*Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022)*, pages 1444–1453
Marseille, 20-25 June 2022


# Holistic Evaluation of Automatic TimeML Annotators

**Mustafa Ocal, Adrian Perez, Antonela Radas, & Mark A. 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, apere946, arada002, markaf}@fiu.edu

**Abstract**

TimeML is a scheme for representing temporal information (times, events, & temporal relations) in texts. Although automatic TimeML annotation is challenging, there has been notable progress, with $F_1$s of 0.8–0.9 for events and time detection subtasks, and $F_1$s of 0.5–0.7 for relation extraction. Individually, these subtask results are reasonable, even good, but when combined to generate a full TimeML graph, is overall performance still acceptable? We present a novel suite of eight metrics, combined with a new graph-transformation experimental design, for holistic evaluation of TimeML graphs. We apply these metrics to four automatic TimeML annotation systems (CAEVO, TARSQI, CATENA, and CLEARTK). We show that on average 1/3 of the TimeML graphs produced using these systems are inconsistent, and there is on average 1/5 more temporal indeterminacy than the gold-standard. We also show that the automatically generated graphs are on average 109 edits from the gold-standard, which is 1/3 toward complete replacement. Finally, we show that the relationship individual subtask performance and graph quality is non-linear: small errors in TimeML subtasks result in rapid degradation of final graph quality. These results suggest current automatic TimeML annotators are far from optimal and significant further improvement would be useful.

**Keywords:** TimeML, Timelines, Evaluation Metrics

## 1. Introduction

TimeML is an annotation scheme for representing temporal information in natural language texts. Automatic TimeML annotation is a challenging task, not only because it comprises three subtasks (time expression detection, event detection, and TimeML relation extraction), but also because temporal information is often implicit in the commonsense meaning of the language. There are several systems that can automatically generate TimeML graphs by combining the output of components that each focus on one of the individual subtasks. When evaluated in isolation, the components perform well on event detection (0.80–0.85 $F_1$ score) and time expression detection (0.82–0.93 $F_1$ score), while automatic relation extraction trails both in performance (0.50–0.63 $F_1$) and coverage (4–11 out of 25 relation types extracted). Despite these respectable individual performances, the question remains of how the performance of the individual components relates to overall performance, and how that affects a logical next task, namely, timeline extraction. Limited coverage of relation types is especially problematic, because while a reported $F_1$ for relation extraction might be respectable, even good, all extant systems only extract a subset of relation types, and how these omissions effect later tasks has not been evaluated. Furthermore, running each component in isolation and then combining the results does not attend to question of consistency, which is critical to the utility of the final TimeML graphs.

We present a new suite of methods for evaluating TimeML graphs and, consequently, measuring the *holistic* performance of existing TimeML annotation suites. Our evaluation methods include four graph-based metrics (relation distribution, number of closure links, edit-distance from gold standard, and the overall consistency of the graph) and four timeline-based metrics (timeline length, missing times and events, subordination structure, and temporal indeterminacy). We combine five of these metrics with a novel graph-transformation experimental design that allows us to investigate how the metrics vary as TimeML graphs degrade.

In our experiments, we used the TimeBank corpus (Pustejovsky et al., 2003) which is the original TimeML reference corpus. We evaluated four automatic TimeML annotation systems: TARSQI (Verhagen et al., 2005), CLEARTK (Bethard, 2013), CAEVO (Chambers et al., 2014), and CATENA (Mirza and Tonelli, 2016). These systems collectively represent the state-of-the-art in automatic TimeML graph annotation.

The paper is organized as follows. First, we review TimeML, prior work on automatic TimeML annotation, and approaches for timeline extraction (§2). Next, we explain our suite of metrics in detail (§3), and we present our experimental evaluation, including our novel experimental design involving transforming gold standard graphs step-by-step into automatically generated graphs to probe how the metrics vary as graph fidelity degrades (§4). Finally, we discuss the results (§5) and provide a summary of the contributions (§6). We release our code and data to enable reproduction of our results[1].

[1] https://doi.org/10.34703/gzx1-9v95/ZXHACI

## 2. Related Work

### 2.1. TimeML

TimeML is an SGML-based markup language designed to annotate temporal information in natural language texts (Sauri et al., 2006). TimeML is built upon Allen's Interval Algebra, a calculus for temporal reasoning that defines 13 possible relations between time intervals (Allen, 1983). TimeML has five types of objects. An EVENT object is used to represent events and changing states in documents, where an event is a situation that happens or occurs. A TIME object is used to represent temporal expressions such as dates, time intervals, time points, and durations. Temporal expressions can be a single word or a whole phrase.

LINK objects are used to identify relationships between two events, two times, or between an event and a time. There are three types of TimeML links: Temporal Links (TLINK), Aspectual Links (ALINK), and Subordination Links (SLINK). A TLINK object represents temporal relationship between events and times, of which there are 14 types: BEFORE, AFTER, SIMULTANEOUS, IDENTITY, I_BEFORE, I_AFTER, INCLUDES, IS_INCLUDED, DURING, DURING_INVERSE, ENDS, ENDED_BY, BEGINS, and BEGUN_BY. An ALINK object represents an aspectual relationship between an event and one of its subevents, of which there are five types: INITIATES, REINITIATES, CULMINATES, TERMINATES, and CONTINUES. An SLINK object represents counterfactual, conditional, or possible events, as well as marking factuality. There are six types: MODAL, FACTIVE, COUNTER_FACTIVE, CONDITIONAL, EVIDENTIAL, and NEGATIVE_EVIDENTIAL. An SLINK does not imply temporal semantics; two events can be related by both a subordinating and temporal link.

Using TimeML annotations, we can represent the temporal information of the text in a graph, called TimeML graph. A TimeML graph is a graph whose nodes are events and times, and its edges are TimeML links. Take the text shown in Example (1), which is a snippet of the TimeML-annotated text from the TimeBank corpus. The TimeML graph corresponding to the snippet is shown in Figure 1, where we can see that nodes of the graph are either events or times, and the edges are TimeML relations (DCT = *Document Creation Time*).

(1) [**DCT:11/02/89**$_1$]: Pacific First Financial Corp. **said**$_2$ shareholders **approved**$_3$ its **acquisition**$_4$ by Royal Trustco Ltd. of Toronto for $27 a share, or $212 million. The thrift holding company **said**$_5$ it **expects**$_6$ to **obtain**$_7$ regulatory **approval**$_8$ and **complete**$_9$ the **transaction**$_{10}$ by **year-end**$_{11}$. [from `WSJ.0006.tml`]

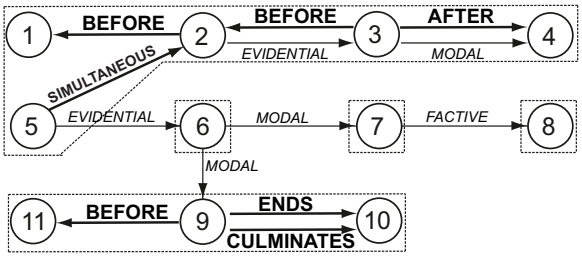


Figure 1: Visualization of the gold-standard TimeBank TimeML graph for Example (1). TLINKs and ALINKs are given in bold, and SLINKs are in italic. Numbers correspond to events and times numbered in the example. The five temporally and aspectually connected subgraphs are separated by dashed lines.

### 2.2. Automatic TimeML Annotators & Data

Creating a full TimeML graph from texts includes three subtasks: temporal expression (time) detection, event detection, and TimeML link extraction. There has been quite a bit of work addressing each of these subtasks individually (Verhagen et al., 2006; Seker and Diri, 2010; Lenzi et al., 2012, for example), but there are only a few systems that can provide integrated capabilities. Of the systems that tackle an individual subtask, there are:

**Time Expressions** HeidelTime recognizes temporal expressions using regular expressions as well as POS tags and handcrafted rules, achieving 0.86 $F_1$ (Strötgen and Gertz, 2010). SUTime, part of the Stanford CoreNLP pipeline, recognizes and normalizes times in documents using regex rules, POS tags, and named entity tags, and achieves 0.92 $F_1$ (Chang and Manning, 2013). SynTime is also a rule-based time recognition tool, additionally using token types to achieves 0.92 $F_1$ (Zhong et al., 2017). PTime generates patterns and selects them using the Extended Budgeted Maximum Coverage (EBMC) model, achieving 0.93 $F_1$ on tweets and 0.87 $F_1$ on a TimeML annotated corpus (Ding et al., 2019).

**Events** NavyTime detects events using POS tags, $n$-grams, lemmas, WordNet events, parse path, and typed dependencies with a MaxEnt Classifier, achieving 0.80 $F_1$ (Chambers, 2013). Sprugnoli and Tonelli (2019) detects events using a CRF classifier with lemma, POS tags, and text genre as features, achieving a 0.83 $F_1$ in historical texts. Multilingual Sequence Tagger (M-LiST) is a deep learning model which uses word alignment, feature encoder, and sequence tagger and achieves 0.86 $F_1$ on event recognition (Goud et al., 2019).

**Temporal Relations** Galvan et al. (2018) presented a bidirectional LSTM-RNN end-to-end neural model for medical domain texts to extract five types of TLINKs: BEFORE, BEGINS, INCLUDES, ENDS and OVERLAP. The system uses event attributes as features and achieves 0.62 $F_1$. Ning et al. (2019) implemented a semi-supervised system combining a structured perceptron algorithm and constraint-driven learning (CoDL) to extract TLINKs from documents. The system achieves 0.67 $F_1$ on event to event relations on five TLINKs: BEFORE, AFTER, INCLUDES, IS_INCLUDED,

and SIMULTANEOUS—notably, the system does not extract event to time or time to time relations.

**Integrated Systems** There are four systems that integrate multiple subtasks. The TARSQI toolkit (Verhagen et al., 2005) takes raw texts as input and recognizes time expressions and normalizes their values using a component called GUTIME. TARSQI recognizes events using a component called EVITA, which is a domain-independent event tagger, and identifies SLINKs, ALINKs, and TLINKs between temporal entities using supervised classification models. TARSQI merges the links using its SPUTLINK component, which applies constraint propagation algorithms to obtain a consistent annotation. TARSQI generates all types of SLINKs and ALINKs. It also generates 9 types of TLINKs: BEFORE, AFTER, INCLUDES, IS_INCLUDED, SIMULTANEOUS, IDENTITY, BEGUN_BY, ENDS, and ENDED_BY.

CLEARTK (Bethard, 2013) ranked first for temporal relation identification on TempEval-2013, and comprises three modules. The first module identifies events in texts and determines their attributes using event text, stem, part-of-speech tags, $n$-grams, and other related features. The second module identifies time expressions as well as time type and value using time-related features and the temporal type of each alphanumeric sub-token of time words. The third module predicts (only) TLINKs between each event and the DCT, between events and times in the same sentence, and between events in the same sentence. CLEARTK generates four types of TLINKs: BEFORE, AFTER, INCLUDES, and IS_INCLUDED.

CAEVO (Chambers et al., 2014) works similarly to TARSQI across four substeps: (1) time expression detection, (2) event detection, (3) temporal relation extraction, and (4) transitive inference over TLINKs. CAEVO uses the SUTIME to extract times, and NAVYTIME to extract events. CAEVO then extracts (only) TLINKs between temporal entities in the same sentence and neighboring sentences using a supervised classifier. It also extracts TLINKs between the document creation time (DCT) and every temporal entity. Finally, it applies transitive rules on TLINKs to extract dense TimeML annotation. CAEVO produces five types of TLINKs: BEFORE, AFTER, INCLUDES, IS_INCLUDED, and SIMULTANEOUS.

Finally, CATENA (Mirza and Tonelli, 2016) is a state-of-the-art sieve-based system for temporal relation extraction which takes texts pre-annotated with times and events and generates TLINKs between all event-event, event-time, DCT-event, and DCT-time pairs using a Support Vector Machine (SVM) and a temporal reasoner. CATENA generates 10 types of TLINKs: BEFORE, AFTER, INCLUDES, IS_INCLUDED, SIMULTANEOUS, BEGINS, BEGUN_BY, ENDS, ENDED_BY, and DURING.

Although there are a number of manually annotated TimeML corpora (Pustejovsky et al., 2003; Finlayson et al., 2014; Minard et al., 2016), in our experiments we used the TimeBank corpus (Pustejovsky et al., 2003) because it is the largest gold-standard TimeML-annotated corpus available, comprising 183 texts, 68,555 words, 7,935 events, 1,414 temporal expressions, and 9,615 TimeML links including all types of TimeML links.

## 2.3. Timeline Extraction Systems

The final four metrics we present below rely on extracting a timeline from the TimeML graph, where a timeline is a structure that gives a total order of events and times. There have been a number of attempts to develop automatic systems for timeline extraction from TimeML annotated texts. Do et al. (2012) provided a supervised machine learning-based system to extract timelines from TimeML annotations using lexical, syntactic, semantic, and event coreference features. The system achieves roughly 73% accuracy for TimeML annotations, but only works on three temporal relation types: BEFORE, AFTER, and SIMULTANEOUS. Kolomiyets et al. (2012) presented two different timeline extraction models: one based on shift-reduce parsing and another on graph parsing. Both models produce a temporal dependency tree (TDT) structure from the text order of the events. Their system achieved 70% accuracy on six temporal relations: BEFORE, AFTER, IDENTITY, INCLUDES, IS_INCLUDED, and OVERLAPS. Leeuwenberg and Moens (2020) presented an approach to extracting timelines from TimeML annotated clinical texts. Their timeline system is built on the system's subtasks of predicting duration of events and anchoring start time points of events. Although the system achieves an accuracy of 77%, the system only deals with BEFORE, AFTER, and OVERLAPS temporal relations.

Instead of these approximate methods, for this work, we use the exact method of Finlayson et al. (2021) to generate timelines. This technique takes a consistent TimeML graph as input, and has three steps to extract a timeline.

**Step 1: Partitioning** Partitioning the TimeML graph is achieved by walking the graph, setting aside all SLINKs. SLINKs do not imply a temporal ordering between nodes, and therefore they do not affect the structure of the timeline. SLINK information is retained only to indicate how to separate timelines are related (as shown by the gray regions in Figure 3). Our running example contains five temporally connected subgraphs, as shown by the dashed lines in Figure 1.

**Step 2: Transformation** Second, each temporally connected subgraph is transformed into a point algebra (PA) graph, which is a constraint graph where the nodes are time points and the edges are one of the two temporal primitive constraints less than ($<$) or equals ($=$). To transform each subgraph to a PA graph, first, each time and event (representing an interval) are replaced by start ($I^-$) and end points ($I^+$), linked by a $<$ relation. Then, each TLINK and ALINK is transformed into primitive constraints according to Table 3 provided in Ocal and Finlayson (2020). Each TLINK and ALINK link can be expressed as no more than two primitive constraints ($<$ , $=$). For example, A BEFORE B is equivalent to $A^- <$

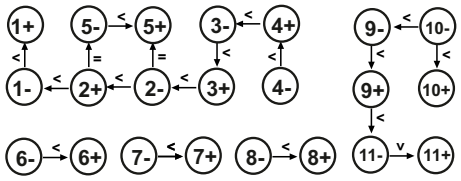


Figure 2: Visualization of the output of the transforming temporally connected subgraphs to PA graph.

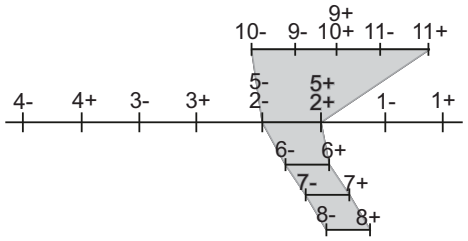


Figure 3: Visualization of the timeline of the gold-standard TimeML graph. Grey areas indicate subordinating timelines.

$A^+ < B^- < B^+$. The PA graph for our example gold-standard TimeML graph is shown in Figure 2.

**Step 3: Solving** Finally, the timeline is generated by solving the point algebra graph using a constraint solver, assigning integers to each time point that satisfies the constraint links. The sorted list of these integers is the order of time points. Because there may be multiple solutions, we take the shortest timeline as the reference. As previously noted, SLINKs are used to identify attachment regions between different timelines. The final set of timelines of the gold-standard TimeML graph is shown in Figure 3. Constraint solving can only find a timeline if the temporal subgraphs are consistent. While all gold-standard graphs in the corrected TimeBank are consistent, we observed temporal inconsistency in many cases in our experiments, as discussed in Section 4.2.

## 3. Metrics

We present eight different metrics: four graph-based metrics, where we assess (1) relation distribution, (2) closure links, (3) edit-distance from the gold standard, and (4) graph consistency; and four timeline-based metrics, where we assess (5) timeline length, (6) missing time points, (7) subordination structure, and (8) temporal indeterminacy. In the descriptions of the metrics below, the *target* graph refers to the graph on which the metric is being computed, namely, the graph being evaluated.

### 3.1. Metric 1: Relation Distribution

This metric computes the difference in the number of TLINK, ALINK, and SLINK relations in the graph in comparison to the gold standard, expressed as a fraction of the number of links in the gold standard. A positive fraction means there are more relations in the target; negative means fewer. This metric can be broken into 28 sub-metrics: 3 metrics measuring TLINK, ALINK, and SLINK categories overall, and 25 metrics each corresponding to each individual relation type (e.g., BEFORE, AFTER, etc.).

### 3.2. Metric 2: Closure Links

This metric is the number of closure links that can be generated from a graph. In temporal algebra, transitive closure is a relationship between three time points $x$, $y$, and $z$ where the temporal link from x to z is inferable from the temporal links $x$ to $y$ and $y$ to $z$. For example, if $x$ is AFTER $y$ and $y$ is AFTER $z$, we can infer $x$ is AFTER $z$. Unlike gold-standard annotations, automatic TimeML annotators use transitive closure to produce more TLINKs. If the existing two links are incorrect, transitive closure can produce another incorrect link. While the existing links are correct, transitive closure generates a link that does not provide any additional information to global order, overall consistency, or the temporal indeterminacy. We exclude links generated by transitive closure when comparing the automatically generated graphs with the gold-standard graphs since the gold-standard graphs do not have transitive closure links.

To compute this metric, we followed Chambers et al. (2014)'s transitive closure rules and implemented a system that counts the extra links (which also allows us to eliminate them during other comparisons). Figure 7 illustrates the case where there are two extra links that are generated by transitive closure: 2−AFTER→4 (since 2 is AFTER 3 and 3 is AFTER 4) and 5−BEFORE→7 (since 5 is IS_INCLUDED 6 and 6 is BEFORE 7).

### 3.3. Metric 3: Edit Distance

The third metric captures the edit distance of the target from the gold standard, represented as a fraction: the number of graphs edits required to transform a target graph into the gold-standard graph divided by the number of nodes and edges in the gold standard. This metric will always be positive. To compute the raw edit distance, we compute the maximum common subgraph between the target and the gold-standard (Bunke and Shearer, 1998). Next, we count both (a) the number of relations and nodes that are present in the target, but not in the gold-standard, and (b) the number of relations and nodes that are present in the gold-standard, but not in the target. The edit distance is the sum of these.

To illustrate this, compare the example gold-standard graph (Figure 1) with the automatic graph computed using CAEVO (Figure 7). The maximum common subgraph between these two graphs comprises Nodes 2–10 and $\text{AFTER}_{3\rightarrow4}$. There are seven relations (all other relations) in the target that are not present in the gold standard, however, two of them are extra links and they are eliminated by the previous process, therefore, we have five. And there are two nodes and thirteen relations

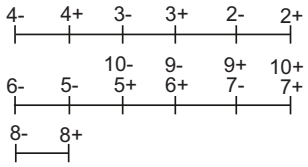


Figure 4: Visualization of the timeline of the CAEVO annotated TimeML graph. Three different timelines for three temporally connected subgraphs.

that are present in the gold standard but not in the target. This means that the edit distance is 20.

### 3.4. Metric 4: Graph Consistency

Barták et al. (2014) showed that a timeline can be extracted if and only if the temporal graph is inconsistent. If the timeline cannot be extracted, that means the graph is consistent. Zaidi (1999) showed that temporal consistency is caused by a cycle of relations which indicates no assignment is possible for the graph.
In this metric, using Algorithm 1 provided in (Ocal and Finlayson, 2020), we calculated the number of inconsistent files of each automatic TimeML annotator's files as well as the total number of inconsistent cycles of relations that automatic annotations have.

### 3.5. Metric 5: Timeline Length

This metric computes the difference in length of the longest timeline of a target TimeML graph compared to that of the gold standard. For example, the longest timeline of the gold-standard graph of the example text (Figure 3) has a length of eight while the longest timeline of the CAEVO annotated graph (Figure 4) has six, which means the target timeline is two units shorter than the gold-standard timeline.

### 3.6. Metric 6: Missing Timepoints

This metric computes the difference between the number of unique timepoints across all timelines of a target graph and that of the gold standard. For example, according to the timeline of the gold-standard graph of the example text (Figure 3), the timeline of the CAEVO annotated graph (Figure 4) misses four time points which are $1-$, $1+$, $11-$, and $11+$.

### 3.7. Metric 7: Subordination Structure

This metric computes the number of subordinated timelines, which can be compared with the number of subordinated timelines in the gold standard.

### 3.8. Metric 8: Temporal Indeterminacy

One of the most useful features of the timeline extraction technique in Ocal and Finlayson (2020) is its ability to measure temporal *indeterminacy*. Temporal indeterminacy arises because, in many cases, TimeML graphs lack enough information to specify a unique total ordering of time points. This can be a natural function

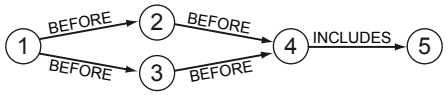


Figure 5: A temporal graph that results in temporal indeterminacy. While the global ordering of 1, 4, and 5 are fixed, the relative order of 2 and 3 are indeterminant.

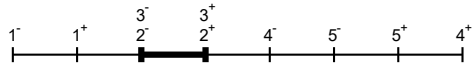


Figure 6: Visualization of an indeterminant graph. Between 2 and 3, there's a temporal indeterminacy.

of linguistic ambiguity, and does not necessarily imply an incorrect graph. A simple example of temporal indeterminacy is shown in Figure 5. In this TimeML graph, the order of time points would be $1^- < 1^+ < 2$ and $3 < 4^- < 5^- < 5^+ < 4^+$. However, the order between 2 and 3 is not fixed. There could be fourteen possible TLINK types between 2 and 3. The corresponding timeline is shown in Figure 6, where indeterminant time points on timeline are highlighted in bold.
Ocal and Finlayson (2020) provide an algorithm for computing temporal indeterminacy given all possible solutions to a temporal graph (Algorithm 2, p. 2153). This relies on the ability of most constraint solvers to produce both the smallest solution as well as all possible solutions for a constraint graph. The algorithm determines whether the order of a pair of adjacent timepoints are the same in all possible timelines as in the smallest timeline. If they are not always adjacent or in the same order, this pair is marked as indeterminate time points, allowing visualization of indeterminate sections of the timeline, as shown in Figure 6 for the temporal graph shown in Figure 5.

## 4. Experimental Results

### 4.1. Generating TimeML graphs

The TimeBank corpus (Pustejovsky et al., 2003) has 183 texts in three formats: raw text, pre-annotated texts, and gold-standard annotated texts. Gold-standard annotated texts are files where each TimeML component (events, times, links, etc.) is labeled with TimeML tags while pre-annotated texts have only TimeML tags for events and times. Later work showed that 30 texts had errors and provided corrections (Derczynski and Gaizauskas, 2012); we used the corrected versions.
To generate baseline TimeML graphs for evaluation, we applied CAEVO, CLEARTK, and TARSQI to the raw TimeBank texts to generate TimeML-annotated texts. Only TARSQI has the capability of producing SLINKs and ALINKs between events. Because CATENA only generates TLINKs (it does not detect events and times) we give pre-annotated texts as input. Similar to CAEVO and CLEARTK, CATENA also can't generate SLINKs or

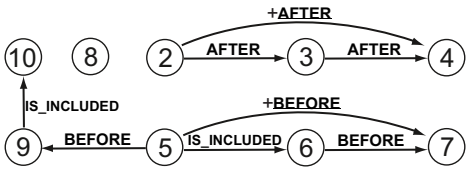


Figure 7: The TimeML graph generated by CAEVO when run on Example (1). Numbers correspond to events and times numbered in the example. Underlined TLINKs indicate closure links.

ALINKs between events. We provided an illustration of the TimeML graph that is extracted from the CAEVO annotation of our example text (Example (1)) in Figure 7. Comparing Figures 1 and 7, several important differences are evident. First, the gold-standard graph is a single connected graph while the generated graph comprises three separate graphs. Second, the generated graph contains no SLINKs and ALINKs because CAEVO, like nearly all automatic TimeML annotation systems, does not generate those link types. Third, the generated graph has more TLINKs than the gold-standard graph, because automatic annotators use transitive closure to generate links consistent with automatically detected links. For example, in Figure 7, event 2 (*said*) is BEFORE event 3 (*approved*), which is BEFORE event 4 (*acquisition*); consequently, CAEVO adds a BEFORE link between events 2 and 4.

## 4.2. Results of Graph-based Metrics

First, we analyzed how many types of TLINK automatic TimeML graphs have and how many total TLINKs they contain. Second, we calculated the closure links and set them aside for comparison. Then, as it's described in Section 3.3, we calculated the average distance between automatic graphs and gold-standard graphs. The results are shown in Table 2.

**Metric 1: Relation Distribution** The distribution of TLINK types in the graphs is shown in Table 1. We note that automatic graphs have many more BEFORE and AFTER links than the gold-standard graphs. While only 35.9% of gold-standard links are BEFORE and AFTER, this number goes up to 76.9% in CAEVO graphs, 78.4% in TARSQI graphs, 69.8% in CATENA graphs, and 76.8% in CLEARTK graphs. This means the automatic annotators tend to generate BEFORE and AFTER TLINKs more than any other TLINK type, meaning they miss a large portion of TLINK types.

We note that TimeML graphs of automatic annotations don't have every TLINK type. The state-of-the-art TLINK detection system - CATENA - produces 10 types of TLINKs while CLEARTK only produces four types of TLINKs. This means they're missing a significant amount of temporal relations. We also note that although automatic systems produce less types of TLINKs, they produce more total TLINKs than gold-standard. While gold-standard graphs have only 6,418 TLINKs, CLEARTK and CAEVO graphs have more than 9,000 TLINKs. One of the reasons for that is automatic systems use transitive closure when it produces TLINKs. As we mentioned earlier, while automatic TimeML annotators use transitive closure as a feature in their system, gold-standard annotations do not necessarily include closure links. This results in automatic annotations having more TLINKs than gold-standard annotations. However, even without closure, automatic annotators generate many more TLINKs than gold-standard annotations. For example, CATENA annotated graphs have two times more TLINKs than gold-standard TLINKs. This suggests automatic annotators generate many incorrect TLINKs.

**Metric 2: Closure Links** We calculated the extra links implied by temporal closure. As shown in Table 2, CAEVO generated 876 extra links, 9.7% of its total links. In the meantime, TARSQI generates 268 extra links (4.2%), CATENA generates 1085 (6.4%), and CLEARTK generates only 67 (0.7%).

**Metric 3: Edit Distance** We calculated the edit distance between gold-standard graphs and automatic graphs. Table 2 shows that on average, each automatic graph requires more than 77 steps to generate gold-standard graphs. Although CATENA is the state-of-the-art for TLINK identification, CATENA graphs are the most distant graphs among all automatic graphs. This is because CATENA produces a large number of TLINKs, most of which are incorrect.

**Metric 4: Graph Consistency** In some cases, we cannot extract timelines from target graphs because they are temporally inconsistent. Inconsistency in the original gold-standard TimeBank annotations are the result of annotators' mistakes, and for the timeline metrics later we use corrected TimeBank gold-standard files where each inconsistency is manually fixed by Ocal and Finlayson (2020). For this metric, we measured how many inconsistent graphs exist in the 183 annotated texts. As mentioned previously, there were 30 inconsistent files in the original gold-standard TimeBank corpus; we use the corrected versions for comparison in our experiments. Although CATENA has the highest $F_1$ score for TLINK detection, CATENA annotations have 159 inconsistent files out of 183 files. We note that TARSQI annotations introduce less inconsistency than the gold-standard annotations.

## 4.3. Results of Timeline-based Metrics

**Metric 5: Timeline Length** Some automatic annotators compute shorter timelines than gold-standard timelines while others output longer timelines. We report Root Mean Square Error (RMSE) measures for timeline length over the corpus for each system, we can clearly see that RMSE scores for automatic annotations are high. We also note that timelines of CATENA annotations are much different than timelines of other annotations. The main reason automatic annotation timelines are longer than gold-standard annotation timelines is that automatic annotations produce high number of BEFORE and

| Annotations | BEFORE | AFTER | I_BEFORE | I_AFTER | SIMULTANEOUS | IDENTITY | INCLUDES | IS_INCLUDED | BEGINS | BEGUN_BY | ENDS | ENDED_BY | DURING | DURING_INV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **GS** | **21.9%** | **14.0%** | **0.5%** | **0.6%** | **10.5%** | **11.6%** | **9.1%** | **21.1%** | **1%** | **1.1%** | **1.2%** | **2.8%** | **4.7%** | **0.02%** |
| CAEVO | 43.8% | 33.1% | - | - | 1.9% | - | 2.3% | 18.8% | - | - | - | - | - | - |
| TARSQI | 47.6% | 30.8% | - | - | 0.8% | 9.2% | 3% | 8.4% | - | 0.1% | - | 0.03% | - | - |
| CATENA | 44.5% | 25.3% | - | - | 6.6% | - | 3.9% | 18.6% | 0.2% | 0.2% | 0.2% | 0.3% | 0.2% | - |
| CLEARTK | 53.8% | 23% | - | - | - | - | 2.8% | 20.4% | - | - | - | - | - | - |

Table 1: Metric 1: Relation Distribution (TLINKs only).

| Annotations | # of TLINK types | Metric 1: Total TLINKs | Total ALINKs | Total SLINKs | Metric 2: Closure Links | Metric 3: Avg. Edit Distance to GS Graphs | Metric 4: # of Inconsistent Main Graph |
|---|---|---|---|---|---|---|---|
| **Gold-Standard** | **14** | **6,418** | **265** | **2,932** | **-** | **-** | **-** |
| CAEVO | 5 | 9,062 | - | - | 876 | 125.26 | 72 |
| TARSQI | 8 | 6,366 | 93 | 1,325 | 268 | 77.54 | 9 |
| CATENA | 10 | 16,875 | - | - | 1,085 | 138.47 | 159 |
| CLEARTK | 4 | 9,015 | - | - | 67 | 94.49 | 34 |

Table 2: Metrics 1–4: Summary Counts, Closure Links, Edit Distances, and Graph Consistency.

AFTER relations (See §4.2).

**Metric 6: Missing Timepoints** Our results show that although automatically generated timelines have a similar number of time points to gold-standard timelines, the RMSE is high. This is because state-of-the-art systems detect incorrect events and times and miss a large number of correct ones. Since CATENA uses pre-annotated texts, we didn't include it in the table.

**Metric 7: Subordination Structure** As discussed, SLINKs are critical for timeline extraction: without them we will include events that may not happen in the real world in the main timeline. Among state-of-the-art TimeML annotators, only TARSQI detects SLINKs between events. Based on TARSQI's RMSE score for the number of subordinated branches, our result shows that TARSQI misses a significant number of SLINKs.

**Metric 8: Temporal Indeterminacy** We measured 67.9% overall indeterminacy score on gold-standard annotations. The reason why the indeterminacy score is already high for gold-standard annotation is that natural language texts usually don't specify the order of some events. However, as we can see in Table 3, the automatic annotators increased indeterminacy significantly. CAEVO annotations have 92.2% indeterminacy score, while CLEARTK annotations have 91% indeterminacy.

## 4.4. Effect of TimeML Graph Degradation on Metrics 4–8

We also wanted to investigate how errors in the detection of events, times, and relations affect the quality of resultant timelines. To do this we take advantage of our ability to transform one graph into another by incrementally removing or adding nodes and links (the total number of steps between gold-standard graph and automatically generated graph is captured by the edit distance metric). To measure this, we begin with the gold-standard graph and transform it into the automatically generated step-by-step in a random order, at each step either (i) removing a link or node that is not present in the automatic graph, or (ii) adding a link or node that is not present in the gold-standard graph[2]. After each editing step we run Metrics 4–8 on the new graph, allowing us to chart the change in those metrics as the graphs degrade. We can average these measures over multiple random sequences of edits.

We start with the 183 gold-standard graphs and transform them into the 734 automatically generated graphs (734 = 183*4, one generated graph for each of the four systems). We show the results for Metric 4 (Inconsistency) and Metric 8 (Indeterminacy) in Figure 8 and 9, respectively. As can be seen in Figure 8, for CAEVO, CATENA, and CLEARTK, the number of inconsistent graphs increases non-linearly: the number of inconsistencies increases relatively slowly and linearly until a breakpoint is reached (roughly 35–55% changed), after which the number of inconsistent graphs increases rapidly. TARSQI is an outlier to this pattern, we suspect because TARSQI imposes consistency checks in its pipeline; ultimate TARSQI only generates 9 inconsistent graphs across all texts.

In contrast, we can see that the increase in indeterminacy follows a smoother, more linear pattern for CAEVO, TARSQI, and CLEARTK. CATENA is the outlier here, because CATENA generates an extremely large number of links (see Table 2, CATENA's Metric 1, which is 16,875, more than twice the other systems), and so when gradually adding in these links indeterminacy temporarily surges, before falling back to the ultimate level found in the automatically generated graphs.

Note that this graph transformation technique does work for Metrics 5, 6, and 7, we do not show those results for several reasons. First, Metric 5 (average main timeline length) and Metric 6 (average number of time points) do not reveal any interesting patterns: the graphs are mainly flat with some noise. This means that when

[2]Naturally we do not add a missing link before the relevant nodes are present, nor do we remove a node if the relevant links are still present.

| Annotations 183 files | Metric 5: Avg. Main. Timeline Len. (RMSE) | Metric 6: Avg. Time Pts / Timeline (RMSE) | Metric 7: Avg. Sub. Branches (RMSE) | Metric 8: % Indeterminacy |
|---|---|---|---|---|
| **Gold-Standard** | **8.61** | **130.79** | **16.02** | **67.9%** |
| CAEVO | 8.32 (3.10) | 138.41 (10.40) | 0 (N/A) | 92.2% |
| TARSQI | 6.49 (3.85) | 100.39 (37.24) | 7.28 (17.09) | 80.7% |
| CATENA | 12.85 (7.56) | -* | 0 (N/A) | 66.7% |
| CLEARTK | 9.62 (3.04) | 94.55 (14.96) | 0 (N/A) | 91% |

Table 3: Metric 5–8: Timeline Length, Missing Timepoints, Subordination Structure, and Temporal Indeterminacy (* excluded since it uses the gold-standard events & times)

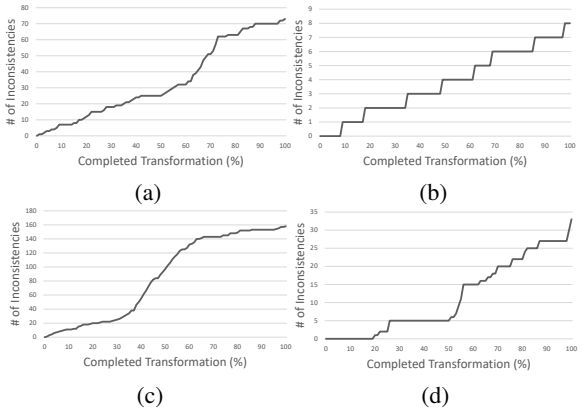


Figure 8: Increase in inconsistency during gradual stepwise transformation of gold-standard graphs (left side of each chart) to automatically generated graphs (right side). Charts represent the number of inconsistent graphs for (a) CAEVO, (b) TARSQI, (c) CATENA, and (d) CLEARTK. Each X-axis indicates the percentage of completed transformation, and each Y-axis indicates raw number of graphs that are inconsistent. The scale for each Y-axis is different because the total number of inconsistent graphs generated by each system is different.

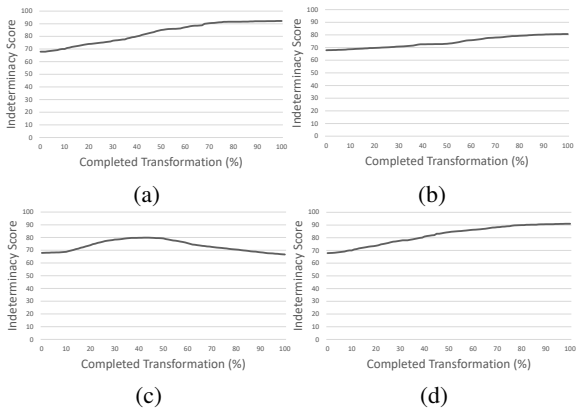


Figure 9: Change in average indeterminacy score during gradual, stepwise transformation of gold-standard graphs (left side of each chart) to automatically generated graphs (right side). Charts represent the average indeterminacy across all graphs for (a) CAEVO, (b) TARSQI, (c) CATENA, and (d) CLEARTK. Each X-axis indicates the percentage of completed transformation, and each Y-axis indicates the average indeterminacy score. Each Y-axis has the same scale.

the graphs are transformed, the average main timeline length and average number of time points stays roughly constant. For Metric 7, only TARSQI generates subordinating links, and we could discern no pattern: its graph is extremely noisy, as the number of subordinated branches should almost directly correlate with the number of subordinating links, and so addition or removal of those links will immediately change that metric.

## 5. Discussion

In this paper we presented a suite of evaluation metrics for automatic TimeML annotators and we evaluated four mainstream state-of-the-art automatic TimeML annotators. The results showed that automatic TimeML annotators generate many inconsistent annotations, which does not necessarily correlate with the raw performance of the detector: for example, even though CATENA has the best $F_1$ performance for TLINK extraction (Mirza and Tonelli, 2016), it has 159 inconsistent files, many more than TARSQI, which has worse TLINK detection performance but only 9 inconsistent files.

Unlike the gold-standard annotations, automatic annotations have much more BEFORE and AFTER links (70–80% of total TLINKs). This is mainly because the automatic TimeML annotators generate only limited number of TLINK types, therefore, many links are mislabeled. Three of the four of the automatic TimeML annotators do not generate SLINKs, and so they don't distinguish real-life events from possible, conditional, or modal subordinated events in their timelines. The system that does generate SLINKs, TARSQI, misses more than half of the ALINKs and SLINKs.

As expected, mislabeled events, times, and links resulted in graphs that are much different than gold-standard graphs (78–139 edits distant). We investigated how various metrics will change during gradual transformation from the gold-standard to the automatically generated. This showed that, for inconsistencies, the relationship is not linear, in particular, that beyond a certain point inconsistencies rapidly increase.

## 6. Contributions

Our contributions in this paper are three-fold. **First,** we presented eight metrics for evaluating the quality of temporal graphs—four graph-based and four timeline-

based—and used these metrics to evaluate four mainstream, state-of-the-art temporal analysis systems. **Second,** we showed that, beyond a certain point, small errors in the detection of events, times, or relations result in rapid degradation of the inconsistency of the final graph. **Finally,** we showed that current automatic TimeML annotators are far from optimal and significant further improvement is potentially needed. We release our code and data to enable reproduction of the results[3].

## 7. Acknowledgements

We gratefully acknowledge valuable discussions with Dr. Karine Megerdoomian, Akul Singh, and Emmanuel Garcia. This work was partially funded by research grants TO-134841, TO-135998, and TO-139837 to FIU from MITRE Corporation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of MITRE Corporation. This work was also partially supported by DARPA Award HR001121C0186 under the INCAS Program.

## 8. Bibliographical References

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

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

Bethard, S. (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.

Bunke, H. and Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern recognition letters*, 19(3-4):255–259.

Chambers, N., Cassidy, T., McDowell, B., and Bethard, S. (2014). Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

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

Chang, A. and Manning, C. D. (2013). Sutime: Evaluation in tempeval-3. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 78–82.

Derczynski, L. and Gaizauskas, R. (2012). Analysing temporally annotated corpora with cavat. *arXiv preprint arXiv:1203.5051*.

[3] https://doi.org/10.34703/gzx1-9v95/ZXHACI

Ding, W., Gao, G., Shi, L., and Qu, Y. (2019). A pattern-based approach to recognizing time expressions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6335–6342.

Do, Q. X., Lu, W., and Roth, D. (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.

Finlayson, M., Halverson, J., and Corman, S. (2014). The n2 corpus: A semantically annotated collection of islamist extremist stories. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 896–902, Reykjavik, Iceland, May. European Language Resources Association (ELRA).

Finlayson, M. A., Cremisini, A., and Ocal, M. (2021). Extracting and aligning timelines. *Computational Analysis of Storylines: Making Sense of Events*, page 87.

Galvan, D., Okazaki, N., Matsuda, K., and Inui, K. (2018). Investigating the challenges of temporal relation extraction from clinical text. In *Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis*, pages 55–64.

Goud, J., Goel, P., Antony, A. J., and Shrivastava, M. (2019). Leveraging multilingual resources for open-domain event detection. In *Proceedings 15th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, pages 76–82.

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.

Leeuwenberg, A. and Moens, M. F. (2020). Towards extracting absolute event timelines from english clinical reports. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2710–2719.

Lenzi, V. B., Moretti, G., and Sprugnoli, R. (2012). Cat: the celct annotation tool. In *LREC*, pages 333–338. Citeseer.

Minard, A.-L., Speranza, M., Urizar, R., Altuna, B., van Erp, M., Schoen, A., and van Son, C. (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, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Mirza, P. and Tonelli, S. (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.

Ning, Q., Feng, Z., and Roth, D. (2019). A structured

learning approach to temporal relation extraction. *CoRR*, abs/1906.04943.

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

Pustejovsky, J., Hanks, P., Saurí, R., See, A., Gaizauskas, R., Setzer, A., Radev, D., Sundheim, B., Day, D., Ferro, L., and Lazo, M. (2003). The timebank corpus. *Proceedings of Corpus Linguistics*, 01.

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`.

Seker, S. E. and Diri, B. (2010). Timeml and turkish temporal logic. In *IC-AI*, volume 10, pages 881–887.

Sprugnoli, R. and Tonelli, S. (2019). Novel event detection and classification for historical texts. *Computational Linguistics*, 45(2):229–265, June.

Strötgen, J. and Gertz, M. (2010). Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324.

Verhagen, M., Mani, I., Saurí, R., Littman, J., Knippen, R., Jang, S. B., Rumshisky, A., Phillips, J., and Pustejovsky, J. (2005). Automating temporal annotation with tarsqi. In *ACL*, pages 81–84.

Verhagen, M., Knippen, R., Mani, I., and Pustejovsky, J. (2006). Annotation of temporal relations with tango. In *LREC*, pages 2249–2252.

Zaidi, A. K. (1999). On temporal logic programming using petri nets. *Trans. Sys. Man Cyber. Part A*, 29(3):245–254, May.

Zhong, X., Sun, A., and Cambria, E. (2017). Time expression analysis and recognition using syntactic token types and general heuristic rules. In *ACL*.