HeidelTime: High Quality Rule-based Extraction and Normalization of Temporal Expressions

Jannik Strötgen
Institute of Computer Science
University of Heidelberg
Heidelberg, Germany
stroetgen@uni-hd.de

Michael Gertz
Institute of Computer Science
University of Heidelberg
Heidelberg, Germany
gertz@uni-hd.de

Abstract

In this paper, we describe HeidelTime, a system for the extraction and normalization of temporal expressions. HeidelTime is a rule-based system mainly using regular expression patterns for the extraction of temporal expressions and knowledge resources as well as linguistic clues for their normalization. In the TempEval-2 challenge, HeidelTime achieved the highest F-Score (86%) for the extraction and the best results in assigning the correct value attribute, i.e., in understanding the semantics of the temporal expressions.

1 Introduction

Temporal annotation of documents, i.e., the extraction and chronological ordering of events, is crucial to many NLP applications, e.g., text summarization or machine translation. In this paper, we describe our system HeidelTime for the extraction and normalization of temporal expressions in English documents. It was the best-performing system in Task A for English of the TempEval-2 challenge[1]. The purpose of this challenge was to evaluate different systems for temporal tagging as well as event and temporal relation extraction since a competitive evaluation helps to drive forward research, and temporal annotation is important for many NLP tasks (Pustejovsky and Verhagen, 2009). The annotation scheme for temporal expressions, events, and relations is based on TimeML, the ISO standard for temporal annotation[2].

Before using temporal information in other applications is possible, the first task to solve is to extract and normalize temporal expressions (Task A of the challenge, annotated as Timex3). There are two types of approaches to address this problem: rule-based and machine learning ones. We decided to develop a rule-based system since normalization can then be supervised in a much easier way. Furthermore, respective systems allow for modular extensions.

Although we only participated in Task A, we do not consider the extraction and normalization of temporal expressions in isolation, but use temporal information in combination with other extracted facts, e.g., for the exploration of spatio-temporal information in documents (Strötgen et al., 2010). One of our primary objectives is therefore to develop a system that can be used in other scenarios without any adaptations. Thus, we implement HeidelTime as a UIMA[3] (Unstructured Information Management Architecture) component to integrate the system into our existing document processing pipeline. Another advantage of our temporal tagger is that the user can choose between a precision- and a recall-optimized rule set. In the TempEval-2 challenge, both rule sets achieved top scores in the extraction (F-scores of 86%) and the precision-optimized set achieved the best results for assigning the correct value attributes to the temporal expressions (85% accuracy).

The remainder of the paper is structured as follows: The system architecture is outlined in the next section. In Section 3, we present the evaluation results of HeidelTime in comparison to other systems that participated in the challenge. We conclude our paper in Section 4.

2 System Architecture

In this section, the system architecture of HeidelTime is explained. First, UIMA and our UIMA-based document processing pipeline are detailed, followed by a description of the extraction and normalization tasks, the functionality of the rules

[1] http://semeval2.fbk.eu/
[2] http://www.timeml.org/
[3] http://uima.apache.org/

Proceedings of the 5th International Workshop on Semantic Evaluation, ACL 2010, pages 321–324,
Uppsala, Sweden, 15-16 July 2010.

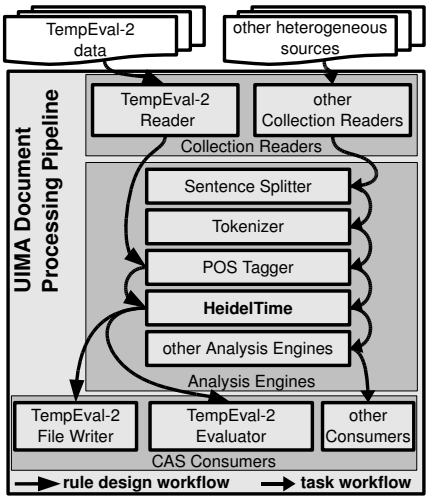

Figure 1: UIMA pipeline with two workflows, one for rule design and one for using HeidelTime.

and the post-processing steps.

2.1 Document Processing Pipeline

HeidelTime is developed as a UIMA component so that we are able to integrate our temporal tagger into our existing document processing pipeline. It is an extension of the temporal tagger we already use for the extraction and exploration of spatio-temporal information in documents (Strötgen et al., 2010). UIMA is widely used for processing unstructured content such as audio, images, or text. Different components can be combined to create a pipeline of modular tools, and all components use the same data structure, the Common Analysis Structure (CAS). This allows to combine tools that were not originally built to be used together, an advantage we are using for preprocessing tasks as well.

In general, a UIMA pipeline consists of three types of components, a Collection Reader for accessing the documents from a source and initializing a CAS object for each document. The analysis of the documents is performed by Analysis Engines that add annotations to the CAS objects. Finally, CAS Consumers are used for final processing, e.g., for storing the annotated information in a database or performing an evaluation.

In Figure 1, the document processing pipeline for designing and using our temporal tagger HeidelTime is depicted. The design workflow (left arrows) contains the TempEval-2 Reader, which reads the TempEval-2 data, initializes a CAS object for each textual document and adds the annotated data to the CAS. For the test set of the temporal expression task, these include the sentence and token information, and for the training set also the gold standard Timex3 entities. Next, the OpenNLP part-of-speech tagger[4] is used, which assigns the corresponding part-of-speech (POS) tag to each token. The information about sentences, tokens, and POS tags is then used by our temporal tagger HeidelTime for extracting and normalizing temporal expressions mentioned in the documents. The CAS Consumer TempEval-2 File Writer is used for creating the files needed for applying the scorer and which had to be submitted for evaluation. During the rule development phase of HeidelTime, the CAS Consumer TempEval-2 Evaluator was used, which compares the gold standard Timex3 annotations with the Timex3 annotations extracted by HeidelTime, resulting in lists of true positives, false positives, and false negatives. These lists were then used for adapting existing or creating new rules.

On the right-hand side of Figure 1, a workflow for using HeidelTime in other scenarios is shown. This workflow reflects the fact that temporal tagging is just one intermediate component of our document processing pipeline. Here, the documents have to be split into sentences and tokens using the two analysis engines Sentence Splitter and Tokenizer. The POS tagger and HeidelTime are used in the same way as described for the other workflow. In addition, other Analysis Engines can be used, e.g., for combining the extracted temporal information with spatial information. Finally, CAS Consumers are used, e.g., for storing the spatio-temporal information in a database.

2.2 Extraction and Normalization Tasks

Every temporal expression te can be viewed as a three-tuple $te_i = \langle e_i, t_i, v_i \rangle$, where e_i is the expression itself as it occurs in the textual document, t_i represents the type of the expression, and v_i is the normalized value. There are four possible types, namely *Date*, *Time*, *Duration*, and *Set*. The normalized value represents the temporal semantics of an expression as it is specified by the markup language TimeML, regardless of the expression used in the document. The goal of HeidelTime is to extract for every temporal expression the expression e_i and to correctly assign the type and value attributes t_i and v_i, respectively.

For this, HeidelTime uses hand-crafted rules,

[4] http://opennlp.sourceforge.net

Expression resources	$reMonth$ = "(...|June|July|...)"
	$reSeason$ = "(...|summer|...)"
Normalization functions	$normMonth$("June") = "06"
	$normSeason$("summer") = "SU"

Table 1: Examples for extraction and normalization resources for months and seasons.

which are grouped into four types, namely the four possible types of temporal expressions. More precisely, every rule is a triple of an expression rule, a normalization function and the type information. The extraction rules mainly consist of regular expression patterns. However, other features can be used as well, e.g., a constraint what part-of-speech the previous or next token has to have. HeidelTime contains resources for both the extraction and the normalization tasks of the rules. For instance, there are resources for weekdays, months, or seasons, which are realized as regular expressions and can be accessed by multiple extraction rules. In addition, there are knowledge resources for the normalization of such expressions. Examples are given in Table 1.

Algorithm 1 illustrates how rules are used in HeidelTime. First, the rules are applied to every sentence of a document, and extracted timexes are added to the CAS object. Then, two post-processing steps are executed to disambiguate underspecified values and to remove invalid temporal expressions from the CAS. This functionality is detailed in the next sections with a focus on the linguistic clues for the normalization task.

Algorithm 1 ApplyRules.

```
foreach sentence in document
    addDatesToCAS(date_rules, CAS);
    addTimesToCAS(time_rules, CAS);
    addDurationsToCAS(dur_rules, CAS);
    addSetsToCAS(set_rules, CAS);
end foreach
foreach timex3 in CAS
    disambiguateValues(CAS);
end foreach
removeInvalidsFromCAS(CAS);
```

2.3 Functionality of HeidelTime

There are many ways to textually describe temporal expressions, either explicitly, implicitly or relatively (Schilder and Habel, 2001). The extraction for all temporal expressions works in the same way, but assigning the value attributes has to be done differently. Explicit temporal expressions are fully specified, i.e., the value attribute can directly

explicit temporal expressions
$date\_r1 = (reMonth)_{g1}\ (reDay)_{g2},\ (reFullYear)_{g3}$
norm_r1(g1,g2,g3) = g3-normMonth(g1)-normDay(g2)
implicit temporal expressions
$date\_r2 = (reHoliday)_{g1}\ (reFullYear)_{g2}$
norm_r2(g1,g2) = g2-normHoliday(g1)

Table 2: Extraction parts and normalization parts of two sample rules.

be assigned using the corresponding normalization function of the rule. For example, the explicit expression *March 11, 1982* can be extracted with the rule date_r1 of Table 2 containing the resources $reMonth$, $reDay$, and $reFullYear$ (regular expressions for possible month, day and year tokens of a date phrase, respectively). The matched tokens can be accessed using the group ids so that the normalization function can be called with the extracted tokens resulting in the value 1982-03-11.

The value attribute of implicit expressions can be assigned once the implicit temporal semantics of such expressions is known. Holidays, for example, can be extracted using date_r2 with the resource $reHoliday$ and normalized using the knowledge resource for normalization as shown in Table 2. An example is *Independence Day 2010* to which the value 2010-07-04 is assigned.

The normalization of relative expressions for which a reference time is needed is the most challenging task. Examples are *last June*, just *June* in phrases such as *in June*, or *year-earlier* in *the year-earlier results*. To such expressions, HeidelTime assigns the values in an underspecified format depending on the assumed reference time and disambiguates them in a post-processing step. The underspecified values for the examples are UNDEF-last-June, UNDEF-June, and UNDEF-REF-last-year, respectively. For the first two examples, the document creation time (dct) is assumed to be the reference time while for the last example the previously mentioned date is used for reference. In news texts (as used in TempEval-2) the dct is meaningful while other documents may not contain such a reference time. Then, the previously mentioned date is used for all underspecified values. The disambiguation of such expressions is detailed in the next section.

2.4 Post-Processing

The first post-processing step is to disambiguate underspecified value attributes (see Algorithm 1). If the value starts with UNDEF-REF, the pre-

viously mentioned date is used for disambiguation, otherwise the document creation time (dct) if meaningful. The value UNDEF-last-June of the previous section is disambiguated by calculating the June before the dct. More complex are even less underspecified values like UNDEF-June. Here, linguistic knowledge is used to disambiguate which June is meant: The tense of the sentence is determined by using the part-of-speech information of the tokens and checking the semantics of the verbs in the sentence. This method identifies whether a sentence is past, present, or future tense. E.g., the tense of the sentence *In June, new results will be published* will be determined to be future tense and the new value UNDEF-next-June can be assigned instead of UNDEF-last-June if past tense was identified. Such values are then disambiguated using the methods described above.

If the reference time is assumed to be the previously mentioned date all previous extracted Timex3 are checked to be of the type *Date*. The value v_{ref} of the closest previously mentioned date is then used for further disambiguation. For example, UNDEF-REF-last-year is calculated by subtracting one year from v_{ref}. This can result in a specific day but also in a specific quarter if the last mentioned timex was a quarter.

The last post-processing step is to remove all extracted timex annotations that are invalid. Invalid are all expressions that are included in other expressions. For instance, having the phrase *June 11* the whole phrase is found by a rule as well as just *June*. Since *June* is in *June 11*, it is removed.

3 Evaluation

In this section, we outline the evaluation of HeidelTime and compare our results with other systems that participated in the TempEval-2 challenge Task A for English. For this challenge, we developed two rule sets, one precision- and one recall-optimized set, reflecting the user's choice between precision and recall. The first set consists of 43 rules, 25 for dates, and 6 for times, durations, and sets, respectively. The recall-optimized rule set contains two more rules, one for dates and one for durations. These rules are very general and thus negatively influence precision.

Our results for the extraction in the two runs are shown in Figure 2 together with the results of the other participating systems. As one can see, both our runs achieved the best F-score results (86%) with a precision of 90% (82%) and a recall of 82% (91%) for the two sets.

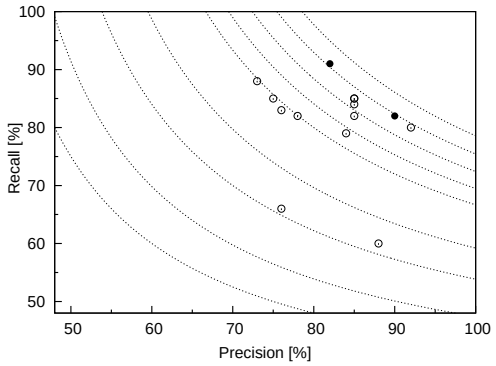

Figure 2: Performance of participating systems with an F-score contour for reference. Our runs are shown as full circles.

HeidelTime, with the precision-optimized rule set, was the best system in assigning the value attributes (85% values are assigned correctly). In addition, the type attribute was correctly assigned to 96% of the extracted expressions.

4 Conclusions

HeidelTime achieves high quality results for the extraction and normalization of temporal expressions. The precision-optimized rule set achieved the best results for interpreting the semantics of the temporal expressions. In our opinion, this aspect, i.e., assigning the correct value attribute, is crucial since the value is used for further analysis of the documents, e.g., when ordering events or doing a temporal analysis of documents.

The rule-based approach makes it possible to include further knowledge easily, e.g., to assign temporal information directly to historic events.

References

James Pustejovsky and Marc Verhagen. 2009. SemEval-2010 Task 13: Evaluating Events, Time Expressions, and Temporal Relations (TempEval-2). In *Proceedings of the Workshop on Semantic Evaluations (SEW-2009)*, pages 112–116. ACL.

Frank Schilder and Christopher Habel. 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*, pages 65–72. ACL.

Jannik Strötgen, Michael Gertz, and Pavel Popov. 2010. Extraction and Exploration of Spatio-Temporal Information in Documents. In *GIR '10*, pages 1–8. ACM.