Lang Resources & Evaluation (2013) 47:269–298
DOI 10.1007/s10579-012-9179-y

ORIGINAL PAPER

Multilingual and cross-domain temporal tagging

Jannik Strötgen · Michael Gertz

Published online: 8 May 2012

Abstract Extraction and normalization of temporal expressions from documents are important steps towards deep text understanding and a prerequisite for many NLP tasks such as information extraction, question answering, and document summarization. There are different ways to express (the same) temporal information in documents. However, after identifying temporal expressions, they can be normalized according to some standard format. This allows the usage of temporal information in a term- and language-independent way. In this paper, we describe the challenges of temporal tagging in different domains, give an overview of existing annotated corpora, and survey existing approaches for temporal tagging. Finally, we present our publicly available temporal tagger HeidelTime, which is easily extensible to further languages due to its strict separation of source code and language resources like patterns and rules. We present a broad evaluation on multiple languages and domains on existing corpora as well as on a newly created corpus for a language/domain combination for which no annotated corpus has been available so far.

Keywords Temporal information · Temporal tagger · Named entity recognition · Named entity normalization · TIMEX2 · TIMEX3

1 Introduction

Temporal information is prevalent in many kinds of documents, and its extraction and normalization from documents are important preprocessing steps for many natural language processing and understanding tasks. For example, in information

J. Strötgen · M. Gertz (✉)
Institute of Computer Science, Heidelberg University, Heidelberg, Germany
e-mail: gertz@informatik.uni-heidelberg.de

J. Strötgen
e-mail: stroetgen@informatik.uni-heidelberg.de

retrieval, temporal information can be used, among others, for temporal clustering of documents along timelines and querying a document collection using temporal constraints. Alonso et al. (2007, 2011) give an overview of the value of temporal information and discuss current research trends in temporal information retrieval. In research areas requiring rich natural language understanding, such as information extraction, document summarization, and question answering, temporal information is often utilized as well. For example, in topic detection and tracking, it helps to identify new unreported events and to assign documents to already detected events (see, e.g., Allan 2002; Makkonen et al. 2003).

Independent of the specific task, all applications using temporal information mentioned in text documents rely on high quality temporal taggers, which extract temporal expressions from documents and normalize them. Due to its importance for many tasks, temporal tagging has become an active research field over the past few years. This resulted in the development of standards for temporal annotation such as TIDES TIMEX2 (Ferro et al. 2005) and the markup language TimeML (Pustejovsky et al. 2003a), and also in the creation of annotated corpora, e.g., the TimeBank corpus (Pustejovsky et al. 2003b). In addition, several temporal taggers were developed and competitions were organized where temporal taggers were evaluated. However, many temporal taggers are adapted to a specific domain (the news domain) and support only one language (often English).

In this paper, we present our temporal tagger, called HeidelTime, which achieves high quality results for the extraction and the normalization of temporal expressions not only in news documents but also in narrative-style documents. In addition, HeidelTime is developed as a rule-based system with a strict separation between the source code and all resources such as patterns, normalization information, and rules. This allows the simple development of resources for additional languages. In addition, due to the modular structure of the resources, HeidelTime can be extended with task-specific modules. This combination of domain-independence, multilinguality, extensibility, modifiability, and the high quality of both the extraction and the normalization of temporal expressions is what distinguishes HeidelTime from other temporal taggers. So far, we developed resources for English and German[1] and, in this paper, we detail evaluation results for both languages. For this, we used publicly available corpora and developed a German corpus. This corpus as well as HeidelTime itself and several additional tools, e.g., for corpora preparation and evaluation, are made available to the public.[2]

The remainder of the paper is structured as follows: In Sect. 2, after a discussion of the different types of temporal expressions and how they occur in documents, we discuss different annotation standards. In Sect. 3, we give an overview of time-annotated corpora and briefly describe the creation of our new corpus. In addition, we survey temporal taggers by presenting their methods. Then, in Sect. 4, HeidelTime and its system architecture are presented, and we show how

[1] Recently, we were able to add resources for Dutch. These were developed and kindly provided by Matje van de Camp (Tilburg University).

[2] HeidelTime, the German corpus as well as additional scripts and components are publicly available at http://dbs.ifi.uni-heidelberg.de/heideltime/. Thus, all our evaluation results are reproducible.

HeidelTime is integrated into our text mining pipeline. The evaluation results on different corpora and languages are presented in Sect. 5. Finally, we conclude our paper and describe ongoing work in Sect. 6.

2 Temporal information in documents

In many types of text documents, there is a lot of temporal information. The most simple type of temporal information is the document creation time or the time when the document was last modified. This information is usually directly accessible through the metadata of a document. In addition, in several domains, documents often contain many temporal expressions directly in the text. For example, news documents and Wikipedia articles typically contain several temporal expressions. In the context of our work on event-centric document similarity (Strötgen et al. 2011), we analyzed the English Wikipedia featured articles[3] and the cross-language linked German ones with respect to the contained temporal expressions. We chose these documents because they are grouped into categories allowing a category-based analysis. As the numbers in Table 1 show, temporal expressions are frequent in all kinds of Wikipedia articles with documents of categories such as Biographies, Education, and Warfare containing many more temporal expressions than documents about, e.g., Video gaming and Biology, which still contain about 39 temporal expressions on average in English language articles.

The temporal expressions in the text of documents can describe different temporal phenomena such as a point in time or a time interval. In Sect. 2.1, we give an overview of these different types of temporal expressions and illustrate them using some examples. In Sect. 2.2, we detail possible linguistic realizations of temporal expressions and point out the different challenges for their extraction and normalization. Finally, we present different standards for the annotation of temporal expressions in Sect. 2.3.

2.1 Types of temporal expressions

Following TimeML (Pustejovsky et al. 2005), the standard markup language for temporal annotation, we group temporal expressions into four categories: date, time, duration, and set. *Time* and *date* expressions (e.g., "11 a.m." or "July 29, 2003") can be placed on a timeline—although at different granularities. Exceptions are generically used expressions and vague expressions, e.g., "several days later". Here, one can only specify the temporal relation to a reference time if this is known. While expressions of the type *duration* are used to inform about the length of an interval (e.g., "three years" in "They have been traveling through Europe for three years"), expressions of the type *set* provide information about the periodical aspect of an event (e.g., "twice a month" in "They go out for lunch twice a month"). Note that durations and sets may additionally be anchored to a specific point in time (Pustejovsky et al. 2005).

[3] http://en.wikipedia.org/wiki/Wikipedia:Featured_articles.

Table 1 Average number of temporal expressions per document of the English Wikipedia featured articles grouped by category. The numbers in parentheses are for the German cross-language linked documents

Category	Timexes
Biographies	166.0 (237.2)
Education	148.6 (20.1)
Sport and recreation	148.3 (161.2)
Geography and places	138.9 (78.0)
Warfare	130.0 (214.9)
History	119.2 (67.4)
Business, economics, and finance	109.9 (52.3)
Culture and society	104.2 (41.7)
Law	96.5 (22.4)
Politics and government	95.3 (47.3)
Religion, mysticism, and mythology	90.4 (48.6)
Transport	88.8 (44.5)
Art, architecture, and archaeology	85.3 (28.8)
Royalty, nobility, and heraldry	84.0 (73.0)
Mathematics	82.1 (23.0)
Engineering and technology	80.4 (42.9)
Awards, decorations, and vexillology	80.1 (34.7)
Literature and theatre	80.0 (31.9)
Meteorology	73.9 (90.9)
Media	68.5 (35.0)
Music	66.7 (23.1)
Physics and astronomy	65.1 (43.4)
Language and linguistics	64.2 (32.4)
Health and medicine	57.0 (26.2)
Geology and geophysics	55.3 (29.4)
Food and drink	54.7 (66.8)
Philosophy and psychology	54.7 (16.7)
Computing	54.2 (27.4)
Animals	53.1 (23.1)
Chemistry and mineralogy	43.8 (18.6)
Video gaming	39.3 (26.3)
Biology	39.1 (20.2)

Due to the fact that temporal expressions can be normalized, a standardized value can be associated with each temporal expression. Thus, regardless of the used term or language, every temporal expression referring to the same point in time or carrying the same meaning can be normalized to the same value in some kind of standard format such as the ISO 8601 standard for temporal information. For example, assuming a document's creation time is February 27, 2011 (2011-02-27 according to ISO 8601), all the expressions "the day before yesterday", "two days ago", "last Friday", or "February 25th" refer to the same point in time and thus can be normalized to 2011-02-25. These examples show that temporal semantics,

e.g., referring to a specific point in time, can be expressed in many different ways, an aspect we will describe in more detail in the next section.

2.2 Realizations of temporal expressions

There are many ways to realize temporal expressions in text documents. The way of realization directly influences the level of difficulty for the normalization of a temporal expression. According to Pustejovsky et al. (2003a), there are three major types of temporal expressions: (1) fully specified temporal expressions, (2) underspecified temporal expressions, and (3) durations. However, since we already use the categories date, time, duration, and set for temporal expressions, we prefer to further distinguish similar to Schilder and Habel (2001) and according to our previous work (Alonso et al. 2007; Strötgen et al. 2010) between explicit, implicit, and relative expressions.

- *Explicit expressions*: All expressions that are fully specified and can thus be normalized without any further knowledge are defined as explicit temporal expressions. In Fig. 1, explicit temporal expressions are marked with transparent boxes. Note that the granularity of the expression does not matter. For example, the expressions of the granularity day "May 22, 1995" and of granularity month "April 1985" can be normalized directly to "1995-05-22" and "1985-04", respectively.
- *Implicit expressions*: Implicit temporal expressions can be normalized once their implicit temporal semantics is known. Examples are names of holidays and events that can directly be associated with a point or interval in time. Whether such implicit expressions are identified and correctly normalized by a temporal tagger highly depends on the underlying knowledge base. A simple implicit expression is "Christmas 2005" since Christmas refers to December 25. Thus, it can be normalized to "2005-12-25". A more complex example is "Labor day

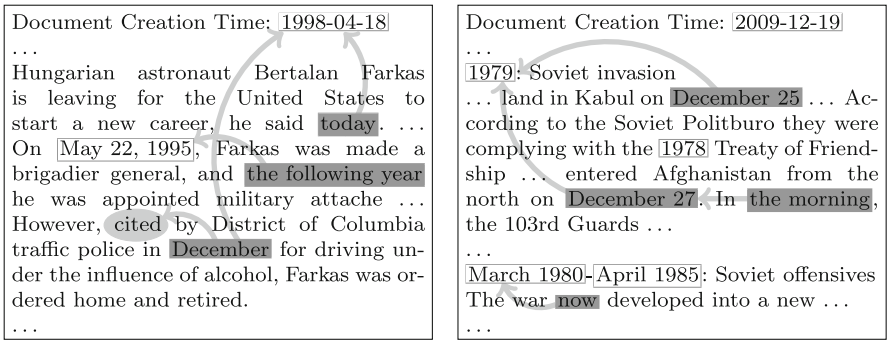

Fig. 1 Examples of temporal expressions in a news document from the TimeBank corpus (**a**) and in a narrative document from the WikiWars corpus (**b**) with explicit (*transparent boxes*) and relative (*solid boxes*) expressions. Arrows indicate what kind of context information is needed to normalize the temporal expressions and thus point to the reference times. In one case, the tense of the sentence is needed, which is indicated by the arrow "December" to "cited"

2009", which can be normalized to "2009-09-07" if the temporal tagger knows that Labor Day can always be mapped to the first Monday in September; this can then be calculated for the specific year, in this case 2009. An example of an event specified as an implicit temporal expression is "soccer world cup final 2010", which took place on July 11, 2010. Of course, the knowledge base of a temporal tagger is extensible with respect to such events in an almost infinite manner. For this, it is important that a temporal tagger can easily be extended.

- *Relative expressions*: The main characteristic of relative expressions is that they cannot be normalized without further context information. In Fig. 1, relative expressions are marked with solid boxes. For expressions such as "today" and "the following year" but also for underspecified expressions such as "December" or "December 25", a reference time has to be known to anchor the expression. This reference time can either be the document creation time (e.g., for "today"), or another temporal expression in the document (e.g., for "the following year"). Sometimes, also the temporal relation to the reference time has to be known. For instance, in Fig. 1a, "December" cannot be normalized without knowing the relationship to the reference time. In such cases, often the tense of the sentence can be used to determine this relationship. That is, usually, present and future tense refer to an upcoming point in time while past tense refers to a previous point in time. In the example, the tense is past tense ("cited"), and thus "December" is normalized to "1997–12".

Independent of the type of an expression, the normalization task of a temporal tagger is to assign the same value to all expressions carrying the same semantics or referring to the same point in time. While this is straightforward in some cases, it is a difficult task in other cases. For example, in Fig. 1b, identifying "1979" and not "1978" as reference time for "December 27" is challenging since both expressions are explicit and of the granularity type year. In addition, different strategies have to be developed for news and narrative documents. For example, although "December" (Fig. 1a) and "December 25" (Fig. 1b) occur in a similar way, in the first case, the reference time is the document creation time, and in the second case, it is a previously mentioned temporal expression in the document. We will discuss these challenges, with a particular focus on the difficulties of determining the reference time, in Sect. 4.1 when presenting HeidelTime's approach for normalizing temporal expressions. In the next section, we present the two most popular annotation standards for temporal expressions in documents.

2.3 Annotating temporal expressions

Currently, there are two annotation standards used for annotating temporal expressions in documents: The TIDES TIMEX2 standard[4] and TimeML,[5] a specification language for temporal annotation containing TIMEX3 tags for

[4] http://fofoca.mitre.org/.

[5] http://www.timeml.org/.

temporal expressions. Both standards present guidelines for the annotation of temporal expressions, including how to determine the extents of expressions and their normalizations. In both cases, the normalization is defined according to the ISO 8601 standard for temporal information with some extensions.

The TIDES annotation guidelines are based on the principles that temporal expressions should be tagged "if a human can determine a value for [it]", and that the value "must be based on evidence internal to the document" (Ferro et al. 2001). For the normalization, the TIMEX2 tag may contain the following attributes (Ferro et al. 2005):

- VAL: a normalized form of the date/time
- MOD: captures temporal modifiers
- ANCHOR_VAL: a normalized form of an anchoring date/time
- ANCHOR_DIR: the relative direction between VAL and ANCHOR_VAL
- SET: identifies expressions denoting sets of times

TimeML is based on the TIDES standard and was developed to capture further types of temporal information in documents: events, relationships between events and temporal expressions, and relationships between two events. For this, in addition to a tag for temporal expressions (TIMEX3), TimeML contains tags for annotating events, temporal links, and temporal signals (Pustejovsky et al. 2003a, 2005). Due to this extension of annotating temporal information, there are significant changes between TIMEX2 and TIMEX3. These changes affect the attribute structure as well as the exact use. For example, specific types of pre- and post-modifications of temporal expressions are part of TIMEX2 tags while in TimeML, they are outside TIMEX3 tags. Such constructs are handled using the newly introduced tags for annotating relations between temporal expressions and events. In addition, TIMEX3 tags cannot be nested. However, TIMEX3 tags with no extent are introduced, e.g., to deal with unspecified time points, which are needed to anchor durations. The most important attributes of TIMEX3 tags are[6]:

- *type*: defines whether the expression is of type date, time, duration, or set
- *value*: a normalized form of the expression
- *mod*: captures temporal modifiers
- *quant*: specifies the quantity of set expressions
- *freq*: specifies the frequency of set expressions
- *beginpoint*: anchor begin of a duration
- *endpoint*: anchor end of a duration

While the attribute *type* is newly introduced in TIMEX3, the attributes *value* and *mod* are similar to the VAL and MOD attributes of TIMEX2. These two attributes already capture a large part of the information of temporal expressions, and for many expressions, the *value* attribute is the only attribute that is needed for normalization. Although the different attributes and definitions of extents between

[6] The details of the attributes are described in the TimeML annotation guidelines including further attributes, e.g., to capture the function of a temporal expression in a document. For details, see http://www.timeml.org/.

TIMEX2 and TIMEX3 are significant, the annotations for many temporal expressions are often very similar so that an automated conversion often works reasonably well (see, e.g., Saquete Boro 2010).

3 Literature review

In the last few years, there have been significant efforts in developing approaches to annotate temporal information in documents, which led to the annotation standards of TIMEX2 and TIMEX3 as described in the previous section. In addition to annotation guidelines, these efforts resulted in a couple of annotated corpora, and some challenges were organized to evaluate temporal taggers on these corpora. There were some early contests with tasks on temporal information extraction without normalization, e.g., the MUC (Message Understanding Conference) named entity recognition tasks in 1995 and 1997 (Grishman and Sundheim 1995; Chinchor 1997). In Sect. 3.1, we give an overview of corpora annotated with respect to the annotation standards of TIMEX2 or TIMEX3 and competitions based on these corpora. In Sect. 3.2, we survey existing temporal taggers by describing their methods for the extraction and normalization tasks. Often, they are evaluated on the presented corpora so that we are able to compare their quality.

3.1 Time-annotated corpora

There are a couple of corpora annotated with TIMEX2 or TIMEX3 tags. Based on the TIDES TIMEX2 annotation standard, the corpora for the ACE (Automatic Content Evaluation) time expression and normalization (TERN) contests in 2004, 2005, and 2007 were created.[7] Although all ACE corpora are annotated using TIMEX2 tags, different versions of the annotation guidelines were used. The changes, however, are not significant. The ACE TERN 2004 training data as well as the evaluation data released by the Linguistic Data Consortium (LDC) consist of English news documents. The documents of the ACE 2005 and 2007 corpora are not all from the news domain, but additionally contain conversations, discussions, and Web blogs. Unfortunately, only the ACE 2005 training corpus has officially been released, yet. For the evaluation of temporal taggers, a script is provided to measure precision, recall, and f-score for the extraction and normalization tasks.[8] Details about the evaluation measures used in temporal tagging are given in Sect. 5.

The TimeBank corpus was developed during the workshop Time and Event Recognition for Question Answering Systems (TERQAS) in 2002 as a reference corpus for TimeML (Pustejovsky et al. 2003b). Thus, TimeBank contains TIMEX3 tags for temporal expressions, and events and temporal relations are also annotated. The TimeBank 1.2 version released by LDC[9] consists of 183 news documents.

[7] The 2004 and 2005 training sets and the 2004 evaluation set are released by LDC (LDC2005T07, LDC2006T06, LDC2010T18); see: http://www.ldc.upenn.edu/.

[8] The TERN evaluation script is available at http://fofoca.mitre.org/tern.html.

[9] TimeBank 1.2 is released by LDC (LDC2006T08); see: http://www.ldc.upenn.edu/.

For the TempEval-2 competition with tasks on temporal expression, event, and relation extraction and normalization, TimeBank was used as the underlying corpus for the English training data. The evaluation data consist of newly annotated documents. In addition, corpora for Spanish, Italian, French, Korean, and Chinese were developed and are publicly available together with the used annotation guidelines and a script for evaluation.[10]

The ACE corpora as well as the TimeBank and TempEval-2 corpora consist of news and news-style documents for which the document creation time plays an important role. These documents are relatively short and contain only a few temporal expressions. Thus, the temporal discourse structure is very limited. These facts motivated Mazur and Dale (2010) to create a corpus of narratives containing more complex temporal phenomena. They developed WikiWars,[11] a corpus of 22 documents containing parts of Wikipedia articles about famous wars in history. Temporal expressions are annotated according to the TIDES TIMEX2 standard, and the corpus is created in the ACE TERN style so that the ACE TERN evaluation scripts can be used for evaluation.

The WikiWars corpus is the only publicly available, temporal annotated corpus containing narratives. To be able to evaluate our multilingual temporal tagger (Sect. 4) in more than one language and not only on news documents but also on narratives, we developed the WikiWarsDE corpus (Strötgen and Gertz 2011).[12] It contains parts of the 22 German Wikipedia articles about the same famous wars as the English corpus. For the annotation, we followed the suggestions of the WikiWars developers. Thus, we used the annotation tool Callisto,[13] annotated the corpus according to the TIDES TIMEX2 standard, and published the corpus in the same style as the WikiWars corpus and the ACE corpora. Some statistics of WikiWarsDE as well as of other publicly available corpora are shown in Table 2. Compared to the other corpora, WikiWars and WikiWarsDE contain long documents with many temporal expressions and thus a complex temporal discourse structure.

3.2 Temporal taggers

As described in Sect. 2, the task of temporal tagging can be split into two subtasks, the extraction and the normalization of temporal expressions. The extraction task is to correctly identify temporal expressions and their boundaries. It can thus be seen as a typical classification problem of deciding whether a token is part of a temporal expression or not. For this, approaches range from rule-based to machine learning strategies. The normalization of temporal expressions is to assign temporal expressions a value in some standard format and thus it is a more challenging

[10] The TempEval-2 data are available at http://timeml.org/site/timebank/timebank.html. While TempEval-2 had a task for the extraction and normalization of temporal expressions, the first TempEval evaluation challenge concentrated on tasks for identifying temporal relations. Thus, we do not consider the corpus of the first TempEval here.

[11] WikiWars is available at http://www.timexportal.info/wikiwars/.

[12] WikiWarsDE is publicly available at http://dbs.ifi.uni-heidelberg.de/heideltime/.

[13] http://callisto.mitre.org/.

Table 2 Statistics of the WikiWarsDE corpus and other publicly available or released corpora. Tokens/Timex and Timex/Document of TempEval-2 corpora for other languages are similar to English (en) and Spanish (es) TempEval-2 corpora

Corpus	Docs	Tokens	Timex	Tokens/Timex	Timex/Document
ACE04 en train	863	306,463	8,938	34.3	10.36
ACE04 en eval	192	54,614	1,828	29.9	9.52
ACE05 en train	599	259,889	5,469	47.5	9.13
TimeBank 1.2	183	78,444	1,414	55.5	7.73
TempEval2 en train	162	53,450	1,052	50.8	6.49
TempEval2 en eval	9	4,849	81	59.9	9.00
TempEval2 es train	173	58,423	1,093	53.5	6.32
TempEval2 es eval	17	4,278	91	47.0	5.35
WikiWars	22	119,468	2,671	44.7	121.41
WikiWarsDE	22	100,699	2,240	45.0	101.81

and complex task as described in Sect. 2.2. This task is addressed by almost all temporal taggers in a rule-based way. In summary, existing temporal taggers use either a combination of machine learning and rule-based techniques for the extraction and normalization of temporal expressions or solely rule-based methods.

One of the first temporal taggers is TempEx (Mani and Wilson 2000). It is a simple, rule-based system that uses TIMEX2 tags, although the normalization functionality is limited. Based on this temporal tagger, GUTime was developed as reference tool for TimeML using TIMEX3 tags.[14] GUTime is one of the most widely used temporal taggers. It is part of the TARSQI toolkit consisting of components for the extraction of events, temporal expressions, and temporal relationships (Verhagen and Pustejovsky 2008). GUTime has been evaluated on the TERN 2004 training data and achieves competitive results. We detail the evaluation results of all temporal taggers introduced in this section and compare their quality with our temporal tagger HeidelTime in Sect. 5 since they are often evaluated on different corpora and with respect to different evaluation measures.

The best performing system of the ACE TERN 2004 competition was Chronos (Negri and Marseglia 2005). It uses a relatively large rule set containing 1,000 hand-crafted rules to capture information needed for normalization. All systems of the TERN 2004 competition performing the extraction and the normalization are rule-based while all systems performing no normalization are machine-learning ones, e.g., the ATEL system, which uses SVM classifiers (Hacioglu et al. 2005). Motivated by this observation, Ahn et al. (2005a, b) raise the question whether decoupling recognition from normalization may improve a temporal tagger's quality for extraction and normalization and show that decoupling might help. In addition, it is shown that one may split the normalization task into smaller subtasks and

[14] http://timeml.org/site/tarsqi/modules/gutime/index.html.

address some of them with machine learning techniques using a cascaded approach while only a smaller set of composition rules has to be applied (Ahn et al. 2007).

Another temporal tagger separating the tasks of extraction and normalization is the DANTE tagger (Mazur and Dale 2009). The extraction is done using a JAPE grammar (Java Annotation Pattern Engine), and the normalization is performed in a rule-based manner. The system annotates temporal expressions according to the TIMEX2 guidelines and was one of the systems that participated in the ACE 2007 competition where it achieved competitive results. The developers of DANTE point out the challenges of normalizing temporal expressions when processing narratives instead of news documents. They developed the first temporal annotated corpus containing narratives, as described in the previous section (Mazur and Dale 2010).

Recently, the participants of TempEval-2 developed several temporal taggers using the TIMEX3 annotation standard. Eight teams submitted results for temporal tagging in English and three teams for Spanish. The other languages were not addressed by any of the participating teams. For example, Saquete Boro (2010) used the TERSEO tagger and a transducer to translate its TIMEX2 tags into TIMEX3 tags. This method shows that such a transformation works reasonably well. In addition to the TempEval-2 results, UzZaman and Allen (2011) evaluate their temporal tagger, which is based on conditional random fields, on the TimeBank corpus and compare their results with Boguraev and Ando (2005) and Kolomiyets and Moens (2009). While the former system is based on a cascaded finite-state grammar, the latter uses a maximum entropy classifier. We present these evaluation results in Sect. 5 together with our results on the different corpora.

Another important point in temporal tagging is multilinguality. Negri et al. (2006) show that one can use automatic translations of extraction rules from one language into another. These are then assigned to a language-independent normalization format. Finally, the normalization can be performed for all languages in the same way. Recently, a semi-automated translation of a TimeML annotated corpus into another language was proposed by Costa and Branco (2010). They show that fast porting with relatively small manual effort is possible. The annotated corpus in a new language can then be used as training data for machine learning approaches.

4 HeidelTime

While there are a couple of temporal taggers as described in the previous section, there is a lack of publicly available temporal taggers, which can be used for processing different languages and domains with high accuracy in both the extraction and the normalization of temporal expressions. Thus, for our research on multilingual temporal information extraction and exploration (Strötgen et al. 2010; Strötgen and Gertz 2010b), we developed HeidelTime, a temporal tagger satisfying the following requirements:

A. Extraction and normalization should be of high quality.
B. High quality results should be achieved across domains.
C. Further languages should be integrable without modifying the source code.

D. The architecture should allow the integration of new modules, e.g., for additional implicit expressions.
E. When needed, adding and modifying rules should be simple.

Although there are some promising machine learning approaches for the extraction of temporal expressions, we developed HeidelTime as a rule-based system for the following reasons: (1) the divergence of temporal expressions is very limited compared to other named entity recognition and normalization tasks, e.g., the number of persons and organizations as well as the variety of names referring to these entities are probably infinite, (2) the normalization is hardly solvable without using rules, (3) resources for additional languages can be added without the need of an annotated corpus, and (4) the knowledge base can be extended in a modular way, e.g., for adding events and their temporal information such as "soccer world cup final 2010" that took place on July 11, 2010. Furthermore, for the ability to easily add and modify rules (req. E), we developed a well-defined rule syntax (see Sect. 4.1.2). As annotation format, HeidelTime uses the TimeML annotation standard of TIMEX3 tags for temporal expressions. Nevertheless, due to the similarities between TIMEX3 and TIMEX2, the tags can be converted into TIMEX2 as well—although not all attributes are supported. Similar to the transformation from TIMEX2 to TIMEX3 described by Saquete Boro (2010), though the other way around, we used this property to be able to evaluate HeidelTime on corpora annotated with TIMEX2.

As a first official evaluation, we participated in the TempEval-2 task of extracting and normalizing English temporal expressions. HeidelTime achieved the best results for both the extraction and the normalization task (English) (Strötgen and Gertz 2010a). Although detailed evaluation results are presented later in Sect. 5, this verifies that requirement A is satisfied. How the remaining requirements B, C, and D are met is explained in Sect. 4.1 when describing HeidelTime's system architecture. Then, in Sect. 4.2, we present our UIMA-based text mining pipeline of which HeidelTime is one component. Finally, in Sect. 4.3, we detail how HeidelTime's language resources were developed.

4.1 HeidelTime's architecture

An overview of HeidelTime's system architecture is given in Fig. 2. The most important feature is the strict separation between the algorithmic part, i.e., the source code, and the resources for patterns, rules, and normalization information. The resources are organized in a modular way. When new resources are added to HeidelTime, they are automatically loaded by HeidelTime whenever they are named and built according to HeidelTime's conventions. Thus, the requirement of extensibility is satisfied (req. D). In addition, only the resources are language-dependent. Thus, when integrating a new language, only these have to be adapted (req. C).[15]

As detailed in Sect. 3.2, temporal tagging can be split into two tasks, the extraction and the normalization of temporal expressions. For the extraction, HeidelTime mainly uses regular expressions that can make use of pattern resources. However,

[15] Due to this feature, we were able to include Dutch language resources recently developed at Tilburg University, see, http://dbs.ifi.uni-heidelberg.de/heideltime/.

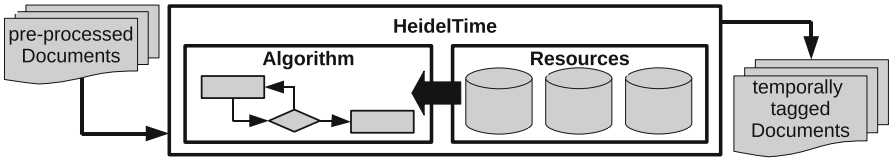

Fig. 2 HeidelTime's system architecture with algorithm (source code) and resources

other constraints can be set as well, e.g., the part-of-speech tag of a specific token in the expression itself or before or after the temporal expression. For the normalization, we use normalization resources containing mappings between an expression and its value in standard format. Furthermore, linguistic clues are applied to normalize ambiguous expressions. For example, the tense of a sentence may indicate the temporal relation between an expression and its reference time. The difficulties of normalizing temporal expressions in different domains were described in Sect. 2.2. To allow cross-domain temporal tagging (req. B), HeidelTime distinguishes between two document types: "news" and "narratives". On the one hand, all documents for which the document creation time is crucial are summarized as "news", e.g., news documents, conversations, but also documents like blog entries. On the other hand, "narratives" refer to documents for which the document creation time is usually irrelevant, e.g., Wikipedia articles and all other kinds of narrative-style documents. In addition to news- and narrative-style documents, a third type of documents exist, namely documents in which temporal expressions cannot be normalized to real points in time but to a "document-internal" timeline. This behavior is typical for literary and scientific documents. For example, in a biomedical text, when describing clinical trials there are often expressions such as "three days later" for which the reference time is not a real date but a "time point zero" in the document, i.e., such documents have their own time frame. The temporal expression "three days later" could thus be normalized to "three days after point zero". While the present version of HeidelTime supports normalization strategies for news- and narrative-style documents, we are currently working on integrating the additional normalization strategy for "closed time frame" documents.

In the following, we present HeidelTime's resources (Sect. 4.1.1), describe the syntax of the rule language (Sect. 4.1.2), and explain HeidelTime's algorithm for temporal tagging of documents (Sect. 4.1.3).

4.1.1 HeidelTime's resources

HeidelTime's resources are read and interpreted by the algorithm and organized in a directory structure. For every language, three directories are used, representing the three resources (1) pattern resources, (2) normalization resources, and (3) rule resources. Within these directories, every resource item is represented as a file in which one can easily modify the resource or include comments and examples without influencing the resource itself. In the following, we describe the three resources in detail:

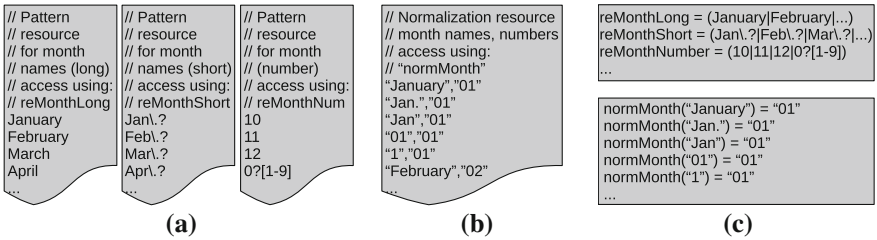

Fig. 3 Pattern resource files for different expressions for month names and numbers (**a**) and normalization resource file for month expressions (**b**). (**c**) represents how their information is used in HeidelTime after being read and translated by HeidelTime's resource interpreter

- *Pattern resources*: Pattern resources are used to create regular expressions, which can be accessed by every rule. This allows to use category names instead of listing all items every time the category is needed in a rule. For example, there are patterns for month names, names of weekdays, and number words. The pattern resource files contain one disjunct per line and the regular expression is built by HeidelTime's resource interpreter when reading the resources. Figure 3a shows examples of pattern resource files, and Fig. 3c (upper part) how they are translated by the resource interpreter.
- *Normalization resources*: Normalization resources contain normalized values of expressions included in the pattern resources. These values often correspond to the ISO standard for temporal information. They are used when HeidelTime assigns a value to a temporal expression, i.e., when interpreting the temporal expression. The normalization resource files are read by HeidelTime's resource interpreter, and for every file, a hash map is created. The files contain one key/value pair in each line. An example of a normalization resource file is given in Fig. 3b. How it is used in HeidelTime is shown in the lower part of Fig. 3c.
- *Rule resources*: The rule resources contain the rules for the extraction and the normalization of temporal expressions. For every type (date, time, duration, and set), there is one file. In the extraction part and the normalization part of the rules, one can use the pattern resources and the normalization resources, respectively. In addition, one can define further constraints, such as specific part-of-speech tags at a specific position, or modify the extent of a temporal expression. Similar to the other resources, the rule resources are read by HeidelTime's resource interpreter and hash maps are created for the extraction, the normalization, and for all further constraints. However, due to the complexity, the details of the rule resources and the syntax of the rule language are explained in Sect. 4.1.2.

The strict separation between the source code and the resources as well as the directory structure of the resources allow the easy integration of new languages to HeidelTime. Additional modular extensions can be integrated by adding further

extraction and normalization resources, e.g., for event expressions, which can be mapped to some point or interval in time. While the pattern and normalization resources can be created as described above, the rules are developed according to the syntax of the rule language, which is described next.

4.1.2 HeidelTime's rule syntax

In general, HeidelTime considers every temporal expression as a three-tuple $te_i = \langle e_i, t_i, s_i \rangle$, with the expression itself (e_i), the type of the expression (t_i, with $t_i \in \{date, time, duration, set\}$), and the normalized semantic attributes of the expression (s_i). Note that s_i does not only consist of the TIMEX3 attribute *value*, but of all attributes that are subject to normalization, e.g., the *mod* attribute. However, for better readability, we start explaining HeidelTime's rule syntax with the focus on the *value* attribute.

The goal of HeidelTime is, for each temporal expression te_i in a document, to identify the expression e_i and its type t_i and to correctly normalize its semantics s_i. For this, we developed a rule syntax according to which all rules have to be specified in the rule resources. While the rules are written in separate files, the source code only needs to know how to read and interpret the rules. A single rule has to contain the following components:

- RULENAME: Assigning a name to each rule allows to retrace which rule extracted which temporal expression. This is useful for several tasks, e.g., calculating statistics of the occurrences of different realizations of temporal expressions, and performing a detailed error analysis and improving rules.
- EXTRACTION: Every rule contains an extraction part describing the regular expression pattern that has to be matched. In this part, one can use the pattern resources described above. In addition, one may use parentheses to group parts of the expression, which is important for the normalization of the expressions.
- NORM_VALUE: This part defines the normalized value of an expression. One can use the normalization resources described in the previous section and refer to parts of an expression using the groups defined in the extraction part of the rule. In addition, the following functions can be applied:
  - SUBSTRING(x,i,j): returns a substring of the string x starting at character position i and having the length j.
  - LOWERCASE(x): converts all characters of the string x to lower case.
  - UPPERCASE(x): converts all characters of the string x to upper case.
  - SUM(x,y): adds the integer y to the integer x.

The three components rulename, extraction, and norm_value are required and thus part of every rule. To access the pattern resources in the extraction part and the normalization resource in the norm_value part, we use the percent sign (%). For example, to access the pattern resource reMonthLong (see Fig. 3a), one writes "%reMonthLong" in the extraction part of a rule. To distinguish between resources

and functions in the norm_value part, function words are surrounded by percent signs, e.g., "%LOWERCASE%(x)".

An example for a simple rule, which extracts date expressions such as "January 25, 2009" or "March 11, 1999" and normalizes their values according to the TimeML standard format (2009-01-25 and 1999-03-11 for the two examples), can be written as:

```
RULENAME="date_r1",
    EXTRACTION="%reMonthLong %reDayNumber, %reYear4Digit",
    NORM_VALUE="group(3)-%normMonth(group(1))-%normDay(group(2))"
```

Note that every pattern resource in the extraction part of the rule counts as one parenthesis pair for the group function. Thus, group(1), group(2), and group(3) refer to reMonthLong, reDayNumber, and reYear4Digit, respectively. To allow for similar expressions to be matched with the same rule, one can easily extend the rule to match abbreviated month names (reMonthShort) and ordinal numbers (reDayNumberTh):

```
RULENAME="date_r1",
    EXTRACTION="(%reMonthLong|%reMonthShort) " +
                    "(%reDayNumberTh|%reDayNumber), %reYear4Digit",
    NORM_VALUE="group(7)-%normMonth(group(1))-%normDay(group(4))"
```

For some linguistic phenomena, one needs to specify further constraints to correctly extract and normalize temporal expressions. For this, we define the following parts that can be added to a rule.

- POS_CONSTRAINT(group(x):y:): the part of speech tag of the group x of the matched expression must be equal to y.
- OFFSET(group(x)-group(y)): instead of extracting the complete matched expression as temporal expression, the extent starts with the beginning of group x and ends with the end of group y of the matched expression.
- NORM_MOD: the attribute *mod* is defined here.
- NORM_QUANT: the attribute *quant* is defined here.
- NORM_FREQ: the attribute *freq* is defined here.

To correctly normalize some expressions, in addition to the *value* attribute of a temporal expression the attributes *mod*, *quant*, and *freq* have to be set according to the normalization standards. The parts of a rule norm_mod, norm_quant, and norm_freq are used to set the values of these attributes of a temporal expression in addition to the *value* attribute. All the functions defined for the norm_value part (see above) can be used here as well. To clarify the use of these further parts of a rule, we give example rules to change the extent, to normalize further attributes, and to set a part-of-speech constraint.

```
RULENAME="date_r2",
    EXTRACTION="%reYear4Digit(-| and )%reYear2Digit",
    NORM_VALUE="%SUBSTRING%(group(1),0,2)group(3)",
    OFFSET="group(3)-group(3)"
RULENAME="date_r3",
    EXTRACTION="%rePartWords([ ]?)%reYear4Digit",
    NORM_VALUE="group(3)",
    NORM_MOD="%normPartWords(group(1))"
RULENAME="date_r1_negative",
    EXTRACTION="%reYear4Digit ([\S]+)",
    NORM_VALUE="REMOVE",
    POS_CONSTRAINT="group(2):NNS:"
```

The rule date_r2 matches expressions such as "1990–1995" and extracts the temporal expression with the extent "95" for which the value is set to "1995". The normalization is done using the substring function in the norm_value part of the rule. Without using the offset part of the rule, the whole expression "1990–1995" would wrongly be extracted as one temporal expression.

In the extraction part of the rule date_r3, the pattern resource rePartWords is used. It contains expressions such as "the beginning of", "the end of", and "mid-" and their normalized values are defined in the normPartWords resource. This rule extracts expressions like "mid-2002" and "the beginning of 1999" and normalizes their values to 2002 and 1999, respectively. In addition, the *mod* attribute is normalized as defined in the normPartWords resource. In these examples, the *mod* attributes are "MID" and "START", respectively.

The rule date_r1_negative is an example of a *negative rule*. Negative rules are used to prevent phrases to be matched as temporal expressions. For example, a simple rule with the extraction part being "%reYear4Digit" matches every four digit number in a text. Although a four digit number often refers to a year, sometimes it is used as a numeral for a count noun. In such cases, one wants that four digit number to be blocked for the positive rule. This task is performed by the rule date_r1_negative. It extracts a four digit number followed by an arbitrary token. However, this arbitrary token must have the part-of-speech tag "NNS" as defined by the pos_constraint part of the rule. A part-of-speech tagger assigns the "NNS" tag to plural nouns. Thus, this rule extracts phrases such as "2000 soldiers" or "1900 miles". The value "REMOVE" is assigned to such expressions, which is interpreted by the algorithm as just defined. The details of how the algorithm handles negative rules and the "REMOVE" value are described in the next section.

Note that the rule date_r1_negative may wrongly match expressions that refer to a year, e.g., in "the 2000 celebrations" or "the 2005 treaties". Without further knowledge, solving these ambiguity problems is a tough challenge. The expressions

could either refer to 2000 different celebrations and 2005 different treaties, respectively, or to the year 2000 celebrations and the treaties concluded in the year 2005.

Using the rule syntax described so far, it is possible to extract and normalize temporal expressions that are explicitly mentioned in the text. Implicit expressions can already be extracted as well if the required resources for the normalization are available. However, as mentioned in Sect. 2.2, temporal information is often expressed in an underspecified way and thus relative to other expressions. While the extraction part is similar to the ones for explicit expressions, the normalization has to be performed differently. For this, we set the values to expressions starting with "UNDEF". Depending on the type of text that is processed (news or narratives) and the characteristics of the temporal expression, the reference time is determined. While the details for this normalization are explained in the next section, the syntax for the underspecified normalization is defined according to one of the formats

- UNDEF-%normUnit(x)-REST
- UNDEF-(this|next|last)-%normUnit(x)-REST

with normUnit containing normalized values of expressions such as day, month, and year. "REST" represents the rest of the temporal expression, which is already normalized or might be empty. The first case is used if the relation to the reference time is unknown, e.g., in phrases like "In August". Here other methods have to be used to identify the relation to the reference time. The second case is used if the relation is known. For example, "last month" gets the value "UNDEF-last-month". In addition, in the second case, "REST" may represent a calculation function of the form "-(MINUS|PLUS)-y". Two example rules for extracting and normalizing relative temporal expressions using such a calculation function are:

```
RULENAME="date_r4",
    EXTRACTION="([\d]+) %reUnit ago",
    NORM_VALUE="UNDEF-this-%normUnit(group(2))-MINUS-group(1)"
RULENAME="date_r5",
    EXTRACTION="(%reMonthLong|%reMonthShort)" +
                    "(%reDayWordThl%reDayNumberThl%reDayNumber)"
 NORM_VALUE="UNDEF-year-%normMonth(group(1))-%normDay(group(4))"
```

The rule date_r4 matches expressions like "10 months ago" and normalizes them to underspecified values. For the given example, the value is set to "UNDEF-this-%normUnit(month)-MINUS-10", which results in "UNDEF-this-month-MINUS-10". The rule date_r5 matches expressions such as "December 15th" and sets the value, for this example, to "UNDEF-year-%normMonth ("December")-%norm-Day("15th")", which results in an underspecified value of "UNDEF-year-12-15".

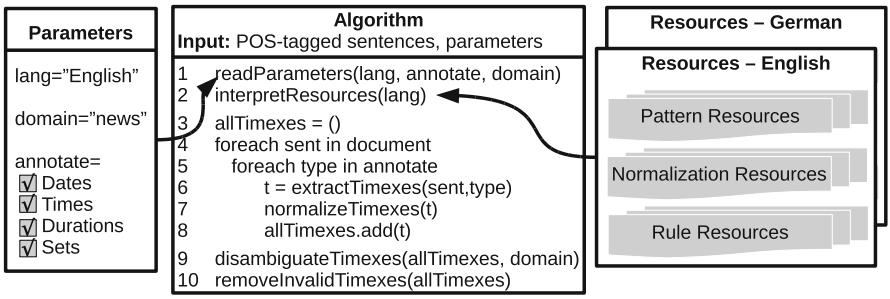

Fig. 4 HeidelTime's algorithm reading parameters and resources

The final values for such expressions are then calculated internally in HeidelTime's disambiguation phase, as described in the next section.

4.1.3 HeidelTime's algorithm

As show in Fig. 4, HeidelTime expects as input part-of-speech tagged sentences and user-specified parameters defining which types of expressions are to be annotated (parameter *annotate*) and which language and domain are used (parameters *lang* and *domain*, respectively). In an initialization phase, the parameters are read (line 1) and the resources of the corresponding language are interpreted by HeidelTime's resource interpreter (line 2) as described in Sect. 4.1.1. Then, HeidelTime performs the extraction and normalization of temporal expressions by running the following phases: (1) the extraction phase, (2) the normalization phase, (3) the disambiguation phase, and (4) the cleaning phase. In Fig. 4, these phases are called in lines 6, 7, 9, and 10, respectively. The extraction and normalization phases are called for every sentence (line 4) and for every annotation type (line 5).

During the extraction phase, the extraction parts of the rules are searched in the sentences. During the normalization phase, the—possibly underspecified—normalized values are assigned to the extracted expressions. In the previous section, we detailed the syntax of the rule language and described that further constraints (pos_constraint, offset) may have to be satisfied in the extraction phase, and further attributes (mod, freq, and quant) may have to be normalized in the normalization phase.

After all sentences are processed, underspecified and ambiguous temporal expressions are subject to analysis in the disambiguation phase. For this, all extracted expressions, which are part of other temporal expressions, are removed. For example, in the phrase "... On January 24, 2009, ...", HeidelTime's rules match the expressions (1) "January 24, 2009", (2) January 24, (3) January, and (4) "2009", but all expressions except the longest one (i) are removed. If overlapping expressions are extracted, e.g., "late Monday" and "Monday morning", only the first expression is currently annotated. However, in addition, the user is informed about overlapping expressions since these indicate that the rules could be improved. In the given example, a rule for expressions such as "late Monday morning" should

be added to the rule set. In the next step, all remaining temporal expressions are searched for values starting with "UNDEF". For these expressions, the reference time and the relation to the reference time are determined, and the values are disambiguated according to this information. In the cleaning phase, all invalid temporal expressions are deleted, i.e., expressions identified by negative rules and thus expressions with the value "REMOVE". Since all shorter expressions within these expressions have already been deleted in the disambiguation phase, the task of negative rules to block parts of expressions for other rules is correctly performed in the cleaning phase.

To further detail the disambiguation phase, we use the two examples of Fig. 1. In the news document (Fig. 1a) and the narrative document (Fig. 1b), the expression "December" and "December 25" are normalized to "UNDEF-year-12" and "UNDEF-year-12-25" in the normalization phase, respectively. During the disambiguation phase, these have to be fully specified. For narrative documents, HeidelTime assumes the last mentioned temporal expression of the type date to be the reference time. Thus, the value of the expression "December 25" is correctly normalized to "1979-12-25". For news documents, HeidelTime assumes the document creation time to be the reference time. Thus, the relation to the document creation time has to be identified using the tense information of the sentence. This is done by determining the part-of-speech tags of the verbs in the sentence. If past tense is determined, the year of the value will be set to the year of the previous December of the document creation time. If present or future tense is identified it will be set to the year of the December after the document creation time. In the example, the document creation time is "1998-04-28", i.e., the value of the expression "December" is correctly disambiguated to "1997–1912" since the tense of the sentence (the verb "cited") is determined as past tense.

4.2 HeidelTime as UIMA component

Due to the need for linguistic preprocessing such as sentence splitting and part-of-speech tagging and due to the fact that temporal tagging should be done independent of the format of the input data, it is useful to have HeidelTime as one component of a document processing pipeline.

For this, we use the Unstructured Information Management Architecture (UIMA), a system architecture to process unstructured content of any type (e.g., text or images).[16] Using UIMA, it is possible to combine different tools originally not built to be used together since all components are based on the same data structure, the Common Analysis Structure (CAS). In addition, the types of annotations that may be added to a CAS object are defined in a type system. For example, a type *Sentence* may contain the start and end position of a sentence in the document, while a type *token* may additionally contain a *part-of-speech* feature. Due to these characteristics of UIMA, we can use existing components for linguistic preprocessing without running into integration problems—an often occurring

[16] See http://uima.apache.org/.

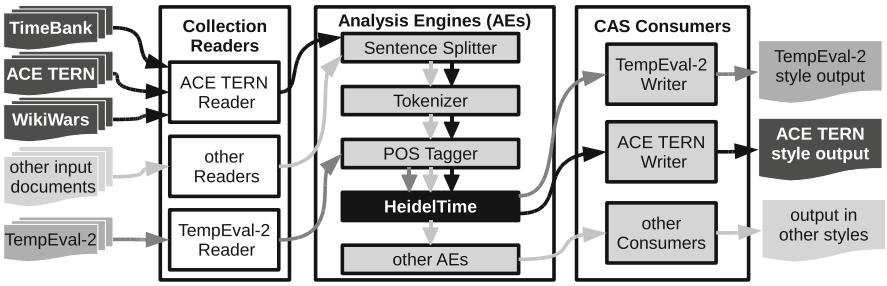

Fig. 5 Components of the UIMA-based text mining pipeline showing three workflows: Processing of ACE TERN style documents (*black*), TempEval-2 style documents (*dark gray*), and other data with other goals than temporal tagging evaluation (*light gray*)

problem due to incompatibilities and heterogeneous tools. In addition, this allows other users of HeidelTime to use other tools for preprocessing.

In Fig. 5, our document processing pipeline is illustrated. In general, a UIMA pipeline consists of three types of components: A *Collection Reader* to read the input data from an arbitrary source, to initialize a CAS object for every document subject to analysis, and to set the document text, i.e., the text that is directly processed by the next type of components, the *Analysis Engines*. These perform the analytical part in the processing pipeline, i.e., analyze the documents, extract information, and add annotations of the extracted or derived information to the CAS objects. Finally, *CAS Consumers* perform the final processing such as writing annotations into a database or writing files containing annotations extracted by the Analysis Engines.

We developed a Collection Reader to read the ACE TERN style input data.[17] This *ACE TERN Reader* can be used to access the ACE TERN corpora, the WikiWars and WikiWarsDE corpora, and the TimeBank corpus. As described in Sect. 3.1, the WikiWars corpus is made publicly available in the TERN format as is our WikiWarsDE corpus. In addition, we converted the TimeBank corpus into this format to be able to run the ACE TERN scripts for evaluation of this corpus as well. The *ACE TERN Reader* sets the document text as well as the document creation time. The format of the TempEval-2 corpus differs from the TERN format and, in addition, sentence and token annotations are directly provided. Since this information is used by the official TempEval-2 evaluation scripts, one should use the provided sentence and token information when evaluating a temporal tagger on the TempEval-2 corpus. For this, we developed the *TempEval-2 Reader*, which annotates sentence and token information in addition to the document text and the document creation time. Thus, when processing TempEval-2 documents the Analysis Engines *Sentence Splitter* and *Tokenizer* are not applied while they are used when ACE TERN style data is processed. In both cases, the *Part-of-Speech Tagger* and *HeidelTime* are used before the CAS Consumers *ACE TERN Writer* or *TempEval-2 Writer* are applied to create result files in the ACE TERN or TempEval-

[17] Our UIMA components as well as conversion scripts described in this section are available at http://dbs.ifi.uni-heidelberg.de/heideltime/.

2 format, respectively. These files can directly be used to run the official evaluation scripts provided by the organizers of the competitions.

In addition to the ACE TERN workflow (black) and the TempEval-2 workflow (dark gray), a general workflow is depicted in Fig. 5 (light gray). This workflow demonstrates that independent of the format or type of the original documents, the tasks of sentence splitting, tokenization, part-of-speech tagging, and temporal tagging are always performed in the same way. Only the Collection Reader depends on the source. Of course, further analysis engines can be added to the pipeline and CAS Consumers are used depending on the tasks that one wants to perform afterwards. For example, we built a system called TimeTrails for the exploration of events in documents based on the spatial and temporal information occurring together in the sentences of documents. For this, we run our pipeline using a Collection Reader that crawls Wikipedia articles. Then, the Analysis Engines are applied as shown in Fig. 5. Additional Analysis Engines are used for geo-tagging and the extraction of co-occurrences of temporal and spatial expressions. Finally, a CAS Consumer writes all extracted pairs of spatial and temporal expressions and thus all events into a database, which is used as knowledge base for the visualization and exploration components of TimeTrails (Strötgen and Gertz 2010b).

In summary, using HeidelTime as a UIMA component allows users to perform all kinds of tasks that deal with temporal information extracted from documents. Additionally, existing or newly developed components can be integrated into the pipeline without running into any integration problems. In our case, for linguistic preprocessing, we use the sentence splitter, tokenizer, and part-of-speech tagger components of the UIMA DKPro repository (Gurevych et al. 2007) with the part-of-speech tagger being a wrapper for the TreeTagger (Schmid 1994). We selected these components due to their applicability to multiple languages. Of course, these components can be replaced by other tools performing the same tasks.

4.3 Resource development process

In this section, we describe the resource development process for HeidelTime's English and German resources. In the context of TempEval-2, we developed HeidelTime's first version of English resources using the TempEval-2 training data, which corresponds to the TimeBank corpus (Verhagen et al. 2010). We developed a precision- and a recall-optimized rule set, but later dropped the recall-optimized rule set. For processing narrative-style documents, we then added the second normalization strategy to HeidelTime and adapted the pattern, normalization, and rule resources. However, these modifications were not performed using an annotated corpus but in the context of our work on spatio-temporal document exploration (Strötgen and Gertz 2010b). For this, we manually checked the results on some Wikipedia articles. The result of this work corresponds to the current version of HeidelTime's English resources. Thus, for the development of the English resources we did not use any of the other temporally annotated corpora, which are used for the evaluation described in the next section.

HeidelTime's German resources were developed after the English ones and we started with translating the English pattern and normalization resources as well as the English words directly occurring in the English rules. Then, for our work on multilingual document similarity (Strötgen et al. 2011), we used some German Wikipedia articles to improve the German rules. However, at this point in time, we had not yet developed WikiWarsDE, and thus, we did not use the WikiWarsDE corpus for the development of the German resources. Since the normalization strategies, the rule syntax, and the English resources were already available, the development of the German resources was straightforward and took only a few days.

Since making HeidelTime publicly available, we keep on receiving feedback with suggestions on how to improve the rules. We will regularly update HeidelTime's resources to further improve HeidelTime's quality for extracting and normalizing temporal expressions on different domains. In the next section, we present the evaluation results of the current version of HeidelTime.

5 Evaluation

After describing evaluation measures in Sect. 5.1, we present the evaluation results for HeidelTime on publicly available corpora in Sect. 5.2. We also give the results of the temporal taggers described in Sect. 3.2 on these corpora. This allows to compare the quality of HeidelTime with existing systems. As described in the previsous section, we did not optimize HeidelTime's resources to the different corpora. Besides adding the timestamp formats used in the corpora as document creation times, we did not use the evaluation corpora to develop HeidelTime—except the TimeBank corpus, on which the TempEval-2 training data was based.

Note that HeidelTime uses TIMEX3 for annotating temporal expressions, but except TimeBank and TempEval-2, the publicly available corpora are annotated with TIMEX2. Due to the differences, we performed a simple conversion from TIMEX3 to TIMEX2 similar to (Saquete Boro 2010) for being able to evaluate HeidelTime on the TIMEX2-annotated corpora as well. In addition, we transformed the format of the TimeBank corpus into the ACE TERN format for being able to run the ACE TERN evaluation scripts as described in Sect. 4.2. We did not change the TIMEX3 annotations on this corpus but directly evaluated HeidelTime's TIMEX3 annotations.

5.1 Evaluation measures

When evaluating temporal taggers, the subtasks of extraction and normalization can be measured separately. Although the evaluation of the normalization could include all the attributes of the temporal expressions, we concentrate on the value (VAL) attribute, which is the most important attribute of temporal expressions. For both tasks, the measures of precision, recall, and f-score are widely used, e.g., the evaluation scripts of the ACE TERN and TempEval-2 competitions calculate these

measures. However, while the ACE TERN script measures the scores at the expression level, the TempEval-2 script calculates them at the token level. In addition, when using the measures at the expression level, one can distinguish between strict and lenient matching. While the strict match means a complete match between the gold standard and the system's expression, for a lenient match it is sufficient that one single character overlaps. Precision (P), recall (R), and f-score (F) are calculated according to the following formulas, with *true positives* (TP) being the number of expressions correctly identified as temporal expression by the system, *false positive* (FP) being the number of expressions wrongly identified as temporal expressions by the system, and *false negative* (FN) being the number of temporal expressions that were missed by the system:

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN} \quad F = \frac{2 \times P \times R}{P + R}$$

The normalization can be evaluated either with respect to all expressions in the gold standard or to all expressions correctly identified by the system. While the second method is used by the ACE TERN and the TempEval-2 scripts, we argue similar to Ahn et al. (2005b) that the first one is more meaningful. For the sake of completeness, we give the following evaluation results for HeidelTime on all corpora:

- lenient: extraction (lenient) only
- strict: extraction (strict) only
- value: value, based on correctly identified expressions only
- len+val: extraction (lenient) and value normalization
- str+val: extraction (strict) and value normalization

5.2 Evaluation results

As a first official evaluation, we participated in the TempEval-2 task of extracting and normalizing temporal expressions in English documents. Here, we achieved the best results for both the extraction and the normalization. Table 3 shows the results of TempEval-2 separated from other evaluation results since this evaluation is done on a token level. Thus, the results are not directly comparable to the other corpora that are evaluated on an expression level. In addition to the *value* attribute, the *type* attribute was evaluated. In the competition, we participated with two rule sets, HeidelTime-1 and HeidelTime-2, a precision- and a recall-optimized rule set. The publicly available version of HeidelTime contains only one rule set for English and one for German. While we described the resource development process in the previous section, some statistics on the rule sets and the number of pattern and normalization resources are given in Table 4. The results on the TempEval-2 corpus with the publicly available English rule set are shown in Table 3.[18]

In Table 5, HeidelTime's evaluation results on the other publicly available corpora described in Sect. 3.1 are presented. In addition, if available, evaluation

[18] Results slightly differ from HeidelTime-1 due to some bug fixes.

Table 3 Results of TempEval-2 (Verhagen et al. 2010) and HeidelTime's publicly available version

	P	R	F	Value	Type
HeidelTime-1	90	82	86	85	96
HeidelTime-2	82	91	86	77	92
TRIOS	85	85	85	76	94
TRIPS	85	85	85	76	94
TipSem	92	80	85	65	92
Edinburgh	85	82	84	63	84
KUL Run2	85	84	84	55	91
USFD2	84	79	82	17	90
TERSEO	76	66	71	65	98
HeidelTime	90.7	86.0	88.3	86.0	96.0

Table 4 Number of rules and resources of HeidelTime's publicly available version

Rules	English	German
Date	59	48
Time	20	8
Duration	22	17
Set	13	8
Negative rules	12	12
Total	126	93
Resources	**English**	**German**
Pattern	27	27
Normalization	16	16

results of the temporal taggers surveyed in Sect. 3.2 are shown for comparison purposes. On the ACE TERN 2004 training data (Table 5a), HeidelTime achieves better results than GUTime for which only the f-scores are published. Table 5b shows the evaluation results of some systems on the ACE TERN 2004 evaluation corpus. The best systems participating in the challenge are Chronos (extraction and normalization) and ATEL (extraction only). Similar results were achieved by the cascaded machine learning approach to interpreting temporal expressions (ARR). The results of the TERSEO system indicate that an automatic rule translation works reasonably well since this system uses automatic rule translation (TERSEO-1) and, in addition, automatically extracted rules from an annotated corpus (TERSEO-2).

Three taggers were evaluated on the TimeBank-1.2 corpus (Table 5d), although only evaluation results for the extraction were given. HeidelTime achieves better results than all the other three taggers. Furthermore, we present results for the normalization and the combination of extraction and normalization demonstrating the high quality of HeidelTime's normalization approach.

The developers of the WikiWars corpus evaluated their temporal tagger DANTE on the ACE TERN 2005 training data (Table 5c) and the WikiWars corpus (Table 5e) with two rule sets. On both corpora, HeidelTime significantly

Table 5 Evaluation results of HeidelTime and other taggers on different corpora

(a) Results on ACE TERN 2004 training data

	Lenient			Strict			Value			Len+val			Str+val		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
GUTime[a]			85			78			82						
HeidelTime	94.5	80.5	86.9	85.7	73.0	78.8	85.2	85.4	85.3	80.5	68.6	74.1	75.2	64.1	69.2

(b) Results on ACE TERN 2004 evaluation data

	Lenient			Strict			Value			Len+val			Str+val		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Chronos[b]	97.6	88.0	92.6	88.5	79.8	83.9	87.5	87.0	87.2						
ATEL[c]	97.8	89.4	93.5	91.9	84.0	87.8									
ARR[d]	92.9	81.3	86.7	87.8	76.9	81.9	91.0	88.7	89.9						
TERSEO-1[e]	67.3	72.8	69.9	77.0	62.0	69.0	75.7	73.5	74.6						
TERSEO-2[f]	95.4	78.6	86.2	68.7	56.7	62.1	68.6	70.9	69.8						

(c) Results on ACE TERN 2005 training data

	Lenient			Strict			Value			Len+val			Str+val		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
DANTE[g]	71	87	78	53	65	58				34	42	37	30	36	33
DANTE[h]	88	93	90	75	79	77				63	67	65	57	60	58
HeidelTime	88.5	76.8	82.2	75.3	65.4	70.0	74.0	76.2	75.1	65.4	56.8	60.8	60.9	52.9	56.6

(d) Results on TimeBank 1.2

	Lenient			Strict			Value			Len+val			Str+val		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
BA-05[i]	85.2	95.2	89.6	77.6	86.1	81.7									
KM-09[j]	87.2	83.6	85.2	86.6	79.6	82.8									
UA-11[k]	95.4	86.5	90.7	86.5	78.5	82.3									
HeidelTime	90.7	91.5	91.1	83.7	84.4	84.1	86.2	86.2	86.2	78.3	78.9	78.6	73.5	74.1	73.8

(e) Results on WikiWars

	Lenient			Strict			Value			Len+val			Str+val		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
DANTE[l]	90	75	82	42	35	38				22	18	20	19	16	17
DANTE[m]	98	99	99	95	95	95				59	60	59	58	59	58
HeidelTime	93.9	82.6	87.9	86.0	75.7	80.5	89.5	90.1	89.8	84.1	73.9	78.7	79.6	70.0	74.5

Table 5 continued

(f) Results on WikiWarsDE

	Lenient			Strict			Value			Len+val			Str+val		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
HeidelTime	98.5	85.0	91.3	92.6	79.9	85.8	87.0	87.0	87.0	85.7	74.0	79.4	82.5	71.2	76.5

[a] http://timeml.org/site/tarsqi/modules/gutime/index.html

[b] Negri and Marseglia (2005)

[c] Hacioglu et al. (2005)

[d] Ahn et al. (2007)

[e] Negri et al. (2006)

[f] Saquete Boro (2010)

[g] Mazur and Dale (2010) initial performance

[h] Mazur and Dale (2010) improved rule set

[i] Boguraev and Ando (2005), with lenient as identical right boundaries instead of overlap

[j] Kolomiyets and Moens (2009)

[k] UzZaman and Allen (2011)

[l] Mazur and Dale (2010) initial performance

[m] Mazur and Dale (2010) improved rule set

outperforms DANTE's initial rule set in the extraction and the normalization tasks. However, since we did not make any corpus-specific adaptations, DANTE with its adapted rules achieves a much better recall. A simple error analysis on the ACE TERN 2005 training corpus showed that there are a couple of frequently occurring annotated expressions that are not covered by HeidelTime's rule set, such as age information expressions (e.g., "19" in "Bassem Takrouri, 19, lived ..."), incomplete timestamps (e.g., "????-??-??T12:26:00"), and simple but unspecific expressions (e.g., "once", "then", "new", and "ex"). While rules covering these types of temporal expressions could easily be added to HeidelTime's rule set, some of them are not part of the TIMEX3 annotations. Note that the corpus is annotated according to the TIDES TIMEX2 guidelines and HeidelTime is developed using TIMEX3 annotations. In addition, as described in Sect. 4.3, we did not use the evaluation corpora for adapting HeidelTime's rule set at all. On the WikiWars corpus, HeidelTime achieves much better results for the normalization than DANTE with its adapted rules. While HeidelTime distinguishes between news and narrative-style documents, DANTE normalizes temporal expressions independent of the domain, i.e., using the document creation time. HeidelTime's much better results for the normalization indicate that there is a need to apply different normalization strategies for news and narrative style documents.

In summary, these results demonstrate that HeidelTime achieves very good results for the extraction and the normalization on both, the news domain and on narratives. Finally, we evaluate HeidelTime on the German WikiWarsDE corpus (Table 5f) to demonstrate its ability to achieve high quality results on corpora of different languages. We cannot compare HeidelTime with any other tagger on this

corpus, but the results look very promising since they are similar to the results on the English corpus.

6 Conclusions and ongoing work

In this paper, we presented an overview of the tasks of extracting and normalizing temporal expressions, surveyed temporally annotated corpora and existing temporal taggers, and introduced our multilingual temporal tagger HeidelTime. A tough challenge in the normalization task of temporal tagging is to correctly identify the reference time of so-called relative temporal expressions (e.g., "today", In "November"). Without knowing their reference time, these expressions cannot be normalized. Depending on the domain of the processed documents, different strategies to determine the reference time are needed. As we showed by surveying existing temporal taggers, there are hardly any approaches to apply temporal taggers on other domains than the news domain. In addition, existing temporal taggers lack the possibility to simply add rules for task-dependent temporal expressions or to adapt a temporal tagger to a new language without modifying their source code. Motivated by these observations, we developed our temporal tagger, called HeidelTime, a rule-based system that strictly separates between the source code and resources like rules. This architectural feature allows to simply (1) add or modify rules, (2) integrate new modules, and (3) develop resources for new languages. Furthermore, HeidelTime pursues different strategies for normalizing temporal expressions—depending on the domain of the documents that are to be processed. Using publicly available corpora, we were able to demonstrate the high quality results of HeidelTime on different domains. In addition, we showed the extensibility to further languages by evaluating HeidelTime on a newly developed corpus for German.

To further improve temporal tagging, we are currently working on adding further clues to our normalization strategies. For example, a repeatedly occurring problem in narratives are temporal expressions that refer to background information but do not belong to the main plot of the narrative. Thus, when identifying these expressions, they should not be used as a candidate for the reference time. In the document shown in Fig. 1b, one would like to identify "1978" as such an expression. By making available HeidelTime's current version with its rule sets for English, German, and Dutch, the newly developed corpus WikiWarsDE, and several further UIMA components and scripts, we provide valuable contributions to the community. Furthermore, our evaluation results are reproducible, and we are going to maintain HeidelTime and provide the most recent versions including new rule sets for further languages.

References

Ahn, D., Adafre, S. F., & de Rijke, M. (2005a). Extracting temporal information from open domain text: A comparative exploration. *Journal of Digital Information Management, 3*, 14–20.

Ahn, D., Adafre, S. F., & de Rijke, M. (2005b). Towards task-based temporal extraction and recognition. In G. Katz, J. Pustejovsky, & F. Schilder (Eds.) *Annotating, extracting and reasoning about time and events*, Dagstuhl, Germany, no. 05151 in Dagstuhl Seminar Proceedings.

Ahn, D., van Rantwijk, J., & de Rijke, M. (2007) A cascaded machine learning approach to interpreting temporal expressions. In Proceedings of human language technologies: The annual conference of the North American chapter of the association for computational linguistics, pp. 420–427.

Allan, J. (Ed.) (2002). *Topic detection and tracking: Event-based information organization*. Norwell, MA: Kluwer Academic Publishers.

Alonso, O., Gertz, M., & Baeza-Yates, R. (2007). On the value of temporal information in information retrieval. *SIGIR Forum, 41*(2), 35–41.

Alonso, O., Strötgen, J., Baeza-Yates, R., & Gertz, M. (2011). *Temporal information retrieval: Challenges and opportunities*. In Proceedings of the 1st international temporal web analytics workshop (TWAW 2011), pp. 1–8.

Boguraev, B., & Ando, R. K. (2005). TimeBank-driven TimeML analysis. In G. Katz, J. Pustejovsky, & F. Schilder (Eds.) *Annotating, extracting and reasoning about tme and events*, no. 05151 in Dagstuhl Seminar Proceedings.

Chinchor, N. A. (1997). Overview of MUC-7/MET-2. In Proceedings of the 7th conference on message understanding (MUC 1997).

Costa, F., & Branco, A. (2010). Temporal information processing of a new language: Fast porting with minimal resources. In Proceedings of the 48th annual meeting of the association for computational linguistics (ACL '10), pp. 671–677.

Ferro, L., Mani, I., Sundheim, B., & Wilson, G. (2001). *TIDES temporal annotation guidelines—version 1.0.2*. Technical report, The MITRE Corporation.

Ferro, L., Gerber, L., Mani, I., Sundheim, B., & Wilson, G. (2005). *TIDES 2005 standard for the annotation of temporal expressions*. Technical report, The MITRE Corporation.

Grishman, R., & Sundheim, B. (1995). Design of the MUC-6 evaluation. In Proceedings of the 6th conference on message understanding (MUC 1995).

Gurevych, I., Mühlhäuser, M., Müller, C., Steimle, J., Weimer, M., & Zesch, T. (2007). Darmstadt knowledge processing repository based on UIMA. In Proceedings of the first workshop on unstructured information management architecture at biannual conference of the society for computational linguistics and language technology.

Hacioglu, K., Chen, Y., & Douglas, B. (2005). Automatic time expression labeling for english and chinese text. In Proceedings of the 6th international conference on intelligent text processing and computational linguistics (CICLing '05), Springer, pp. 548–559.

Kolomiyets, O., & Moens, M.-F. (2009). Meeting tempeval-2: Shallow approach for temporal tagger. In Proceedings of the workshop on semantic evaluations: Recent achievements and future directions (DEW '09), pp. 52–57.

Makkonen, J., Ahonen-myka, H., & Salmenkivi, M. (2003). Topic detection and tracking with spatio-temporal evidence. In Proceedings of 25th European conference on information retrieval research (ECIR '03), pp. 251–265.

Mani, I., & Wilson, G. (2000). Robust temporal processing of news. In Proceedings of the 38th annual meeting on association for computational linguistics (ACL '00), pp. 69–76.

Mazur, P., & Dale, R. (2009). The DANTE temporal expression tagger. In Proceedings of the 3rd language and technology conference, pp. 245–257.

Mazur, P., & Dale, R. (2010). WikiWars: A new corpus for research on temporal expressions. In Proceedings of the conference on empirical methods in natural language processing (EMNLP '10), pp. 913–922.

Negri, M., & Marseglia, L. (2005). *Recognition and normalization of time expressions: ITC-irst at TERN 2004*. Technical report.

Negri, M., Saquete, E., Martínez-Barco, P., & Muñoz, R. (2006). Evaluating knowledge-based approaches to the multilingual extension of a temporal expression normalizer. In Proceedings of the workshop on annotating and reasoning about time and events (ARTE '06), pp. 30–37.

Pustejovsky, J., Castaño, J. M., Ingria, R., Sauri, R., Gaizauskas, R. J., Setzer, A., Katz, G., & Radev, D. R. (2003a). TimeML: Robust specification of event and temporal expressions in text. In: New Directions in Question Answering, pp. 28–34.

Pustejovsky, J., Hanks, P., Sauri, R., See, A., Gaizauskas, R., Setzer, A., Radev, D., Sundheim, B., Day, D., Ferro, L., Lazo, M. (2003b). The TIMEBANK corpus. In Proceedings of corpus linguistics 2003, pp. 647–656.

Pustejovsky, J., Knippen, R., Littman, J., & Sauri, R. (2005). Temporal and event information in natural language text. *Language resources and evaluation, 39*(2–3), 23–164.

Saquete Boro, E. (2010). ID 392:TERSEO + T2T3 transducer. A systems for recognizing and normalizing TIMEX3. In Proceedings of the 5th international workshop on semantic evaluation (SemEval '10), pp. 317–320.

Schilder, F., & Habel, C. (2001). From temporal expressions to temporal information: Semantic tagging of news messages. In Proceedings of the ACL-2001 workshop on temporal and spatial information processing, pp. 65–72.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In Proceedings of the international conference on new methods in language processing.

Strötgen, J., & Gertz, M. (2010a). HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In Proceedings of the 5th international workshop on semantic evaluation (SemEval '10), pp. 321–324.

Strötgen, J., & Gertz, M. (2010b). TimeTrails: A system for exploring spatio-temporal information in documents. In Proceedings of the 36th international conference on very large data bases (VLDB 2010), pp. 1569–1572.

Strötgen, J., & Gertz, M. (2011). WikiWarsDE: A German corpus of narratives annotated with temporal expressions. In Proceedings of the conference of the German society for computational linguistics and language technology (GSCL 2011), pp. 129–134.

Strötgen, J., Gertz, M., & Popov, P. (2010). Extraction and exploration of spatio-temporal information in documents. In Proceedings of the 6th workshop on geographic information retrieval (GIR '10), pp. 1–8.

Strötgen, J., Gertz, M., & Junghans, C. (2011) An event-centric model for multilingual document similarity. In Proceeding of the 34rd international ACM SIGIR conference on research and development in information retrieval (SIGIR'11), pp. 953–962.

UzZaman, N., & Allen, J. (2011). Event and temporal expression extraction from raw text: First step towards a temporally aware system. *International Journal of Semantic Computing, 4*(4), 487–508.

Verhagen, M., & Pustejovsky, J. (2008). Temporal processing with the TARSQI toolkit. In *Coling 2008: Companion volume: Demonstrations*, pp. 189–192.

Verhagen, M., Sauri, R., Caselli, T., & Pustejovsky, J. (2010). SemEval-2010 task 13: TempEval-2. In Proceedings of the 5th international workshop on semantic evaluation (SemEval '10), pp. 57–62.