the duration unit.

3. **time patterns**: time-specific patterns over text similar to patterns accepted by Java's DateFormat and JodaTime's DateTimeFormat. These patterns allow for a more human-readable format than using standard regular expressions.

   For instance, the pattern below can be used to recognize variations of ISO 8601 date/time patterns.

   ```
   { ruleType: "time",
     pattern: /yyyy-?MM-?dd-?'T'HH(:?mm(:?ss([.,]S{1,3})?)?)?/ }
   ```

   An appropriate **Time** object is created based on the parts of the pattern matched.

Both composite rules and filtering rules can be specified as patterns over tokens. Below is an example rule that combines the pattern **Date** *at* **Time** into one temporal object.

```
{ ruleType: "composite",
  pattern: ( ( [ { temporal::IS_TIMEX_DATE } ] )
             /at/  ( [ { temporal::IS_TIMEX_TIME } ] ) ),
  result:  TemporalCompose(INTERSECT,
                           $0[0].temporal,
                           $0[-1].temporal) }
```

For instance, we can specify the following rule to indicate that if a potential temporal expression is a single word like *fall* and the part of speech tag is not a noun, then it should not be resolved to a temporal object.

```
{ ruleType: "filter",
  pattern: ( [ { word:/fall|spring|second|march|may/ }
           & !{ tag:/NN.*/ } ] ) }
```

In addition to specifying the pattern and the resulting object, it is also possible to specify a priority for each rule. When multiple rules can be triggered for a sequence of tokens, a rule is selected based on the priority of the rule, followed by the length of the matched sequence, and finally the order in which the rule was specified.

2.4. Limitations

As a rule-based system, SUTIME is limited by the coverage of the rules for the different types of temporal expressions that it recognizes. Some of the known limitations are:

- Handling of ambiguous phrases is poor. One of the biggest limitations of SUTIME, and other rule-based systems, is their brittleness when faced with ambiguous language. Consider the phrase *The water from the spring was fresh and clear*. Although it is clear that *spring* is not a time expression, a rule-based system could incorrectly identify it as a season. Probabilistic models could help resolve such ambiguities. If we first ran a named entity recognizer, e.g., the Stanford NER system (Finkel et al., 2005), we could design patterns taking into account whether a word's NER tag is **DATE**.

- Resolving relative expressions can be difficult. Even with a reference date, there can be inherent ambiguity about the time point to which a relative time expression refers. For instance, given a reference date of **2011-09-19**, a Monday, it is unclear whether *Friday* refers to **2011-09-16** or **2011-09-23**. In addition, SUTIME resolves all temporal expressions to one reference date, the document date. In some cases, it may be more appropriate to resolve a particular temporal expression to a nearby date in the text instead.

- Holidays are not supported. SUTIME does not currently recognize time expressions relating to holidays, such as *Christmas* or *Halloween*.

- Support for temporal ranges is poor. For instance, the expression *from 3 to 4 p.m.* is incorrectly identified as **15:57:00**, while the expression *12-13 March 2011* is identified just as **2011-03**.

- Non-whole numbers such as *a half* are not recognized. Due to limitations of the numeric normalizer used by SUTIME, non-whole numbers are not recognized. Consequently, SUTIME cannot correctly interpret temporal expressions such as *a year and a half ago*.
- Patterns are language specific. SUTIME is limited to extracting temporal expressions from English text. In order to recognize temporal expressions in other languages, separate rules will need to be developed.

3. Evaluation

We evaluated SUTIME's performance on TempEval-2 Task A (Verhagen et al., 2010), which consists of two parts: identifying the extents of a temporal expression and then providing the correct TIMEX3 type and value attributes for each recognized expression. For the evaluation of extents, token-based precision, recall, and F_1 are used. For the evaluation of attributes, only tokens that are correctly identified as being part of a temporal expression are considered. The official evaluation for TempEval-2 includes the percentage of correct guesses for both the type and the value attributes.

3.1. Other systems

We compare SUTIME's performance with several other systems for the TempEval-2 Task A in English. Table 1 gives the results for all systems on the evaluation set.

- GUTime (Mani, 2004):[6] a Perl temporal tagger provided by Georgetown University, part of the TARSQI toolkit (Verhagen and Pustejovsky, 2008) for temporal processing. Although GUTime generates TIMEX3 annotations, the format is not the same as that used in TempEval-2. GUTime is an extension of the TempEx tagger (Mani and Wilson, 2000), which is targeted for ACE TIMEX2. Annotations provided by GUTime incorporate some TimeML TIMEX3 extensions. We use simple rules to map from the output of GUTime to a format that is compatible with the TempEval-2 scorer.
- HeidelTime (Strötgen and Gertz, 2010):[7] the best-performing system from SemEval-2 — also a rule-based system. HeidelTime1 is optimized for precision, while HeidelTime2 is tuned for recall. HeidelTime* corresponds to results from the publicly available version of HeidelTime (from December 2011).
- TRIPS/TRIOS (UzZaman and Allen, 2011):[8] the second best system from SemEval-2, which uses a conditional random field (CRF) for recognizing temporal expressions, and a rule-based system for normalizing temporal expressions.

3.2. Discussion

Experimental results show that both the rule-based system (HeidelTime) and the probabilistic system (TRIPS/TRIOS) were as effective (with similar F_1 scores) for identifying temporal expressions. This validates our intuition that most temporal patterns can be captured effectively with rules.

	Extents			*Attribute*	
System	P	R	F_1	type	value
GUTime	0.89	0.79	0.84	0.95	0.68
SUTime	0.88	**0.96**	**0.92**	0.96	0.82
TRIPS/TRIOS	0.85	0.85	0.85	0.94	0.76
HeidelTime1	**0.90**	0.82	0.86	**0.96**	**0.85**
HeidelTime2	0.82	0.91	0.86	0.92	0.77
HeidelTime*	0.57	0.89	0.70	0.96	0.85

Table 1: TempEval-2; English evaluation set.

SUTIME has the highest overall F_1 and the highest recall in discovering temporal expressions. Compared to HeidelTime2, the high recall system, SUTIME has both higher precision and higher accuracy for the attribute type and value. However, compared to HeidelTime1, SUTIME has a lower precision and lower accuracy for the attribute type and value. Note that using the attribute scores of the official TempEval-2 scorer to compare the systems can be misleading since the attributes are scored only for tokens correctly identified as belonging to a temporal expression. Because the attribute scores are computed using a total that is different for each system, it is inappropriate to compare the attribute scores across systems.

For example, consider a system that only marks one token as belonging to a temporal expression. Assuming that the token had the correct type and attribute, the system would achieve an accuracy of 100% on the type and value attributes (although the F_1 of identified extents would be extremely low). From this example, we see that is it difficult to only compare the attribute scores without considering the scores for identifying extents.

3.3. Revised Scoring of Attributes

To address the attribute scoring problem, we compare the systems using a modified attribute scoring method where we compute the precision, recall, and F_1 for each attribute based on the number of temporal expressions with the correct attribute, the total number of temporal expressions in the gold, and the total number of temporal expressions in the system response. Table 2 gives the revised attribute scores based on the following formulas:

$$P_{\text{attr}} = \#\,\text{Correct}_{\text{attr}}/\#\,\text{Mentions}_{\text{resp}}$$
$$R_{\text{attr}} = \#\,\text{Correct}_{\text{attr}}/\#\,\text{Mentions}_{\text{gold}}$$
$$F_{1\,\text{attr}} = 2P_{\text{attr}}R_{\text{attr}}/(P_{\text{attr}} + R_{\text{attr}})$$

The number correct is determined by matching each response mention against a gold mention and checking if the attribute matches.

Using the revised attribute scores, SUTIME has the highest recall and F_1 for both the type and value attributes on the English evaluation set. HeidelTime has the best precision. Of the systems listed, GUTime has the lowest attribute value scores, which could be partially due to the incomplete coverage of our rules for converting its output to match the TempEval-2 format.

[6] timeml.org/site/tarsqi/toolkit/index.html
[7] dbs.ifi.uni-heidelberg.de/index.php?id=129
[8] www.cs.rochester.edu/u/naushad/temporal

	Type			*Value*		
System	P	R	F_1	P	R	F_1
GUTime	0.85	0.79	0.82	0.59	0.55	0.57
SUTime	0.84	**0.94**	**0.89**	0.71	**0.78**	**0.74**
HeidelTime*	**0.87**	0.89	0.88	**0.76**	0.77	0.76

Table 2: Revised attribute scores for English evaluation set.

3.4. Error Analysis

We performed error analysis on the TempEval-2 training set. Performance of SUTIME on this training set, using the official scorer, is given in Table 3. The revised attribute scores are given in Table 4.

	Extents			*Attribute*	
System	P	R	F_1	type	value
GUTime	**0.88**	0.71	0.79	0.92	0.67
SUTime	0.87	**0.90**	**0.89**	0.92	0.77
HeidelTime*	0.58	0.82	0.68	**0.96**	**0.85**

Table 3: TempEval-2; English training set.

	Type			*Value*		
System	P	R	F_1	P	R	F_1
GUTime	0.79	0.71	0.75	0.57	0.51	0.54
SUTime	0.78	**0.84**	0.81	0.65	0.70	0.67
HeidelTime*	**0.84**	**0.84**	**0.84**	**0.73**	**0.74**	**0.73**

Table 4: Revised attribute scores for English training set.

Below, we give some examples of the types of errors that SUTIME makes.

True errors made by SUTIME. Most of the errors are due to the limitations of SUTIME, as noted before. Since SUTIME cannot recognize fractions such as *one half*, for the phrase *a minute and a half*, instead of marking the entire phrase as a temporal expression with a normalized value of **PT1M30S**, it only recognizes *a minute* as a temporal expression and gives it a normalized value of **PT1M**.

In another case, SUTIME has trouble with ambiguous words. For instance, in the phrase *...where Orangemen march...*, SUTIME incorrectly identifies *march* as referring to the month March. In addition, SUTIME will always try to resolve *tomorrow* literally to the day after the document date, even in phrases such as *the people of tomorrow*. Also, despite its best efforts, SUTIME often makes mistakes when resolving a relative date. Given a reference date of **1990-08-16**, SUTIME has trouble determining whether *September* should refer to **1989-09** or **1990-09**. Some other examples of temporal expressions that are difficult for SUTIME to resolve correctly are given below:

Expression	*SUTime*	*Correct*
a year ago	1988-10-27	1988-Q3
the full year	P1Y	1989
more than two thousand years	P2000Y	P2L
months ago	1989-09-30	PXM

Despite its high recall, there are still many phrases that SUTIME misses, such as *the latest period*, which happened to refer to **1989-Q3**. This particular case also requires more advanced semantic understanding to be able to recognize and infer that the *period* being referred to is a **Quarter**, and the last one that occurred is **1989-Q3**.

Different normalization. In some cases, the normalization selected by SUTIME was different from that of the gold annotation, but the two could be regarded as equivalent.

Expression	*SUTime*	*Correct*
the last twenty four hours	PT24H	P1D
the late 1970s	197X	197
more than two thousand years	P2000Y	P2L

In the TempEval-2 annotation, **Q** is used for **Quarters**, **E** for decades, **C** for centuries, and **L** for millennia. Since these are non-standard, they are not used by SUTIME.

Temporal expression not marked by annotators. In some cases, our system actually recognized many temporal expressions that were not marked by annotators.

Expression	*Type*	*Value*
annual	SET	P1Y
each July	SET	XXXX-07
20th century	DATE	20XX

Errors in annotations. In a few rare cases, the disagreement appears to have been due to errors in annotation.

Expression	*SUTime*	*Correct*
a few minutes	PTXM	PXM
July last year	1997-07	1997-06

We would like to note that document **ed980111.1130.0089** appears to have an incorrect reference date of **2010-03-24** (the actual date should be **1998-01-11**), causing SUTIME to evaluate incorrect values for *Yesterday* and *today*.

Imprecise annotation. The week number is often not marked by annotators, whereas SUTIME will typically attempt to guess a week.

Expression	*SUTime*	*Correct*
this week	1989-W43	1989-WXX

Type Mistakes. SUTIME has difficulty distinguishing when an expression should be marked as a **Duration** versus a **Date**. There also appear to be problems with annotators often marking temporal expressions for a **Date** as **Time**.

Expression	*SUTime*	*Correct*
[in] five years	DURATION	DATE
January	DATE	TIME
third-quarter	DATE	TIME

4. Conclusion

We presented SUTIME, a temporal tagger which provides a practical and extensible state-of-the-art system for extracting time expressions. SUTime can be used as a basic component for building temporally aware systems and for investigating problems requiring temporal information, such as event extraction, temporal ordering of events, and question answering. By incorporating SUTIME in Stanford CoreNLP, we provided easy access to temporal information alongside other levels of NLP annotation. SUTIME can also serve as a strong baseline for comparing future temporal taggers.

5. Acknowledgements

We gratefully acknowledge the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government. Angel X. Chang is supported by a SAP Stanford Graduate Fellowship. We thank Valentin I. Spitkovsky, Gabor Angeli, Manolis Savva and the anonymous reviewers for helpful comments on draft versions of this paper.

6. References

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370.

Jerry R. Hobbs, Douglas E. Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. 1997. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. pages 383–406.

Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 69–76.

Inderjeet Mani. 2004. Recent developments in temporal information extraction. In *Proceedings of RANLP03*, pages 45–60. John Benjamins.

James Pustejovsky, Jos Castao, Robert Ingria, Roser Saur, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003. TimeML: Robust specification of event and temporal expressions in text. In *in Fifth International Workshop on Computational Semantics (IWCS-5*.

Jannik Strötgen and Michael Gertz. 2010. HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324.

Naushad UzZaman and James F. Allen. 2011. Event and temporal expression extraction from raw text: first step towards a temporally aware system. *International Journal of Semantic Computing*.

Marc Verhagen and James Pustejovsky. 2008. Temporal processing with the TARSQI toolkit. Coling 2008: Companion volume: Demonstrations.

Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 57–62. Association for Computational Linguistics.