Deriving Chronological Information from Texts through a Graph-based Algorithm

Cosmin Adrian Bejan
Human Language Technology Research Institute
The University of Texas at Dallas
Richardson, TX 75083-0688, USA
ady@hlt.utdallas.edu

Abstract

We propose a method of deriving chronological order of events in natural language texts by constraining temporal boundaries associated to events and projecting them on a timeline. The algorithm for constraining event temporal boundaries employs deductive inferences on graph structures that encode temporal information extracted from texts. To this end, we consider the task of extracting events from texts. The results obtained show improvement over the previous event extraction systems when evaluated on the same corpus.

Introduction

When a story is written, multiple events are described. They occur at different times and, because of this, each topic has its own chronology. Our algorithm of deriving chronological order of events builds graphical representations of temporal elements extracted from texts and, based on the temporal relations that exist between events and time expressions in graphs, it adjusts the temporal boundaries associated to each event. In our implementation, we used the temporal elements annotated in TimeBank, a corpus with TimeML annotations. TimeML (Pustejovsky *et al.* 2005) is a specification language for annotating time expressions, events, and temporal relations holding between these elements.

Placing events on a timeline is useful in various applications such as question answering, information extraction and multidocument summarization. In question answering, for instance, the timeline can play the role of a temporal index of events over the entire document collection. Using this functionality, a question answering system not only is able to answer simple questions such as what event(s) happened at a specific time interval, but it also can answer complex temporal questions that require solving the temporal order of events across different documents.

Algorithm for Projecting Events on a Timeline

(Step1) Building a Graph-based Representation: We define a *time-event graph* associated to a text as a temporal representation of events and time expressions encoded in the text. In this graph, the nodes correspond to events and time expressions while the directed edges correspond to TLINK relations. The TLINK (or Temporal Link) relations represent a class of temporal relations defined in TimeML that can link two events or an event and a time expression.

The idea of representing temporal information using graph structures is not new. (Allen 1983) proposed as framework for reasoning with time an algebra of temporal intervals, in which the elements of the algebra represent a disjunction of 13 relations[1] between temporal intervals. Allen also proposed a constraint propagation algorithm for determining the deductive closure of the temporal relations. In order to avoid difficult search problems along temporal chains of nodes, we convert all the TLINK relations of time-event graphs into Allen's relations as described in (Verhagen 2005), and then we applied a simplified version of his closure algorithm.

(Step2) Transforming Interval-based Temporal Relations into Point-based Relations: We reduce the problem of finding chronological order of events to the problem of assigning temporal intervals to events in a text. Therefore, we associate to every event e in a time-event graph a temporal interval by specifying its starting and ending points, denoted as $S(e)$ and $E(e)$ respectively. The purpose of this algorithm is to approximate $S(e)$ and $E(e)$ as accurate as possible, and, in the same time, to preserve all temporal constraints in which the event e is involved.

In order to independently constrain these temporal boundaries, we transform the interval-based relations from time-event graphs into point-based relations. For this operation, we consider the framework proposed by (Vilain, Kautz, & van Beek 1990) using the set of binary relations $\{<, =, >\}$ defined in time point algebra. After this process is executed, every event node e that is related to a time node t will have associated a set of point-based temporal relations $R(e_t)$.

(Step3) Deriving Temporal Boundaries of Events: We present a constraint propagation algorithm for deriving temporal boundaries of events. The algorithm (illustrated in Figure 1) is applied to every event node e from a time-event

[1]The Allen's 13 relations are: BEFORE ($<$), AFTER ($>$), MEETS (M), MET-BY (MI), OVERLAPS (O), OVERLAPPED-BY (OI), STARTS (S), STARTED-BY (SI), DURING (D), CONTAINS (DI), FINISHES (F), FINISHED-BY (FI), EQUALS ($=$).

```
1. Expansion Step
1.1 for every time node t, $t \xrightarrow{r} e$
    $S(e_t)$ = the smallest time point that satisfies temporal relations in $R(e_t)$
    $E(e_t)$ = the largest time point that satisfies temporal relations in $R(e_t)$
2. Contraction Step
2.1 for all time nodes t, $t \xrightarrow{=} e_t$
    $S(e) = agreement_t\{S(e_t) \mid S(e) = S(e_t)\}$
    $E(e) = agreement_t\{E(e_t) \mid E(e) = E(e_t)\}$
    $L.add(S(e) = S(e_t))$
    $L.add(E(e) = E(e_t))$
    $stop\ searching\ for\ S(e)\ and\ E(e)$
2.2 for all time nodes t, $t \xrightarrow{M} e_t$
    $S(e) = agreement_t\{E(e_t) \mid S(e) = E(e_t)\}$
    $L.add(S(e) = E(e_t))$
    $stop\ searching\ for\ S(e)$
2.3 for all time nodes t, $t \xrightarrow{MI} e_t$
    $E(e) = agreement_t\{S(e_t) \mid E(e) = S(e_t)\}$
    $L.add(E(e) = S(e_t))$
    $stop\ searching\ for\ E(e)$
2.4 for all time nodes t, $t \xrightarrow{S} e_t$
    $S(e) = agreement_t\{S(e_t) \mid S(e) = S(e_t)\}$
    $L.add(E(e) > max_t\{E(e_t) \mid E(e) > E(e_t)\})$
    $stop\ searching\ for\ S(e)$
2.5 for all time nodes t, $t \xrightarrow{F} e_t$
    $E(e) = agreement_t\{E(e_t) \mid E(e) = E(e_t)\}$
    $L.add(S(e) < min_t\{S(e_t) \mid S(e) < S(e_t)\})$
    $stop\ searching\ for\ E(e)$
2.6 for all time nodes t, $t \xrightarrow{SI} e_t$
    $S(e) = agreement_t\{S(e_t) \mid S(e) = S(e_t)\}$
    $E(e) = min_t\{E(e_t) \mid E(e) < E(e_t)\}$
    $L.add(E(e) < min_t\{E(e_t) \mid E(e) < E(e_t)\})$
    $stop\ searching\ for\ S(e)$
2.7 for all time nodes t, $t \xrightarrow{FI} e_t$
    $E(e) = agreement_t\{E(e_t) \mid E(e) = E(e_t)\}$
    $S(e) = max_t\{S(e_t) \mid S(e) > S(e_t)\}$
    $L.add(S(e) > max_t\{S(e_t) \mid S(e) > S(e_t)\})$
    $stop\ searching\ for\ E(e)$
2.8 for all time nodes t, $t \xrightarrow{O} e_t$
    $S(e) = max\{max_t\{S(e_t) \mid S(e) > S(e_t)\}, S(e)\}$
    $L.add(S(e) > max_t\{S(e_t) \mid S(e) > S(e_t)\})$
    $L.add(S(e) < min_t\{E(e_t) \mid S(e) < E(e_t)\})$
    $L.add(E(e) > max_t\{E(e_t) \mid E(e) > E(e_t)\})$
2.9 for all time nodes t, $t \xrightarrow{OI} e_t$
    $E(e) = min\{min_t\{E(e_t) \mid E(e) < E(e_t)\}, E(e)\}$
    $L.add(S(e) < min_t\{S(e_t) \mid S(e) < S(e_t)\})$
    $L.add(E(e) > max_t\{S(e_t) \mid E(e) > S(e_t)\})$
    $L.add(E(e) < min_t\{E(e_t) \mid E(e) < E(e_t)\})$
2.10 for all time nodes t, $t \xrightarrow{DI} e_t$
    $S(e) = max\{max_t\{S(e_t) \mid S(e) > S(e_t)\}, S(e)\}$
    $E(e) = min\{min_t\{E(e_t) \mid E(e) < E(e_t)\}, E(e)\}$
    $L.add(S(e) < max_t\{S(e_t) \mid S(e) > S(e_t)\})$
    $L.add(E(e) < min_t\{E(e_t) \mid E(e) < E(e_t)\})$
2.11 for all time nodes t, $t \xrightarrow{D} e_t$
    $L.add(S(e) < min_t\{S(e_t) \mid S(e) < S(e_t)\}$
    $L.add(E(e) > max_t\{E(e_t) \mid E(e) > E(e_t)\})$
2.12 for all time nodes t, $t \xrightarrow{<} e_t$
    $S(e) = max\{max_t\{E(e_t) \mid S(e) > E(e_t)\}, S(e)\}$
    $L.add(S(e) > max_t\{E(e_t) \mid S(e) > E(e_t)\})$
2.13 for all time nodes t, $t \xrightarrow{>} e_t$
    $E(e) = min\{min_t\{S(e_t) \mid E(e) < S(e_t)\}, E(e)\}$
    $L.add(E(e) < min_t\{S(e_t) \mid E(e) < S(e_t)\})$
```

Figure 1: Algorithm for approximating boundaries of an event e.

1 Syntactic constituent label;
2 Binary feature indicating whether the constituent spans one or multiple words;
3 The word, lemma and part of speech of the constituent head;
4 Ternary feature indicating whether the part of speech of the constituent head is verb, noun or adjective;
5 Binary feature indicating whether the lemma of the constinuent head is a nominalization or not. To extract this feature, we built a list with all nominalizations iterating over all WordNet verbs;
6 Constituent adjacent word and part of speech unigrams;
7 Binary feature indicating whether the lemma of the constituent head is in the list of events and their corresponding WordNet hyponyms created in the preprocessing step using all events from the training set;
8 Binary feature indicating whether the constituent strictly identifies a name;
9 Binary feature indicating whether the constituent identifies a partial name.

Figure 2: Features for extracting events from texts.

Event Extraction Systems	*Event Detection*		*Event Classification*	
	1.1	*1.2*	*1.1*	*1.2*
(Sauri et al., 2005)	—	80.12	—	—
(Boguraev and Ando, 2005)	80.30	—	64.00	—
Our system	81.11	82.94	77.54	78.27

Table 1: Systems results for event detection and classification.

graph. It consists of two main steps: an *expansion step* and a *contraction step*. In the expansion step, because a disjunction of convex temporal relations can exist between an event e and a time t, the interval e_t associated to the $e \xrightarrow{r} t$ edge is expanded as much as possible. On the other hand, in the contraction step, the event boundaries of an event e are restrained as much as the temporal relations that link time nodes t to e permit. The contraction of boundaries is performed by deduction rules applied to every type of relation that exist between the interval determined by $<S(e_t), E(e_t)>$ and t. In cases where the relations determine exactly one of event boundaries or both (steps 2.1 – 2.7), we use the $agreement$ procedure to establish the most specific time expression among all time nodes involved in the temporal relations. The algorithm also checks the temporal consistency among time constraints stored in list L.

Event Extraction

To extract temporal information from texts and to encode it into time-event graphs, we need to develop frameworks trained on resources with TimeML annotations. We describe a learning framework for detecting and classifying events from TimeBank. In this framework, we consider a learning instance as being associated to a constituent from a syntactic parse tree. This approach takes advantage of the syntactic information encoded into parse trees that can be explored by various features. The features we implemented in our event extraction system are described in Figure 2.

We report results for event detection and classification and compare them with results obtained by other two event extraction systems described in (Boguraev & Ando 2005) and (Sauri *et al.* 2005). In our experimental settings, like (Sauri *et al.* 2005), we considered only events expressed as verbs, nouns and adjectives. Also, we performed experiments using SVM and maximum entropy classification models, and, like (Boguraev & Ando 2005), we evaluated the results using micro-averaged F-measure in a 5-fold cross validation scheme. The systems results are listed in Table 1. In order to compare our results with the two previous systems we run experiments on two versions of TimeBank. For event detection, we obtained the best results when using SVM models, whereas the maximum entropy classifiers obtained the best results for event classification. Our system was shown to surpass the results of previous event extraction systems.

Conclusions

We presented a method for placing events on the time dimension. We first described how to build temporal graph structures, then how to obtain point-based relations and, finally, how to apply deductive rules to approximate event boundaries. To automatically build graph structures, we also implemented a framework for extracting events from texts.

References

Allen, J. F. 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM* 26(1):832–843.

Boguraev, B., and Ando, R. K. 2005. TimeML-Compliant Text Analysis for Temporal Reasoning. In *Proceedings of IJCAI-2005*.

Pustejovsky, J.; Ingria, B.; Sauri, R.; Castano, J.; Littman, J.; Gaizauskas, R.; Katz, G.; and Mani, I. 2005. The Specification Language of TimeML. In *The Language of Time: A Reader*.

Sauri, R.; Knippen, R.; Verhagen, M.; and Pustejovsky, J. 2005. Evita: A Robust Event Recognizer for QA Systems. In *Proceedings of HLT/EMNLP*.

Verhagen, M. 2005. Temporal Closure in an Annotation Environment. In *Language Resources and Evaluation*, volume 74.

Vilain, M.; Kautz, H.; and van Beek, P. 1990. Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report. In *Readings in Qualitative Reasoning about Physical Systems*.