Temporal Evaluation

Naushad UzZaman and James F. Allen
Computer Science Department
University of Rochester
Rochester, NY, USA
{naushad, james}@cs.rochester.edu

Abstract

In this paper we propose a new method for evaluating systems that extract temporal information from text. It uses temporal closure[1] to reward relations that are equivalent but distinct. Our metric measures the overall performance of systems with a single score, making comparison between different systems straightforward. Our approach is easy to implement, intuitive, accurate, scalable and computationally inexpensive.

1 Introduction

The recent emergence of language processing applications like question answering, information extraction, and document summarization has motivated the need for temporally-aware systems. This, along with the availability of the temporal annotation scheme TimeML (Pustejovsky et al., 2003), a temporally annotated corpus, TimeBank (Pustejovsky et al., 2003) and the temporal evaluation challenges TempEval-1 (Verhagen et al., 2007) and TempEval-2 (Pustejovsky and Verhagen, 2010), has led to an explosion of research on temporal information processing (TIP).

Prior evaluation methods (TempEval-1, 2) for different TIP subtasks have borrowed precision and recall measures from the information retrieval community. This has two problems: First, systems express temporal relations in different, yet equivalent, ways. Consider a scenario where the reference annotation contains $e_1<e_2$ and $e_2<e_3$ and the system identifies the relation $e_1<e_3$. The traditional evaluation metric will fail to identify $e_1<e_3$ as a correct relation, which is a logical consequence of the reference annotation. Second, traditional evaluations tell us how well a system performs in a particular task, but not the overall performance. For example, in TempEval-2 there were 6 subtasks (event extraction, temporal expression extraction and 4 subtasks on identifying temporal relations). Thus, different systems perform best is different subtasks, but we can't compare overall performance of systems.

We use temporal closure to identify equivalent temporal relations and produce a single score that measures the temporal awareness of each system. We use Timegraph (Miller and Schubert, 1990) for computing temporal closure, which makes our system scalable and computationally inexpensive.

2 Related Work

To calculate the inter-annotator agreement between annotators in the temporal annotation task, some researchers have used semantic matching to reward distinct but equivalent temporal relations. Such techniques can equally well be applied to system evaluation.

Setzer et al. (2003) use temporal closure to reward equivalent but distinct relations. Consider the example in Figure 1 (due to Tannier and Muller, 2008). Consider graph K as the reference annotation graph, and S_1, S_2 and S_3 as outputs of different systems. The bold edges are the extracted relations and the dotted edges are derived. The traditional matching approach will fail to verify B<D is a correct relation in S_2, since there is no explicit edge between B and D in reference annotation (K). But a metric using temporal closure would create all implicit edges and be able to reward B<D edge in S_2.

[1] Temporal closure is a reasoning mechanism that derives new implied temporal relations, i.e. makes implicit temporal relations explicit. For example, if we know A *before* B, B *before* C, then using temporal closure we can derive A *before* C. Allen (1983) demonstrates the closure table for 13 Allen interval relations.

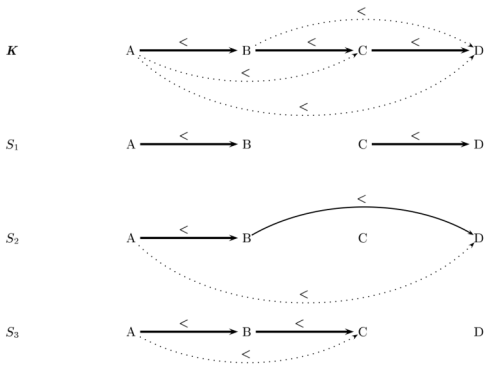

Figure 1: Examples of temporal graphs and relations

Setzer et al.'s approach works for this particular case, but as pointed by Tannier and Muller (2008), it gives the same importance to all relations, whereas some relations are not as crucial as others. For example, with K again as the reference annotation, S_2 and S_3 both identify two correct relations, so both should have a 100% precision, but in terms of recall, S_3 identified 2 explicit relations and S_2 identified one explicit and one implicit relation. With Setzer at al.'s technique, both S_2 and S_3 will get the same score, which is not accurate. Tannier and Muller handle this problem by finding the core[2] relations. For recall, they consider the reference core relations found in the system core relations and for precision they consider the system core relations found in the reference core relations. They noted that core relations do not contain all information provided by closed graphs. Hence their measure is only an approximation of what should be assessed. Consider the previous example again. If we are evaluating graph S_2, they will fail to verify that B<D is a correct edge.

We have shown that both of these existing evaluation mechanism reward relations based on semantic matching, but still fail in specific cases.

3 Temporal Evaluation

We also use temporal closure to reward equivalent but distinct relations. However, we do not compare against the temporal closure of reference annotation and system output, like Setzer et al., but we use the temporal closure to verify if a temporal relation can be derived or not. Our precision and recall is defined as:

Precision = (# of system temporal relations that can be verified from reference annotation temporal closure graph / # of temporal relations in system output)

Recall = (# of reference annotation temporal relations that can be verified from system output's temporal closure graph / # of temporal relations in reference annotation)

The harmonic mean of precision and recall, i.e. fscore, will give an evaluation of the temporal awareness of the system.

As an example, consider again the examples in Figure 1, with K as reference annotation. S_1 and S_3 clearly have 100% precision, and S_2 also gets 100% precision, since the B<D edge can be verified through the temporal closure graph of K. Note, our recall measure doesn't reward the B<D edge of S_2, but it is counted for precision. S_1 and S_3 both get a recall of 2/3, since 2 edges can be verified in the reference temporal closure graph. This scheme is similar to the MUC-6 scoring for coreference (Vilain et al., 1995). Their scoring estimated the minimal number of missing links necessary to complete co-reference chain in order to make it match the human annotation. Here in both S_1 and S_3, we are missing one edge to match with the reference annotation; hence 2/3 is the appropriate score. Precision, recall and fscore for all these system output are shown in Table 1.

System	Precision	Recall	Fscore
S_1	2/2=1	2/3=0.66	0.8
S_2	2/2=1	1/3=0.33	0.5
S_3	2/2=1	2/3=0.66	0.8

Table 1: Precision, recall and fscore for systems in Figure 1 according to our evaluation metric

4 Implementation

Our proposed approach is easy to implement with an existing temporal closure implementation. We preferred Timegraph (Miller and Schubert, 1990) over Allen's interval closure algorithm (Allen, 1983) because Timegraph has been shown to be more scalable[3] to larger problems (Yampratoom

[2] For relation $R_{A, B}$ between A and B, derivations are $R_{A, C}$, $R_{B, C}$, $R_{A, D}$, $R_{B, D}$. If the intersection of all these derived relations equals $R_{A, B}$, it means that $R_{A, B}$ is not a core relation, since it can be obtained by composing some other relations. Otherwise, the relation is a core, since removing it tends to loss of information.

[3] Allen's temporal closure takes $O(n^2)$ space for *n* intervals, whereas Timegraph takes O(n+e) space, where *n* is the number of time points[3] and *e* is the number of relations between them. In terms of closure computation, without

and Allen, 1993). Furthermore, the additional expressive power of interval disjunction in Allen (1983) does not appear to play a significant role in temporal extractions from text.

A Timegraph G = (T, E) is an acyclic directed graph in which T is the set of vertices (nodes) and E is the set of edges (links). It is partitioned into chains, which are defined as sets of points in a linear order. Links between points in the same chain are in-chain links and links between points in different chains are cross-chain links. Each point has a numeric pseudo-time, which is arbitrary except that it maintains the ordering relationship between the points on the same chain. Chain and pseudo-time information are calculated when the point is first entered into the Timegraph. Determining relationship between any two points in the same chain can be done in constant time simply by comparing the pseudo-times, rather than following the in-chain links. On the other hand, relationship between points in different chains can be found with a search in cross-chain links, which is dependent on the number of edges (i.e. number of chains and number of cross-chain links). A metagraph keeps track of the cross-chain links effectively by maintaining a metanode for each chain, and using a cross-chain links between metanodes. More details about Timegraph can be found in Miller and Schubert (1990) and Taugher (1983).

Timegraph only supports simple point relations (<, =, ≤), but we need to evaluate systems based on TimeML, which is based on interval algebra. However, single (i.e., non-disjunctive) interval relations can be easily converted to point relations[4].

For efficiency, we want to minimize the number of chains constructed by Timegraph, since with more chains our search in Timegraph will take more time. If we arbitrarily choose TimeML TLINKs (temporal links) and add them we will create some extra chains. To avoid this, we start with a node and traverse through its neighbors in a systematic fashion trying to add in chain order. This approach decreases number of nodes+edges by 2.3% in complete TimeBank corpus, which eventually affects searching in Timegraph.

Next addition is to optimize Timegraph construction. For each relation we have to make sure all constraints are met. The easiest and best way to approach this is to consider all relations together. For example, for interval relation X *includes* Y, the point relation constraints are: x1<y1, x1<y2, x2>y1, x2>y2, x1<x2 and y1<y2. We want to consider all constraints together as, x1 < y1 < y2 < x2 and add all together in the Timegraph. In Table 2, we show TimeML relations and equivalent Allen's relation[5], then equivalent representation in point algebra and finally point algebra represented as a chain, which makes adding relations in Timegraph much easier with fewer chains. These additions make Timegraph more effective for TimeML corpus.

TimeML relations	Allen relations	Equivalent in Point Algebra	Point Algebra represented as a chain
Before	Before	x1<y1, x1<y2, x2<y1, x2<y2	x1 < x2 < y1 < y2
After	After	x1>y1, x1>y2, x2>y1, x2>y2	y1 < y2 < x1 < x2
IBefore	Meet	x1<y1, x1<y2, x2=y1, x2<y2	x1 < x2 = y1 < y2
IAfter	MetBy	x1>y1, x1=y2, x2>y1, x2>y2	y1 < y2 = x1 < x2
Begins	Start	x1=y1, x1<y2, x2>y1, x2<y2	x1 = y1 < x2 < y2
BegunBy	StartedBy	x1=y1, x1<y2, x2>y1, x2>y2	x1 = y1 < y2 < x2
Ends	Finish	x1>y1, x1<y2, x2>y1, x2=y2	y1 < x1 < x2 = y2
EndedBy	FinishedBy	x1<y1, x1<y2, x2>y1, x2=y2	x1 < y1 < y2 = x2
IsIncluded, During	During	x1>y1, x1<y2, x2>y1, x2<y2	y1 < x1 < x2 < y2
Includes	Contains	x1<y1, x1<y2, x2>y1, x2>y2	x1 < y1 < y2 < x2
Identity & Simultaneous (=)	Equality	x1=y1, x1<y2, x2>y1, x2=y2	x1 = y1 < x2 = y2

Table 2: Interval algebra and equivalent point algebra

disjunction Allen's algorithm computes in $O(n^2)$, whereas Timegraph takes O(n+e) time, *n* and *e* are same as before.

[4] Interval relation between two intervals X and Y is represented with points x1, x2, y1 and y2, where x1 and y1 are start points and x2 and y2 are end points of X and Y. Temporal relations between interval X and Y is represented with point relation between x1,y1; x1,y2; x2,y1 and x2,y2.

[5] We couldn't find equivalent of Overlaps and OverlappedBy from Allen's interval algebra in TimeML relations.

5 Evaluation

Our proposed evaluation metric has some very good properties, which makes it very suitable as a standard metric. This section presents a few empirical tests to show the usefulness of our metric.

Our precision and recall goes with the same spirit with traditional precision and recall, as a result, performance decreases with the decrease of information. Specifically,

i. if we remove relations from the reference annotation and then compare that against the full reference annotation, then recall decreases linearly. Shown in Figure 2.

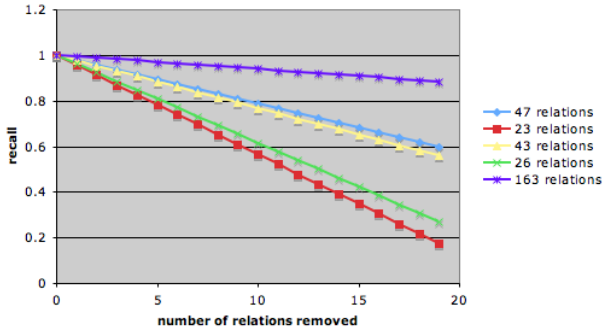

Figure 2: For 5 TimeBank documents, the graph shows performance drops linearly in recall by removing temporal relations one by one.

ii. if we introduce noise by adding new relations, then precision decreases linearly (Figure 3).

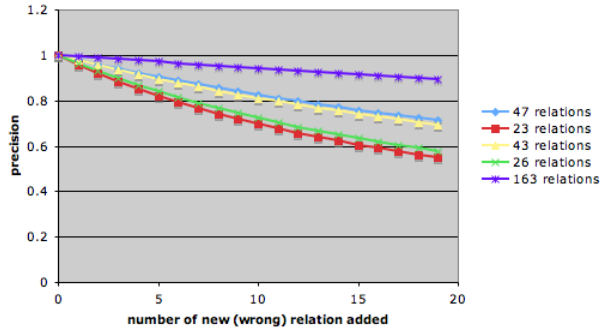

Figure 3: For 5 TimeBank documents, the graph shows performance drops linearly in precision by adding new (wrong) temporal relations one by one.

iii. if we introduce noise by changing existing relations then fscore decreases linearly (Figure 4).

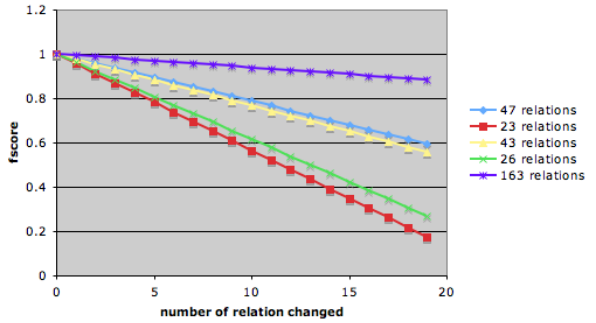

Figure 4: For 5 TimeBank documents, the graph shows performance drops linearly in fscore by changing temporal relations one by one.

iv. if we remove temporal entities (such as events or temporal expressions), performance decreases more for entities that are temporally related to more entities. This means, if the system fails to extract important temporal entities then the performance will decrease more (Figure 5).

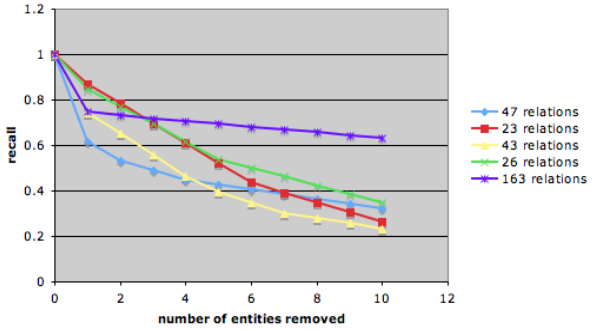

Figure 5: For 5 TimeBank documents, performance drop in recall by removing temporal entities.

Temporal entities related with a maximum number of entities are removed first. It is evident from the graph that performance decreased more for removing important entities (first few entities).

These properties explain that our final fscore captures how well a system extracts events, temporal expressions and temporal relations. Therefore this single score captures all the scores of six subtasks in TempEval-2, making it very convenient and straightforward to compare different systems.

Our implementation using Timegraph is also scalable. We ran our Timegraph construction algorithm on the complete TimeBank corpus and found that Timegraph construction time increases linearly with the increase of number of nodes and edges (= # of cross-chain links and # of chains) (Figure 6).

The largest document, with 235 temporal relations (around 900 nodes+edges in Timegraph)

only takes 0.22 seconds in a laptop computer with 4GB RAM and 2.26 GHz Core 2 Duo processor.

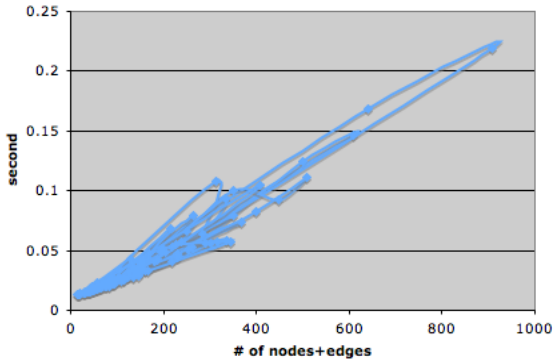

Figure 6: Number of nodes+edges (# of cross-chain links + # of chains) against time (in seconds) for Timegraph construction of all TimeBank documents.

We also confirmed that the number of nodes + edges in Timegraph also increases linearly with number of temporal relations in TimeBank documents. , i.e. our Timegraph construction time correlates with the # of relations in TimeBank documents (Figure 7).

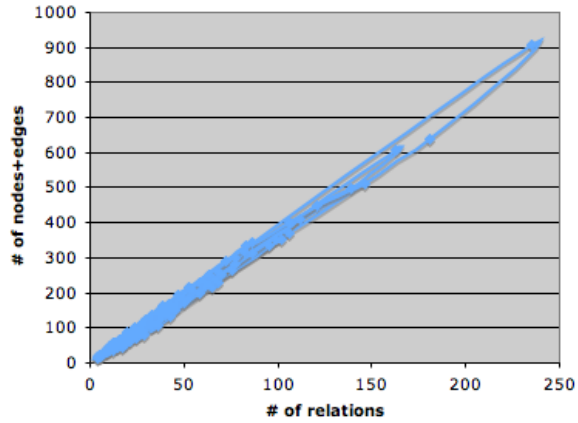

Figure 7: Number of temporal relations in all TimeBank documents against the number of nodes and edges in Timegraph of those documents

Searching in Timegraph, which we need for temporal evaluation, also depends on number of nodes and edges, hence number of TimeBank relations. We ran a temporal evaluation on TimeBank corpus using the same document as system output. The operation included creating two Timegraphs and searching in the Timegraph. As expected, the searching time also increases linearly against the number of relations and is computationally inexpensive (Figure 8).

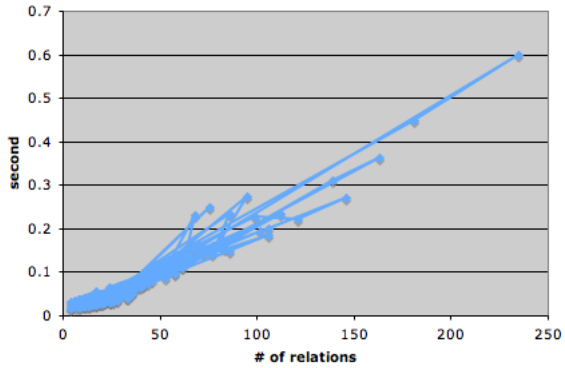

Figure 8: Number of relation against time (in seconds) for all documents of TimeBank corpus.

6 Conclusion

We proposed a temporal evaluation that considers semantically similar but distinct temporal relations and consequently gives a single score, which could be used for identifying the temporal awareness of a system. Our approach is easy to implement, intuitive and accurate. We implemented it using Timegraph for handling temporal closure in TimeML derived corpora, which makes our implementation scalable and computationally inexpensive.

References

James F. Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* **26**, 832-843.

S. Miller and L. Schubert. 1990. Time revisited. *Computational Intelligence* **6**, 108-118.

J. Pustejovsky, P. Hanks, R. Sauri, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro and M. Lazo. 2003. The TIMEBANK corpus. Proceedings of the Corpus Linguistics, 647–656.

James Pustejovsky, Jos M. Castao, Robert Ingria, Roser Sauri, Robert J. Gaizauskas, Andrea Setzer, Graham Katz and Dragomir R. Radev. 2003. TimeML: Robust Specication of Event and Temporal Expressions in Text. . Proceedings of the New Directions in Question Answering.

James Pustejovsky and Marc Verhagen. 2010. SemEval-2010 task 13: evaluating events, time expressions, and temporal relations (TempEval-2). Proceedings of the Workshop on Semantic

Evaluations: Recent Achievements and Future Directions.

A Setzer, R Gaizauskas and M Hepple. 2003. Using semantic inferences for temporal annotation comparison. Proceedings of the Fourth International Workshop on Inference in Computational Semantics (ICOS-4), 25-26.

X Tannier and P Muller. 2008. Evaluation Metrics for Automatic Temporal Annotation of Texts. Proceedings of the Proceedings of the Sixth International Language Resources and Evaluation (LREC'08).

J. Taugher. 1983. An efficient representation for time information. *Department of Computer Science*. Edmonton, Canada: University of Alberta.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz and James Pustejovsky. 2007. SemEval-2007 Task 15: TempEval Temporal Relation Identification. Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007).

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. Proceedings of the MUC6 '95: Proceedings of the 6th conference on Message understanding.

Ed Yampratoom and James F. Allen. 1993. Performance of Temporal Reasoning Systems. *TRAINS Technical Note 93-1*. Rochester, NY: University of Rochester.