jTLEX: a Java Library for TimeLine EXtraction

Mustafa Ocal, Akul Singh, Jared Hummer, Antonela Radas, & Mark Finlayson
Knight Foundation School of Computing and Information Sciences
Florida International University
CASE Building, Room 362, 11200 S.W. 8th Street, Miami, FL USA 33199
{mocal001,asing118,jhumm001,arada002,markaf}@fiu.edu

Abstract

jTLEX is a programming library that provides a Java implementation of the TimeLine EXtraction algorithm (TLEX; Finlayson et al., 2021), along with utilities for programmatic manipulation of TimeML graphs. Timelines are useful for a number of natural language understanding tasks, such as question answering, cross-document event coreference, and summarization & visualization. jTLEX provides functionality for (1) parsing TimeML annotations into Java objects, (2) construction of TimeML graphs from scratch, (3) partitioning of TimeML graphs into temporally connected subgraphs, (4) transforming temporally connected subgraphs into point algebra (PA) graphs, (5) extracting exact timeline of TimeML graphs, (6) detecting inconsistent subgraphs, and (7) calculating indeterminate sections of the timeline. The library has been tested on the entire TimeBank corpus, and comes with a suite of unit tests. We release the software as open source with a free license for non-commercial use.

1 Introduction

TimeML is a standardized temporal annotation scheme for annotating temporal information in texts (Pustejovsky et al., 2003a). TimeML annotations can be used to build temporal graphs, where nodes are events and temporal expressions (i.e., times), and edges are temporal relations. TimeML annotations can be generated using automatic analyzers (Chambers et al., 2014), manual annotation (Minard et al., 2016), or some combination of the two.

While temporal graphs derived from texts can be deeply informative, they usually only encode partial orderings of events and times. For many NLP tasks such as text summarization and visualization (Liu et al., 2012), question-answering (Saquete et al., 2004), and knowledge representation (Galton, 2009), a total order of all events and times (i.e., a timeline) is often more useful. Unfortunately, timelines are rarely explicit in texts and usually cannot be read off from texts directly. There have been a few prior attempts to extract timelines from temporal graphs, however, these works have certain limitations: they do not handle all TimeML relations, they do not separate "real-life" events and subordinated events, and they do not deal with multiple possible temporal orders.

These problems were addressed by an approach called TLEX (TimeLine EXtraction; Finlayson et al., 2021). TLEX presented a CSP-based solution that extracts *exact* timelines from a TimeML graph. TLEX also detects inconsistent TimeML subgraphs as well as temporal indeterminacy. TLEX is a formally correct method and the experimental evaluation on four different TimeML corpora showed that it has 100% accuracy for extracting timelines (Finlayson et al., 2021). TLEX has been used for several NLP tasks such as corpus validation, evaluating temporal dependency parsers, and narrative representation (Ocal et al., 2022a,b; Ocal and Finlayson, 2020). To enable better access to this approach for the community, we have implemented jTLEX, an open-source Java implementation of TLEX for other researchers in the field to use. We present jTLEX in this paper.

jTLEX provides several types of functionality. In its canonical usage, jTLEX takes a TimeML annotated file as input, then (1) parses the annotations into TimeML objects, (2) builds a TimeML graph, (3) partitions the TimeML graph into temporally connected graphs to separate real-life events and subordinated events, (4) transforms the temporally connected graphs into point algebra (PA) graphs, and (5) solves the PA graphs to extract a timeline. If a timeline cannot be extracted, meaning the graph is temporally inconsistent, (6) it detects the minimum inconsistent subgraph and returns it to the annotator to fix it. Finally, if the order of events and times are indeterminant (multiple possible order-

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics
System Demonstrations, pages 27–34
May 2-4, 2023

ing), (7) it calculates the temporal indeterminacy. These steps correspond to steps of the TLEX algorithm as described in Finlayson et al. (2021).

We have tested jTLEX on the entire TimeBank corpus (Pustejovsky et al., 2003b) which is a reference corpus for TimeML that contains 183 TimeML annotated news articles. For each file, jTLEX took no longer than 1 second to extract all timelines on a currently available, standard consumer laptop (3.0 GHz Intel Core i7-1185G7 with 32GB of RAM) Our demonstration system as well as a screencast video demonstrating our system is available[1].

2 Library Overview

2.1 User Input

jTLEX processes and allows the manipulation of all the information available in a TimeML annotation. jTLEX can read in preexisting TimeML annotations from a `.tml` file, accept TimeML annotations directly via a JSON-style TimeML encoding, or accept the raw text of TimeML annotations as a Java String object. TimeML annotations can be generated via manual annotation by following the TimeML annotation guide (Sauri et al., 2006) or by using state-of-the-art TimeML annotators such as TARSQI (Verhagen et al., 2005), ClearTK (Bethard, 2013), CAEVO (Chambers et al., 2014), or CATENA (Mirza and Tonelli, 2016). Note that there are certain limitations when using automatic TimeML annotators such as information loss and temporal inconsistency (Ocal et al., 2022a). Fortunately, jTLEX can detect inconsistencies and help users to fix them as explained in Section 2.7.

2.2 TimeML Parser

TimeML is an SGML-based annotation scheme to annotate temporal information in texts. TimeML defines tags for events (<EVENT>), temporal expressions (<TIMEX>), temporal signals (<SIGNAL>), event instances (<MAKEINSTANCE>), and links between events and times, namely temporal link (<TLINK>), subordinated link (<SLINK>), and aspectual link (<ALINK>). Each tag has attributes that contain information about a TimeML object. For example, the "`polarity`" attribute of <MAKEINSTANCE> indicates whether an event is negated and it contains either `POS` or `NEG` values.

jTLEX provides a TimeML parser that can parse TimeML annotations into a set of TimeML Java objects, including events, times, and links. Additionally, it can strip the TimeML tags and return the raw text. The TimeML annotation guide identifies optional and non-optional attributes for each TimeML tag. If the input to jTLEX is missing a non-optional attribute, the parser returns an error message to the user about which attribute is missing for which object. Therefore, jTLEX's TimeML parser can be used to check compliance of annotations to the TimeML standard.

2.3 Graph Constructor

A TimeML graph is a graph where nodes are events and times, and edges are TimeML links (as shown in Figure 1). After jTLEX parses a TimeML annotations into the TimeML objects (events, times, links, etc.), it builds a TimeML graph. Any information about the graph can be then programmatically queried, such as the set of links, the set of nodes, a link by ID, a node by ID, a list of incoming or outgoing links, and much else.

jTLEX allows users to directly modify the TimeML graph if they wish. Users can add or remove links or nodes to the graph, and can also build their own custom graph by creating an empty graph and adding events, times, and links. The graph implementation has a method that returns a JSON output of the graph. This allows users to take advantage of existing graph visualization system such as React Flow (So, 2018) to inspect the TimeML annotations.

2.4 Partitioner

As mentioned in Section 2.2, there are three types of TimeML links. While <TLINK> and <ALINK> provide information about the temporal order of events and times, <SLINK> is used for contexts introducing possible (modal), counterfactual, or conditional relations between two events. An example is shown below.

(1) Amanda *forgot* to *buy* coffee.

The example indicates a counterfactual relationship between *forgot* and *buy*. The event of *buy* never happened in the world described in the text, i.e., the "real world", and therefore, it needs to be separated. As described in the TLEX paper, jTLEX implements this by partitioning a TimeML graph into temporally connected subgraphs. The partitioner has two steps: jTLEX walks the graph to partition the TimeML graph into connected subgraphs. Then

[1] https://cognac.cs.fiu.edu/jtlex/

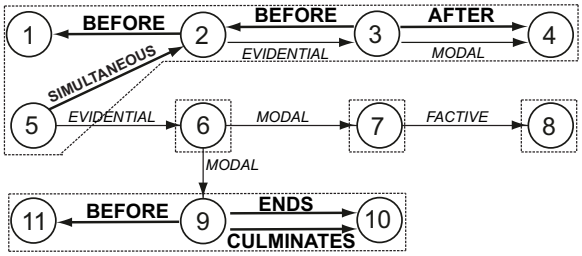

Figure 1: Visualization of partitioning the TimeML graph for wsj_0006.tml from the TimeBank corpus. TLINKs and ALINKs are given in bold, and SLINKs are in italic. jTLEX partitions the TimeML graph into five temporally and aspectually connected subgraphs, as shown by dashed lines.

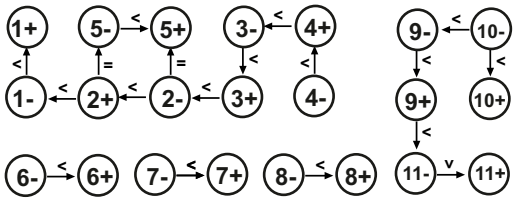

Figure 2: Visualization of the output of the transforming temporally connected subgraphs in Figure 1 into the PA graph.

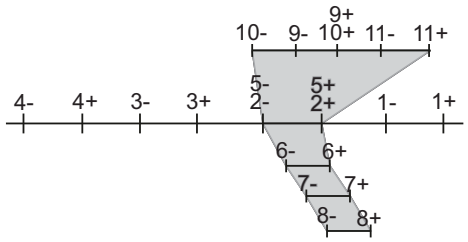

Figure 3: Visualization of the timeline of the TimeML graph in Figure 1. Grey regions indicate subordinating links between timelines.

it further partitions these connected subgraphs into subgraphs connected only with temporal and aspectual links. We name the subgraph(s) that contains "real world" events as *main* subgraph(s), and subgraphs that connect to main subgraphs via subordination links as *subordinated* subgraphs. A visual example of this process is shown in Figure 1.

2.5 Transformer

A point algebra (PA) graph is a graph where nodes are time points and edges are primitive temporal constraints such as {<, =}. As prescribed by the TLEX algorithm, jTLEX transforms each temporally connected subgraph into a PA graph. Each node and link in the subgraph can be expressed in time points and constraints. Assume we have two events (A and B) and A is *BEFORE* B. This can be represented in a PA graph as follows: $A^- < A^+ < B^- < B^+$, where $-$ and $+$ indicate the start and end time points of a node, respectively. An example of the transformation of the TimeML graph in Figure 1 into a PA graph is shown in Figure 2.

This transformation is necessary because the timeline is generated by solving the temporal constraint satisfaction problem (TCSP) represented by the PA graph, as discussed in Section 2.6.

2.6 Solver

After the transformer transforms each temporally connected subgraph into a PA graph, jTLEX assigns integers to each time point in the graph using Java Constraint Programming (JaCoP) library (Kuchcinski and Szymanek, 2013). The library then obtains a timeline after sorting the assigned integers. JaCoP is an open-source java library that offers a rich set of primitive, logical, conditional, and global constraints as well as configurable solution search methods. jTLEX by default produces the smallest solution, which starts at 1 and which represents each time point difference as a difference of 1. Users can also use jTLEX to produce a random solution.

When run over all the PA graphs, jTLEX produces an *exact* trunk-and-branch *timeline* where the trunk is the main timeline corresponding to the main subgraph and branches are subordinated timelines corresponding to the subordinated subgraphs as shown in Figure 3. Therefore, the main timeline consists the global order of "real world" events and times, while subordinated branches consist subordinated events. Users can retrieve the length of the timeline, the first and last time points, the main timeline, subordinated branches, the number of subordinated branches, the number of time points, and the list of attachment time points where subordinated branches are connected to the main timeline. Users can also retrieve the JSON output of the timeline, and therefore they can visualize the timeline using a third-party graph visualizer application.

2.7 Inconsistency Detector

As described in the TLEX work, the solver can only extract a timeline of the TimeML annotation if the annotation is temporally consistent. If the integer assignment is not possible, then it means the TimeML graph has temporal inconsistency (Barták

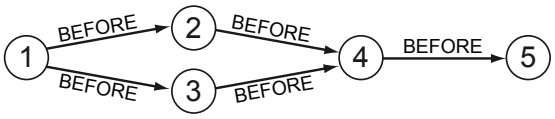

Figure 4: An example graph that has temporal indeterminacy.

et al., 2014). jTLEX provides an inconsistency detector to detect inconsistent cycles in the TimeML graph.

The inconsistency detection algorithm detects self-loops, where the start and end node of an edge is the same such as *A-AFTER->A*. This can be removed automatically as users request. For other non-self-loop cycles, it detects the minimum inconsistent subgraph in the TimeML graph. Further, in the case of two or more inconsistent subgraph overlaps (have a shared edge), it can distinguish the subgraphs and return each inconsistent cycle to the users. A jTLEX output of an inconsistent cycle is shown in Section 3.

Users can use jTLEX to check if their annotation is temporally consistent. If the annotation is not consistent, jTLEX returns the links that cause inconsistency, therefore, users can fix the inconsistent annotation. This could be used as a corpus validation after NLP researchers create a corpus.

2.8 Indeterminacy Calculator

In most cases, natural language texts lack enough information to specify a full ordering, meaning there could be multiple different global orderings of the same events and times. Figure 4 shows a TimeML graph with just such a temporal indeterminacy. The graph indicates a global order of $1^- < 1^+ < 2$ and $3 < 4^- < 4^+ < 5^- < 5^+$, but does not specify the relative order of 2 and 3.

jTLEX uses its own algorithm to measure temporal indeterminacy in a timeline. The algorithm extracts all possible timelines of the graph and compares the shortest timeline with all other possible timelines. More precisely, it checks whether every neighboring time point pair in the shortest timeline is a neighboring time point in other timelines. If they are not always adjacent, the order of that pair is indeterminate. With this result, we can actually represent indeterminate sections on the timeline as shown in Figure 5. This allows jTLEX to represent multiple different orderings in a single timeline.

Using jTLEX, users can retrieve the indeterminant sections of a timeline. jTLEX also provides a

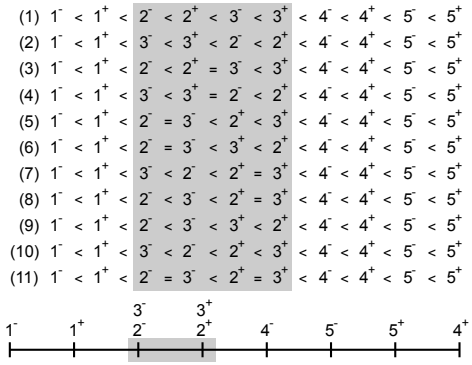

Figure 5: Illustration of the indeterminacy calculation process. The algorithm extracts all possible timelines. For the graph in Figure 4, between 2 and 3, there are 11 possible ordering. This temporal indeterminacy is shown in the grey area in the timeline. Therefore, multiple possible ordering can be represented in a single timeline.

standard way of scoring the amount of indeterminacy present in particular timeline.

3 Use Cases

A user guide and license information can be found on the jTLEX website[2]. Here we illustrate an approach for one of the TimeML annotations of the TimeBank corpus, called `wsj_0006.tml`. This file and the rest of the corpus can be obtained from LDC website[3]. The following text shown in Example (2), is a snippet of the TimeML-annotated text of `wsj_0006.tml`. The TimeML graph corresponding to the snippet text is shown in Figure 1, where we can see that the nodes of the graph are either events or times, and the edges are TimeML relations. Event instance IDs and timeIDs are given in square brackets (DCT = DOCUMENT CREATION TIME).

(2) [DCT:11/02/89$_{1[t9]}$]: Pacific First Financial Corp. **said**$_{2[ei73]}$ shareholders **approved**$_{3[ei74]}$ its **acquisition**$_{4[ei75]}$ by Royal Trustco Ltd. of Toronto for $27 a share, or $212 million. The thrift holding company **said**$_{5[ei76]}$ it **expects**$_{6[ei77]}$ to **obtain**$_{7[ei78]}$ regulatory **approval**$_{8[ei79]}$ and **complete**[$_{9[ei80]}$ the **transaction**$_{10[ei81]}$ by **year-end**$_{11[t10]}$.

Users can read the file and create the TimeML graph as follows:

[2] https://cognac.cs.fiu.edu/jtlex/
[3] https://catalog.ldc.upenn.edu/LDC2006T08

```
1 Link: {ID = 1, LinkTag = TLINK, Syntax
      = "", Temporal Relation = BEFORE,
     Origin = null
2     Signal: {Id = sid12, String = "by"
         }
3     Related to event - Timex: {tID =
         t10, Type = DATE, Value =
         1989-12-31 , Mod = null,
         Temporal Function = true,
         AnchorID = t9, Begin Point = t0
         , End Point = t0, Quantity =
         null, Frequency = null}
4     Event Instance - Event Instance:
5     {ID = eiid80, Tense = PRESENT,
         Aspect = NONE, Part of Speech =
          VERB, Polarity = POS, Modality
          = "null", Cardinality = "null"
         , Signal = null
6         EVENT: eid = e7, class =
             ASPECTUAL, stem = complete}
7     }
```

Listing 1: jTLEX parser output for printing the information about the first link of the graph.

```
File tmlFile = new File(fName);
ITimeMLGraph graph = GraphReader.
    TimeMLGraph(tmlFile);
```

Here, `fName` is the path to the file. Users can retrieve any information about the graph such as links (all or one by ID), nodes (all or one by ID), incoming links, outgoing links, JSON output, number of nodes, number of links, number of link types, etc. Using the following code, users can retrieve the information of the first link:

```
System.out.print(graph.getLinkById(1));
```

The output will be as shown in Listing 1. As can be seen, jTLEX provides all the available information in the TimeML annotation about the link and its components using the TimeML parser.

After the TimeML graph is created, users can create a `TLEX` object to perform the timeline extraction including partitioning, transforming, solving, inconsistency detection (if the graph is inconsistent), and temporal indeterminacy. Creating the tlex object is as follows:

```
TLEX tlex = new TLEX(graph);
```

Users, can retrieve the *exact* trunk-and-branch timeline structure using:

```
tlex.getTimeline();
```

The output will be as shown in Listing 2. As can be seen, jTLEX returns the *main* timeline, subordinated timelines, and the attachment points for each subordinated timeline.

```
Main Timeline: {                          1
    eiid75- = 1                           2
    eiid75+ = 2                           3
    eiid74- = 3                           4
    eiid74+ = 4                           5
    eiid73- = 5                           6
    eiid76- = 5                           7
    eiid73+ = 6                           8
    eiid76+ = 6                           9
    t9- = 7                               10
    t9+ = 8                               11
}                                         12
Attachment Points: {eiid77->eiid78,       13
    eiid77->eiid80, eiid76->eiid77,
    eiid78->eiid79}
Subordinated Timelines: {                 14
[eiid81- = 1, eiid80- = 2, eiid81+ =      15
    3, eiid80+ = 3, t10- = 4, t10+ =
    5],
[eiid79- = 1, eiid79+ = 2],               16
[eiid78- = 1, eiid78+ = 2],               17
[eiid77- = 1, eiid77+ = 2]}               18
```

Listing 2: jTLEX timeline output for the `wsj_0006.tml` file.

```
[Graph Type: Main Graph                   1
Nodes Count = 2                           2
Links count = 2                           3
TLinkType: 2                              4
ALinkType: 0                              5
SLinkType: 0                              6
Nodes:                                    7
eiid2048, t57                             8
Links: (From -> To)                       9
(t57 BEFORE eiid2048)                     10
(eiid2048 BEFORE t57)                     11
]                                         12
```

Listing 3: jTLEX inconsistent subgraph output for the `wsj_1011.tml` file.

Since the graph of `wsj_0006.tml` is consistent, jTLEX's inconsistency detection method returns an empty set. We illustrate the inconsistency detection algorithm using a temporally inconsistent file from the TimeBank corpus, called `wsj_1011.tml`.

After running the method for graph construction and creating the tlex object, users can simply call the method `tlex.getInconsistentSubGraphs()` and retrieve the inconsistent cycle. For this file, jTLEX returns the output show in Listing 3. As can be seen from the output, jTLEX returns the inconsistent subgraph along with the information about the subgraph.

4 Related Work

As we discussed in Section 1, TimeML is a standardized temporal markup language in the NLP community. Therefore, many tools have been de-

veloped for TimeML-related tasks. We can classify existing TimeML tools into two categories: producing TimeML annotations and analyzing TimeML annotations. NLP researchers have concentrated tools of the first type, in particular for both full automatic TimeML annotation—such as TARSQI (Verhagen et al., 2005), ClearTK (Bethard, 2013), and CAEVO (Chambers et al., 2014)—and tools of the automatic annotation of sub-parts of TimeML—such as Evita (Saurí et al., 2005) and NavyTime (Chambers, 2013) for event detection, GATE-Time (Derczynski et al., 2016) and SUTime (Chang and Manning, 2012) for temporal expression recognition, and CATENA (Mirza and Tonelli, 2016) and LCC-TE (Min et al., 2007) for temporal relation extraction.

There are only a small number of tools that evaluate TimeML annotations. Tango is a Java TimeML parser tool to parse the TimeML annotated documents and create a TimeML graph (Verhagen et al., 2006). Tango allows users to modify the graph and it checks the temporal consistency of the graph using temporal closure. Tango was used to evaluate the TimeBank corpus, however, Tango did not report any inconsistency on the TimeBank files. Using the <TIMEX> values, Tango displays the graph in a timeline form, where each section contains a <TIMEX> and the events connected to the <TIMEX>, however, it doesn't provide the global order of events. Similarly, TBOX (Verhagen, 2007) also generates a TimeML graph from a TimeML annotation, but it further removes the temporal closure links to display a simplified TimeML graph. TBOX displays each event in a box shape and places each box based on the temporal relation to present the timeline (e.g., if event A is before event B, then box-A would be on the left of box-B). However, this representation could be problematic considering temporal indeterminacy is already high in TimeML annotations.

TimeML-strict is a Java validation tool that parses TimeML annotations and validates them whether they follow strict TimeML annotation guide rules (Derczynski et al., 2013). It also fixes missing document creation time (DCT) and <TEXT> tags in the annotations. CAVaT is a Python tool that parses TimeML annotations and prints out the quantitative results such as distributions of the TimeML objects (Derczynski and Gaizauskas, 2012). Further, it detects self-loops as well as disconnectivity in the TimeML graphs. CAVaT detects the temporal inconsistency of the graph using inconsistent disclosure. If the graph is inconsistent, it returns the last added constraint to the inconsistent cycle. Determining the entire inconsistent cycle based on one edge is very difficult for the annotators considering the graph size. CAVaT detects 30 inconsistent files in the TimeBank corpus. However, CAVaT's inconsistency detection algorithm only deals with TLINKs and ignores ALINKs and SLINKs. Later, (Ocal et al., 2022b) showed that by taking ALINKs into consideration, the TimeBank corpus actually has 65 inconsistent files.

In addition to these TimeML tools, NLP researchers have also developed ML-based approaches to extract timelines from TimeML annotations (Mani et al., 2006; Do et al., 2012; Kolomiyets et al., 2012; Leeuwenberg and Moens, 2020). However, these approaches have certain limitations such as they do not deal with all temporal links (at most 6 out of 13), they do not distinguish the real-life events and subordinated events, and they do not handle temporal indeterminacy.

Unlike other tools and approaches, in this work we provide an open-source implementation of TLEX, a method for extracting *exact* timelines from a TimeML annotation. Like prior approaches, TLEX—and by extension, jTLEX—offers a TimeML parser and a graph constructor. However, it goes further by separating subordinated events from real-life events, presenting the global order of events and times in a novel trunk-and-branch timeline structure, detecting inconsistencies automatically and helping users to fix them, representing multiple different orders in a single timeline, and measuring the indeterminacy score.

5 Discussion

We perform an extensive evaluation of the TLEX algorithm using the jTLEX output in our forthcoming paper (Ocal and Finlayson, 2023). We perform sampling evaluation using Simple Random Sampling (Saunders et al., 2009, p. 222), which allows us to check the correctness of a specific feature of a set of n members randomly selected from a population with size N to obtain an estimate of the correctness of that feature over all the data. Sampling evaluation shows that jTLEX achieved 100% accuracy on time point ordering and indeterminacy identification with 95% confidence (Ocal and Finlayson, 2023).

Because jTLEX can detect temporal errors in the

annotations and measure temporal indeterminacy, it can be used to evaluate automatic TimeML tools and manual TimeML annotations. An example of using jTLEX to evaluate automatic TimeML annotators can be found in Ocal et al. (2022a), and for using it to evaluate gold-standard TimeML annotations, see Ocal et al. (2022b).

6 Conclusion

We presented jTLEX, an open-source Java library that, for the first time, allows the programmatic extraction of exact timelines from TimeML annotated texts using a standard Java API. jTLEX provides many useful methods for the TimeML community such as TimeML parsing, graph extraction, timeline extraction, inconsistency detection, and temporal indeterminacy calculation. jTLEX can for used on any TimeML annotations in any domain of natural language. We release jTLEX as an open source library that is free for non-commercial use[4].

References

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

Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Second joint conference on lexical and computational semantics (* SEM), volume 2: proceedings of the seventh international workshop on semantic evaluation (SemEval 2013)*, pages 10–14.

Nathanael Chambers. 2013. Navytime: Event and time ordering from raw text. Technical report, NAVAL ACADEMY ANNAPOLIS MD.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Angel X Chang and Christopher D Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *Lrec*, volume 3735, page 3740.

Leon Derczynski and Robert Gaizauskas. 2012. Analysing temporally annotated corpora with cavat. *arXiv preprint arXiv:1203.5051*.

Leon Derczynski, Hector Llorens, and Naushad UzZaman. 2013. Timeml-strict: clarifying temporal annotation. *arXiv preprint arXiv:1304.7289*.

Leon Derczynski, Jannik Strötgen, Diana Maynard, Mark A Greenwood, and Manuel Jung. 2016. Gatetime: Extraction of temporal expressions and event. In *10th Language Resources and Evaluation Conference*, pages 3702–3708. European Language Resources Association (ELRA).

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL'12)*, pages 677–687.

Mark A Finlayson, Andres Cremisini, and Mustafa Ocal. 2021. Extracting and aligning timelines. In *Computational Analysis of Storylines: Making Sense of Events*, page 87. Cambridge University Press.

Antony Galton. 2009. Spatial and temporal knowledge representation. *Earth Science Informatics*, 2(3):169–187.

Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. 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.

Krzysztof Kuchcinski and Radoslaw Szymanek. 2013. JaCoP: Java constraint programming solver. http://jacop.cs.lth.se/.

Artuur Leeuwenberg and Marie-Francine Moens. 2020. Towards extracting absolute event timelines from english clinical reports. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2710–2719.

Shixia Liu, Michelle X. Zhou, Shimei Pan, Yangqiu Song, Weihong Qian, Weijia Cai, and Xiaoxiao Lian. 2012. Tiara: Interactive, topic-based visual text summarization and analysis. *ACM Trans. Intell. Syst. Technol.*, 3(2):25:1–25:28.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (ICCL-ACL'06)*, pages 753–760. Sydney, Australia.

Congmin Min, Munirathnam Srikanth, and Abraham Fowler. 2007. Lcc-te: a hybrid approach to temporal relation identification in news text. In *Proceedings of the fourth international workshop on semantic evaluations (SemEval-2007)*, pages 219–222.

Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Begona Altuna, Marieke Van Erp, Anneleen Schoen, and Chantal Van Son. 2016. Meantime, the newsreader multilingual event and time corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4417–4422.

[4] https://cognac.cs.fiu.edu/jtlex/

Paramita Mirza and Sara Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 64–75.

Mustafa Ocal and Mark Finlayson. 2020. Evaluating information loss in temporal dependency trees. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2148–2156, Marseille, France. European Language Resources Association.

Mustafa Ocal and Mark A Finlayson. 2023. Tlex: A formally correct method for extracting exact timelines from timeml temporal graphs. *Journal of Artificial Intelligence Research (JAIR)*, Manuscript submitted for publication.

Mustafa Ocal, Adrian Perez, Antonela Radas, and Mark Finlayson. 2022a. Holistic evaluation of automatic timeml annotators. In *Proceedings of the Language Resources and Evaluation Conference*, pages 1444–1453, Marseille, France. European Language Resources Association.

Mustafa Ocal, Antonela Radas, Jared Hummer, Karine Megerdoomian, and Mark Finlayson. 2022b. A comprehensive evaluation and correction of the timebank corpus. In *Proceedings of the Language Resources and Evaluation Conference*, pages 2919–2927, Marseille, France. European Language Resources Association.

James Pustejovsky, José Castaño, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003a. TimeML: robust specification of event and temporal expressions in text. In *Fifth International Workshop on Computational Semantics (IWCS-5)*, pages 1–11.

James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Rob Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. 2003b. The TimeBank corpus. In *Proceedings of Corpus Linguistics Conference*, pages 647–656. Lancaster, UK.

E. Saquete, P. Martínez-Barco, R. Muñoz, and J. L. Vicedo. 2004. Splitting complex temporal questions for question answering systems. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

M.N.K. Saunders, Philip Lewis, and Adrian Thornhill. 2009. *Research Methods for Business Students, Fifth Edition*. Prentice Hall, New York.

Roser Saurí, Robert Knippen, Marc Verhagen, and James Pustejovsky. 2005. Evita: a robust event recognizer for qa systems. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 700–707.

Roser Sauri, Jessica Littman, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. 2006. TimeML annotation guidelines, version 1.2.1. https://catalog.ldc.upenn.edu/docs/LDC2006T08/timeml_annguide_1.2.1.pdf.

Preston So. 2018. React. In *Decoupled Drupal in Practice*, pages 313–334. Springer.

Marc Verhagen. 2007. Drawing timeml relations with tbox. In *Annotating, extracting and reasoning about time and events*, pages 7–28. Springer.

Marc Verhagen, Robert Knippen, Inderjeet Mani, and James Pustejovsky. 2006. Annotation of temporal relations with tango. In *LREC*, pages 2249–2252.

Marc Verhagen, Inderjeet Mani, Roser Saurí, Jessica Littman, Robert Knippen, Seok Bae Jang, Anna Rumshisky, John Phillips, and James Pustejovsky. 2005. Automating temporal annotation with tarsqi. In *ACL*, pages 81–84.