2012 19th International Symposium on Temporal Representation and Reasoning

Merging Temporal Annotations

Hector Llorens♣
University of Alicante
Alicante, Spain
hllorens@dlsi.ua.es

Naushad UzZaman♣
University of Rochester
Rochester, New York, USA
naushad@cs.rochester.edu

James Allen
University of Rochester
Rochester, New York, USA
james@cs.rochester.edu

***Abstract*—In corpus linguistics obtaining high-quality semantically-annotated corpora is a fundamental goal. Various annotations of the same text can be obtained from automated systems, human annotators, or a combination of both. Obtaining, by manual means, a merged annotation from these, which improves the correctness of each individual annotation, is costly. We present automatic algorithms specifically for merging temporal annotations. These have been evaluated merging the annotations of three state-of-the-art systems on the gold standard corpora and the correctness of the merged annotation improved over that of individual annotations and baseline merging algorithms.**

***Keywords*-Temporal Information Processing; Temporal Reasoning; Corpus Linguistics**

I. Introduction

Recently, there has been significant research on automatically extracting temporal information. This task involves extracting events, temporal expressions and temporal relations from natural language text. Much research in this area is focused around TempEval competitions [1], [2]. Extracted temporal information enables automated temporal reasoning, question answering and deep language understanding.

In this work, we explore whether or not the correctness of temporal annotations can be improved by merging multiple annotations of the same text using automated algorithms. Our hypothesis is that merging annotations may improve over the performance of individual automated systems in various ways:

- Building a high recall system: If we take different systems, some can fail to extract some elements and others can have better coverage for those particular elements. By merging, the missing elements can be covered increasing the recall.
- Building a high precision system: By taking votes from multiple systems, we can ignore some low-confidence information.
- Building an overall better system (balanced precision and recall): Increasing the recall by combining annotations will decrease the precision and increasing the precision by ignoring low-confidence information will decrease the recall. However, a balanced combination may help to improve performance in general.

The underlying difficulty of merging annotations is that it cannot be ensured that the result will be better than the individual annotations. When merging annotations it is unknown whether or not an annotated element is correct. It is crucial therefore that the merging algorithms use some heuristics to merge different elements.

In this work, we propose algorithms to merge multiple temporal annotations, which consider the system/annotator performance and the temporal annotation particularities (e.g., temporal relation consistency). These algorithms are evaluated to demonstrate the hypothetical benefits introduced above.

The paper is structured as follows. First, we describe different merging techniques employed in various natural language processing (NLP) applications. Then, we introduce characteristics of temporal annotation. Next, we present our merging algorithms. Finally, we report and discuss our experimental results and draw the conclusions.

II. Related work

In NLP, a range of merging techniques have been employed for improving different tasks such as part-of-speech tagging or syntactic parsing.

Sjobergh [3] and Brill & Wu [4] combined different part-of-speech taggers. Reidel et al. [5] combined bio-molecular event extraction systems. Swift et al. [6] combined parsing output from different sources.

The techniques used in these papers are:

- Simple voting: this is the most common technique [3], [4]. This is equivalent to weighted voting using same weight for all the inputs.
- Weighted voting: this normally involves giving higher weights to better systems. Sometimes the systems also output confidence score for each individual prediction. These scores for predictions have been also used as weights for voting [3].
- Stacked merging: there are two main approaches for stacked merging: (i) using the output of one system as input feature of another system [6], [5]; and (ii) taking all system outputs and training a new model with these outputs [3].

In our merging, we exploit simple and weighted voting techniques with specific heuristics to merge temporal annotations. Due to the complexity of temporal annotation, we did not explore stacked merging.

III. Temporal Annotation

Corpus-based temporal information processing has been fueled by different temporal annotation proposals including

♣The first and the second author contributed equally to this paper.

1530-1311/12 $26.00 © 2012 IEEE
DOI 10.1109/TIME.2012.14

TimeML [7], TIDES [8], STAG [9], MUC [10].

Our merging algorithms are based on the TimeML[1] scheme, which is the current standard in the task. TimeML annotates temporal expressions (or timexes) and events (hereafter referred to as entities); and their temporal relations. Example (1) illustrates a simplified TimeML annotation.

```
(1) He <EVENT eid="e1">came</EVENT> back <TIMEX3
 tid="t1">yesterday</TIMEX3>. He was <EVENT
 eid="e2">working</EVENT> in New York <TIMEX3
 tid="t2">three days ago</TIMEX3>.
 <TLINK eventID="e1" relatedToTime="t1" relType=
 "IS_INCLUDED" /> <TLINK ...
```

Temporal annotation makes temporal information explicit. For example in this case, it shows that the time of the event *came* is included in *yesterday* time-span, that *came* takes place after working, etc. Below we explain the specific attributes of each element.

An event in TimeML is the linguistic expression of something that happens or a dynamic property that is true for a period of time. Events include several event attributes like class, tense, and aspect, with class the most important.

A timex is a linguistic representation of a time point or period. The TimeML attributes for timex are type (date, time, duration or set) and value (a normalized value of the expression). Obtaining the correct value depends on the correct annotation of the words representing the timex.

A temporal relation can be established between an event and a timex, between two events, or between two timexes. Its most important attribute is the relation type, whose possible values (after, is_included, etc.) are derived from Allen's Interval Algebra [11]. A specific issue of temporal annotation is the consistency of the temporal relations and the temporal closure [12]. Ideally, all the explicitly annotated relations and those implicitly inferred by closure must be valid. E.g., if in an annotation event A is before event B and event B is before event C, it can be inferred that event A is before event C. Annotating that A is after C is, therefore, not consistent.

For merging temporal annotations, we propose algorithms that take into account the particularities and restrictions of these temporal elements.

IV. Merging Temporal Annotations

Our goal is to take a set of TimeML annotations of the same text from different sources –either automated systems, human annotators, or both– and automatically obtain a merged annotation. In particular, we aim to get a merged annotation which is better than individual annotations in terms of correctness – i.e., get the correctly annotated elements of the annotations and avoid the incorrect ones. Since it is unknown which elements of each annotation are correct, the problem is to find heuristics useful, in particular, for merging temporal annotations.

We present two approaches to this problem: bottom-up and top-down. Each approach is defined based on how the elements in the annotation are merged. Bottom-up first merges the entities (events and temporal expressions) from all the sources and then merges the temporal relations from the sources that contain those entities. Top-down merges the temporal relations first and then based on the merged temporal relations, the entities.

In both approaches, we use voting to decide which elements to keep in the merged annotation. In particular, we use weighted voting to be able to give higher weights to better sources to ensure that a priori better annotations are preferred. We only include an element in the merged annotation if its voting is above a threshold (merging threshold). The thresholds, for entities and relations, are customizable in our algorithms, i.e., these can be modified to setup different merging configurations (e.g., threshold 0% is equivalent to unifying all annotations or threshold 50% to selecting by majority).

Before running the merging algorithms we normalize the annotated elements. TimeML annotation output from different sources may have different elements (e.g., annotation A has an event which is missed in annotation B). In order to merge annotations we first need to identify which elements are the same in different annotations. We therefore developed an algorithm which gives the same id to the same elements from different sources.

Entity normalization: We consider entities to be the same when there is a partial overlapping (or inclusion) of the text between entities of two different sources, e.g. if one source identifies Sunday and another source identifies Sunday morning, we will identify both of these as same entity. In order to carry out this task, the algorithm uses the entity position in text.

Relation normalization: A temporal relation links two entities with a relation type. We give the same id to the relations from different sources that relate the same two entities (i.e., with the same id). If the entities are related in different order, e.g. one source has e1 e2 r1 and another source has e2 e1 reverse(r1), then we will make both e1 e2 r1.

Normalized TimeML annotations from different sources are the input of our merging algorithms.

A. Bottom-up merging

Merging entities: In bottom-up, entities are merged first. We consider the weighted votes from all annotations to decide if an entity is kept or not. We keep the entity if the voting is above a predefined threshold. The approach to select the entities and their main attribute values[2] is explained in Algorithm 1. In the case of multi-token entities, the algorithm also selects the extent of the entities – tokens included in the entity. Since obtaining the correct attribute value often requires considering the correct entity extent, for each entity, after selecting which attribute value to keep in the merged annotation,the algorithm checks which sources suggested that attribute value and selects the extent and other attributes by voting only from these sources.

Merging relations: After getting the merged entities from Algorithm 1, we merge, from all sources, the relations which

[1] timeml.org

[2] Main attribute is class for events and value for timex.

```
// sources = all sources for merging
// entities= all entities from all sources
// w[s] is the weight for source s

foreach entity e ∈ entities do
    foreach source s ∈ sources do
        if e ∈ s then
            entity_weight[e] += w[s]

merged_entities = {}
foreach entity e ∈ entities do
    foreach source s ∈ sources do
        if e ∈ s and entity_weight[e] > threshold
        and e ∉ merged_entities then
            value = get_attribute_value(e, attribute, s)
            attribute_value_weight[e][value] += w[s]
            merged_entities.append(e)
```

Algorithm 1: Calculating weight for entities

link those entities. In order to merge them we distinguish two matching levels: **triples** and **doubles**. We define relation triple as {entity1, entity2, relation_type} (complete match) and relation double as {entity1, entity2} (only entities match). We use triples and doubles to calculate the weighted voting as described in Algorithm 2. Doubles are useful because they represent when two entities are related, but triples are much more important because they represent how they are related. In the algorithm, to sort relations in terms of triple weights first and then double weights, given n sources, we multiply the weight of each triple by (n+1). This gives more weight to one triple than n doubles. Note that n doubles is the maximum number of doubles that can be found given n sources, since a pair of entities {e1, e2} can only have one category (e.g. {e1, e2, BEFORE}).

```
// n = number of sources
// relations = relations whose entities are contained
//             in merged entities (algorithm 1)
// get_double_from_triple(triple t) = remove relation category from t
// e.g., get_double_from_triple(e1, e2, BEFORE) = e1, e2

foreach triple t ∈ relations do
    foreach source s ∈ sources do
        if t ∈ s then
            weight_triple[t] += w[s] * (n+1)
            d = get_double_from_triple(t)
            weight_double[d] += w[s]

foreach triple t ∈ relations do
    d = get_double_from_triple(t)
    relation_weight[t] = weight_triple[t] + weight_double[d]
merged_relations = {}
foreach triple t ∈ relations do
    if relation_weight[t] > threshold and consistent(t) then
        merged_relations.append(t)
```

Algorithm 2: Calculating the weight for relations

Example (2) illustrates how the algorithm merges the relations from three sources (S1, S2, S3) with different weights.

```
(2)  w[S1] =4, w[S2]=3, w[S3] = 2
  triples[S1]: {e1, e2, BEFORE}, {e2, t1, AFTER}
  triples[S2]: {e1, e2, BEFORE}, {e2, t1, ENDS}
  triples[S3]: {e1, e2, AFTER}, {e5, e6, AFTER}
  relation_weight[{e1, e2, BEFORE}] =
      (w[S1]+ w[S2])*4 + w[S1] + w[S2] + w[S3] = 37
  relation_weight[{e2, t1, AFTER}] =
      w[S1]*4 + w[S1]+w[S2] = 23
  relation_weight[{e2, t1, ENDS}] =
      w[S2]*4 + w[S1]+ w[S2] = 19
  relation_weight[{e1, e2, AFTER}] =
      w[S3]*4 + w[S1] + w[S2] + w[S3] = 17
  relation_weight[{e5, e6, AFTER}] = w[S3]*4 + w[S3] = 10
```

With Algorithm 2, we will get a sorted list of relations, i.e. highly voted relations on top. This allows removing the less voted inconsistent ones. We assume that temporal annotations should be consistent and then also the merging output.[3]

As explained in section 3, to check temporal consistency we need to calculate the temporal closure. This is a computationally complex task. Our merging algorithms check temporal consistency using Timegraph [13], which creates a temporal structure in which temporal consistency can be efficiently checked.

The less voted relations that are inconsistent with the already added relations (more voted) are eliminated from the list. Finally, the consistent merged relations that are above the established merging threshold are included in the merged output.

B. Top-down merging

The top-down merging approach first merges the temporal relations. We apply the same algorithm described for bottom-up merging in Algorithm 2, but instead of starting with relations of merged entities, we start with all relations from all sources. For merging entities, we by-default include the entities that participate in the merged relations. For the rest of the entities, we apply the entity merging approach described in Algorithm 1.

V. Evaluation and Discussion

In this section, at first, we describe the corpora, the annotation sources, and the evaluation metrics that we used for our experiments. Finally, we present and discuss the results.

A. Corpora

We consider TimeBank 1.2 and AQUAINT[4], which are the available TimeML compliant corpora. We use both corpora together and we perform 10-fold cross validation for all the experiments, i.e., we report the mean of 10 complementary experiments using 90% training and 10% test.

B. Annotations being Merged

We take, as annotation sources, the annotations of three systems from TempEval-2, which participated in all the tasks and obtained the best scores. These are TIPSem and TIPSemB [14]; and TRIOS [15]. These systems were built for TempEval-2 task, where temporal relation identification task was not

[3]Even knowing that some gold standard annotations are erroneously inconsistent in terms of closure.

[4]http://timeml.org/site/timebank/

included, and the relation categorization task was simplified limiting the number of relation categories to 6. For the experiment presented here we need the systems to obtain complete TimeML annotations. This means identifying temporal relations and obtaining their categorization in the full set of TimeML relation categories. In order to meet these requirements, we extended the systems in two respects: (i) we added a module to identify which entities to pair for temporal relations; and (ii) we trained the relation categorization model on the full set of TimeML relations –as annotated in TimeBank and AQUAINT– instead of the reduced set used in TempEval-2.

C. Evaluation Metric

The evaluations can be divided into entity evaluation (for events and temporal expressions) and relation evaluation.

Entity Evaluation

To evaluate entity performance, we use Precision, Recall and F1 score.

$$\textbf{precision} = \frac{true\_positives}{true\_positives + false\_positives}$$

$$\textbf{recall} = \frac{true\_positives}{true\_positives + false\_negatives}$$

$$\textbf{F1} = \frac{2 * precision * recall}{precision + recall}$$

There are two main ways of calculating these scores for entities entity-based score and token-based score (used in TempEval-2). Between them, we preferred entity-based. Example (3) shows the difference between both scoring methods.

(3) Imagine a text with 3 correct timexes and 2 annotations of such timexes (1 pos, 0 neg):

timexes:	the	beginning	of	October	1999 |	2005 |	yesterday
annot1	0	1	1	1	1	1	0
annot2	0	0	0	1	1	1	1

annot1 score: 5/7 tokens 2/3 entities (relaxed match)
annot2 score: 4/7 tokens 3/3 entities (relaxed match)

The entity-based score shows the proportion of timexes recognized, while token-based focuses on the number of tokens and does not differentiate if a system is missing tokens or entire entities.

For entity attributes, we report the attribute recall. In TempEval-2, the attribute performance was reported as attribute accuracy – calculated as the matching attributes out of matching reference and system entities. If an annotation matched only one entity and gets its attribute correct, then it gets 100% accuracy; just as one annotation that has matched all the entities and attributes. This makes comparing attribute performance between annotations difficult. In order to make comparison easier, we used the attribute recall – calculated as the number of matching attributes and entities out of total reference entities. Attribute recall is equivalent to the multiplication of entity recall and attribute accuracy.

Relation Evaluation

To evaluate relations, we use the new evaluation metric presented by [16]. This metric captures the temporal awareness of an annotation in terms of precision, recall and F1 score. Temporal awareness is defined as level performance of an annotation as identifying and categorizing temporal relations, which implies the correct recognition and classification of the temporal entities involved in the relations. Unlike TempEval-2 relation score, where only categorization is evaluated for relations, this metric evaluates how well pairs of entities are identified, how well the relations are categorized, and how well the events and temporal expressions are extracted.

Considering, System$^+$ = System Closure[5] and Reference$^+$ = Reference Closure, the precision, recall and F score can be summarized with the following formulas.

$$\textbf{precision} = \frac{|System \cap Reference^+|}{System}$$

$$\textbf{recall} = \frac{|Reference \cap System^+|}{Reference}$$

$$\textbf{F1} = \frac{2 * precision * recall}{precision + recall}$$

D. Results and Discussions

The objective of this evaluation is to analyze whether or not the application of our algorithms leads to a merged temporal annotation which is better than the individual annotations used to obtain it.

We evaluate various voting configurations for merging annotated elements. Each configuration sets a voting threshold which has to be reached by an element to be kept in the merged annotation. In some experiments we use weighted voting – not all the annotation sources have the same weight. In order to assign weights to individual annotations, we measured their temporal awareness F1 score in a separated subset of the data (TempEval-2 test documents in TimeBank). When it is needed, we use their performance as weight, i.e., 32% for TIPSem, 29% for TIPSemB, and 27% for TRIOS.

As baseline configurations, we consider union and intersection. In union, all the annotations have the same weight (33%) and the threshold is 0%, i.e., only 1 vote is required and then all the elements are kept. In intersection, all the annotations have the same weight (33%) and the threshold is 99.9%, i.e., all the votes are needed to keep an element.

As proposal, we include three merging configurations. Firstly, majority (MAJ), where we give same weight to all annotations (33% – simple voting); and the threshold is set to 50%, i.e., majority is required (2 votes). Secondly, performance (PERF), where we give each annotation expected correctness (system performance) as the weight and the threshold is set to the lowest weight (27%). In this configuration, we give preference to the better annotations as regards their performance. Finally, balanced (BLNC), where we give the best annotation its performance as weight (32%) and the other annotations a lower and equal weight (27%); the threshold here is also the lowest weight (27%). In this configuration we give preference to the best annotation and for the rest we

[5]Closure graph contains all possible relations that can be inferred using closure property. For example, if we know A<B and B<C, then we can infer that A<C, i.e. the closure graph will contain A¡C.

require at least 2 votes to include their elements in the merged annotation.

In PERF and BLNC, we expect a better balance between precision and recall because these will include the majority of the elements of the better annotation and only those elements of the worst annotations that get some support (more than one vote).

Below we report on the results for bottom-up approach for event, timex, and relation merging. For each of the cases, we compare the performance of our different merging approaches against baselines, and then we compare the results of the individual annotations (systems) against PERF approach, which obtained the best average results.

EVENT: For event extraction, our PERF configuration obtained the best results, closely followed by BLNC. All our merging configurations are also better than our baselines. Performance in F1 is reported in Table 1.

Table I
MERGING PERFORMANCE FOR EVENT EXTRACTION

	union	intersection	MAJ	BLNC	PERF
F1	84.9	64.6	85.1	86.7	**86.9**

Furthermore both PERF and BLNC outperform all individual annotations that we merge. Performance in F1 is reported in Table 2.

Table II
SYSTEMS VS. PERF MERGING FOR EVENT EXTRACTION

	TIPSem	TIPSemB	TRIOS	PERF
F1	86.6	85.5	75.1	**86.9**

The mean positive difference in the 10-folds between PERF and TIPSem was 0.10 with a mean standard deviation of 0.07, which supports our hypothesis that merged annotation improves over individual annotations.

TIMEX: For timex extraction, PERF performed the best among our configurations, closely followed by BLNC. However, the union baseline obtained the best results for timex extraction task. This is due to the fact that, in general, union obtains high recall and it can get high F1 if it can maintain a high precision at the same time. For timex extraction, individual systems have high precision and then union performs very well. The F1 of the baselines and our merging configurations are reported in Table 3 and a further details for union are discussed in the next subsection in Table 8.

Table III
MERGING PERFORMANCE FOR TIMEX EXTRACTION

	union	intersection	MAJ	BLNC	PERF
F1	**91.6**	60.6	86.1	89.4	89.5

Both PERF and BLNC outperform all individual systems that we merge. Performance in F1 is reported in Table 4.

Table IV
SYSTEMS VS. PERF MERGING FOR TIMEX EXTRACTION

	TIPSem	TIPSemB	TRIOS	PERF
F1	88.3	86.1	86.0	**89.5**

The mean positive difference between PERF and TIPSem was 0.10 with a mean standard deviation of 0.08, which supports our hypothesis that merging improves also in timex annotation.

RELATIONS: For relations (temporal awareness), our BLNC configuration performed the best. MAJ performed better than PERF here, which we will explain in detail later. All our merging configurations are better than our baselines in temporal awareness. Performance in F1 is reported in Table 5.

Table V
MERGING PERFORMANCE FOR TEMPORAL AWARENESS

	union	intersection	MAJ	BLNC	PERF
F1	29.35	16.98	30.90	**31.38**	30.65

All our merging configurations outperform all the individual annotations that we merge. Performance in F1 is reported in Table 6.

Table VI
SYSTEMS VS. PERF MERGING FOR TEMP. AWARENESS

	TIPSem	TIPSemB	TRIOS	PERF
F1	29.59	28.41	24.75	**30.65**

For temporal awareness, the mean positive difference in the 10-fold between the best system (TIPSem) and PERF was 0.15 with a mean standard deviation of 0.03.

To the best of our knowledge this is the first time Temporal Awareness metric is used. The scores obtained seem low, but this new metric measures the global performance of a complex task, temporal information processing. Unlike TempEval-2 relation accuracy, which only measures accuracy on categorizing pre-identified relations in six categories, temporal awareness measures the general performance – the identification of relations and its categorization in thirteen categories. Temporal awareness score depends on the temporal relation categorization performance, which depends on temporal relation identification, which depends on extraction of events and temporal expressions. We checked the gold data against which we evaluate annotations and it is worth mentioning that the low scores are also due to the incompleteness and the inconsistencies of these corpora. This shows the complexity of the task, prevents systems to achieve high results, and justifies the low agreement between human annotators reported for TimeBank in temporal relations (0.55).

DETAILED EXPERIMENTS: We also report the detailed scores to better analyze the merging results.

EVENT: We detail the event performance of best individual system (TIPSem) and average best merging configurations (BLNC, PERF). For event attributes (class recall), both our best merging configurations outperform TIPSem. Table 7 includes event precision (P), recall (R), F1 score (F1), class accuracy (class ac) and class recall (class R).

Table VII
DETAIL EVENT PERFORMANCE FOR BEST INDIVIDUAL SYSTEM AND BEST MERGING CONFIGURATIONS

	P	R	F1	class ac.	class R
TIPSem	87.5	85.7	86.6	82.7	70.8
BLNC	87.0	86.5	86.7	82.8	71.6
PERF	85.8	88.0	86.9	82.7	72.8

TIMEX: In this case it is worth detailing the UNION baseline in addition to the best individual system (TIPSem) and best merging configurations (PERF, BLNC). In timex attribute (value recall), both our best merging configurations outperform the best individual system; and UNION performs the best as it is shown in Table 8.

Table VIII
DETAIL TIMEX PERFORMANCE FOR BEST INDIVIDUAL SYSTEM AND BEST MERGING CONFIGURATIONS

	P	R	F1	class ac.	class R
TIPSem	94.6	82.8	88.3	70.5	58.4
PERF	93.9	85.7	89.5	71.7	61.4
BLNC	94.7	84.7	89.4	72.0	61.0
UNION	91.0	92.4	91.6	70.2	64.9

Since the precision of the annotations being merged is high the key point for timex is improving recall. The best way to do so is by a union merging.

RELATIONS: Table 9 shows the detail temporal awareness performance of best individual system (TIPSem) and all the merging configurations.

Table IX
DETAIL TEMPORAL AWARENESS PERFORMANCE FOR BEST INDIVIDUAL SYSTEM AND BEST MERGING CONFIGURATIONS

	P	R	F1
TIPSem	26.46	33.62	29.59
BLNC	28.68	34.71	31.38
PERF	26.55	36.35	30.65
MAJ	35.27	27.54	30.90

Table 5 showed previously that MAJ performs the second best. But we can get the insight from Table 9 that the improvement was due to high precision. Table 9 also shows the strengths of each of our configurations. MAJ has high precision, PERF has high recall and BLNC is a good balance between precision and recall. Depending on the need, we can use the best configuration for our purpose to get the best out of our merging algorithms.

ADDITIONAL EXPERIMENTS: We did few additional experiments to make a better analysis.

Experiment 1: We checked if merging only two systems, instead of three, still improves over individual systems. We experimented the merging of two systems in PERF configuration, which is equivalent to BLNC when merging only two systems. The results are reported in Table 10.

Table X
PERFORMANCE OF MERGING TWO SYSTEMS

	TIPSemB -TRIOS	TIPSem -TIPSemB	TIPSem -TRIOS
F1	30.12	31.15	31.33

In all two-system combinations we have found that our merged annotations outperform the individual best system. TIPSem and TIPSemB are the two best individual systems. It could be expected that their combination would lead to the best result. However, TIPSem-TRIOS combination performed better. This is because TIPSem and TRIOS are the most different systems and their annotations are more complementary (better for merging), while TIPSemB is a variant of TIPSem and their annotations are more similar.

Experiment 2: We repeated all the previous experiments using top-down approach, but we did not find any notable differences for any task between top-down and bottom-up. The comparison temporal awareness is reported in Table 11.

Table XI
COMPARISON BETWEEN TOP-DOWN AND BOTTOM-UP

	MAJ	BLNC	PERF
Bottom-up	30.9	31.38	30.65
Top-down	31.23	31.18	30.52

Experiment 3: For all the experiments, we only considered the relations that are consistent, i.e. maintains the closure property. We evaluated our configurations keeping all the relations, i.e. including inconsistent relations. The performance is reported in Table 12.

Table XII
PERFORMANCE WITH/WITHOUT INCONSISTENT RELATIONS FOR BLNC

	MAJ	BLNC	PERF
without inconsistent	28.68	34.71	31.38
with inconsistent	27.09	35.25	30.60

We found a slightly better recall when including inconsistent relations since we are considering more relations, but overall performance decreases because of lower precision.

VI. CONCLUSION

In this paper, we presented algorithms to merge temporal annotations. It is hard to merge multiple annotations of the same text, since we do not know which one is good and which one is bad, and therefore the merged annotation is not ensured to be better than all the individual annotations being merged.

In the experiments, we merged annotations from three state-of-the-art systems and our merging strategies improved over the state-of-the-art performance. The improvement found is not high but merged annotations improved individual annotations in all our experiments. We consider therefore that merging is useful because we can expect a result at least slightly better than the best individual annotation.

The proposed automatic merging avoids the costly task of manually merging annotations. Furthermore, using different merging configurations, we can achieve high recall, high precision or balanced annotations.

The TimeML merging tool kit developed is available for download at (if the paper is accepted, the url will be included in the camera ready version). It is an open-source software and holds a double license (GPL and Apache2) to fit the needs of different potential users. The tool can be tuned to get a high recall, high precision, or balanced merged annotations. Depending on the need, users can choose the appropriate configuration.

In the future we plan to exploit the merging algorithms for annotating temporal corpora automatically and semi-automatically. We plan to apply merging also to human annotations in addition to automated annotations. In this case, we expect the following benefits as initial hypothesis:

- Merging different human annotations: When we have different human annotations of the same text, it is hard to merge them manually. An automated merger can be used to merge the output easily.
- Increasing human annotation coverage: Temporal annotation is a hard task even for humans[6]. Human annotators can often miss some useful information. By merging human annotation with state-of-the-art automated annotations, we can increase the coverage.
- Annotating large corpora: Finally, with merging algorithms that improve over state-of-the-art systems, we can automatically annotate a large corpus that could not be annotated by manual means. In addition, if we use high recall merging, it can be easily improved with human review mainly by removing information. In particular, we plan to apply our merging algorithms to annotate a large corpus with high recall –a silver standard corpus–, and with human review we will create the gold standard corpus almost by removing incorrect information exclusively.

ACKNOWLEDGMENTS

This work has been supported by the Spanish Government, in projects TIN-2009-13391-C04-01, MESOLAP TIN2010-14860, PROMETEO/2009/119 and ACOMP/2011/001. This work is also supported in part by NSF grant 1012205, and ONR grant N000141110417.

[6]Inter-annotator agreement:
http://timeml.org/site/timebank/documentation-1.2.html#iaa

REFERENCES

[1] M. Verhagen, R. Gaizauskas, M. Hepple, F. Schilder, G. Katz, and J. Pustejovsky, "Semeval-2007 task 15: Tempeval temporal relation identification," in *Proceedings of the 4th International Workshop on Semantic Evaluations*. Prague: ACL, 2007, pp. 75–80.

[2] M. Verhagen, R. Saurí, T. Caselli, and J. Pustejovsky, "Semeval-2010 task 13: Tempeval-2," in *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: ACL, 2010, pp. 57–62.

[3] J. Sjobergh, "Combining POS-taggers for improved accuracy on Swedish text," in *Proceedings of the NoDaLiDa*, 2003.

[4] E. Brill and J. Wu, "Classifier combination for improved lexical disambiguation," in *Proceedings of the 17th international conference on Computational linguistics*, vol. 1, 1998, pp. 191–195.

[5] S. Riedel, D. McClosky, M. Surdeanu, A. McCallum, and C. Manning, "Model Combination for Event Extraction in BioNLP 2011," in *Proceedings of the NLP in Biomedicine ACL 2011 Workshop (BioNLP 2011)*, 2011.

[6] M. Swift, J. F. Allen, and D. Gildea, "Skeletons in the parser: Using a shallow parser to improve deep parsing," in *Proceedings of COLING*, 2004.

[7] J. Pustejovsky, J. M. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, and G. Katz, "TimeML: Robust Specification of Event and Temporal Expressions in Text," in *IWCS-5*, 2003.

[8] L. Ferro, L. Gerber, I. Mani, B. Sundheim, and G. Wilson, "TIDES 2005 Standard for the Annotation of Temporal Expressions," MITRE, Tech. Rep., 2005.

[9] A. Setzer and R. Gaizauskas, "Annotating Events and Temporal Information in Newswire Texts," in *LREC 2000*, 2000, pp. 1287–1294.

[10] R. Grishman and B. Sundheim, "Message understanding conference- 6: A brief history," in *COLING*, 1996, pp. 466–471.

[11] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[12] M. Verhagen and J. Pustejovsky, "Temporal processing with the tarsqi toolkit," in *Proceedings of the ACL: Demonstration Papers*, 2008, pp. 189–192.

[13] S. Miller and L. Schubert, "Time revisited," *Computational Intelligence*, vol. 6, no. 1, pp. 108–118, 1990.

[14] H. Llorens, E. Saquete, and B. Navarro-Colorado, "TIPSem (English and Spanish): Evaluating CRFs and Semantic Roles in TempEval-2," in *Proceedings of the 5th International Workshop on Semantic Evaluation*. ACL, 2010, pp. 284–291.

[15] N. UzZaman and J. F. Allen, "TRIPS and TRIOS system for TempEval-2: Extracting temporal information from text," in *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: ACL, 2010, pp. 276–283.

[16] N. UzZaman and J. Allen, "Temporal evaluation," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: ACL, 2011, pp. 351–356.