Language Resources and Evaluation (2005) 39: 211–241
DOI 10.1007/s10579-005-7884-5

Temporal Closure in an Annotation Environment

MARC VERHAGEN
Computer Science Department, Brandeis University, Waltham, MA, USA

1. Introduction

News articles typically present a story that develops over time. Events and times are introduced and the reader understands what the sequence of events is. Simple questions like “What happened after the kidnapping?” can only be answered if information about events and the temporal relations between events is available, and a document needs to be annotated automatically or manually to provide this information.

This article focuses on how a temporal closure component can be embedded in a temporal annotation environment. Temporal closure takes known temporal relations in a text and derives new implied relations from them, in effect making explicit what was implicit. A temporal closure component helps to create an annotation that is complete and consistent.

The assumption here is that explicit temporal annotation is required for natural language processing applications like question answering and text summarization. Take for example the question-answering task. We want to be able to answer questions that involve events occurring at certain times or events happening in a certain order. As an example consider the following fragment from a 1998 Associated Press newswire.

(1) Turkey (AP) – Some 1500 ethnic Albanians marched Sunday in downtown Istanbul, burning Serbian flags to protest the killings of ethnic Albanians by Serb police in southern Serb Kosovo province. The police barred the crowd from reaching the Yugoslavian consulate in downtown Istanbul, but allowed them to demonstrate on nearby streets.

This text is easy to understand and we all know what happened and when things happened. But what do we exactly need to know when we answer specific questions? Take the three questions below.

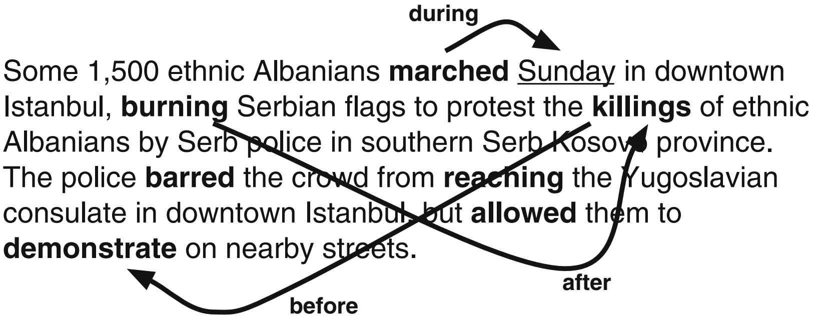
(2) What happened on Sunday?
(3) Were Serbian flags burned before the killings?
(4) Were ethnic Albanians killed during the demonstration?

The first one is the simplest. The text contains a temporal expression *Sunday* and an event-denoting verb right next to it. It is not too much of a stretch to put these two together. Question (3) is more complicated. But if we have encoded that a pattern “doing X to protest Y” implies that X is during Y or after Y, then we can give an answer. Question (4) is even harder because it requires ordering of events that do not occur in near proximity to each other, and there are no obvious markers that give us the needed information.

In any case, a document can be marked up to provide the information that we need. We can add tags to the text that mark the events and time expressions as well as the temporal relations between them. The text in example (1) can be marked up with temporal information as in Figure 1. The event *marched* is now explicitly anchored to the time expression *Sunday*. In addition, the events in the pairs *burning*-*killings* and *killings*-*demonstrate* are now ordered relative to each other. Once all events are anchored and ordered we can effectively create a timeline and graphically display the sequence of events in the document.

TimeML (Pustejovsky *et al.*, 2003) is an XML-compliant annotation language for temporal information. It uses a basic ontology of expressions denoting temporally relevant entities: time expressions are identified by the TIMEX3 tag and events are identified by the EVENT tag. The ontology also includes temporal relations between events and time expressions. The TLINK tag consumes no input and encodes temporal relations. One of its attributes contains the kind of temporal relation between two events or times. Some of the values allowed are before, simultaneous, includes, and begin.

A document could be processed automatically to achieve the results in Figure 1 and add the appropriate markup. The last couple of years have spawned significant research on event extraction (Aone and Ramos-Santacruz, 2000), extraction of time expressions (Mani and Wilson, 2000), and event anchoring and ordering (Filatova and Hovy, 2001; Schilder and Habel, 2001; Mani *et al.*, 2003). This research is promising and some components, most notably the recognition of time expressions, are of a high quality. Nevertheless, it is too early to



Some 1,500 ethnic Albanians **marched** Sunday in downtown Istanbul, **burning** Serbian flags to protest the **killings** of ethnic Albanians by Serb police in southern Serb Kosovo province. The police **barred** the crowd from **reaching** the Yugoslavian consulate in downtown Istanbul, but **allowed** them to **demonstrate** on nearby streets.

Figure 1. Enriching a text with temporal information.

comfortably use state-of-the-art automatic generation of temporal relations because it does not yet exhibit high enough precision and recall.[1]

Manual annotation needs to be a part of the total annotation effort given the precision and recall figures for machine annotation. But manual annotation comes with its own set of practical challenges. The task is a complex one, characterized by high density, low markup speed, hard-to-avoid inconsistencies, and low inter-annotator agreement.

The high density is due to the fact that the set of possible temporal relations is essentially quadratic to the number of events and time expressions in a document. If a document has N events and time expressions, then there are $N(N-1)/2$ possible temporal relations. A typical document contains about 50 temporal objects, which implies 1225 possible temporal relations. Larger documents with about 150 time objects (events and time expressions) have over 10,000 relations. An annotation that contains all temporal relations is clearly impractical by human means alone.

Annotation of temporal relations requires more reflection than, for example, annotation of part-of-speech tags and is, therefore, slower. Syntactic tags and many semantic tags like entity tags or event tags can be added in a strictly linear fashion. Temporal relations are different because they require us to specify attributes of pairs of objects, and the objects involved may not be close to each other in the text. Annotating a mid-sized newspaper article can take up to an hour.

Experience with consistency checking tools showed that it is hard to annotate a one-page document without introducing inconsistencies. An inconsistency can occur because the choice for a particular temporal relation often restricts choices down the road. For example, if an annotator decides that X is before Y and Y is before Z, then the choice of temporal relations between X and Z is constrained. But even trained annotators are liable to introduce relations that clash with previous choices. This is sometimes the result of vague or ambiguous temporal relations between events, and sometimes the result of a fuzzy interpretation of a particular relation (for example, does X includes Y mean that X and Y may share a beginning point or not). But often plain fatigue is to blame.

A manually annotated document is sparse in temporal information given the high density potential. More often than not, two annotators choose to add different temporal relations simply because the space they can pick from is so large, as depicted in Figure 2.

It is unreasonable to expect that an annotator adds much more than a hundred links in a two-page document. On average, annotators annotate about 1–5% of all possible links and in only about 10% of the cases two annotators chose to add temporal relations between the same two-time objects.[2]

The quality of an annotated document and the quality of an annotation specification and its guidelines are often measured by comparing the

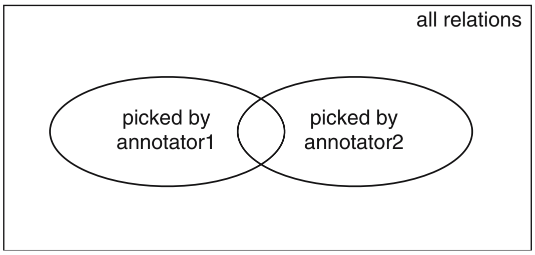


Figure 2. Two annotators marking up different temporal relations.

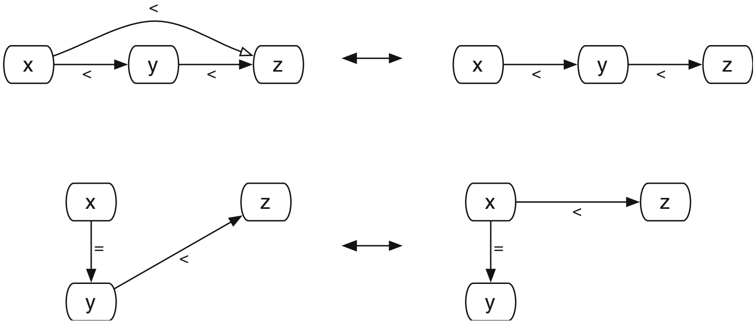


Figure 3. Two pairs of identical annotations.

annotations of two or three annotators, and low inter-annotator agreement scores are perceived by many to indicate an ill-defined annotation task (Hirschman *et al.*, 1998; Setzer, 2001). The sparsity of temporal annotation depresses inter-annotator agreement.

Another complication is that, unlike, for example, part-of-speech annotation, temporal annotations need to be compared at the semantic level and not the syntactic level. The two pairs of temporal networks in Figure 3 should be considered pair wise identical because they convey the same meaning, even though they do not contain the same temporal facts.

Some way is needed to compare temporal networks in a meaningful manner. Temporal closure can be used to map semantically identical annotations onto syntactically identical annotations.

1.1. Mixed-Initiative Annotation

So neither machine nor human can produce a high-quality annotation that is consistent and complete. Manual temporal annotation is expensive and time-consuming, and clearly impractical if it is to deliver a complete annotation. Fully automatic temporal annotation is not yet up to the task and exhibits precision and recall figures that are not high enough.

The solution is to let both human and computer do what they do so well. The human can quickly see how events relate in time without there being any single clear and explicit textual marker; the computer can deal with large amounts of tedious data and skillfully perform simple reasoning tasks. Mixed-initiative temporal annotation is a hybrid approach that goes some way towards meeting the practical challenges mentioned above. It was pioneered in the Alembic Workbench (Day *et al.*, 1997) and includes a range of modules:

- automatic pre-processing for those tasks that have high precision, most notably recognition of events and time expressions
- manual annotation
- a user-assisted temporal closure algorithm
- machine learning techniques

Here, we focus on the closure component. Temporal closure makes it easier to create a consistent temporal annotation because it constrains choices and finds inconsistencies in a set of relations added before closure was applied. In addition, temporal closure can be employed in a user-assisted mode where the user is asked to fill in temporal relations and the machine continues to add facts after each user-added relation.

I will show that this approach makes it feasible to achieve a near complete annotation because temporal closure will derive about 95% of the temporal relations.

In the following sections, I will first describe two annotation efforts that used a model-theoretic approach or an explicit temporal closure component for the purpose of annotation comparison (Section 2). Section 3 introduces SputLink, a temporal closure module intended to be embedded in an annotation environment. In essence, SputLink is based on a restricted interval algebra. This theoretical background is presented in Section 3.1. SputLink itself, as well as its embedding in an annotation environment, is described in Sections 3.2 and 3.3. Finally, Section 4 gives statistics from SputLink at work, including number and kind of links added (Section 4.1), inter-annotator agreement (Section 4.2), and data on user-assisted closure (Section 4.3).

2. Previous Approaches

Graham Katz and Fabrizio Arosio (Katz and Arosio, 2001) proposed a simple temporal annotation language for intra-sentential precedence and inclusion relations between verbs. Their language has labels $<$ and $>$ for precedence relations and $\subseteq$ and $\supseteq$ for inclusion relations. Each sentence also includes an indexical reference to the speech time which can be temporally related to the verbs. Annotations are provided with a model theoretic interpretation and consistency of annotations is defined in terms of satisfiability in models. Katz and Arosio report results from an experiment where two annotators annotated 50 complex sentences. The annotations were syntactically identical in only 70% of the cases, but they were semantically consistent in 85% of the cases.

Andrea Setzer and Robert Gaizauskas aimed to capture temporal information in newswire texts. To that end, they defined an annotation language and a set of annotation guidelines called STAG: Sheffield Temporal Annotation Guidelines (Setzer, 2001; Setzer and Gaizauskas, 2001). Events and times are connected using five temporal relations: before, after, includes, included and simultaneous. The fifth relation, simultaneous, is rather fuzzy and means something like "roughly at the same time".

Setzer and Gaizauskas used a deductive closure component to get more reliable inter-annotator agreement figures. They use a domain $E \cup T$ (events and time expressions) which has three binary relations defined on it. B, I and S are the sets of all pairs in the domain that are assigned the before, includes and simultaneous relations, respectively. Ten inference rules were derived from the formal properties of the STAG relations; simultaneous is an equivalence relation while before, includes and their inverses are transitive, asymmetric and irreflexive. Three of the rules are shown below.

(5) 1. $\forall\, x, y, z \in (E \cup T) : (x, y) \in S \Rightarrow (y, x) \in S$
2. $\forall\, x, y, z \in (E \cup T) : (x, y) \in B \wedge (y, z) \in B \Rightarrow (x, z) \in B$
3. $\forall\, x, y, z \in (E \cup T) : (x, y) \in B \wedge (y, z) \in S \Rightarrow (x, z) \in B$

Now standard precision and recall measures can be applied to the domain after computing the deductive closure of B, T and S. Measuring the inter-annotator agreement was Setzer and Gaizauskas' main application for the deductive closure component, but they proceeded to use it to increase the number of temporal facts in a text. They introduced two stages of annotation. In the first stage, the annotator would manually markup explicit and implicit temporal relations in the text. In the second stage, relations are normalized and all inferences that can be drawn are added to the set of relations. Then the system enters a loop where the user is prompted to specify the relation between two events and time expressions that have not yet been related. Each time the user adds a fact, the closure component tries to add new inferences. This loop continues until all relations are specified.

This inferencing approach is not sound due to the fuzzy nature of the simultaneous relation. Setzer acknowledges this in her thesis and gives an example that illustrates this, here presented in Figure 4. Event x is before event y and event y is simultaneous with event z ("roughly at the same time"), yet event x is not before event z, thereby violating rule 3. These incorrect inferences are not necessarily a problem when the closure component is used

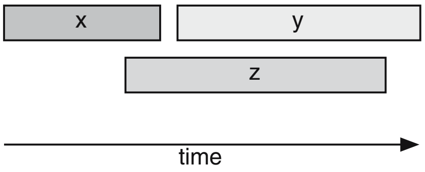


Figure 4. An incorrect inference in Setzer's closure algorithm.

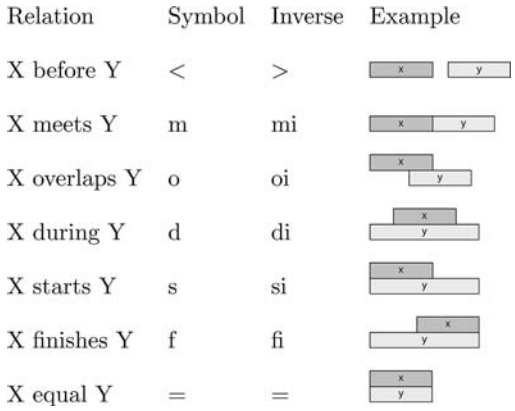
to more correctly measure inter-annotator agreement. But if temporal closure is used to achieve a complete annotation then it should not be at the price of lower precision.[3]

3. Implementing Temporal Closure

3.1. Intervals, Points, and Neighborhoods

Allen's interval calculus (Allen, 1983, 1984) has been very influential in the field of temporal reasoning. Its starting point is the acknowledgement that there are thirteen basic temporal relationships between two intervals, as depicted in Figure 5, which shows seven relations and the inverses of six of them.

The temporal relations between intervals can be maintained in a graph where the nodes are the intervals and the arcs are labeled by arbitrary disjunctions over the 13 basic relations. Allen assumes that the network always maintains complete information about how its intervals could be related. When a new temporal relation between two intervals is added, all consequences are generated by computing the transitive closure of the temporal relations. Each new fact adds a constraint about how its two intervals could be related, which may in turn introduce new constraints between other intervals through the transitivity rules governing the temporal relations.



Relation	Symbol	Inverse	Example
X before Y	<	>	
X meets Y	m	mi	
X overlaps Y	o	oi	
X during Y	d	di	
X starts Y	s	si	
X finishes Y	f	fi	
X equal Y	=	=	

Figure 5. The thirteen basic relations.

A fragment of Allen's 13× 13 transitivity table that models the transitive behavior of all relation pairs is given in Table I. The composition operator $\odot$ is used as another way to denote lookup in the transitivity table; $r_1 \odot r_2$ is the cell (r_1, r_2). If a new fact $\langle i$ during $j \rangle$ is added, and j is before k, then it is inferred from the table that i must be before k. The new constraint can be a disjunction, for instance, if the arc $\langle i, j \rangle$ is labeled $\{<\}$ and the arc $\langle j, k \rangle$ is labeled {d}, then the arc $\langle i, k \rangle$ can be constrained to the set {< o m d s}. In any case, the new fact is added to the network, possibly introducing further constraints on the relationships between other intervals.

Allen's constraint propagation algorithm is given in Figure 6. In this algorithm, R (i, j) is the new basic relation or set of basic relations just added between i and j, and N (i, j) is the existing set of basic relations between i and j. It is easy to see that the time complexity of Allen's algorithm is $O(N^3)$ where N is the number of intervals. Adding one arc to the network is linear and the number of modifications that can be made is 13 times the number of binary relations between all nodes, which is $O(N^2)$.

A problem with the constraint propagation procedure is that while it does not generate inconsistencies it does not detect all inconsistencies in its input. That is, it is sound but not complete. The algorithm never compares more than three arcs at a time and there are temporal networks where each subgraph of three arcs is consistent but where there is no consistent labeling for the whole graph (an example is given in (Allen, 1983)). The algorithm becomes exponential when complete consistency checks are incorporated.

A tractable restricted subset of the interval algebra was proposed by Marc Vilain, Henry Kautz and Peter van Beek (Vilain *et al.*, 1990). They used relations on points to restrict the $2^{13} = 8192$ different labels that interval algebra allows. Each interval can be represented as a pair of points where one precedes the other. For example, the interval X could be rewritten as $x_1 - x_2$, where x_1 is the begin point, x_2 is the end point and $x_1 < x_2$. All basic relations can also be rewritten using precedence and equality relations on begin and end points, as shown for a few of the basic relations in Table II.

Table I. The transitive behavior of basic relations

	<	>	d	di
<	<	all	< o m d s	<
>	all	>	> oi mi d f	>
d	<	>	d	all
di	< o m di fi	>oi di mi si	o oi d s f di si fi =	di

```
Add R(i,j):
 add R(i,j) to ToDo
 while notEmpty ToDo do
     get next R(i,j) from ToDo
     N(i,j) := R(i,j)
     foreach node k do
         R(k,j) := N(k,j) ∩ Constraints(N(k,i),R(i,j))
         if R(k,j) ⊂ N(k,j) then
            add R(k,j) to ToDo
         R(i,k) := N(i,k) ∩ Constraints(R(i,j),N(j,k))
         if R(i,k) ⊂ N(i,k) then
            add R(i,k) to ToDo

Constraints(R1,R2):
 Result := ∅
 foreach r1 in R1 do
     foreach r2 in R2 do
         Result := Result ∪ r1 ⊙ r2
 return Result
```

Figure 6. Allen's constraint propagation algorithm.

Table II. Mapping interval relations to point relations

X before Y	$x_2 < y_1$
X starts Y	$x_1 = y_1 \wedge x_2 < y_2$
X during Y	$x_1 > y_1 \wedge x_2 < y_2$
X overlap Y	$x_1 < y_1 \wedge x_2 > y_1 \wedge x_2 < y_2$

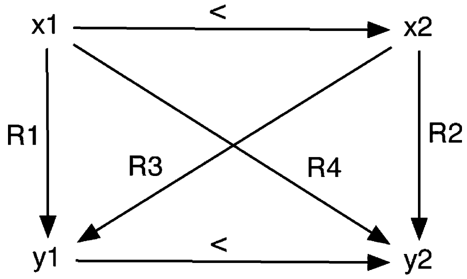


Figure 7. Decomposing an interval relation.

The point algebra is defined by the four point relations between the beginning and end of two intervals. Any basic relation between intervals can be represented by defining the four relations RI through R4 as shown in Figure 7. The labels R1 through R4 on the point relations are taken from the set $\{< = >\}$, so instead of thirteen basic relations there are now only three. An interesting table emerges when all thirteen basic relations from the interval algebra are ordered according to the point relations assigned to the four relations above, see Figure 8.

A convex relation is a relation between the four points where the following labels are allowed: $\{<\}$, $\{=\}$, $\{>\}$, $\{< =\}$, $\{> =\}$, and $\{< = >\}$. Convex relations map to disjunctions of interval relations, but not all disjunctions of interval relations can be expressed by a convex relation. For expample, $x_2 < = \ y_1$ maps to the disjunction $\{< m\}$, but there is no convex relation that covers the disjunction $\{< si >\}$, as can easily be verified by inspecting Figure 8. Ordering and restricting the unlimited disjunctions of Allen this way gives us a set of 82 convex relations. Schilder (1997) ordered these in a hierarchy based on the subset relation.

This restricted point algebra of convex relations can be mapped to a subset of Allen's interval algebra by simply translating the point relation assignments to disjunctions of basic relations between intervals, using Figure 8. This interval algebra, with 82 rather than 2^{13} possible labels, has the property that detecting inconsistencies is now tractable. Indeed, (Vilain *et al.*, 1990) proved that Allen's constraint propagation algorithm is sound and complete if the reduced set of labels is adopted.

Christian Freksa (Freksa, 1992) proposes another subset of the interval algebra. He argues that Allen does not introduce a good mechanism for

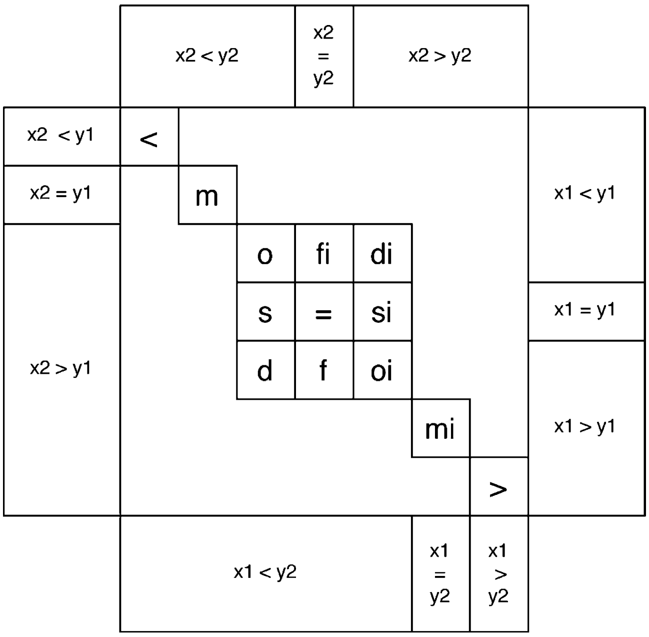


Figure 8. Interval relations and point relations.

coarse temporal information because his disjunctions of basic relations are not at all restricted. In addition, Allen's representation and algorithm become more complex when less information is available on the arcs. Coarse temporal information is needed to properly describe indefinite temporal information in discourse, as exemplified in example (6) below.[4]

(6) "Mary stared^{e1} at Peter. He gave^{e2} her pizza back."

Event *e*l can occur before *e*2, it can meet *e*2 or it can overlap with *e*2. Allen's scheme needs the disjunction {< m o} to capture this information and requires a loop over the transitivity table to compute how constraints propagate through the network. To more concisely capture this kind of coarse temporal knowledge, Freksa introduced the notion of conceptual neighborhood. Two relations between intervals are conceptual neighbors if they can be directly transformed into one another by deforming the intervals (that is, shortening or lengthening), as in Figure 9.

The before and meet relations are conceptual neighbors, but before and overlap are not because the transformation is indirect via the relation meet. The 13 basic temporal relations can be ordered in a network according to

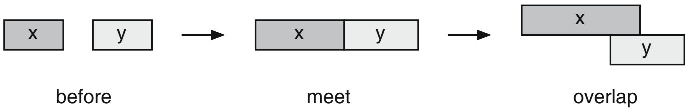


Figure 9. Deforming intervals.

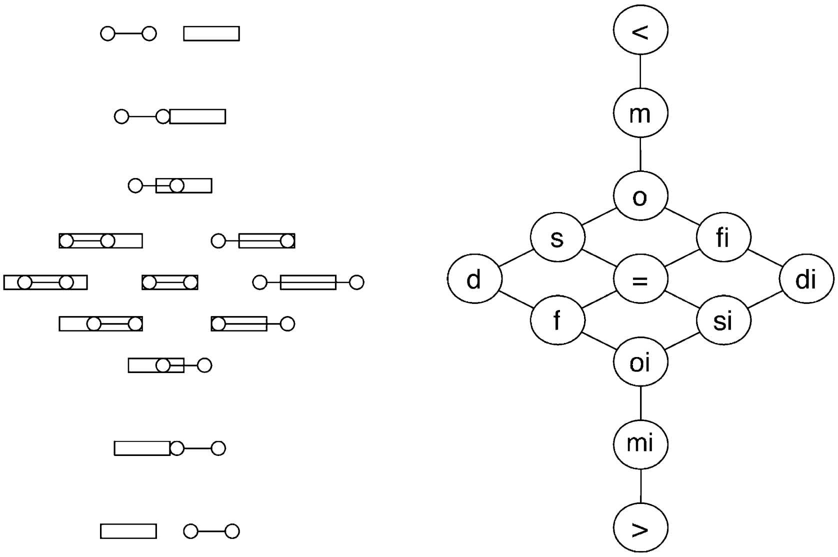


Figure 10. The basic relations in their neighborhood.

Table III. Two neighborhoods

Label	Mnemonic	Allen	Point relations
tt	tail to tail with	fi = f	$x_2 = y_2$
oc	older contemporary of	o fi di	$x_1 < y_1 \wedge x_2 > y_1$

their conceptual neighborhood. The result is the graph in Figure 10 that looks a lot like the table with point relation assignments in Figure 8.

The lines between the relations represent the direct one-step transformations of the intervals. A conceptual neighborhood is defined as a set of relations that are path-connected through conceptual neighbor relations. For example, the set {< m o fi =} is a conceptual neighborhood but {< o} is not. Note that all convex relations are conceptual neighborhoods but that the reverse is not true. Figure 8, by the way, presents another way of defining convex relations. A convex relation *Rel* has a top element r_1 and a bottom element r_2 such that $Rel = \{r | r_i \subseteq r \subseteq r_2\}$. So {oi =} is not a convex relation because by the definition above f and si should also be included.

Freksa continues by identifying ten conceptual neighborhoods that are the basis for coarse temporal reasoning. He selected the neighborhoods in such a way that finer relations (the Allen relations) can be expressed as conjunctions of the coarse relations. Two examples of these neighborhoods are shown in Table III. Freksa then creates the transitivity table for these ten relations and shows that using this table generates the same inferences as Allen's transitivity table, the difference only being that Allen's algorithm creates disjunctions when reasoning over coarse information whereas

```
Add R(i,j):
 add R(i,j) to ToDo
 while notEmpty ToDo do
     get next R(i,j) from ToDo
     N(i,j) := R(i,j)
     foreach node k do
         R(k,j) := N(k,j) $\cap \odot$(N(k,i),R(i,j))
         if R(k,j) $\subset$ N(k,j) then
            add R(k,j) to ToDo
         R(i,k) := N(i,k) $\cap \odot$(R(i,j),N(j,k))
         if R(i,k) $\subset$ N(i,k) then
            add R(i,k) to ToDo
```

Figure 11. SputLink's Constraint Propagation Algorithm.

Table IV. Transitivity table for point relations

	<	=	>	?
<	<	<	?	?
=	<	=	>	?
>	?	>	>	?
?	?	?	?	?

Freksa's uses conjunctions when reasoning over fine information. Finally, he creates a 29 × 29 table that is closed under neighborhood-based reasonings; that is, composition of any two of the 29 neighborhoods results in one of the 29 neighborhoods. These 29 relations are a subset of the 82 convex relations defined by (Vilain *et al.*, 1990) and therefore the algebra inherits the tractability of the point algebra with convex relations.

3.2. Launching SputLink

SputLink is an implementation of Allen's interval algebra but it restricts the set of possible labels using insights from point algebra. Rather than using $\{<\}$, $\{=\}$, $\{>\}$, $\{<\}$, $\{> =\}$, and $\{< = >\}$ as allowed labels for point relations, SputLink only uses $\{<\}$, $\{=\}$, $\{>\}$, and $\{< = >\}$. As a result, the set of possible labels is limited to 29 elements, which are the same relations that Freksa identified. These relations between intervals can be plotted in a hierarchy by using the subset relation. This hierarchy is similar to, yet smaller than the hierarchy presented in (Schilder, 1997).

The core SputLink constraint propagation algorithm is presented in Figure 11. It is very similar to Allen's algorithm in Figure 6. The main difference is that there is no Constraints procedure that loops over a 13 × 13 composition table of basic relations but a single lookup in a 29 × 29 composition table of convex relations. This table can simply be computed by applying Allen's original algorithm to all 29 × 29 combinations of the restricted set of labels. Alternatively, all combinations of interval relations can be decomposed into point relations. Assume we have three intervals, x_1 - x_2, $y_1 - y_2$ and z_1–z_2, and point relations between x_i and y_j and point relations between y_k and z_l. The algorithm in Figure 11 can be applied to this graph using the composition Table IV and the resulting point relations between points in $x_1 - x_2$ and $z_1 - z_2$ can be mapped to interval relations and put in the composition table.

3.2.1. *Intervals and Points*

Taking an interval-based approach assumes that intervals are the primitives for the purpose of temporal closure over the annotation. Allen originally

Table V. Mapping TimeML relations to basic relations

TimeML relation	Allen relation	Relations between points
A before B	<	$a_2 < b_1$
A after B	>	$a_1 < b_2$
A ibefore B	m	$a_2 = b_1$
A iafter B	mi	$b_2 = a_1$
A includes B	di	$a_1 < b_1 \wedge a_2 < b_2$
A is_included B	d	$a_1 > b_1 \wedge a_2 < b_2$
A identity B	=	$a_1 = b_1 \wedge a_2 = b_2$
A simultaneous B		
A holds B		
A held_by B		
A begins B	s	$a_1 = b_1 \wedge a_2 < b_2$
A begun_by B	si	$a_1 = b_1 \wedge a_2 > b_2$
A ends B	f	$a_1 > b_1 \wedge a_2 = b_2$
A ended_by b	fi	$b_1 > a_1 \wedge a_2 = b_2$

claimed that even events or time expressions that look like points-in-time can in fact be treated as very short intervals and that interval-based reasoning was more efficient than point-based reasoning. Antony Galton (Galton, 1990) argued that the neglect of time instants results in a formalism that is too crude for representing facts about continuous change. To amend that, points and intervals need to be treated on equal footing.

SputLink as described above has no concept of points but the 29 × 29 composition table can easily be expanded to allow for temporal relations between points and intervals and relations between points. For example, to take care of point-interval relations we can take the square in Figure 7 with four relations between points and reduce it to a triangle with three point relations. We can then create eight convex relations between point and interval: five basic ones (before, starts, in, ends, and after), and three disjunctions ({before, starts, in}, for when the point comes before the end point of the interval, {in, ends, after}, for when the point comes after the begin point of the interval, and {before, starts, in, ends, after}, the totally underspecified relation). The 8 × 8 composition table can be filled in the same way as the 29 × 29 composition table for interval-interval relations.

3.2.2. *Intervals, Events and Times*

Note that to initialize the algorithm we need to map an annotation graph with TimeML objects into a graph with intervals and relations between intervals (for the moment ignoring the points-in-time). This amounts to reducing events to intervals. In other words, we abstract away from all properties of events (and times) and view them as time intervals only for the sake of the algorithm. All TimeML relations are mapped onto Allen relations, as given in Table V.

Note that there is no need for a mapping to the basic relations o and oi since TimeML has no overlap relation. Another thing to realize is that TimeML relations are not intended to be as precise as Allen relations. There is a certain amount of fuzziness built into some of the relations, although this fuzziness is not even close to the fuzziness of STAG's simultaneous relation. As a result, a TimeML closure engine that uses the precise relations behind the screens may introduce incorrect links in a similar way as some of the inference rules of Gaizauskas and Setzer. How often this happens is an empirical question.

Also note that it is not trivial to translate back from Allen relations to TimeML relations since there are four relations that are mapped onto the = relation: identity, simultaneous, holds, and its inverse held_by. Now, which TimeML relation should be assigned to a relation type if the closure component generates a constraint on a temporal link that includes Allen's equal relation? The closure component is separated as much as possible from the component that translates back to TimeML relation types. The closure component reduces events and time expressions (timexes) to intervals whose only characteristics are their begin and end points. The choice of the TimeML relation type depends on factors beyond the position of the events or timexes in the partial order of time points: it depends on the types of intervals that are linked.

3.3. Embedding SputLink

SputLink's closure algorithm does not run in a vacuum. It is embedded in a mixed-initiative temporal annotation environment. This section explores some of the issues about how the algorithm interacts with other components and how the algorithm can be employed to achieve consistency and (near) completeness.

There are basically two ways for the algorithm to be embedded: (i) as a separate stage in the annotation and (ii) as a process that constantly runs in the background.

In the first case, manual or automatic annotation occurs before any activity from the temporal closure component, which runs in a separate stage afterwards. This is the approach taken by (Setzer, 2001). She also introduced a user-prompting stage where the user is asked to fill in a relation type for a link that has none. This is followed by another application of the closure component. The cycle continues until all event/timex pairs are visited. Note that the first time that closure runs, it may

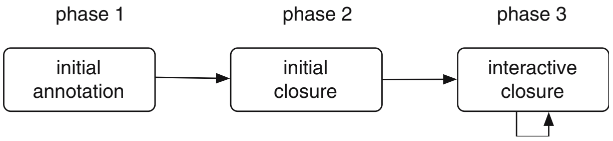


Figure 12. The three phases of annotation.

discover inconsistencies. So we have three stages of annotation: (i) a phase of TLINK-annotation by the annotator, (ii) a phase of initial temporal closure, and (iii) a phase of interactive closure centered around a user-prompting and closure loop. This three-phase approach is presented graphically in Figure 12.

In the second case, temporal closure runs each time a temporal relation gets added or further constrained. This has the advantage that the annotation is guaranteed to be consistent at any time, but it may not always be possible to use this mode, for example, when some temporal links are generated automatically in a pre-processing stage.

3.3.1. *User-Prompting and Text-Segmented Closure*

Closure by itself does not guarantee anything near completeness. Consider the strategy where a manual annotation stage is followed by application of the closure module. This strategy was used in the creation of the TimeBank corpus (Day *et al*., 2003). TimeBank annotators typically mark up only 1% of all possible temporal links and closure ramps this up to just over 5%.

User-prompting as used by (Setzer, 2001) does guarantee completeness. Yet this kind of user-prompting requires the annotator to fill in relations between events that may be separated by large expanses of text. The solution is to constrain user-prompting using text-segmented closure. The basic idea is that we relax the requirement (or strong wish) for completeness and settle for local completeness, which is defined informally as follows:

(7) A locally complete temporal annotation of a document is an annotation where each event is linked to all events and time expressions within its local context and where each time expression is linked to all events within its local context.

This relaxed completeness does not require the annotator to fill in all the relations that the closure algorithm cannot derive axiomatically. Instead, the only relations that the annotator would be prompted for are relations between events and timexes that are adjacent in the text. A segment is defined as a sequence of *N* time objects (events or time expressions) or sentences, where sentence boundaries are defined by punctuation markers. For example, a segment could consist of three sentences. Segments overlap; for instance,

Table VI. Number of global and local links

Time objects	Segments	Global links	Local links
50	5	2500	500
100	10	10,000	1000
200	20	40,000	2000

with three-sentence segments, the first segment of a document contains the first three sentences, the second segment contains sentences two through four, and so forth.

The annotator in the prompting phase is faced with a sliding window that moves down the text. The window starts out covering just the first segment and the user is prompted for new relation types inside this window. Each time the annotator adds a relation, temporal closure computes the consequences (including the non-local ones). The cycle continues until all event and timex pairs in the first segment are specified. Then the window will slip down one sentence. A benefit of this is that the annotator always has easily available all she needs to determine the relation type, no scrolling is required.

It is interesting to compare the number of global links to the number of local links. Table VI has a few example figures that illustrate the difference between those numbers in a document.

The number of global links is bound to $O(N^2)$ whereas the number of local links is bound to $O(n^2s)$, where N is the number of time objects in a document, n the number of time objects in a segment and s the number of segments. This means that the number of local links is linear if the segment size is fixed. By extension, temporal annotation using text-segmented closure is a linear task. As mentioned before, text-segmented closure is not globally complete, only locally complete. But, as we will see later in this chapter, text-segmented closure with user-prompting easily can derive more than 90% of all possible links.

Note, by the way, the distinction between the complexity of the annotation task and the complexity of the closure algorithm. The first task is linear, whereas the second is cubed to the number of time objects. This seems like a fair division of labour.

4. SputLink and the Real World

In this section, I examine more closely the claims made about temporal closure, showing that temporal closure detects inconsistencies and that, when coupled with user-prompting, it makes a near-complete annotation feasible. More specifically, I will present data on (i) the number of links added, (ii) the increase in average link span, (iii) the number of inconsistencies detected in an annotated corpus, and (iv) the impact of closure on inter-annotator agreement. In addition, I will investigate how the user-prompting in text-segmented closure helps the annotation task. Throughout this section, some fledgeling comparisons with Setzer's closure component are offered.

The data on number of links added and average link span in Section 4.1 and the inter-annotator agreement figures in Section 4.2 are all relative to phases one and two of the annotation. The user-prompting evaluation results in Section 4.3, on the other hand, also include phase three.

4.1. Closing TimeBank

A wealth of TimeML data is available in the TimeBank corpus (Day *et al.*, 2003). TimeBank consists of 182 documents with 7962 events, 1422 timexes and 5681 TLINKS. Applying closure to TimeBank delivers solid statistics on numbers of links generated and the non-local nature of TLINKS after closure. It also provides examples of how temporal closure identifies inconsistencies. In all sections except the section on inconsistencies only a subset of the 182 TimeBank documents was used: the 32 documents that contained inconsistencies were excluded from the sample.

4.1.1. *Links Added*

A first obvious characteristic of a corpus after initial closure is that its number of TLINKS has increased. But what is always in the back of our mind is the loftier goal of a complete or near-complete annotation. This section explores how much initial closure gets us closer to that goal.

Running initial temporal closure over TimeBank more than quadruples the number of TLINKS, as shown in Table VII. The density column deserves some explanation. It is convenient to have a measure that reflects how complete an annotation is. Using recall for this purpose has proven to be a tad confusing, so here I'll use the term 'density'. The density of an annotation is the percentage of TLINKS relative to all possible TLINKS in the corpus. An annotation is complete if its density is 100%. After closure, the density of TimeBank goes up from 1.28% to 5.30% and closure ends up being responsible for almost 76% of all links.

Table VII. Links added during the first two phases of TimeBank annotation

	Links	Links/doc	Share (%)	Density (%)
Added by initial annotation	4243	28.3	24.2	1.28
Added by initial closure	3306	88.7	75.8	4.02
Total	17549	117.0	100.0	5.30

Table VIII. Density after initial closure relative to the largest subgraph

Ratio	Docs	Nodes/doc	% Derived	Density (%)
0.00–0.25	53	35.9	67.1	2.5
0.25–0.50	60	24.6	75.4	9.3
0.50–0.75	34	24.1	85.1	19.2
0.75–1.00	3	14.0	83.5	53.8

Table IX. Comparing Setzer's closure module to SputLink.

	Docs	Links per document			Density (%)
		Annotated	Derived	% Derived	
Setzer	1	14.0	25.0	64.1	20.5
SputLink 45	25	12.6	24.6	66.2	18.8

It needs to be said that the average density after initial closure hides massive variation across documents, especially amongst smaller documents. For example, the observed post-closure density for texts with less than 25 time objects ranges from 2% to 57%. The parameter responsible for this variation is the size of the largest subgraph. Suppose we have a graph with eight events and two ways of carving it up into two subgraphs: [{el e2 e3 e4}, {e5 e6 e7 e8}] and [{el e2 e3 e4 e5 e6 e7}, {e8}]. Closure can never derive a link that connects two subgraphs because there already needs be a path that connects the two events or timexes, so the number of links derived by closure is bound by the maximum number of links for the individual subgraphs. In the first case, the maximum number of links is 6 + 6 = 12, in the second case it is 21 + 0 = 21. In general, annotations with the largest subgraphs are favored to derive more links by closure because the number of links is quadratic to the number of time objects.

The size of the largest subgraph is measured as the ratio of the size of the subgraph and the total number of time objects. For example, if an annotation graph has 28 events and timexes and the largest subgraph contains 12 elements, then the ratio is 12/28 = 0.43. Table VIII shows that indeed the size of the largest subgraph has a major impact on the density after initial closure, and this impact cannot be explained by adjusting for document size. This means that an annotation strategy that maximizes the size of the largest subgraph is more likely to achieve 'higher density with fewer user-added links. This issue will be taken up again in Section 4.3.

Andrea Setzer (2001) also provides some data on percentage of links derived by closure. A comparison of Setzer's data with SputLink is given in

Table IX. For this table, a subset of TimeBank was used; only the 45 documents with sizes similar to Setzer's document (that is, between 15 and 25 time objects) were used.

It looks like there is no big difference in how many links are derived by initial closure. However, a comparison is pretty much meaningless given the very small size of Setzer's sample and the observation made previously that there is a large variation hidden in the averages.

4.1.2. *Average Link Span*

In this section, I present statistics that show that temporal closure adds non-local TLINKS to the annotation and that these links were mostly absent from the initial annotation. The annotators who marked-up TimeBank seemed to converge on similar annotation strategies, linking events to other events and timexes that were close in textual proximity. Take for example the TimeBank fragment in (8) and the TLINKS added by the annotator in (9).[5]

(8) DCT: 02 – 27 – 980802EST^{t25}

Both U.S. and British officials **filed**^{e12} objections to the court's jurisdiction in 1995^{t23}, **claiming**^{e13} Security Council **resolutions**^{e2o} **imposed**^{el4} on Libya to **force**^{e15} the suspects' extradition **overruled**^{el6} a 1971^{t24} Convention which gives Libya the right to try the men.

(9) < TLINK event = e12 relatedToTime = t23 reltype = IS_INCLUDED >
< TLINK event = e12 relatedToEvent = e13 reltype = IS_INCLUDED >
< TLINK event = e20 relatedToEvent = e12 reltype = BEFORE >
< TLINK event = e14 relatedToEvent = e20 reltype = SIMULTANEOUS >

The four TLINKS shown are all the TLINKS that the annotator added for events and timexes in the sentence above; there were no TLINKS from events in this sentence to events elsewhere in the document. The fragment exhibits two kinds of TLINKS: a local anchoring of the *filed* event to the time expression *1995* and three TLINKS that establish local orderings of events. What is interesting are the TLINKS that are not there. There are no global anchorings from events to time expressions in other sentences and there is no ordering of events with events outside the sentence. What we have here is a subgraph in the annotation with the nodes {e12 e13 e14 e20 t23}.

The average link span is the textual span between the two events or timexes that are linked; it is measured by the number of sentence boundaries that are crossed. A sentence is delimited by punctuation and may include a main clause and an embedded clause. If all TLINKS are intra-sentential, as in the example above, then the average link span is 0; if all TLINKS cross one sentence boundary, then the average link span is 1. Table X has the links spans for TimeBank.

The baseline is the average link span if all links were annotated, that is, a complete TimeBank annotation would have an average link span of just over 13. For TimeBank, the average link span after initial annotation is 2.42. The number between brackets reflects the average link span when all links to the document creation time (DCT) are taken out. For TimeBank, taking out the DCT makes a big difference: 2.42 vs. 0.88. This means that most cross-sentence links involve global anchoring to the DCT and that there is no significant global anchoring to other time expressions and no significant global ordering of events.

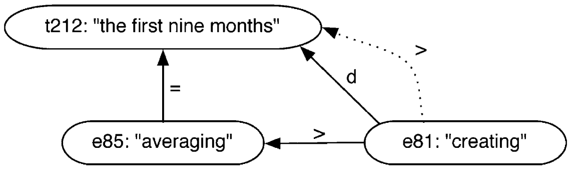
After initial closure, the link span goes up from 2.42 to 6.89. This figure reveals that initial temporal closure adds a whole group of non-local links that are systematically missed by the annotators. Average link span of non-DCT links before closure does not vary a lot across document sizes, but after closure it is higher for larger documents.

4.1.3. *Inconsistencies*

An inconsistency occurs when the relation type r_1 of a TLINK $(x\ r_1\ y)$ clashes with the relation type r_2 of a TLINK $\langle x\ r_2\ y\ \rangle$, where $r1 \neq r_2$ and $\langle x\ r_2\ y\ \rangle$ is derived by closure from $\langle x\ r_3\ q\rangle$ and $\langle q\ r_4\ y\ \rangle$. An example from TimeBank[6] is displayed in Figure 13.

Table X. Link spans for TimeBank

	Link span	
Before closure	2.42	(0.88)
After closure	6.89	
Baseline	13.35	



After accounting for a small downward revision Friday to December's figures, the economy has been **creating**^{e81} jobs at a rate of 358,000 a month for the last four months and over 381,000 over the last three months after **averaging**^{e85} 242,000 for the first nine months^{t212} of 1997.

Figure 13. Example inconsistency from initial TimeBank annotation.

One of the motivations for temporal closure is that it can catch inconsistencies like the one in Figure 13 where two TLINKS, one added by initial annotation and one added by initial closure, are incompatible. The dotted line represents a TLINK ⟨e81 < t212⟩, which was derived by closure from ⟨e81 < e85⟩ and ⟨e85 = t212⟩. Clearly, ⟨e81 < t212⟩ clashes with ⟨e81 d t212⟩, an event cannot be both during and before a certain time period. In this particular case, the annotator decided that "averaging" and "the first nine months" are simultaneous, but then continued stating that "creating" is both after one and during the other.

All TimeBank articles were checked for inconsistencies using SputLink. There were 32 documents with an inconsistency. Manual inspection showed that all inconsistencies could be reduced to three annotator-added TLINKS between three time objects. About half the inconsistencies were intra-sentential, the others crossed 1–20 sentence boundaries, with the vast majority only crossing one or two sentences. Those spanning more than two sentences almost always included the document creation time.

SputLink can obviously help resolve the inconsistencies and thereby increase the number of correct TLINKS by 32 and marginally improve TLINK-precision, which, given the 5681 TLINKS in TimeBank, would increase by 0.56%.

The number of 32 inconsistencies is small considering the size of TimeBank. But a couple of small-scale experiments with an early version of SputLink[8] have shown that more inconsistencies can pop up in the user-prompting phase, when the annotators are asked to reflect on temporal relations that are much less clear than those that they volunteer in an initial round of markup. This particular experiment showed that about four to five inconsistencies are generated during the user prompting stage of a document with about 40 time objects. This has not been quantified thoroughly though.

4.2. Inter-Annotator Agreement

Inter-annotator agreement (IAA) gives a hint as to how well-defined an annotation task is: low IAA indicates an ill-defined task. However, as noted in Section 1, a comparison of two TimeML annotations needs to take into account that two annotations can be syntactically different yet semantically the same. Temporal closure maps identical semantic annotations onto identical syntactic annotations and therefore has the potential to increase IAA scores, as was claimed previously by Katz and Arosio (2001) and Setzer (2001).

To calculate IAA, I adopt (Setzer, 2001), who, following (Hirschman *et al.*, 1998), used pairwise comparisons of precision and recall figures.[9] For each text, one annotator is taken as the key and standard precision and recall, as defined in (10), are calculated with the other annotator as the response. Then annotators swap their key and response status and P&R are calculated again. Finally, we average over the two sets of data.

Table XI. IAA scores for eight documents

Document	Inter-annotator agreement				
	Time objects	Links		Relations	
AP900822-0016	87.1	25.8	27.1	17.8	20.3
APW19980428.0729	83.1	35.9	42.9	14.4	8.6
APW19980510.0720	78.9	24.1	16.7	16.1	10.9
CNN19980104.1600.0033	68.6	8.7	14.7	0.0	2.9
NYT19980212.0025	66.7	16.1	10.9	6.4	4.4
PRI19980218.2000.0431	80.9	0.0	0.0	0.0	0.0
SJMN91-06338157	79.5	31.6	36.3	6.3	16.1
WSJ910627-0102	75.5	20.6	20.4	8.6	13.3

(10) $$\text{Precision} = \frac{\text{matches in response}}{\text{entities in response} \times 100}$$

$$\text{Recall} = \frac{\text{matches in response}}{\text{entities in key}} \times 100$$

The IAA data in this section are obtained from an experiment at Brandeis University. Eighteen documents[10] were each annotated by two people using the Alembic Workbench. The annotators had no prior exposure to Alembic and had no background in linguistics. They each received about 2 hours of training. Of the 18 documents, 10 were taken out of the sample because closure generated inconsistencies or because one of the two annotators did not add any TLINKS at all. The IAA measures before and after closure are in Table XI. The time objects column contains the IAA for the presence of an event or timex at some text extent. This is rather liberal because it is considered a match when the same extent is annotated as an event by one annotator and as a timex by the other. The links column is similar to the time objects column and reflects whether two text extents were connected by a TLINK by the two annotators. The relations column is sensitive to the relation type of the two TLINKS: a match requires the two relation types to be the same. The first percentage inside the links and relations columns reflects the IAA after the initial annotation phase, the second percentage the IAA after the initial closure.

As expected, inter-annotator agreement before closure is low, varying from 0% to 36%, with the average hovering around 20%. There is also considerable disagreement amongst annotators on the relation type. There were 104 instances where two annotators created a TLINK connecting the same text objects, but in only 50 of those the annotators added the same relation type attribute. IAA for relation type on average is about 9%, and about 48% if links that do not occur in both annotations are ignored.

What was not expected is that initial temporal closure has no obvious effect on inter-annotator agreement: it goes up for some documents but down for others. It may be unintuitive that temporal closure can actually lower inter-annotator agreement, especially given the assertion that temporal closure maps semantically identical but syntactically different annotations to syntactically identical annotations. Consider Figure 14 for an example of how closure can reduce IAA. The solid arrows are TLINKS added in the initial annotation phase. The two annotations have one TLINK in common and differ on the other: IAA is 50%. Now consider the dotted line which represents a third TLINK added after initial closure. IAA is now down to 33%. In general, closure can generate both TLINKS that raise IAA and TLINKS that push down IAA and there are many annotation configurations where the latter is more prevalent than the former.

So there is no evidence that initial temporal closure has a positive (or negative) influence on IAA. This is contrary to results from (Katz and Arosio, 2001) who reported that annotations were syntactically identical in 70% of the cases and semantically identical in 85% of the cases. It is not clear however whether they measured the same thing as I'm measuring here: (Katz and Arosio, 2001) only looked at annotation within a sentence and used a very small set of temporal relations. A comparison with (Setzer, 2001) is not possible because she does not provide IAA figures after temporal closure. It should also be noted that the Brandeis experiment was performed by a very diverse group of naive, unpaid and possibly unmotivated annotators. A larger-scale experiment with less naive annotators is sorely needed.

But even given these tentative results, we can still speculate with good cause that IAA scores will go up when two annotations both have a sufficiently high density. In that case the IAA figures in the links column have no choice but to go up, with the numbers in the relations column probably following in their wake.

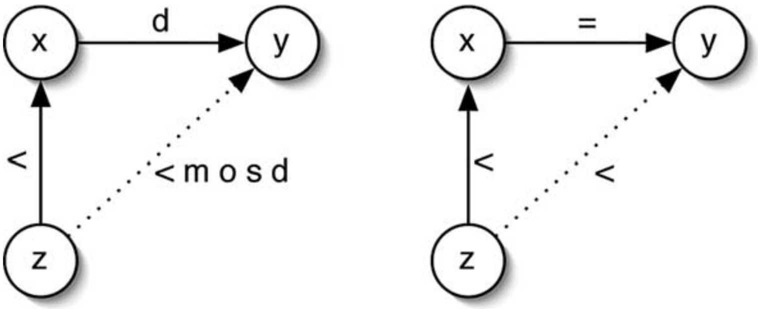


Figure 14. Example of how closure can reduce IAA.

4.3. Text-Segmented Closure

Initial closure does not provide a complete annotation: only one in 20 potential links in TimeBank are made available after initial closure. The data in this section concentrate on phase three of the annotation: the interactive closure with user-prompting. I will show that a near-complete annotation can be obtained in linear time without having to ask the annotator to supply non-local temporal relations.

Some of the data in this section are derived using a human annotator actually answering the prompting. But the vast majority of data was collected using a simulation. In this simulation, an added component to SputLink stands in for the user and provides the relation type. This relation type is generated randomly, but relative weights were used to model the observation that some relations are more frequent than others. Relation type distribution data from TimeBank provided the relative frequencies of TimeML relations. The only exception was the unknown relation, which was added to TimeML predominantly to allow for underspecification in the prompting phase. Distribution data for this relation type were gleaned from the manual prompting experiments, which indicated that about 24% of user prompts result in the addition of an unknown relation type.

The simulation was set up because there were not nearly enough data to make any significant comparative statements. Properly evaluating the claims about text-segmented closure, the optimal segment size and the optimal prompting strategy would require a large-scale annotation effort with a medium-sized group of annotators (about 5–10) annotating a range of articles using an array of different user-prompting setups. There were simply not enough resources to do this and the next best option was to set up a simulation, with some human assisted closure as a sanity check. Using a simulation is acceptable in the present case because, unlike with inter-annotator agreement or precision, the data that I'm trying to generate here are quantitative; that is, I'm interested in how many TLINKS are derived, not what TLINKS.

Consider the data in Table XII, which was filled using a simulation of user-prompting with a segment size of three sentences where the segments always included the document creation time.

As you can see, initial and interactive closure together derive a little over 94% of all TLINKS (7.6 + 86.7), which is significantly higher than the 76% share of derived TLINKS after phase 2 (cf. Table VII). This particular local prompting setup delivers a global density of just over 94%. Note that all that's needed to achieve a density of 94% is an average of about 22 + 24 = 46 user-specified TLINKS per document. The massive variation in density we saw before user-prompting is not observed here. About 95%

of all documents have densities over 80%, 44% have densities over 98% and 20% have a density of 100%.

Let's compare the figures in Table XII to the prompting statistics reported by (Setzer, 2001). She gives closure figures for six newswire articles from the New York Times from 1996. The comparison is in Table XIII.

The difference in link-generating capacity of the two closure engines (82% versus 94%) may tentatively be explained as follows:

1. SputLink has a bigger rule base. Setzer uses an incomplete set of 10 inference rules; SputLink employs a complete relation composition table with 638 entries.
2. The smaller sample size of Setzer's corpus results in a higher variation of closure percentage, potentially skewing the results. For example, one of Setzer's articles only had 67.4% of links derived by closure. Individual data for the other files were not available.
3. Density is 100% for Setzer and 94.2% for SputLink. The last 6% may have taken more prompting-cycles to complete.
4. The simulation skews the results.

The next section on optimal segment size makes it clear that explanation number three is not correct. As for the possibility that the simulation skews the results: manual experiments do indeed suggest that the simulation slightly underreports the number of prompts needed to achieve a certain density, but

Table XII. Number of TLINKS derived with simulated user-prompting

		Links/doc	Share %	Density %
Phase 1	Annotated links	22.1	2.8	2.6
Phase 2	Initial closure	60.7	7.6	7.1
Phase 3	Prompted links	23.8	3.0	2.8
	Interactive closure	695.0	86.7	81.7
Total		801.6	100.1	94.2

Table XIII. Comparison of user-prompting stages

	Setzer		SputLink	
	Links/doc	Share %	Links/doc	Share %
Annotated	18.5	4.0	22.1	2.8
Prompted	63.5	13.8	23.8	3.0
Derived	387.3	82.2	755.7	94.2
Total	469.3	100.0	801.6	100.0

Table XIV. Prompting simulation results for sentence segments

	Number of sentences in segment								
	1	2	3	4	5	6	7	8	9
With DCT									
Density in %	77.5	90.4	94.3	95.3	97.5	98.2	98.4	98.4	99.2
Prompts/doc	17.5	21.9	23.2	24.8	25.9	28.8	26.2	26.2	26.4
Without DCT									
Density in %	28.9	75.7	90.2	95.4	97.4	97.5	98.4	98.4	98.7
Prompts/doc	13.7	21.4	24.0	24.4	25.5	26.2	26.7	26.3	26.3

this is not nearly enough to explain the difference between the link-generating capacity of Setzer's closure algorithm and SputLink.

4.3.1. *Optimal Segment Size*

Two of the parameters that need to be set for text-segmented closure are the segment size unit and the number of units in a segment. There are two obvious choices for the unit: the sentence and the event or timex. The sentence is here again understood as a text extent between two punctuation markers. An additional question is whether the document creation time (which is a bit like a global time expression) should be included in the segment or not. There is always going to be a tradeoff between effort (number of prompts) and result (density), but the ultimate goal is to get a high enough density with a number of user prompts that is feasible for human annotation.

Table XIV shows how link density and number of prompts are related when the sentence is the unit of measurement. The top half of the table presents the figures when the DCT is included in each segment; for the bottom half the DCT is left out of all segments. The data in this table support two significant observations:

1. All that's needed to achieve near-completeness of TimeBank is about 25–30 user prompts per document. Not shown in the table are the numbers for individual documents. Not surprisingly, the amount of prompts is higher for larger documents. But the number of prompts was lower than the number of time objects for all documents which suggests that user prompting in text-segmented closure is linear relative to the document size.
2. Local prompting within a window of three to four sentences supports a density in the mid nineties. This is a nice result because it means that text-segmented closure does not need to degrade to large segments if high density is required. So prompting can remain essentially local. The table also shows that allowing restricted global prompting by including the

Table XV. Prompting simulation results for node segments

	Number of nodes in segment								
	1	2	3	4	5	7	10	15	20
With DCT									
Density in %	62.3	77.4	83.5	86.2	89.3	92.5	94.6	96.7	97.8
Prompts/doc	11.8	18.3	19.6	21.1	22.0	23.2	24.8	28.8	26.3
Without DCT									
Density in %	9.7	35.8	57.9	74.3	79.3	90.5	95.1	97.2	97.8
Prompts/doc	0.0	16.9	19.6	20.9	22.2	23.4	24.5	25.5	26.1

> DCT gives much better results for the smaller segments. This effect evaporates when the segments are larger than three sentences.

This picture does not change when we take segment sizes to be determined by number of nodes (events and timexes) rather than sentences, as displayed in Table XV. Using the node as the unit is in some sense more pure because it will not allow extremely long or short sentences to skew the results. That this indeed happens with sentence-sized segments becomes clear when we look at the range of measurements for individual files. For example, the average density when the segment size is three sentences (with the DCT included) is 94.3%, but this hides the fact that there are considerable variations. Sixteen of the documents had a density below 90%, six below 80%, and one had a density of 17%. If the segment size is set to ten nodes then the spread is much smaller: seven documents with density below 90%, two below 80%, and none below 65%. To frame this in more standard statistical terms we can use the standard deviation σ which indicates how tightly measurements are clustered around the mean. This is defined as

$$\sigma = \sqrt{\frac{\Sigma(x-\mu)^2}{n}} \tag{3}$$

where μ is the mean and n is the number of measurements. The standard deviation for the three-sentence segments is 12.2, for the ten-node segments it is 5.1.

I already discussed that Table XIV shows that user-prompting is linear to the document size in time objects. In the early days of SputLink there was some concern that global user prompting was potentially quadratic to the size of the document because the number of potential links is quadratic to document size. If this were true, then the number of prompts per document would go up much faster than actually happens in Tables XIV and XV. What

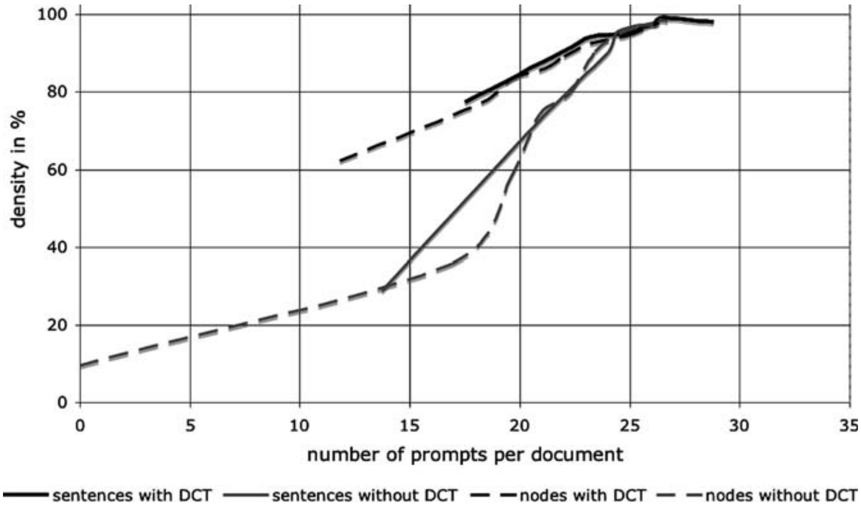


Figure 15. The relation between number of prompts and density.

we see instead is that density and number of prompts go up pretty much evenly and that the relation is quite linear. And, indeed, setting the segment size to infinity results in 100% density with only 27.1 prompts per document. The documents used for this simulation did not show any outliers, in other words, the worst-case scenario of quadratic user-prompting did not occur in any of the documents. Figure 15 illustrates the linearity of the relation between prompts and density.

So the simulation results do not display the worst-case scenario of quadratic user-prompting. This was corroborated by a couple of manual experiments where the segment size was set to infinity and where the annotator was not abused with quadratic user-prompting at all. This result can be explained by the fact that when the density of an annotation increases then the chance that there is a path between any two nodes X and Y also increases. And if there is a path between X and Y, then closure is often able to draw a TLINK directly from X to Y.

All in all, text-segmented closure has proven to be a viable approach to the interactive closure phase of the annotation effort. It provides for near complete annotations with a linear annotation effort while only prompting the user for local temporal relations, which simplifies the annotation task.

5. Conclusion

A temporal closure component can greatly enhance temporal annotation. I introduced SputLink, a temporal closure component based on James Allen's interval algebra and embedded in an annotation environment.

I have shown that SputLink increases the density of an annotation and helps reduce the locality of manual annotation. That is, closure generates temporal relations between events that are not close to each other in the document.

With closure, it is possible to ensure consistency and much easier to achieve near completeness. Densities of over 90% are possible with interactive closure and user-prompting. Experiments showed that temporal annotation is a task that is linear to the size of the document. Text-segmented closure simplifies the annotation process in the sense that the annotator will never be required to specify temporal relations between events that are not close in textual proximity. Yet text-segmented closure does not have a large negative impact on the annotation density.

The main goal in any annotation strategy should be to create a fully connected annotation graph. Running closure over a fully connected graph generates the largest number of inferences. Text-segmented closure is able to achieve that connectedness.

Notes

[1] Precision is defined as the percentage of correct answers. Recall is defined as the percentage of correct answers relative to all possible correct answers. Reported precision and recall figures range from 59% to 94%, depending on the complexity of the temporal annotation subtask.

[2] This is probably a rather pessimistic figure since it is based on a small experiment with naive annotators. Trained annotators that have memorized a solid set of annotation guidelines should choose to add the same links more often.

[3] This begs the empirical question of how often false inferences are drawn. A small number of these false hits and slightly lower precision may be acceptable as long as there would be a significant increase in recall.

[4] This example was taken from (Schilder, 1997).

[5] This fragment was taken from TimeBank article APW19980227-0476.

[6] *Source*: TimeBank document NYT19980206.0460.

[7] Precision is defined in (10) in the next section.

[8] This version did not use a complete composition table and allowed inconsistencies to be generated during the user-prompting phase.

[9] An older standard measure to measure inter-rater agreement is the Kappa coefficient, which adjusts for the number of agreements that would have occurred by chance. This coefficient though is not well suited for annotation tasks that cannot be construed as a pure classification task.

[10] These documents were not from TimeBank but they were taken from the same domain.

References

Allen J. (1983) Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11), pp. 832–843.

Allen J. (1984) Towards a General Theory of Action and Time. *Artificial Intelligence*, 23(2), pp. 123–155.

Aone C., Ramos-Santacruz M. (2000) REES: A Large-Scale Relation and Event Extraction System. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP-2000)*.

Day D., Aberdeen J., Hirschman L., Kozierok R., Robinson P., Vilain M. (1997) Mixed-Initiative Development of Language Processing Systems. In: *Fifth Conference on Applied Natural Language Processing Systems*. Washington D.C., USA, pp. 88–95.

Day D., Ferro L., Gaizauskas R., Hanks P., Lazo M., Pustejovsky J., Saurí R., See A., Setzer A., Sundheim B. (2003). The TimeBank Corpus. Corpus Linguistics.

Filatova E., Hovy E. (2001) Assigning Timestamps to Event-Clauses. In *Proceedings of A CL-EA CL 2001, Workshop for Temporal' and Spatial Information Processing*. Toulouse, France, pp. 88–95.

Freksa C. (1992) Temporal Reasoning Based on Semi-Intervals. *Artificial Intelligence*, 54(1), pp. 199–227.

Galton A. (1990) A Critical Examination of Allen's Theory of Action and Time. *Artificial Intelligence*, 42, pp. 159–188.

Hirschman L., Robinson P., Burger J., Vilain M. (1998) Automatic Coreference: The Role of Annotated Training Data. In *AAAI 1998 Spring Symposium on Applying Machine Learning to Discourse Processing*. Stanford, USA, pp, 1419–1422.

Katz G., Arosio F. (2001) The Annotation of Temporal Information in Natural Language Sentences. In: *Proceedings of ACL-EACL 2001, Workshop for Temporal and Spatial Information Processing*, Toulouse, France, pp. 104–111.

Mani I,, Schiffman B., Zhang J. (2003) Inferring Temporal Ordering of Events in News. In *Proceedings of the Human Language Technology Conference (HLT-NAACL'03)*.

Mani I., Wilson G. (2000) Robust Temporal Processing of News. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL2000)*, New Brunswick, New Jersey, pp. 69–76.

Pustejovsky J., Castaño J., Ingria R., Saurí R., Gaizauskas R., Setzer A., Katz G. (2003) TimeML: Robust Specification of Event and Temporal Expressions in Text. In *IWCS 5 Fifth International Workshop on Computational Semantics*.

Schilder F. (1997) *Temporal Relations in English and German Narrative Discourse*. University of Edinburgh, Edinburgh, UK, Ph.D. thesis.

Schilder F., Habel C. (2001) From Temporal Expressions To Temporal Information: Semantic Tagging Of News Messages. In *Proceedings of ACL-EACL 2001, Workshop for Temporal and Spatial Information Processing*. Toulose, France, pp. 65–72.

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

Setzer A., Gaizauskas R. (2001) A Pilot Study on Annotating Temporal Relations in Text. In *ACL 2001, Workshop on Temporal and Spatial Information Processing*.

Vilain M., Kautz H., Beek P.van (1990) Constraint propagation algorithms: A revised report. In Weld. D.S., Kleer. J.de (eds.), *Qualitative Reasoning about Physical Systems* San Mateo CaliforniaMorgan Kaufman, pp. 373–381.