Temporal Text Extraction and Automated Time-OWL Population

Min Xia[1] and Ju An Wang[2]

Abstract - The Web Ontology Language (OWL) has been widely used as a knowledge representation language for building ontologies in semantic web research. However, it remains a manual process so far to build an OWL file from a natural language query on a domain-specific ontology. In this paper, we present an approach to populate ontological concepts automatically with instances expressed in a natural language query. We focus on temporal expressions and time-related reasoning based on an existing OWL-Time ontology established by Hobbs and Pan [1]. Our approach consists of two steps: First, the time phrases expressed in natural language queries are extracted and translated into a standard format with a domain specific language parser. Second, the standard time expressions are sent to an OWL generator that produces an OWL file for semantic reasoning. With this approach, temporal reasoning can be performed by end users without OWL knowledge using their everyday languages. Various semantic applications can be built with our automatic population approach such as summarizing a story in terms of a time-line, extracting and chronologically ordering events in a narrative report, generating a project plan or time-line based on its user requirements, and scheduling classes, meetings, and events by a software agent, to name just a few.

Keywords: Ontology, OWL-Time, ANTLR

1 INTRODUCTION

In the field of natural language processing (NLP) there is a moving trend to use an ontology to help represent and manipulate syntax and semantics of a natural language. An ontology captures the essential knowledge of a domain, including basic concepts and relations among them, such that the domain knowledge can be automatically processed. NLP tools and applications can use ontologies and knowledge bases built from ontologies as input to generate desired results. A great variety of general-purpose ontologies and domain-specific ontologies exist in literature.

Human being understand a natural language because they recognize its concepts and their meanings. They can reason about cause and effect, resolving explicit or implicit confliction among various constituents with knowledge acquired from learning or experience. For instance, a secretary can arrange multiple meetings without time confliction. A department chair can schedule various classes without confliction in terms of time and location. All these time-related scheduling work can be done with common sense. To automate these tasks by a computer, however, require that the computer program fully understand all the constraints and consequences of different scheduling options. At a minimum, the computer must be able to understand all the time-related queries in a natural language. One solution is to capture all time-related knowledge in an ontology, so that a computer can refer to this ontology to make a schedule or arrange time-sensitive events automatically.

Hobbs and Pan [1, 4] built an OWL Time ontology with applications in semantic web. However, when a user makes query to the ontology, she or he has to populate time phrases and expressions manually into an OWL file before consulting the time ontology. In this paper, we focus on automatically populating the OWL file with temporal expressions and time phrases extracted from sentences in a natural language. Our contribution is to provide a seamless integration between a natural language query related with time and a time-related reasoning through an inference engine associated with the Time ontology.

[1]Southern Polytechnic State University, 1100 S Marietta Pkwy, Marietta, GA 30060,mxia@spsu.edu

[2]Southern Polytechnic State University, 1100 S Marietta Pkwy, Marietta, GA 30060,jwang@spsu.edu

Time plays a very important role in our lives. It is interesting to make a machine capable of recognizing and understanding temporal events and time expressions in natural language text. For example, if "John has a meeting A at 2 pm tomorrow lasting two hours and a meeting B tomorrow at 3 pm lasting 1 hour," will this be possible? By commonsense we know John has a schedule confliction. But how could a machine recognize this? This is partially the challenge of machine intelligence in natural language understanding. In this research we build a model named Time Ontology Population Model (TOPM) and we use it to hook up temporal text and Time Ontology Web Language. The ultimate goal is to make a computer to generate a project plan or time line from a software user requirements specification in natural language. As a first step towards this goal, we have built an ontology with formal representations of a set of concepts within a time domain and the relationships between those concepts. With the existing well-defined time relationship and their associated events, we believe that a software agent could make reasonable project plans, meeting schedules, or other time-related automation a reality. The rest of the paper is organized in the follows: In Section 2, we present our population model; Section 3 presents a simple parser with examples; Section 4 describes the procedure of populating OWL file with time phrases in English; and finally in Section 5, we summarize our work and discuss some future work along this direction.

2 TIME ONTOLOGY POPULATION MODEL

The purpose of Time Ontology Population Model (TOPM) is to populate an OWL file with time phrases extracted from a natural language text. There are three steps to accomplish this: First it identifies the temporal instance from texts in English. Second, it extracts and translates it into pre-defined standard formats that machine can recognize. Third, it populates the time instance into predefined TIME Ontology Web Language for machine understanding.

Assume that there are a set of events, namely `E = {e1, …, en}`, with each `ei` presents the event being described in a natural languag text. Let `C = {c1, …, cn}` be a set of time instances like the start time of an event, the duration of an event, or the end time of an event. For every time instance, we have a pair of entities, for instance, I = [`n1, v1`] corresponding to `c1`. We defined TOPM in three steps: Classification of time phrases and building Time grammar, Translation and Normalization, Populating Normalized time instances in Time Ontology.

(i) **Extracting and building grammar based on classifications**

The natural language text is the initial resource of ontology population and it may contain many entities that are not related to time. There may have a lot of information or entities in a piece of text, and thus machine needs to parse these entities to understand their meaning and relations. Our model focuses on Time concepts, so temporal entities must be recognized and be extracted from the text (e.g. today, after, in, month, etc.). These entities may consist a lot of phrases which are in present time. We adopt the domain specific language tool ANTLR [2] as our parser and translator, so we need to classify all of phrase patterns and build the specific grammar for these patterns based on these temporal entities.

(ii) **Translation and normalization**

The translator will translate and normalize the text instance extracted at step (i) following the standard grammar format. For example, the text "John will have a meeting in the afternoon at 2 clock the day after today" and the reference day is 2008-10-23. The machine will translate this text into "John will have a meeting at 2008-10-24 2:00 PM". Generally speaking, this is to translate temporal entities into some a numerical format based on a reference time. In other words, this is to translate relative time into absolute time. Then the time should be normalized, that is, further changing it into a standard timestamp like 2008-10-24 2:00 PM.

(iii) **Populate the normalized time instances in Time Ontology**

With the existed OWL-Time ontology established by Hobbs and Pan [1, 4], the major relations of time entities used in everyday life have been defined. Our work is to assign time instances I into Time entities C combined with appropriate Event E having a pair if values [`ni, vi`]. For example, in step (ii) we have recognized that time entities from the original text and translated it into a normalized format, such as ,John has a meeting on 2008-10-23. We feed this information into Ontology so that further reasoning could be conducted.

In this paper we only address temporal information extraction, translation and Time Ontology automated population, we do not talk Ontology reasoning on the relations of events. The architecture of our model is depicted as Figure 1 below.

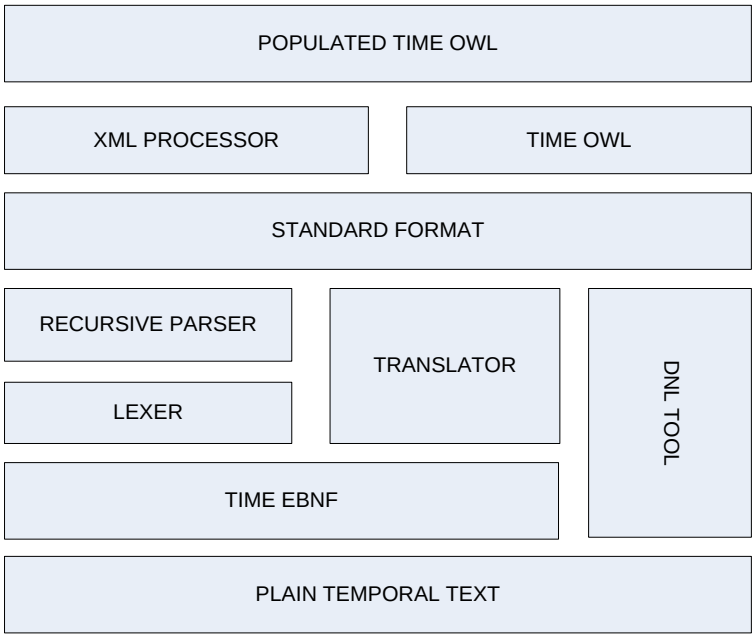

Figure 1 Time Ontology Population Model Structure

3 METHODOLOGY AND EXAMPLES

As indicated previously, our Time Ontology Population Model (TOPM) only handles time concepts with events from an English text, so the short phrases like 'the day after tomorrow' needs to be recognized and extracted from the text for further processing. To this end, we use a domain specific language tool ANTLR [2] to design the time-phrase grammar. ANTLR has the ability to parse and translate texts with its predicated-LL(*)parsing grammars, avoiding ambiguous decisions during parsing. From the linguistic point of view, the time entities under consideration can be divided into prepositions, nouns, and adverbs. Different temporal phrases can be consisted by similar time entities based on the sequence of combinations of these prepositions, nouns, and adverbs. Therefore, we define all of these time entities as TOKENS in the time grammar. We classify phrase types by combinations, and build a number of grammar rules in the time grammar based on how many types of combinations existed in the text. Machine can recognize all kinds of temporal phrases by time grammar rules within ANTLR. Below we give two examples showing that the time entities could be expressed in different temporal phrases with different combinations in the text.

Rule 1: Jason will attend a conference the day [noun] after [prep] tomorrow [noun].

Rule 2: John has attended a meeting two days [noun] ago [adverb].

In our project, we have collected all of the common time entities and the possible combinations of these time entities used in human narratives. In the time grammar, tokens include time entities, grammar rules, and combinations of time entities. After recognizing and extracting temporal phrases from a piece of English text, the machine needs to translate target texts to standard format that will be the input for the Time ontology. We embedded Java code to our defined Time Grammar to do the translation work. Before starting to translate, we need to choose a reference time point as a time baseline. For example, if we choose 2008-10-22 as the reference time (the current time) for the previous examples, the two rules displayed above will be translated to:

Rule 1: Jason will attend a conference on 2008-10-24. (the day after tomorrow.)

Rule 2: John has attended a meeting before 2008-10-20. (two days ago.)

The translated format 2008-10-24 is a standard time format commonly used in modern programming language and operating systems, thus it can be easily understood by the software agents using the Time Ontology.

4 EXPORTING TRANSLATED ENTITIES TO TIME OWL FILES

TIME OWL has defined a framework to describe the relations of all time entities. It expresses the relations of entities by an OWL XML file. All time events can be expressed by TIME OWL expressions. Let us see one class in a TIME OWL file as an example:

```
<time-entry:Time rdf:ID="eventStartTime">
      <time-entry:year rdf:datatype=”Year">null</time-entry:year>
      <time-entry:month rdf:datatype="Month">null</time-entry:month>
      <time-entry:day rdf:datatype="Day">null</time-entry:day>
      <time-entry:hour rdf:datatype="Hour">null</time-entry:hour>
      <time-entry:minute rdf:datatype="Minute">null</time-entry:minute>
  </time-entry:Time>
```

This class describes the event's start time along with some sub-tags for year, month, day, etc. Through our population method, this will be filled with real time values for the event. If the reference time is 2008-10-22, for instance, for the text "Jason will attend a conference the day after tomorrow", we should fill in 2008, 10, and 24 into these sub-tag areas respectively in the previous example. The challenging task is to make the machine understand what does "the day after tomorrow" mean and distinguish what the relations between those temporal tags in the TIME OWL file. Hence we translate the temporal text to a standard format to hook up it with the TIME OWL file. In the example, we translate the English text first to "Jason will attend a conference on 2008-10-24". Then we extract the values of year, month and day. We can easily populate these values into a TIME OWL file with an XML processor. The populated TIME OWL file will be like:

```
<time-entry:Time rdf:ID="conferenceStartTime">
      <time-entry:year rdf:datatype=”Year">2008</time-entry:year>
      <time-entry:month rdf:datatype="Month">10</time-entry:month>
      <time-entry:day rdf:datatype="Day">24</time-entry:day>
      <time-entry:hour rdf:datatype="Hour">00</time-entry:hour>
      <time-entry:minute rdf:datatype="Minute">00</time-entry:minute>
  </time-entry:Time>
```

If we populated all the temporal event information in the TIME OWL file, the reasoning engine of the ontology could be used to derive time plans, schedules, and other time-related conclusions. We will not address the reasoning part in this paper, since there have been many solutions in this area, for instance, Protégé with JESS [9], and Pellet Open Source OWL DL Reasoner [10].

5 CONCLUSION AND FUTURE WORK

Time is critical in our life and time expressions appear in many user requirements and project planning in a software lifecycle. It is challenging for computers of software agents to understand time phrases in an article and take actions accordingly. It is promising to use ontological approach to natural language understanding, especially in processing

time-related natural language texts. With the ontological approach, an OWL file must be created to capture all the time constraints before the ontology inference engine can process the time constraints. So far this OWL file is populated manually, which is a tedious and error-prone process. This paper proposed a solution to automate this population process. Our approach utilizes the domain specific language parser, ANTLR, and translates relative time expressions into absolute time expressions based on an inferred time baseline.

We have built a Time Ontology Population Model (TOPM) for automatically populating TIME Ontology file from an English text containing timing phrases. Our TIME grammar captures the most common time phrases and time expressions in English, but it is certainly worthwhile to see the expressive power of this TIME grammar in real applications. Some fuzzy meaning in a natural language could make such automatic population impossible, for instance, we could not determine exactly on which day the meeting took place in the following situation:

"John had a meeting a few days ago." Or "John will have a meeting in a few days."

It is natural to introduce fuzzy representation and fuzzy reasoning in the language understanding and time OWL file population.

The efficiency of the recognizer and translator is another thing we have to consider if we want to make time queries to the TIME ontology and draw conclusions in a real-time fashion. We are working to design more expressive and reusable rules in the time grammar to enhance its expressive power and the runtime efficiency.

Another future work could be improving the usability of the population by adding spelling/grammar checking and automatic correction. In this way, the system could automatically correct typo or grammatical errors before parsing the English text.

REFERENCES

[1] Time Ontology in OWL: http://www.w3.org/TR/2006/WD-owl-time-20060927/, September 2006.

[2] Terence Parr, "The Definitive ANTLR Reference – Build Domain Specific Languages", Pragmatic Bookshelf, ISBN-10 0-9787392-5-6.

[3] Michael Witbrock, Kathy Panton, Stephen L. Reed, Dave Schneider, Bjørn Aldag, Mike Reimers and Stefano Bertolo. "Automated OWL Annotation Assisted by a Large Knowledge Base". 2004 Workshop on Knowledge Markup and Semantic Annotation at the 3rd International Semantic Web Conference.

[4] Jerry R. Hobbs and Feng Pan. 2004. "An Ontology of Time for the Semantic Web". ACM Transactions on Asian Language Processing (TALIP): Special issue on Temporal Information Processing, Vol. 3, No. 1, March 2004, pp. 66 -85.

[5] Bernardo Magnini, Emanuele Pianta, Octavian Popescu and Manuela Speranza. "Ontology Population from Textual Mentions:Task Definition and Benchmark". Proceedings of the 2nd Workshop on Ontology Learning and Population, pages 26–32,Sydney, July 2006.

[6] Feng Pan. 2007. "Representing Complex Temporal Phenomena for the Semantic Web and Natural Language". Ph.D. Thesis. Computer Science Department, University of Sothern California.

[7] Inderjeet Mani and James Pustejovsky. 2004. "Temporal Discourse Models for Narrative Structure". ACL 2004 Workshop on Discourse Annotation, Barcelona, Spain, 2004.

[8] James Pustejovsky, Robert Knippen, Jessica Littman and Roser Saurí. Forthcoming. "Temporal and Event Information in Natural language Text". Language Resources and Evaluation. 2005, VOL 39; NUMBER 2-3, pages 123-164.

[9] Protégé. http://protege.stanford.edu/, November, 2008.

[10] Pellet: The Open Source OWL DL Reasoner. http://clarkparsia.com/pellet/.

Min Xia

Graduate Student of Computer Science Department, School of Computing and Software Engineering, Southern Polytechnic State University, mxia@spsu.edu

Ju An Wang

Professor of Information Technology Department, School of Computing and Software Engineering, Southern Polytechnic State University, jwang@spsu.edu