Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 2148–2156
Marseille, 11–16 May 2020

Evaluating Information Loss in Temporal Dependency Trees

Mustafa Ocal & Mark A. Finlayson
School of Computing and Information Sciences
CASE Building, Room 362
11200 S.W. 8th Street, Miami, FL 33199 USA
{mocal001, markaf}@fiu.edu

Abstract

Temporal Dependency Trees (TDTs) have emerged as an alternative to full temporal graphs for representing the temporal structure of texts, with a key advantage being that TDTs can be straightforwardly computed using adapted dependency parsers. Relative to temporal graphs, the tree form of TDTs naturally omits some fraction of temporal relationships, which intuitively should decrease the amount of temporal information available, potentially increasing temporal indeterminacy of the global ordering. We demonstrate a new method for quantifying this indeterminacy that relies on solving temporal constraint problems to extract timelines, and show that TDTs result in up to a 109% increase in temporal indeterminacy over their corresponding temporal graphs for the three corpora we examine. On average, the increase in indeterminacy is 32%, and we show that this increase is a result of the TDT representation eliminating on average only 2.4% of total temporal relations. This result suggests that small differences can have big effects in temporal graphs, and the use of TDTs must be balanced against their deficiencies, with tasks requiring an accurate global temporal ordering potentially calling for use of the full temporal graph.

Keywords: Temporal Indeterminacy, TimeML,Temporal Information Extraction

1. Introduction

Extracting a representation of the temporal information in a text is a useful yet challenging task within natural language processing. Representations of temporal information can facilitate **question answering** (Saquete et al., 2004), **information extraction** (Wu et al., 2005), and **summarization** (Liu et al., 2012), among many other tasks.

Researchers have developed several formalisms for representing temporal information expressed in text, each with specific advantages and disadvantages. Temporal representation languages that allow the representation of temporal graphs, such as Allen's temporal algebra (Allen, 1983) or the XML-based Temporal Markup Language (TimeML) (Sauri et al., 2006), are the most flexible schemes for marking of events, temporal expressions, and temporal relations. Temporal graphs expressed in these languages can be used for inference or determining partial orders of events and times. Although some of these languages for temporal graphs are quite expressive (TimeML has 25 relation types) they have certain limitations. First, the generality of temporal graphs makes many operations on them computationally hard. Second, they are less than ideal for visualization purposes, as they are difficult for people to read and understand. Third, they only explicitly provide partial, local orderings of events.

In response to these limitations, Cheng et al. (2007) introduced the idea of the temporal dependency tree (TDT), a more computationally efficient representation where all events are arranged in a tree using only three temporal relation types: BEFORE, AFTER, and OVERLAPS. Both Kolomiyets et al. (2012) and Zhang and Xue (2018a) further improved TDTs with more precise definitions and increased expressivity. TDTs have the advantage that they can be easily computed using adapted dependency parsers, and extracting timelines from TDTs is much easier than from temporal graphs.

Nevertheless, it is intuitive that TDTs should suffer from temporal information loss relative to temporal graphs. TDTs are restricted to a tree form, and so certain temporal relationship that can be expressed in a graph cannot be expressed in the tree (e.g., cycles). Furthermore, all TDT approaches restrict the types of temporal relationships, eliminating even more information. These omissions and restrictions should intuitively result in more indeterminacy in the global ordering of times and events. We demonstrate here a novel method for measuring the indeterminacy of a general temporal graph (a class which includes TDTs), and precisely measure the increase in indeterminacy of TDTs over TimeML graphs. We show that TDTs on average suffer from a 16% increase in indeterminacy on account of their tree form (resulting from the removal of only 2.4% of the temporal relations relative to a TimeML graph), and an additional 14% average increase in indeterminacy on account of their restricted relation set, for an overall 32% average increase in indeterminacy.

The paper is organized as follows. We first discuss related work (Section 2.). We then explain our methods (Section 3.), followed by our results with discussion (Section 4.). We conclude with the contributions of the work (Section 5.).

2. Related Work

2.1. Temporal Algebras & TimeML

Allen's interval algebra was the first attempt to model temporal information in documents (Allen, 1983). It is a calculus for temporal reasoning that defines 13 relations between time intervals. These relations are BEFORE, MEETS, OVERLAPS, STARTS, DURING, and FINISHES, their inverses, and EQUALS. Allen's algebra is what is called a qualitative temporal framework (Barták et al., 2014). In a great deal of later work, Allen's conception of temporal algebras was extended to *quantitative* frameworks, such as Simple Temporal Problems (STPs), Temporal Constraint Satisfaction Problems (TCSPs), Disjunctive Temporal Problems

(DTPs), and Temporal Networks with Alternatives (TNAs) (Barták et al., 2014). Quantitative temporal frameworks allow precise reasoning about the metric temporal distances between time points. Whether qualitative or quantitative, the temporal graphs generated through Allen-like frameworks can be checked for consistency. Allen introduced a constraint propagation algorithm to do just that (Allen, 1983), using his composition table and a shortest path algorithm, although he also showed by counterexample that his algorithm was necessarily incomplete. Later, Vilain et al. (1990) proved that full consistency checking must be NP-complete.

Because of the utility of temporal frameworks for reasoning about time, and also the relevance of time to understanding natural language, researchers have sought to apply these results to text understanding. This requires annotation schemes that would allow a person or a machine to notate the events, time points, and temporal relations expressed in a text.

With regard to expressions of time itself, including expressions of when something happened, how often something occurs, or how long something takes, researchers have developed a sequence of TIMEX annotation schemes (Setzer, 2001; Ferro et al., 2001; Pustejovsky et al., 2003a) for annotating expressions such as *at 4 a.m.* (when), *every 3 weeks* (how often), or *for 2 hours* (how long). Because events are also involved in temporal relations, these approaches developed into schemes for capturing events as well. For example, the Translingual Information Detection, Extraction, and Summarization (TIDES) (Ferro et al., 2001) is an XML-based annotation scheme that integrates TIMEX2 expressions, events, and six types of temporal relations.

Deficiencies in TIDES led to the development of TimeML, another XML-based markup language for annotating temporal information, originally targeted at news articles (Sauri et al., 2006). TimeML added facilities for representing not just Allen's classic temporal relations, but also event coreference (IDENTITY), sub-event structure (aspectual relations) and relations of conditional, hypothetical, or counterfactual nature (subordinating relations). In all there are 25 types of relations (called *links*) in TimeML: 14 temporal, 5 aspectual, and 6 subordinating. TimeML annotations explicitly encode a temporal graph, where the nodes are events or time expressions and edges are temporal links. More precisely, we define a TimeML temporal graph, in accordance with prior work, as follows:

Definition 1. *TimeML Temporal Graph.* A temporal graph is a graph $T = (V_T, E_T)$, where $V_T = \{e_1, e_2, \ldots, e_p, t_1, t_2, \ldots t_q\}$ is a set of events e_i and time expressions t_i, and $E_T = \{l_1, l_2, \ldots, l_r\}$ is a set of temporal links $l_i = (u, v, w)$, a tuple where $u, v \in V_T$ and $w \in \mathscr{L}$, where $\mathscr{L}$ is the set of TimeML link types.

2.2. Temporal Dependency Trees

Temporal dependency trees (TDTs) were first introduced by Cheng et al. (2007), in which they sought to identify temporal relations between events using a sequence labeling model with features from a dependency parse tree. They only used three temporal relations in their model: BEFORE, AFTER, and OVERLAPS. They used an HMM combined with an SVM as a sequence labeling model and achieved 0.75 F_1 score on correctly identifying the link type (as well as the labels NONE or VAGUE) between neighboring event and times.

Later, Kolomiyets et al. (2012) proposed several improvements to TDTs. They re-defined TDTs as a tree where nodes are events and edges are temporal links, and expanded the set of link types to six: BEFORE, AFTER, INCLUDES, IS_INCLUDED, OVERLAPS and IDENTITY. They also proposed a timeline extraction method using TDTs. They evaluated their system with an annotation study: two annotators created TDTs for 100 short narrative texts (fables) and then they measured inter-annotator agreement between them. Using Krippendorff's α as an agreement measure, the annotators achieved agreements of 0.86 for event recognition, 0.82 for link recognition, and 0.70 link label identification. This work generated an annotated corpus of temporal dependency trees, but unfortunately this corpus has not been made public.

Most recently Zhang and Xue (2018a) improved upon Kolomiyets et al.'s approach by including time expressions in the tree structure instead of just events. They used only four relation types (BEFORE, AFTER, OVERLAPS, and INCLUDES), but also included stative events (such as modals), and automatically generated TDTs for a corpus that included both news and narrative genres. They evaluated the reliability of this approach by double annotation, having two annotators manually generate TDTs for 20% of their corpus, achieving F_1 agreement scores of 0.97 on time expression recognition, 0.94 on event recognition, 0.86 on link recognition, and 0.79 on link label identification. Importantly, for the first time, they provided a precise definition of the TDT structure:

Definition 2. *Temporal Dependency Tree.* A temporal dependency tree structure is defined as a 4-tuple (T, E, N, L), where T is a set of [TimeML] temporal expressions, E is a set of [TimeML] events, and N is a set of pre-defined "meta" nodes not anchored to a span of text in the document. [Elements from] T, E, and N are the nodes in the dependency structure, and L is the set of edges in the tree, where each element in L is a temporal link. (Zhang and Xue, 2018a, p. 3099)

A TDT is rooted in a ROOT node which represents the document creation time (DCT). The children of the root are called *meta-nodes*, and represent time points defined relative to the DCT. Three of the meta-nodes defined by Zhang and Xue, and following Sauri et al. (2006), were PAST_REF, PRESENT_REF, and FUTURE_REF. They also defined a fourth ATEMPORAL meta-node for timeless statements. All other nodes in the TDT (descendants of the meta-nodes) represent events or times. Visualizations of a TDT are shown in Figure 1(b) and (c).

As mentioned above, there are 14 TimeML temporal relations. Zhang and Xue mapped the 14 TimeML temporal relations into four relations in their parser, namely BEFORE, AFTER, OVERLAPS$_{tdt}$ and INCLUDES. We use the *tdt* subscript to indicate that their OVERLAPS$_{tdt}$ definition is different than Allen's. In Allen's temporal algebra, when

TimeML Link Types	Zhang and Xue (2018a) Abstracted Link Types
A BEFORE B A IBEFORE B B TERMINATES A	A BEFORE B
A AFTER B A IAFTER B	A AFTER B
A BEGINS B A BEGUN_BY B A ENDS B A ENDED_BY B A SIMULTANEOUS B A IDENTITY B B INITIATES A B CULMINATES A	A OVERLAPS$_{tdt}$ B
A INCLUDES B A DURING_INV B B CONTINUES A	A INCLUDES B
A IS_INCLUDED B A DURING B B REINITIATES A	B INCLUDES A

Table 1: Mapping of 19 TimeML temporal and aspectual link types into 4 abstract temporal link types used by Zhang and Xue (2018a).

two events E_1 and E_2 are related by an OVERLAPS relation (E_1 OVERLAPS E_2), it means the events intersect in time but neither is completely contained in the other. However, in Zhang and Xue's TDTs, OVERLAPS$_{tdt}$ stands for a conjunction of six TimeML temporal relations: BEGINS,BEGUN_BY, ENDS, ENDED_BY, SIMULTANEOUS and IDENTITY.

Zhang and Xue also define a non-temporal DEPENDS_ON link type that is only used to connect each meta-node to their children. Their mapping is shown in Table 1. Since this mapping essentially abstracts the class of temporal relations, we refer to the TDTs defined by (Zhang and Xue, 2018a) as *abstract TDTs*. We refer to TDTs that use the full set of TimeML temporal and aspectual link types as *full TDTs*.

2.3. Corpora

We used three manually annotated TimeML corpora in this work: TimeBank 1.2, the ProppLearner corpus, and the N2 corpus, all in English. TimeBank 1.2 is a collection of news stories from various sources such as Public Radio International, Voice of America, and the New York Times (Pustejovsky and Lazo, 2003). The N2 corpus contains Islamic Extremist stories including Inspire Magazine that were annotated with TimeML, among other things (Finlayson and Corman, 2014). Finally, the ProppLearner corpus was developed to enable the machine learning of Vladimir Propp's morphology of Russian folktales, contains TimeML annotations (Finlayson, 2017). We provide statistical information about the corpora in Table 2.

We use TimeML corpora here because there is no corpus that has both TimeML temporal graphs and TDTs annotated on it. To our knowledge, there is only one publicly available corpus of TDTs, the Temporal Dependency Tree corpus, which is an automatically annotated collection of Chinese news reports and fairy tale stories (Zhang and Xue, 2018). The TDTs in that corpus were generated using an adapted dependency parser (Robaldo et al., 2011), but the texts do not have corresponding TimeML graphs. We use this corpus only to compare the raw indeterminacy of automatically computed TDTs to our TimeML-derived TDTs.

3. Approach

In order to evaluate what is lost when moving from TimeML temporal graphs to TDTs, we need texts which have both TimeML and TDT annotations. This data could be generated in several different ways. First, we could automatically generate TimeML and TDT annotations for texts using TimeML and TDT parsers. This is problematic because of the great deal of noise introduced by even state-of-the-art TimeML and TDT parsers, which then obscures what deficiencies are a result of the TDT representation as opposed to parser performance. Second, we could use a dataset which has gold standard annotations of TimeML and TDTs; unfortunately, as mentioned above, these do not yet seem to exist. There are numerous corpora with gold-standard TimeML annotations, but we were unable to find even a single manually annotated, publicly available TDT corpus that also had TimeML annotations. One corpus (the TDT corpus) has automatically generated TDT annotations, but no TimeML. The third option, which we use here, is to generate TDTs programmatically from gold-standard TimeML annotated texts. This approach guarantees that we will have both a gold-standard TimeML annotations as well as the best possible TDT for every text, so that they can be directly compared.

3.1. Generating TDTs

Because we are automatically generating TDTs from TimeML graphs, and not generating TDTs using a TDT parser, there is a question as to whether the TDTs are faithful to the original TDT scheme. We first provide a brief description of how TDT parsers work, so that it can be seen that our generation algorithm (Algorithm 1) intuitively follows the automatic TDT parsing and thus produces a "best possible" TDT.

A TDT parser, as described by Zhang and Xue (2018b), starts by initializing a TDT with a ROOT node that represents the document creation time (DCT), and which has four children (the *meta-nodes*): PAST_REF, PRESENT_REF, FUTURE_REF, and ATEMPORAL. The parser then proceeds through the text in reading order. Every time the TDT parser detects a new event or time expression, it attempts to find a relationship between that event or time and an existing node in the TDT in a breadth-first manner. If a relationship (link) is found to an existing node (by running a link classifier, e.g., an SVM), the new event or time is added as a child to that node, with the appropriate link type, and removed from further consideration. If no relationship is found, it is set aside and checked for a relationship with each new node that is later added to the tree. If, at the end of the text, there are still nodes that have no relationships

Corpus Name	Texts	Words	Events	Times	Relations	Text Types
ProppLearner	15	18,862	3,438	142	2,778	Hero tales
N2 Corpus	67	28,462	2,345	349	4,854	Religious texts, Magazine
TimeBank	183	68,555	7,935	1,414	8,242	Broadcast news, Newswire, Biography
TDT Corpus	235	n/a	15,783	1,298	17,081	News reports, Fairytale stories
Total	500	115,879+	29,501	3,203	32,955	

Table 2: Summary of the corpora used in the experiments.

to any other nodes in the tree, they are added as children of the ATEMPORAL meta-node.

To illustrate the TDT structure, we introduce here a snippet from the TimeBank 1.2 text *APW19980213.1320.tml* (Pustejovsky et al., 2003b), which we will use as an example throughout the remainder of the paper. In this snippet, underlined text refers to a marked TimeML event or temporal expression, and is labeled with its id (e.g., e_1 or t_{44}) from the annotation. To enhance understandability, we only show events and times related to the FUTURE_REF meta-node, which corresponds to one top-level branch of the resulting TDT.

> *DCT: 1998-02-13$_{t_{41}}$*
>
> *Qantas will almost double$_{e_1}$ its flights between Australia and India by August$_{t_{43}}$ in the search for new markets untouched by the crippling Asian financial crisis. This move$_{e_{28}}$ comes barely a month$_{t_{44}}$ after Qantas suspended$_{e_5}$ a number of services$_{e_{29}}$ between Australia, Indonesia, Thailand and Malaysia in the wake of the Asian economic crisis$_{e_{30}}$. The airline has also cut$_{e_7}$ all flights$_{e_{31}}$ to South Korea.*

The temporal graph for this snippet is shown in Figure 1(a). We can generate a TDT from this temporal graph by following Zhang and Xue's definition and procedure. During the process, we generate two types of TDTs: first, TDTs that use the full set of TimeML temporal and aspectual links (*full* TDTs). Then we apply the abstraction mappings shown in in Table 1 to produce TDTs that use only Zhang and Xue's four types (*abstract* TDTs).

The algorithm for generating full TDTs from TimeML graphs is shown in Algorithm 1. It follows the TDT parsing procedure almost exactly, but rather than using a classifier to determine whether there is a relationship between a new node and the TDT, it queries the TimeML graph. We begin by initializing a FIFO queue with all events and times in the TimeML graph such that they will be returned in text order (line 2). We next initialize an empty TDT with a ROOT node and add all four meta-nodes to the tree as children of the root (lines 3–7). We then pop events or times from the queue (call this event or time n; lines 8–20), first looking through the tree in a breadth-first manner for an event or time to which n is linked (line 11). The presence of a link is determined by querying the TimeML graph. Note that in a consistent TimeML graph no two nodes will be connected by more than one temporal or aspectual link. If n is not found to link to an existing node (line 12), then n is added to the unlinked set for future processing (line 13). If n is found to link to existing node, it is added as a child to that node (line 15) and all unlinked nodes are checked for relationships to the new node (lines 16–20). Any events or times that remain unrelated to any other nodes at end of the text are added as children to the ATEMPORAL meta-node (line 21).

Algorithm 1 Generating a Full TDT

```
 1: procedure GENERATEFULLTDT(G)
Require: G                                   ▷ TimeML graph
Require: N                                   ▷ FIFO queue
Require: U          ▷ set of as-yet unlinked events and times
 2:     N.pushAll(G.V)      ▷ Add all events and times in text order
 3:     T ← ROOT            ▷ initialize TDT with the ROOT node
 4:     T.ROOT.addChild(PAST_REF)   ▷ add meta-nodes
 5:     T.ROOT.addChild(PRESENT_REF)
 6:     T.ROOT.addChild(FUTURE_REF)
 7:     T.ROOT.addChild(ATEMPORAL)
 8:     while !N.isEmpty() do
 9:         n ← N.pop()
10:         for all v ∈ T_v do      ▷ iterate over tree breadth-first
11:             l ← GETLINK(G, n, v)      ▷ identify link, if any
12:             if l = ∅ then
13:                 U.add(n)
14:             else
15:                 v.addChildWithLink(n, l)
16:                 for all u ∈ U do      ▷ check all as-yet unlinked
17:                     l ← GETLINK(G, n, u)
18:                     if l ≠ ∅ then
19:                         n.addChildWithLink(u, l)
20:                         U.remove(u)
21:     ATEMPORAL.addChildren(U)
22:     return T
```

When this procedure is applied to the example temporal graph in Figure 1(a), it produces the full TDT shown in Figure 1(b). As can be seen 3 of the original 11 links in the TimeML graph are omitted in the TDT. One of our main questions is how much temporal information is lost in this process, which we measure directly in Section 4.1.

As mentioned above, during the transforming we iterate events and times in text order. We also experimented with other orders to determine if the order of iteration mattered. While different specific links were omitted for different orders (e.g., reverse text order, or random), it resulted in the same average loss of temporal information. The main reason for this is that to transform a graph into a tree, the algorithm must ultimately remove one edge from every cycle. After generating full TDTs, we generated abstract TDTs by applying the mappings shown in Table 1. Figure 1(c) shows the abstract TDT for the example snippet.

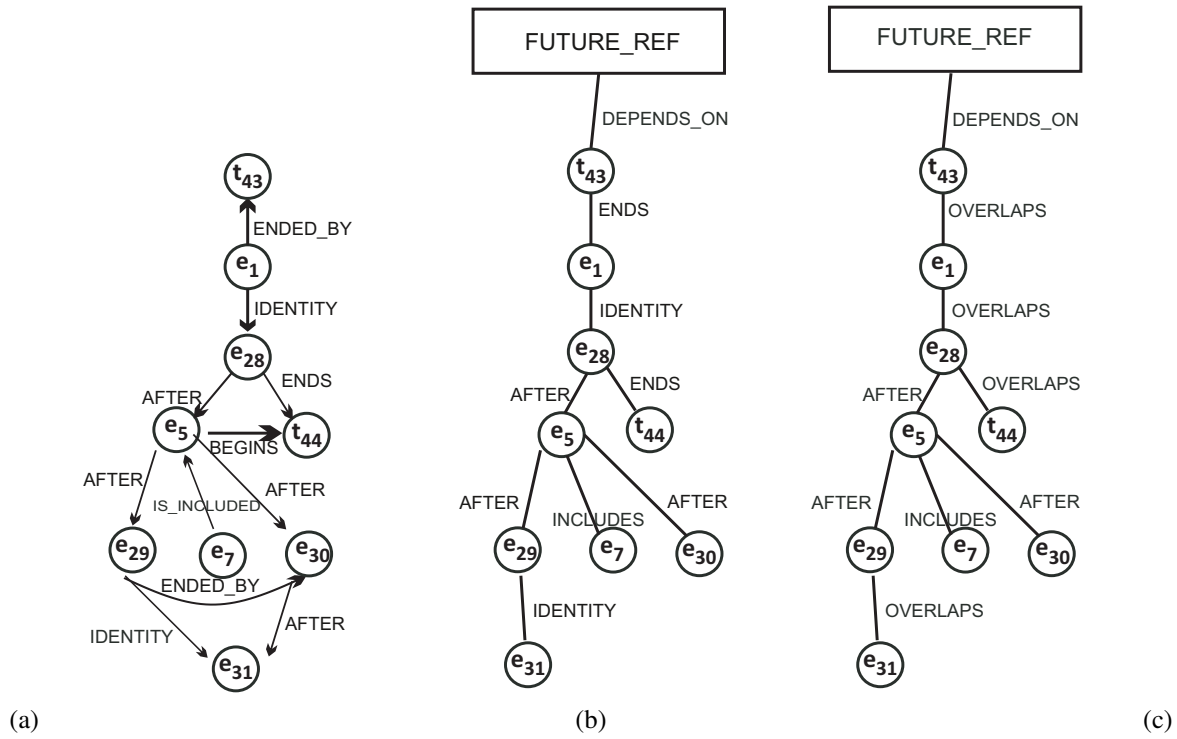

Figure 1: Temporal structures for the example snippet, including (a) the full TimeML temporal graph, (b) a temporal dependency tree which uses all link types (full TDT), and (c) a temporal dependency tree that uses only the abstract link types (abstract TDT).

3.2. Extracting Timelines

We can precisely compare the information lost in moving from TimeML graphs to TDTs by converting both the TimeML graphs and TDTs into a uniform representation, namely, timelines. To extract a timeline, we first translate the TimeML or TDT into a temporal constraint graph using primitive temporal relations, and then solve the graph. This translation is made by substituting each individual event or time interval I by its corresponding pairs of start and end time points I^- and I^+, and replacing each link between intervals with conjunctions of primitive temporal constraints (either $<$ or $=$) between time points as shown in Table 3. In addition to the replacements in the table, we also include for each interval I the constraint that its starting point I^- must be less than its ending point I^+. Following Barták et al. (2014) we can then define a solution to temporal constraint problem as shown in Definition 3.

Definition 3. *Temporal Constraint Problem.* A temporal constraint problem $P = (T, C)$ where $T = \{t_1, t_2, \ldots\}$ is a set of time points, and $C = \{c_1, c_2, \ldots\}$ is a set of temporal constraints between intervals, is *consistent* or *solvable* if and only if a vector of integers $(i_1, i_2, \ldots)$ is a solution $(t_1 = i_1, t_2 = i_2, \ldots)$ of the temporal graph that satisfies all the constraints.

This means that if we can assign integers to the time points such that all temporal constraints are satisfied, then the temporal structure (TimeML temporal graphs, full TDT, abstract TDT, etc.) is consistent and solvable.

To find a solution we used the off-the-shelf Java Constraint Programming (JaCoP) solver (Kuchcinski and Szymanek, 2013). JaCoP is an open source library that offers a rich set of constraint types as well as configurable solution search methods. For this step we use the JaCoP setting that finds the smallest solution, which assures that any differences in the length of timelines are the result of temporal information content, and not choice of solution. If JaCoP is able to find a solution then it means the temporal structure is consistent; otherwise it is inconsistent. We applied this method to the TimeML temporal graphs, full TDTs, and abstract TDTs for the TimeBank corpus, the N2 corpus, and ProppLearner corpus, as well as to the abstract TDTs automatically parsed by Zhang and Xue.

If the TimeML graphs are inconsistent this meant there was an error in the manual annotation; we corrected these graphs by hand using the original text as reference. There were 9, 10, and 18 inconsistent texts in the ProppLearner, N2 Corpus, and TimeBank corpus respectively. Because the TDT Corpus is in Chinese, which we do not speak, we were unable to correct these annotations, and so we discarded inconsistent TDT annotations. Out of 235 TDT corpus texts, 25 were inconsistent and were discarded.

Once we have integer assignments to time points for the TimeML graph or TDT, we can sort these integers to obtain the corresponding timeline. Because the same integer might be assigned to different time points, the length of timelines can be measured in two ways: the number of *time points*, which is directly proportional to the number of events and times, or the number of *time steps*, which is the number of integers in the solution to the temporal constraint problem. For instance, the timelines extracted from the TimeML graph, full TDT, and abstract TDT for our ex-

A BEFORE B	$(A^+ < B^-)$
A AFTER B	$(B^+ < A^-)$
A IBEFORE B	$(A^+ = B^-)$
A IAFTER B	$(B^+ = A^-)$
A BEGINS B	$(A^- = B^-) \wedge (A^+ < B^+)$
A BEGUN_BY B	$(A^- = B^-) \wedge (B^+ < A^+)$
A ENDS B	$(B^- < A^-) \wedge (A^+ = B^+)$
A ENDED_BY B	$(A^- < B^-) \wedge (A^+ < B^+)$
A DURING B	$(B^- < A^-) \wedge (A^+ < B^+)$
A DURING_INV B	$(A^- < B^-) \wedge (B^+ < A^+)$
A INCLUDES B	$(A^- < B^-) \wedge (B^+ < A^+)$
A IS_INCLUDED B	$(B^- < A^-) \wedge (A^+ < B^+)$
A SIMULTANEOUS B	$(A^- = B^-) \wedge (A^+ = B^+)$
A IDENTITY B	$(A^- = B^-) \wedge (A^+ = B^+)$
A INITIATES B	same as B BEGUN_BY A
A REINITIATES B	same as B IS_INCLUDED A
A TERMINATES B	same as B BEFORE A
A CULMINATES B	same as B ENDED_BY A
A CONTINUES B	same as B INCLUDES A
A OVERLAPS$_{tdt}$ B	A (BEGINS $\vee$ BEGUN_BY $\vee$ IDENTITY $\vee$ ENDS $\vee$ ENDED_BY $\vee$ SIMULTANEOUS) B

Table 3: Translation of the TimeML temporal and aspectual links into primitive temporal relations between interval start and end points. For an interval I (an event or a time) the start point of the interval is denoted by I^- and the end point is denoted I^+.

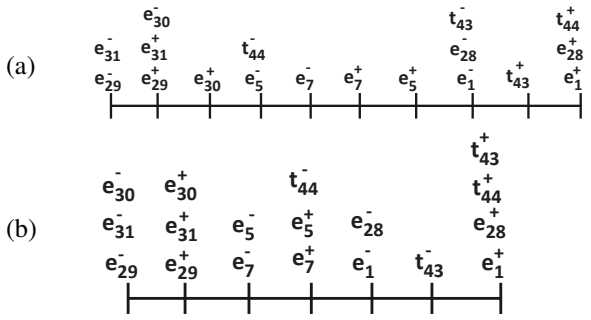

Figure 2: Timelines corresponding to both the (a) TimeML graph and (b) full & abstract TDT.

ample snippet are shown in Figure 2. The timeline for the TimeML graph has 10 time steps and 18 time points. In this example, the TimeML and TDT timelines both have the same number of time points, but the number of time steps in the TDT case is smaller on account of discarded temporal information. In the general case, because TimeML graphs encode subordinating relations which are completely disregarded by TDTs, certain events and times might be removed from the TDT timeline altogether, and so the number of time points in TDT timelines can be smaller than in the equivalent TimeML timeline.

3.3. Computing Indeterminacy

We compare the information loss between timelines by computing the *indeterminacy* of time point orderings relative to the original temporal graph or tree. Temporal graphs or trees often do not have enough information to identify

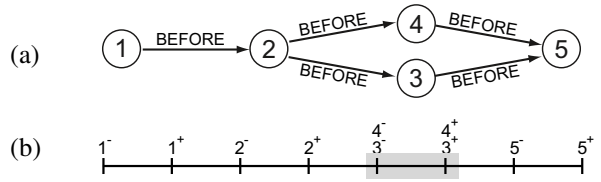

Figure 3: An example of temporal indeterminacy, illustrated by (a) an indeterminate temporal graph and (b) its corresponding minimal timeline solution, with the indeterminate section marked in gray. The relative order of 3 and 4 is indeterminant.

a unique timeline. We call sections of the timeline that have multiple possible solutions *indeterminant*. Similarly, time points or time steps involved in these sections are also called indeterminant.

We give a simple example of a temporally indeterminant TimeML graph in Figure 3. For this temporal graph, the uniquely determined orderings include the first and last sections of the timeline, namely $1^- < 1^+ < 2^- < 2^+ < 3, 4$ and $3, 4 < 5^- < 5^+$. On the other hand, the order of 3 and 4 is indeterminate. There are 11 possible solutions for the ordering of the start and end points of these two intervals.

An algorithm for identifying indeterminacies is shown in Algorithm 2. The algorithm works by comparing all possible timelines, which the JaCoP solver can provide. The algorithm iterates through all adjacent time point pairs in the shortest timeline (lines 3–7), and checks to see if these two points are adjacent in all other timelines (lines 5–6). If they are not, the order of that pair is marked indeterminate (line 7). In practice, it often takes considerable time to compute all possible timelines. In our implementation, to save time, we limited ourselves to 100 random alternative timelines, and as such gives a lower-bound to the indeterminacy. With these results, we can visualize which portions of the shortest timeline are indeterminate, as illustrated in Figure 3. We can thus measure the temporal indeterminacy of the TimeML graphs for the three TimeML corpora, and the TDTs for all four corpora. The amount of indeterminacy can be measured in a number of ways: we can count the number of indeterminate time points or time steps; the number of indeterminate sections; or the fraction of time steps or time points that are indeterminate. Detailed results are presented in Section 4.2. and Table 6.

Algorithm 2 Identifying Indeterminacies

```
1: procedure FINDINDETERMINATESECTIONS(s, T)
Require: s                                   ▷ shortest timeline
Require: T                                   ▷ all other timelines
2:     d ← ∅                                 ▷ map of time point pairs to boolean
3:     for all p ← ⟨t_i, t_{i+1}⟩ ∈ s do     ▷ neighboring time points
4:         d(p) ← false
5:         for all t ∈ T do                  ▷ for all other timelines
6:             if p ∉ t then                 ▷ if the pair are not neighbors
7:                 d(p) ← true               ▷ mark the pair indeterminant
8:     return d
```

Corpus	# TimeML Links	# Links Omitted	% Omitted
ProppLearner	3,900	24	0.6%
N2 Corpus	3,273	98	3.0%
TimeBank	5,058	166	3.3%
Total or Average	12,231	288	2.4%

Table 4: Counts of temporal relations present in the TimeML graphs and omitted in the TDTs.

3.4. Implementation

Our implementation was written in Java, building upon the JaCoP constraint solving tool as mentioned above, and takes as input TimeML (`.tml`), Story Workbench (`.sty`), or TDT (`.tdt`) files. Although the constraint satisfaction problem is in theory NP-complete, our experiments took no longer than than a second on a current, standard consumer laptop (2.4 GHz 4-core Intel i7 3630QM with 8GB of RAM). We release our implementation and the timelines for use by other researchers[1].

4. Results & Discussions

4.1. Omitted Temporal Relations

Using Algorithm 1 we transformed the TimeML graphs from the 265 texts (including corrected texts) in the TimeBank, N2, and ProppLearner corpora into full TDTs. The overall counts of TimeML relations and omitted links is shown in Table 4, and on average 2.4% of temporal relations are omitted. The two reasons for these omissions are (1) tree nodes may only have one parent, and (2) the TDT representation ignores subordinating links. This observation emphasizes that in the general case TDTs cannot represent all of the temporal information in a text.

4.2. Increase in Indeterminacy

After extracting full TDTs from the corpora we generated abstract TDTs as described at the end of Section 3.1., and extracted timelines from the TimeML graphs, full TDTs, and abstract TDTs. Table 5 shows various characteristics of the timelines so extracted, including their average length in terms of both time steps and time points (first and second groups of columns), total number of time points (third group), and average percentage decrease of TDT timeline lengths relative to TimeML timelines in terms of time steps (last group). In the last column group, we see that overall timeline lengths in full and abstract TDTs decrease by anywhere from 3.4% to 14.7% on average.

We applied Algorithm 2 to these timelines to identify indeterminate sections and time points; Table 6 shows the results. We can compare the relative indeterminacy of timelines by computing the percentage of time steps that are assigned an indeterminate time point (last group of columns). Transformation of TimeML graphs into a full TDT increases the temporal indeterminacy by 76%, 16%, and 22% (average 22%) for the ProppLearner, N2, and TimeBank corpora, respectively. These fractions are computed by dividing the numbers in the second-to-last column of Table 6 by those in the third-to-last column. Similarly, transformation of TimeML graphs into abstract TDTs increased indeterminacy by 109%, 51%, and 25% (average 32%). Overall, 11,437 out of 14,671 (78%) time points are indeterminate for abstract TDT timelines and 10,023 out of 14,671 (70%) are indeterminate for full TDT timelines, compared with 8,769 out of 15,623 (56%) for TimeML timelines. Thus, even full TDTs increase temporal indeterminacy significantly compared to TimeML graphs. In contrast to time points, on average 52.2% of time steps in TimeML timelines are indeterminate, compared with 67.2% and 78.1% of time steps in full and abstract TDT timelines. This increase in indeterminacy is potentially important to downstream NLP stages; for example, for a question answering system that is addressing temporal or causal questions, the text may provide enough information to produce a single answer, but a TDT representation may not include all of that information, making it impossible for the QA system to answer unambiguously.

[1]The publicly available code and data is available through the FIU Dataverse, at DOI `https://doi.org/10.34703/gzx1-9v95/29YPVR`

5. Contributions

We have made two contributions in this paper. First, we presented an algorithm that transforms temporal graphs to full or abstracted TDTs. This allows a direct comparison of TimeML and TDT formalisms, because although there a several corpora with TimeML annotations, there are as yet no corpora annotated with both TimeML graphs and TDTs. Second, by transforming TimeML graphs and TDTs into timelines we showed that the TDTs are significantly more temporally indeterminant relative to TimeML graphs: anywhere from 24% to 109% depending on the corpus (average increase of 32%). This increase in indeterminacy is attributable to the omission of a mere 2.4% of temporal relations from TDTs. This suggests that in tasks where information on the global ordering of events and times is important, full TimeML representation is perhaps called for.

6. Acknowledgements

This work was partially funded by ONR under contract N00014-17-1-2983, and by DARPA under contract FA8650-19-C-6017.

7. Bibliographical References

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Barták, R., Morris, R., and Venable, K. (2014). *An Introduction to Constraint-Based Temporal Reasoning*. Morgan & Claypool Publishers.

Cheng, Y., Asahara, M., and Matsumoto, Y. (2007). NAIST.Japan: Temporal relation identification using dependency parsed tree. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval '07)*, pages 245–248, Prague, Czech Republic.

Ferro, L., Gerber, L., Mani, I., Sundheim, B., and Wilson, G. (2001). Tides temporal annotation guidelines, ver. 1.0.2. `http://www.timeml.org/terqas/readings/MTRAnnotationGuide_v1_02.pdf`.

Corpus	Avg. Len. of Main Timeline in Time Steps			Avg. Len. of Main Timeline in Time Points			Total # of Time Points / Corpus			Avg. % ↓ in Timeline Len. in Time Steps	
	TimeML Graphs	Full TDTs	Abs. TDTs	TimeML Graphs	Full TDTs	Abs. TDTs	TimeML Graphs	Full TDTs	Abs. TDTs	Full TDTs	Abs. TDTs
TDT Corpus	-	-	67.2	-	-	72.6	-	-	17,081	-	-
ProppLearner	123.5	108.7	108.7	238.7	230.6	230.6	3,580	3,460	3,460	11.9%	14.7%
N2 Corpus	17.7	17.1	17.1	40.2	35.6	35.6	2,694	2,390	2,390	3.4%	6.7%
TimeBank	9.3	8.5	8.5	51.1	48.2	48.2	9,349	8,821	8,821	8.6%	10.8%
						Total	15,623	14,671	14,671		

Table 5: Characteristics of the timelines extracted from the corrected corpora. The TDT corpus is included for comparison only and includes only abstract TDTs, with inconsistent TDTs excluded (25 texts).

Corpus	Total # Ind. Time Points / Corpus			Total # of Ind. Sections / Corpus			Average # of Ind. Sections / Text			Avg. % of Time Steps in Timelines that are Ind.		
	TimeML Graphs	Full TDTs	Abs. TDTs	TimeML Graphs	Full TDTs	Abs. TDTs	TimeML Graphs	Full TDTs	Abs. TDTs	TimeML Graphs	Full TDTs	Abs. TDTs
TDT Corpus	-	-	11,444	-	-	10,081	-	-	42.9	-	-	63.7%
ProppLearner	1,066	1,892	2,159	432	669	910	36.8	59.5	67.8	29.8%	54.7%	62.4%
N2 Corpus	1,355	1,568	1,816	683	770	944	8.9	11.2	13.0	50.3%	65.6%	76.0%
TimeBank	6,348	6,563	7,462	970	1,079	1,976	6.3	6.3	7.2	67.9%	74.4%	84.6%
Total	8,769	10,023	11,437	2,085	2,518	3,830		**Weighted Avg.**		**61.3%**	**71.1%**	**81.2%**

Table 6: Indeterminacy in timelines extracted from TimeML graphs vs. TDTs. The TDT corpus is included for comparison only and includes only Abstract TDTs, with inconsistent TDTs excluded (25 texts). Sections are defined as unbroken sequences of indeterminate time points or steps. The weighted average was computed by weighting with time points.

Finlayson, M., Halverson, J., and Corman, S. (2014). The N2 corpus: a semantically annotated collection of islamist extremist stories. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)*, pages 896–902, Reykjavik, Iceland.

Finlayson, M. A. (2017). Propplearner: Deeply annotating a corpus of russian folktales to enable the machine learning of a russian formalist theory. *Digital Scholarship in the Humanities*, 32(2):284–300.

Kolomiyets, O., Bethard, S., and Moens, M.-F. (2012). Extracting narrative timelines as temporal dependency structures. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL'12)*, pages 88–97, Jeju Island, Korea.

Kuchcinski, K. and Szymanek, R. (2013). JaCoP: Java constraint programming solver. `http://jacop.cs.lth.se/`.

Liu, S., Zhou, M. X., Pan, S., Song, Y., Qian, W., Cai, W., and Lian, X. (2012). Tiara: Interactive, topic-based visual text summarization and analysis. *ACM Transactions on Intelligent Systems & Technology*, 3(2):25:1–25:28.

Pustejovsky, J., Castaño, J., Ingria, R., Saurí, R., Gaizauskas, R., Setzer, A., and Katz, G. (2003a). TimeML: robust specification of event and temporal expressions in text. In *Proceedings of the 5th International Workshop on Computational Semantics (IWCS-5)*, pages 1–11, Tilberg, The Netherlands.

Pustejovsky, J., Hanks, P., Saurí, R., See, A., Gaizauskas, R., Setzer, A., Radev, D., Sundheim, B., Day, D., Ferro, L., and Lazo, M. (2003b). The TimeBank corpus. In *Proceedings of Corpus Linguistics Conference*, pages 647–656, Lancaster, UK.

Robaldo, L., Caselli, T., Russo, I., and Grella, M. (2011). From italian text to TimeML document via dependency parsing. In *Proceedings of the 12th Computational Linguistics and Intelligent Text Processing (CICLing 2011)*, pages 177–187, Tokyo, Japan.

Saquete, E., Martínez-Barco, P., Muñoz, R., and Vicedo, J. L. (2004). Splitting complex temporal questions for question answering systems. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL '04)*, pages 566–573, Barcelona, Spain.

Sauri, R., Littman, J., Gaizauskas, R., Setzer, A., and Pustejovsky, J. (2006). TimeML annotation guidelines, version 1.2.1. `https://catalog.ldc.upenn.edu/docs/LDC2006T08/timeml_annguide_1.2.1.pdf`.

Setzer, A. (2001). *Temporal Information in Newswire Articles: an Annotation Scheme and Corpus Study*. Ph.D. thesis, University of Sheffield.

Vilain, M., Kautz, H., and van Beek, P. (1990). Constraint propagation algorithms for temporal reasoning: A revised report. In Daniel S. Weld et al., editors, *Readings in Qualitative Reasoning About Physical Systems*, pages 373–381. Morgan Kaufmann, San Francisco, CA.

Wu, M., Li, W., Lu, Q., and Li, B. (2005). CTEMP: A Chinese temporal parser for extracting and normalizing temporal information. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing: Full Papers*, pages 694–706.

Zhang, Y. and Xue, N. (2018a). Neural ranking models for temporal dependency structure parsing. *arXiv, CoRR*, abs/1809.00370.

Zhang, Y. and Xue, N. (2018b). Structured interpretation of temporal relations. *arXiv, CoRR*, abs/1808.07599.

8. Language Resource References

Finlayson, M.A., Halverson, J. and Corman, S. (2014). *The N2 Corpus*. DSpace@MIT, `http://hdl.handle.net/1721.1/85893`.

Finlayson, M.A. (2017). *The ProppLearner Corpus*. DSpace@MIT, `http://hdl.handle.net/1721.1/100054`.

Pustejovsky, J., Hanks, P., Sauri, R., See, A., Gaizauskas, R., Setzer, A., Radev, D., Sundheim, B., Day, D., Ferro, L. and Lazo, M. (2003). *The TimeBank corpus*. The Linguistic Data Consortium, `https://catalog.ldc.upenn.edu/LDC2006T08`.

Zhang, Y. and Xue, N. (2018). *The TDT Corpus*. GitHub, `https://github.com/yuchenz/structured`.