A Robust Shallow Temporal Reasoning System

Ran Zhao Quang Xuan Do Dan Roth
Computer Science Department
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{ranzhao1,quangdo2,danr}@illinois.edu

Abstract

This paper presents a demonstration of a temporal reasoning system that addresses three fundamental tasks related to temporal expressions in text: extraction, normalization to time intervals and comparison. Our system makes use of an existing state-of-the-art temporal extraction system, on top of which we add several important novel contributions. In addition, we demonstrate that our system can perform temporal reasoning by comparing normalized temporal expressions with respect to several temporal relations. Experimental study shows that the system achieves excellent performance on all the tasks we address.

1 Introduction

Performing temporal reasoning with respect to temporal expressions is important in many NLP tasks such as text summarization, information extraction, discourse understanding and information retrieval. Recently, the Knowledge Base Population track (Ji et al., 2011) introduced the temporal slot filling task that requires identifying and extracting temporal information for a limited set of binary relations such as (*person*, *employee_of*), (*person*, *spouse*). In the work of (Wang et al., 2010), the authors presented the Timely Yago ontology, which extracted and incorporated temporal information as part of the description of the events and relations in the ontology. Temporal reasoning is also essential in supporting the emerging temporal information retrieval research direction (Alonso et al., 2011).

In this paper, we present a system that addresses three fundamental tasks in temporal reasoning:

- *Extraction*: Capturing the extent of time expressions in a given text. This task is based on task A in the TempEval-2 challenge (Verhagen et al., 2010). Consider the following sentence:

  Seventy-five million copies of the rifle have been built since it entered production in February 1947.

  In this sentence, *February 1947* is a basic temporal expression that should be extracted by the extraction module. More importantly, we further extend the task to support also the extraction of *complex temporal expressions* that are not addressed by existing systems. In the example above, it is important to recognize and capture the phrase *since it entered production in February 1947* as another temporal expression that expresses the time period of the *manufacturing* event (triggered by *built*.) For the best of our knowledge, this extension is novel.
- *Normalization*: Normalizing temporal expressions, which are extracted by the extraction module, to a canonical form. Our system normalizes temporal expressions (including complex ones) to time intervals of the form [*start point*, *end point*]. The endpoints follow a standard date and time format: *YYYY-MM-DD hh:mm:ss*. Our system accounts for an input reference date when performing the normalization. For example, given March 20^{th}, 1947 as a reference date, our system normalizes the temporal expressions extracted in the example above as follows: [*1947-02-01 00:00:00*, *1947-02-28 23:59:59*] and [*1947-02-01 00:00:00*, *1947-03-20 23:59:59*], respectively.
- *Comparison*: Comparing two time intervals (i.e. normalized temporal expressions). This module identifies the temporal relation that holds be-

Proceedings of the NAACL-HLT 2012: Demonstration Session, pages 29–32,
Montréal, Canada, June 3-8, 2012.

tween intervals, including the *before*, *before-and-overlap*, *containing*, *equal*, *inside* , *after* and *after-and-overlap* relations. For example, when comparing the two normalized time intervals above, we get the following result: [*1947-02-01 00:00:00*, *1947-02-28 23:59:59*] is *inside* [*1947-02-01 00:00:00*, *1947-03-20 23:59:59*].

There has been much work addressing the problems of temporal expression extraction and normalization, i.e. the systems developed in TempEval-2 challenge (Verhagen et al., 2010). However, our system is different from them in several aspects. First, we extend the extraction task to capture complex temporal expressions. Second, our system normalizes temporal expressions (including complex ones) to time intervals instead of time points. Finally, our system performs temporal comparison of time intervals with respect to multiple relations. We believe that with the rapid progress in NLP and IR, more tasks will require temporal information and reasoning, and a system that addresses these three fundamental tasks well will be able to support and facilitate temporal reasoning systems efficiently.

2 The System

2.1 Temporal Expression Extraction

We built the temporal expression extraction module on top of the Heideltime system (Strötgen and Gertz, 2010) to take advantage of a state-of-the-art temporal extraction system in capturing basic expressions. We use the Illinois POS tagger[1] (Roth and Zelenko, 1998) to provide part-of-speech tags for the input text before passing it to HeidelTime. Below is an example of the HeidelTime output of the example in the previous section:

> Seventy-five million copies of the rifle have been built since it entered production in <TIMEX3 tid="t2" type="DATE" value="1947-02">February 1947</TIMEX3>

In this example, HeidelTime captures a basic temporal expression: *February 1947*. However, HeidelTime cannot capture the complex temporal expression *since it entered production in February 1947*, which expresses a period of time from February 1947 until the document creation time. This is actually the time period of the manufacturing event

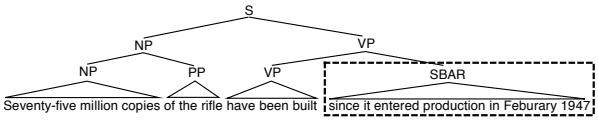


Figure 1: The SBAR constituent in the parse tree determines an extended temporal expression given that *in February 1947* is already captured by HeidelTime.

(triggered by *built*). To capture complex phrases, we make use of a syntactic parse tree[2] as illustrated in Figure 1. A complex temporal expression is recognized if it satisfies the following conditions:

- It is covered by a PP or SBAR constituent in the parse tree.
- The constituent starts with a temporal connective. In this work, we focus on an important subset of temporal connectives, consisting of *since*, *between*, *from*, *before* and *after*.
- It contains at least one basic temporal expression extracted by HeidelTime.

In addition, our extraction module also handles holidays in several countries. For example, in the sentence "The gas price increased rapidly after Christmas.", we are able to extract two temporal expressions *Christmas* and *after Christmas*, which refer to different time intervals.

2.2 Normalization to Time Intervals

Our system normalizes a temporal expression to a time interval of the form [*start point*, *end point*], where *start point* $\leq$ *end point*. Each time endpoint of an interval follows a standard date and time format: *YYYY-MM-DD hh:mm:ss*. It is worth noting that this format augments the date format in TimeML, used by HeidelTime and other existing systems. Our date and time format of each time endpoint refer to an absolute time point on a universal timeline, making our time intervals absolute as well. Furthermore, we take advantage of the predicted temporal value of each temporal expression from the HeidelTime output. For instance, in the HeidelTime output example above, we extract *1947-02* as the normalized date of *February 1947* and then convert it to the interval [*1947-02-01 00:00:00*, *1947-02-28 23:59:59*]. If HeidelTime cannot identify an exact date, month or year, we then resort to our own temporal normalizer,

[1] http://cogcomp.cs.illinois.edu/page/software_view/POS

[2] We use nlparser (Charniak and Johnson, 2005)

which consists of a set of conversion rules, regarding to the document creation time of the input text. An interval endpoint can get infinity value if its temporal boundary cannot be specified.

2.3 Comparison

To compare two time intervals (i.e. normalized temporal expressions), we define six temporal relations: *before*, *before-and-overlap*, *contains*, *equals*, *inside*, *after* and *after-and-overlap*. The temporal relation between two normalized intervals is determined by a set of comparison rules that take the four interval endpoints into consideration. For example, $A = [s_A, e_A]$ *contains* $B = [s_B, e_B]$ if and only if $(s_A < s_B) \wedge (e_A > e_B)$, where s and e are intervals start and end points, respectively.

3 Experimental Study

In this section, we present an evaluation of our extended temporal extractor, the normalizer and the comparator. We do not evaluate the HeidelTime temporal extractor again because its performance was reported in the TempEval-2 challenge (Verhagen et al., 2010), where it achieved 0.86 F_1 score on the TimeBank data sets (Pustejovsky et al., 2003).

3.1 Data Preparation

We focus on scaling up temporal systems to deal with complex expressions. Therefore, we prepared an evaluation data set that consists of a list of sentences containing at least one of the five temporal connectives *since*, *betwen*, *from*, *before* and *after*. To do this, we extract all sentences that satisfy the condition from 183 articles in the TimeBank 1.2 corpus[3]. This results in a total of 486 sentences. Each sentence in the data set comes with the document creation time (DCT) of its corresponding article. The second and the third columns of Table 1 summarize the number of sentences and appearances of each temporal connective.

We use this data set to evaluate the extended temporal extractor, the normalizer and also the comparator of our system. We note that although this data set is driven by our focused temporal connectives, it does not lose the generality of evaluating the normalization and comparison modules because the sentences in this data set also contain many basic temporal expressions. Moreover, there are many cases where the connectives in our data are not actually temporal connectives. Our system is supposed to not capture them as temporal expressions. This is also reflected in the experimental results.

[3]http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2006T08

Connective	# sent.	# appear.	Prec	Rec	F_1
since	31	31	1.0	1.0	1.0
between	32	33	1.0	1.0	1.0
from	340	366	0.8	1.0	0.89
before	33	33	0.8	1.0	0.89
after	78	81	0.72	1.0	0.84
Avg.			0.86	1.0	0.92

Table 1: The performance of our extended temporal extractor on complex expressions which contain at least one of the connectives shown in the first column. These expressions cannot be identified by existing systems.

Module	Correct	Incorrect	Acc
Normalizer	191	16	0.92
Comparator	191	0	1.0

Table 2: The performance of the normalization and comparison modules. We only compare the 191 correctly identified time intervals with their corresponding document creation time.

3.2 Experimental Results

We report the performance of our extended temporal extraction module using precision, recall and F_1 score as shown in the last three columns of Table 1. We evaluate the normalization module on the correctly extracted temporal expressions, including basic expressions captured by HeidelTime and the extended expressions identified by our extractor. A normalization is correct if and only if both time interval endpoints are correctly identified. We study the comparison module by evaluating it on the comparisons of the correctly normalized expressions against the corresponding DCT of the sentences from which they are extracted. Because the normalization and comparison outputs are judged as correct or incorrect, we report the performance of these modules in accuracy (Acc) as shown in Table 2. Overall, the experimental study shows that all modules in our system are robust and achieve excellent performance.

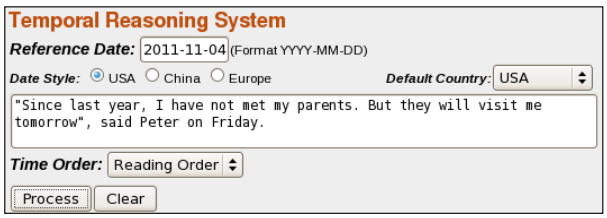
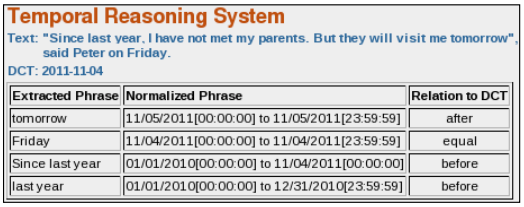


Figure 2: A screenshot of the input panel.



Extracted Phrase	Normalized Phrase	Relation to DCT
tomorrow	11/05/2011[00:00:00] to 11/05/2011[23:59:59]	after
Friday	11/04/2011[00:00:00] to 11/04/2011[23:59:59]	equal
Since last year	01/01/2010[00:00:00] to 11/04/2011[00:00:00]	before
last year	01/01/2010[00:00:00] to 12/31/2010[23:59:59]	before

Figure 3: A screenshot of the output panel.

4 The Demonstration

4.1 Visualization

We have implemented our system in a web-based demo[4]. Figure 2 shows a screenshot of the input panel of the system. The input panel includes a main text box that allows users to input the text, and some other input fields that allow users to customize the system's outputs. Among the fields, the reference date serves as the document creation time (DCT) of the input text. All temporal expressions captured from the text will be normalized based on the reference date and compared also to the reference date as illustrated in Figure 3.

4.2 Script Outline

First, we will give an overview of existing temporal reasoning systems. Then we will introduce the novel contributions of our system. After that, we will go over our web-based demonstration, including (i) the input panel: reference date and the text to be analyzed, and (ii) the output panel: the extracted basic and extended temporal expressions, the normalized intervals, and the comparison results.

5 Conclusions

In this demonstration paper, we introduced a temporal reasoning system that addresses three fundamental problems related to temporal expressions in text, including extraction, normalization and comparison. Our system consists of a temporal expression extractor capable of dealing with complex temporal phrases, a time interval normalizer and a time interval comparator. The experimental study shows that our system achieves a high level of performance, which will allow it to support other systems that require complicated temporal reasoning.

[4]http://cogcomp.cs.illinois.edu/page/demo_view/TempSys

Acknowledgement

This research is supported by the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053 and the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. The second author also thanks the Vietnam Education Foundation (VEF) for its sponsorship. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the VEF, ARL, DARPA, AFRL, or the US government.

References

Omar Alonso, Jannik Strötgen, Ricardo Baeza-Yates, and Michael Gertz. 2011. Temporal information retrieval: Challenges and opportunities. In *TWAW*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2011. Overview of the tac2011 knowledge base population track. In *TAC*.

James Pustejovsky, Jose Castano, Robert Ingria, Roser Sauri, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003. Timeml: Robust specification of event and temporal expressions in text. In *IWCS-5*.

D. Roth and D. Zelenko. 1998. Part of speech tagging using a network of linear separators. In *COLING-ACL, The 17th International Conference on Computational Linguistics*.

Jannik Strötgen and Michael Gertz. 2010. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*.

Yafang Wang, Mingjie Zhu, Lizhen Qu, Marc Spaniol, and Gerhard Weikum. 2010. Timely yago: harvesting, querying, and visualizing temporal knowledge from wikipedia. In *EDBT*.