From Language to Time: A Temporal Expression Anchorer

Benjamin Han, Donna Gates and Lori Levin
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh PA 15213
{benhdj|dmg|lsl}@cs.cmu.edu

Abstract

Understanding temporal expressions in natural language is a key step towards incorporating temporal information in many applications. In this paper we describe a system capable of anchoring such expressions in English: system TEA features a constraint-based calendar model and a compact representational language to capture the intensional meaning of temporal expressions. We also report favorable results from experiments conducted on several email datasets.

1 Introduction

A key ingredient in incorporating temporal information into natural language applications is the normalization, or *anchoring* of temporal expressions, i.e., expressions using temporal terms such as *2005*, *evening*, *tomorrow*, etc. Some of these expressions can be classified into the following categories:

- **Explicit expressions** are the ones that can be immediately anchored; e.g., *June 2005*, *1998 Summer*, etc.
- **Deictic expressions** are the ones that form a specific relation with a *speech time*; e.g., *tomorrow*, *last year*, *two weeks from today*.
- **Relative expressions** are the ones that form a specific relation with a *temporal focus*, i.e., the implicit time central to a discourse; e.g., *from 5 to 7*, *on Friday*, etc.
- **Duration expressions** are the ones that describe certain length in time; e.g., *for about an hour*, *less than 20 minutes*.

Accomplishing this task requires not only the knowledge about how various temporal primitives are related (e.g., *February* in a leap year has 29 *days*), but also how they interact with one another given a description manifested by an expression. In this paper we describe a system capable of automatically anchoring the kinds of expressions listed above[1]. The input to the system TEA (Temporal Expression Anchorer) are English sentences with temporal expressions already identified, and the output is the normalization for each temporal expression. TEA has the following characteristics: (1) it incorporates a constraint-based calendar model to reason with under-specified expressions; (2) it can be extended to deal with new temporal primitives; (3) it captures the *intensional* meaning of temporal expressions using a compact representational language TCNL (Time Calculus for Natural Language).

A system overview of TEA is given in Fig. 1. The Finite-state Parser first takes an input sentence and translates its temporal expressions into TCNL formulae. The temporal references inside the formulae such as focus are then instantiated and any ambiguity is resolved in the Discourse Module. Finally the Evaluator Module takes the set of processed TCNL formulae and evaluate them to give the normalized times. At all stages the Calendar Model provides necessary services such as determining granularity of an expression, comparing two expressions chronologically, and solving constraint satisfaction problems induced by TCNL formulae, etc. Fig. 2[2] shows an example session using *TimeShell*[3], an interactive front-end of TEA, to illustrate this process: the Finite-state Parser first transduces the expression "*yesterday at 3pm*" into its TCNL formula $\{15_{\texttt{hour}}, 0_{\texttt{min}}, \texttt{now}-|1_{\texttt{day}}|\}$, the Discourse Module then rewrites the temporal reference `now` with $\{2006_{\texttt{year}}, \texttt{feb}, 7_{\texttt{day}}\}$ (speech time), and finally the Evaluator solves the formula to its answer `20060206T1500??`.

[1] Notable omissions of the expression types include *recurrence* (e.g., "*3pm every Tuesday and Thursday*") and *rate* expressions (e.g., "*twice a week*"). These are left to the future work.

[2] The output are shown in two formats: TCNL and an ISO8601-like form. In the latter a date is written in the form of YYMMDD and a time point is shown in the form of YYMMDDTHHMMSS; omitted information is indicated by '?'.

[3] `http://telltime.org`

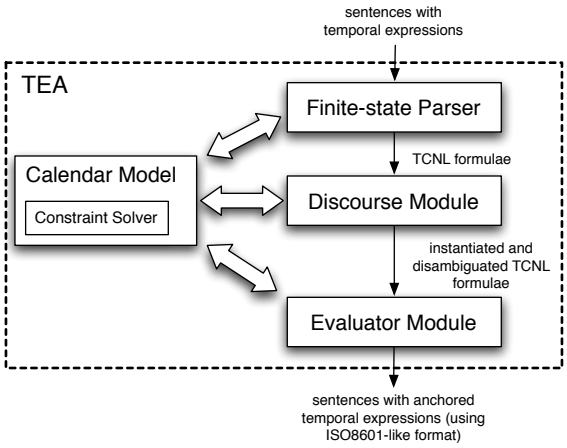

Figure 1. System overview of TEA

```
_ = {2006_year, feb, 3_day}
> yesterday at 3pm
TCNL: {15_hour, 0_min, now - |1_day|}
      = {2006_year, feb, 6_day, 15_hour, 0_min}
ICal+: 20060206T1500??

_ = {2006_year, feb, 6_day, 15_hour, 0_min}
> 2 days after Thanksgiving a year ago
TCNL: {{|4_{thu}| @ {_ - |1_year|, nov}} + |2_day|}
      = {2005_year, nov, 26_day}
ICal+: 20051126

_ = {2005_year, nov, 26_day}
> on Wednesday
TCNL: +{wed} = {104611_week, wed}
ICal+: 20051130
```

Figure 2. Example session in TimeShell

This result becomes the new focus that the Discourse Module then uses to replace the reference '_' in the next formula for the expression "*2 days after Thanksgiving a year ago*". This second formula also demonstrates how TCNL is able to represent the intensional meaning of *Thanksgiving*, which is the 4th Thursday in November. Finally the last expression "*on Wednesday*" is anchored to a specific day thanks to the prefix '+' in the extremely under-specified formula $+\{\texttt{wed}\}$. This instructs the Evaluator to find the first Wednesday possible on or after the focus.

The rest of the paper details TEA in the reversed order outlined above. Sec. 2 first introduces the constraint-based calendar model and its core algorithms. Sec. 3 then gives a concise description of TCNL. These two sections present an updated account to the work reported in [5] and [6]. The Discourse Module and the finite-state parsing are described in Sec. 4 and Sec. 5. Experimental result on email corpora is then reported in Sec. 6. Finally we conclude the paper and outline the future work in Sec. 7.

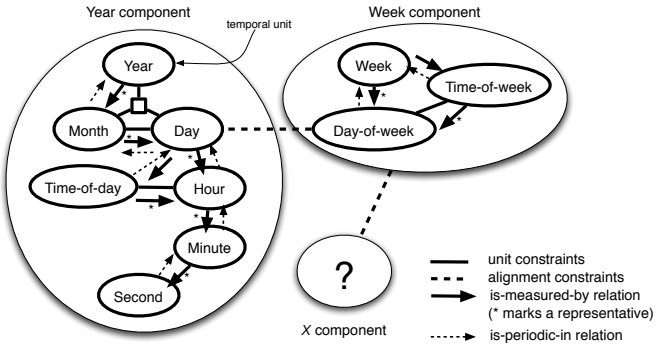

Figure 3. A partial model of the Gregorian calendar

2 Calendar Modeling

A calendar in TEA serves the role of a time ontology; i.e., to encapsulate the relations among a set of temporal primitives, or *temporal units*, so that they can be of use by our representational language TCNL. In literature there has been many works similar in purpose, such as the DAML Time Ontology [7], Calendar Logic [11], and an algebraic representation of granularity systems proposed in [3]. Our modeling of calendars is distinct in that it views a calendar as a *constraint system*, namely we treat a temporal unit as a variable with a discrete, finite and *fully ordered* domain (e.g., the domain of unit `month` consists of $\texttt{jan} < .. < \texttt{dec}$), and a temporal expression in natural language in effect assigns values to some of these units. This modeling has the following advantages: (1) granularity conversion is abstracted and localized in constraints, which makes the model easy to extend; (2) partial assignments to units can be refined by solving the induced Constraint Satisfaction Problems (CSP) [12], thus making the model a perfect fit for reasoning about under-specified expressions (e.g., knowing the date *February 29* implies it is in a leap year). Fig. 3 gives a pictorial view of a partial model for the Gregorian calendar.

The basic building block in our calendar model is a *calendar component*, which is essentially a souped-up constraint network. Multiple components are then *aligned* using constraints (more on this later). Within a calendar component, in addition to the constraints specifying compatible values among the constrained units (e.g., February in non-leap years cannot have 29 days), the units are also partially ordered using a designated *measurement* relation (solid arrows in Fig. 3); e.g., `month` is-measured-by `day`, or $\texttt{month} > \texttt{day}$. A reciprocal relation *periodicity* is also defined (dashed arrows in Fig. 3): if u_1 is-measured-by u_2, u_2 is-periodic-in u_1 *if* iterating through the possible values

of u_2 does not advance the value of u_1; e.g., `dow` (day-of-week: Monday..Sunday) is-periodic-in `week` but *not* periodic in `tow` (time-of-week: weekdays and weekends), because adding one day to Friday will advance the corresponding `tow` from `weekdays` to `weekend`. The periodicity relation then allows us to define the concept of *anchor paths*: a path $< u_n, \ldots, u_1, u >$ is an anchor path of unit u if u_i is-periodic-in u_{i+1} for $i = 1...(n-1)$, and u_n is a maximal unit under the measurement relation. We then say that a set of assignments is *anchored at unit* u if every unit on the anchor path of u has an instantiation (singleton assignment).

The core operations in our calendar model are constraint *propagation* and *distribution*: the former removes incompatible values from the domains of the units (represented by intervals) using the standard AC-3 algorithm[4] [9], and the latter iterates through every consistent set of instantiations using a chronological backtracking algorithm (e.g., given "*Friday the 13th*" find all possible anchored dates). In TEA the distribution algorithm is tailored to instantiate units in a given ordering (usually specified by an anchor path), and it can also start iterating from a given set of instantiations (e.g., iterating through all possible Friday the 13th starting from January 13, 2006). Fig. 4 gives a sketch of the distribution algorithm.

As mentioned earlier a temporal expression essentially gives partial assignments to temporal units. This set of assigned units acts as a *view* into the underlying calendar model; e.g., the view introduced by expression "*February 29*" is $\{\texttt{month}, \texttt{day}\}$. We can then define *granularity* of assignments as the set of *minimal* units of their view under the measurement relation; e.g., the granularity of "February 29" is $\{\texttt{day}\}$. Granularity conversion is therefore implemented as a graph-theoretic operation to find a new view with the desired set of minimal units; e.g., demoting the granularity of "*February 29*" to $\{\texttt{min}\}$ changes its view to $\{\texttt{month}, \texttt{day}, \texttt{hour}, \texttt{min}\}$, while promoting it to $\{\texttt{year}\}$ changes the view to $\{\texttt{year}\}$.

Building on the concepts of anchor paths and granularity, we can then determine the chronological ordering of two sets of assignments. We do so by first comparing two sets of assignments, say a_1 and a_2, on the same anchor path: we say a_1 is earlier than a_2 on an anchor path $< u_n, \ldots, u_1, u >$ iff there exists $i \leq n$ such that $a_1[u_j] < a_2[u_j]$ for all $j = n \ldots i$ (lexicographic comparison). We then say a_1 is *earlier* than a_2 iff there exists a unit u in the union of both granularities such that a_1 is earlier than a_2 on the anchor path of u. E.g., "*7am on February 29, 2004*" (granularity is $\{\texttt{hour}\}$) is earlier than "*afternoon on February 29, 2004*" (granularity is $\{\texttt{tod}\}$, i.e., time-of-day) since the former is earlier than the latter on the anchor path of `hour`.

[4]With *minimum remaining values* heuristics.

distribute(a, $ordering$, a_0)**:**

Input:
  assignments a and starting instantiations a_0
  $ordering$ is a list of units
Output: the next consistent instantiation of all units

$i = 0$; $rest$ = list of units that are not in $ordering$
do:
  $u = ordering[i]$; $backtrack$ = False
  if $a_0[u]$ exists:
    if domain of u has value $a_0[u]$:
      $v = a_0[u]$
    else: $backtrack$ = True
    remove $a_0[u]$
  else:
    if domain of u has any value:
      v = the next available value
    else: $backtrack$ = True

  if $backtrack$ == False:
    $a[u] = v$; remove v from domain of u
    propagate(a)
    if consistent:
      if i == (length of p) - 1:
        if $rest$ is not empty and
            there exists *no* a' from **distribute**(a, $rest$, a_0):
          $backtrack$ = True
        else: yield a
      else: $i = i + 1$
    else: $backtrack$ = True

  if $backtrack$ == True:
    if $i > 0$:
      revert $a[u]$ and domain of u to the state before
       the last assignment
      $i = i - 1$
    else: return

Figure 4. Sketch of the distribution algorithm

Another service provided by our calendar model is the *addition* operation: adding an integer to an instantiated unit; e.g., adding 1 day to unit `day` in the assignments "*February 29*" should result in assignments equivalent to "March 1". The algorithm (shown in Fig. 5)[5] essentially sets up the assignments properly and call the distribution algorithm to accomplish the feat. For certain units, however, a faster operation is possible. A call **add**(a, u, n) can be broken down into **add**(a, u, l) and **add**(a, u', m) where $n = m \cdot k + l$ and $l < k$, if unit u is periodic in u' and every value of u' is compatible with the same set of values of u (the size of the set is k).

Finally the modularity of our calendar model is achieved by allowing multiple calendar components to be related via

[5]This can be easily generalized to the $n < 0$ case.

add(a, u, n)**:**

Input: assignments a, unit u and integer $n > 0$
Output: True iff successful.

p = anchor path of u; $u_0 = p[0]$
a_0 is an empty dictionary
for v in p except u_0:
  $a_0[v] = \min(a[v])$
for every unit v except u_0:
  $a[v]$ = full domain of v

do:
  if there is a next set of assignments from **distribute**(a, p, a_0):
    $n = n - 1$
    if $n == 0$: return True
  else: return False

Figure 5. Sketch of the addition algorithm

alignment constraints. The purpose of such constraints is to establish an *order-preserving bijection mapping* between the instantiations of the two aligned components. The pair of units that are aligned between two components are called the *portal units* of the alignment, and the scope of an alignment is the union of the anchor paths of the portal units. For example, in Fig. 3 the portal units of the alignment between the year component and the week component are `day` and `dow`, and the scope of the alignment constraint is $\{\texttt{year}, \texttt{month}, \texttt{day}, \texttt{week}, \texttt{dow}\}$. Once the non-binary alignment constraint is specified, we can then translate an instantiation of one component to that of the other using the core algorithms described above; e.g., "*February 29, 2004*" is "*Sunday of the 104519th week*". The measurement and periodicity relations can also be extended across aligned calendar components so that operations such as granularity conversion and addition would work properly.

3 Time Calculus for Natural Language (TCNL)

Building on top of the constraint-based calendar model is TCNL, a compact formalism designed to capture the meaning of temporal expressions in natural language. Compared to many other alternatives in literature such as TOP [2], Timex2 [4] and TimeML/Timex3 [13], TCNL has the following characteristics: (1) it is *calendar-agnostic* (or ontology-agnostic); (2) it captures the *intensional* meaning of an expression (e.g., "*yesterday*" is not represented as a fixed date like 2006-02-02 but as a formula $\{\texttt{now}-|1_{\texttt{day}}|\}$; more on this point later); (3) it exposes contextual dependency by using temporal references such as *focus*; and (4) its type system and operators makes granularity conversion and re-interpretation a transparent process.

There are three types of temporal entities in a TCNL representation: *coordinates*, *quantities* and *enumerations*. A coordinate represents a set of assignments to temporal units (Sec. 2); e.g., "*Friday the 13th*" is represented as $\{\texttt{fri}, 13_{\texttt{day}}\}$. Semantically a coordinate represents a time point at a certain granularity even when it is under-specified and can be anchored at multiple possible positions on a time line; e.g, the formula above represents a single Friday on the 13th (of some month). An enumeration, on the other hand, represents a *set* of time points (sets of assignments), including but not limited to intervals; e.g., $[\{\texttt{wed}\},\{\texttt{fri}\}]$ represents "*Wednesday and Friday*" and $[\{\texttt{wed}\}:\{\texttt{fri}\}]$ denotes "*Wednesday to Friday*"[6]. Finally a quantity represents a certain number of temporal units (e.g., $|2_{\texttt{day}}|$ for "*two days*") or coordinates (e.g., $|2_{\{\texttt{fri}, 13_{\texttt{day}}\}}|$ for "*two Fridays the 13th*"). The semantics of a quantity is quite different from that of an interval: the latter must have a starting and an ending point (although they can be under-specified) and must denote a contiguous range on a time line, while a quantity just means a certain amount of "things", and they do not need to be adjacent to one another on a time line (e.g., no two Friday the 13th are adjacent). Following Sec. 2 the granularity of an entity is then defined as the set of minimal units among those appearing in the representation.

More complex temporal entities can be constructed using *operators*. All of the TCNL operators impose type and granularity requirements on their operands; an example is the *fuzzy-shifting operators* $+$ and $-$. In a formula $\{c+q\}$, the operand c must be a coordinate while q must be a quantity. In addition, the granularity of c must be brought to match that of q. Thus the formula $\{\texttt{now}+|1_{\texttt{day}}|\}$ ("*tomorrow*") is evaluated to February 3, 2006 (granularity is $\{\texttt{day}\}$) even when the temporal reference `now` is 10:03pm on February 2, 2006 (granularity is $\{\texttt{min}\}$). Another example is the *ordinal operator* @, which stipulates that the left operand must be a quantity and the right operand must be an enumeration, and that the granularity of the latter must be brought to match that of the former. Evaluating a formula such as $\{|2_{\{\texttt{sun}\}}|@\{\texttt{may}, 2005_{\texttt{year}}\}\}$ ("*the 2nd Sunday in May 2005*"[7]) thus first requires a *type coercion* from $\{\texttt{may}, 2005_{\texttt{year}}\}$ into the correct enumeration $[\{\texttt{sun}, 104580_{\texttt{week}}\}:\{\texttt{sat}, 104585_{\texttt{week}}\}]$. The operator @ then selects the 2nd possible coordinate that is a Sunday using the **find_the_nth** algorithm shown in Fig. 6. A summary of all TCNL operators is given in Table 1.

Temporal entities can also be related with one another using *relations*. A top-level relational term "re'" in a host entity e specifies that (e, e') is in relation r. Based on different type requirements TCNL provides five sets of rela-

[6] An interval represented this way always denotes the *shortest* possible interval; e.g., the example does not denote a time span of more than 7 days.

[7] Mother's day; TEA maintains a database of US holidays using formulae like this.

operator	Type requirement	Granularity requirement	Semantics	Example
$+$ and $-$	C × Q → C	g(LHS) ← g(RHS)	fuzzy forward/backward shifting	{now+|$1_{\texttt{day}}$|} (“*tomorrow*”)
$++$ and $--$	C × Q → C	g(LHS) ← min(g(LHS)∪g(RHS))	exact forward/backward shifting	{now++|$2_{\texttt{hour}}$|} (“*2 hours from now*”)
@	Q × E → C	g(RHS) ← g(LHS)	ordinal	{|$2_{\{\texttt{sun}\}}$|@{may}} (“*the 2nd Sunday in May*”)
&	C × C → C C × E → E E × C → E E × E → E	g(LHS) ← min(g(LHS)∪g(RHS))	distribution	{now &{now+|$1_{\texttt{year}}$|}} (“*this time next year*”) [{$15_{\texttt{hour}}$}&[{wed}:{fri}]] (“*3pm from Wednesday to Friday*”)

Table 1. Summary of operators in TCNL; LHS/RHS is the left/right operand, g(e) returns the granularity of e and $\min(s)$ returns the set of minimal units among s.

find_the_nth(n, c, $start$, end)**:**

Input: integer $n > 0$, coordinate c, $start$ and end
Output: a coordinate if successful, otherwise None.

$iter\_path =$
  union of anchor paths of units in granularity of $start$
sort $iter\_path$ using the measurement relation

do:
  if there is a next set of assignments a
      from **distribute**(c, $iter\_path$, $start$):
    if a is earlier or equal to end:
      $n = n - 1$
      if $n == 0$: return a
  else: return None

Figure 6. Sketch of the find_the_nth algorithm

Relations	Type requirement	Semantics
<, <=, >=, >	Q × Q	shorter-than, shorter-than or equal-to, longer-than or equal-to, and longer-than
<, <=, >=, >	C × C	before, before or equal-to, after or equal-to, and after
b, **s**, **d**, **de**, **f**, **di**	C × E	LHS is before/starting/during/during-equal/finishing/after RHS; **de** is defined as (**s** or **d** or **f**).
b, **s**, **f**, **bi**	E × C	LHS is a maximal interval that is before/starting at/finishing at/after RHS.
b, **m**, **o**, **s**, **d**, **f**, **=**, **fi**, **di**, **si**, **oi**, **mi**, **bi**	E × E	See [1].

Table 2. Summary of relations in TCNL; LHS/RHS is the left/right operand.

tions (Table 2). Some examples are {wed, < now} for “*a past Wednesday*”, {now, **de** {now+|$0_{\texttt{day}}$|}} for “*the rest of today*” and [**s** now] for “*from now on*”[8].

TCNL also provides two temporal references so representations of temporal expressions can use them to expose contextual dependency. We have seen reference ‘now’ used in various examples above: it denotes the *speech time* and usually it is kept constant during a discourse. The other reference available is the *temporal focus*, symbolized by ‘_’ (underscore). It is usually moved around in a discourse depending on which temporal location the discourse is focusing on[9]. A simple example is shown below:

> *After the Challenger accident* ***in ’86*** ({$1986_{\texttt{year}}$}), *shuttle missions were suspended* ***in the next 2 years*** ([_:{_+|$2_{\texttt{year}}$|}]).

[8]This forms an interval starting from now to a pre-defined maximal coordinate.

[9]Another contrast between the two different references is that *dectic expressions* such as “*tomorrow*” use speech time ({now+|$1_{\texttt{day}}$|}) while *relative expressions* such as “*the next day*” use focus ({_+|$1_{\texttt{day}}$|}).

Evaluating the second formula requires instantiating its focus with a previously mentioned time, in this case it is the year 1986. Managing focus movement (or focus tracking) is then relegated to the Discourse Module (to be described in Sec. 4).

It is worthwhile contrasting the use of a temporal focus in TCNL with similar devices adopted in other formalisms. For example, in TimeML/Timex3 the attribute `anchorTimeID` is used in a `TIMEX3` tag to “*introduce the ID of the time expression to which the TIMEX3 markable is temporally anchored*” [13]. An example for expression “*two weeks from next Tuesday*” is shown below:

```
<TIMEX3 tid="t1" type="TIME" value="2002-08-06"
 temporalFunction="true" anchorTimeID="t0">
two weeks from next Tuesday</TIMEX3>
```

The date referred to by `t0` (which was introduced earlier in the discourse) is then used to resolve the expression into the value `2002-08-06`. By contrast the same expression is represented in TCNL as {{_+|$1_{\{\texttt{tue}\}}$|}++|$2_{\texttt{week}}$|}. Using

a placeholder like _ is more akin to the idea of *lazy evaluation*: it allows the same meaning representation to be resolved into different denotations, thus enhancing the modularity of our approach[10]. Another relevant observation is that much of the inner arithmetic evident in the TCNL formula is opaque in the Timex3 representation.

It is not always obvious how one can factor out the effect of focus in a TCNL formula, however. Consider the following example:

> *I am free* ***next week*** ($\{\texttt{now}+|1_{\texttt{week}}|\}$). *How about* ***Friday*** ($\{\texttt{fri}\}$)?

We can evaluate the under-specified expression "*Friday*" here by finding its nearest possible instantiations on or after the focus ("*the next week*"); i.e., we can rewrite $\{\texttt{fri}\}$ into $\{|1_{\{\texttt{fri}\}}|@\{>= \ \_\}\}$ and evaluate the new formula. This step is only necessary, however, if the formula cannot be anchored in its original form (e.g., if the second expression is "*February 3, 2006*", then applying the same rewriting procedure would give us no consistent instantiation). This dilemma motivates the introduction of two *coordinate prefixes*: prefix '+'/'−' leading a coordinate indicates that the formula should be rewritten to find the nearest possible instantiation in the future/past of the focus, *if necessary*. Thus in the example above "*Friday*" should be represented as $+\{\texttt{fri}\}$ instead. Deciding whether to add a prefix or what prefix to add is again a responsibility of a different module.

A second pair of prefixes is also provided to deal with the effect brought by tense/aspect. The "*Friday*" in "*the company announced/will announce on Friday*" can denote a Friday either before or after now. Instead of hard-coding the relation existing between a coordinate with now (e.g., $\{\texttt{fri}, < \texttt{now}\}$ for the past tense), we use prefix '**f**' and '**p**' to mark this relation. Similar to the $+/-$ prefix the insertion of the relational term happens only when necessary.

Finally TCNL provides a handful of functions to represent the meaning of expressions such as "*late 2006*" and "*year-end*". Examples are $\texttt{early}(.)$, $\texttt{mid}(.)$ and $\texttt{late}(.)$ ($C \rightarrow C$).

4 Anchoring in Discourse

Two complications arise before the evaluation of a TCNL formula can commence, and they are handled in the Discourse Module in TEA (Fig. 1). The first is one of ambiguity - an example is ambiguous hour expressions. In TEA the Finite-state Parser produces multiple formulae for expressions like "*at 3*" - in this case they are $+\{3_{\texttt{hour}}\}$ and $+\{15_{\texttt{hour}}\}$[11]. To resolve this ambiguity we simply pick the one that evaluates to a point closer to the focus; e.g., in the expression above if the focus is at 1 pm, $+\{15_{\texttt{hour}}\}$ will be chosen (evaluated to 3 pm on the same day) instead of the other (evaluated to 3 am of the next day).

The second complication is the instantiation of temporal references appearing in a formula (both speech time '**now**' and focus '_'). While instantiating now is straightforward (e.g., experiments reported in Sec. 6 use email time stamp as the speech time), focus instantiation is more challenging. The current implementation of TEA uses a slightly constrained recency-based method for "tracking" focus: we simply pick the most recent formula prior to the one being evaluated to instantiate the focus, *except* when that formula comes from a noun-modifying temporal expression. The exception is motivated by examples such as this:

> *We received a copy of* ***2005*** *report and will send you our analysis* ***by Sunday*** ($\{<= +\texttt{f}\{\texttt{sun}\}\}$). ($\texttt{now} = \{\texttt{feb}, 3_{\texttt{day}}, 2006_{\texttt{year}}\}$)

Clearly $\{<= +\texttt{f}\{\texttt{sun}\}\}$ is not a Sunday in 2005. The conjecture is that a noun-modifying temporal expression usually serves as a temporal *co-reference* instead of introducing a new temporal entity into the discourse, and this reference has a more confined effect in anchoring the subsequent expressions.

We should emphasize that much of the current implementation of the Discourse Module is far from perfect (evident in the results shown in Sec. 6) and requires more work in the future.

5 Parsing Temporal Expressions

The Finite-state Parser in TEA (Fig. 1) first identifies all verb chunks in an input sentence and associates each temporal expression with its nearest verb chunk. Each expression with the tense/aspect information of its associated verb chunk is then used to build a TCNL formula. The formula-building process essentially breaks down an expression and constructs the representation bottom-up. The compositional nature of TCNL makes this a relatively painless process: for example, given an expression "Friday last week", the TCNL formulae for "Friday" ($\{\texttt{fri}\}$) and "last week" ($\{\texttt{now}-|1_{\texttt{week}}|\}$) are first built and then combined to give the final representation ($\{\texttt{fri}, \{\texttt{now}-|1_{\texttt{week}}|\}\}$). Note that we are not required to produce a single "normalized" representation for every equivalent expression; e.g., the formula $\{\texttt{now}-|1_{\{\texttt{fri}\}}|\}$ parsed from "last Friday" would be evaluated to the same date. This makes grammar development a much easier task[12].

[10] We should also emphasize that one of the design goals for TCNL is to make it easier for machines to generate these representations.

[11] Recall from Sec. 3 the prefix + indicates that the formula should be anchored in the future of focus.

[12] The grammar used in the experiments reported in Sec. 6 has 100 rules, including many rules for major US holidays.

Another interesting note is that the interpretation of an expression can be affected by the *granularities* of its sub-expressions. Take the following pair of expressions for example:

"*Tuesday before Christmas*"
= $\{\texttt{tue}, < \{|25_{\texttt{day}}|@\{\texttt{dec}\}\}\}$

"*Tuesday before 6pm*"
= $\{< \{\texttt{tue}, 18_{\texttt{hour}}\}$, **de** $\{\texttt{tue}\}\}$

Both of the expressions share the same "X before Y" pattern, but their interpretations are different (see Table 2 for relation **de**). The key to discriminate the two is to compare the granularities of X and Y: if Y if at a *higher* granularity (Sec. 2) then the first interpretation should be adopted. This observation has persuaded us to provide mechanisms for "estimating" the granularity of a formula (without first evaluating it) and making it available to the parser[13].

6 Experiments and Results

We have tested the effectiveness of TEA over time on several *email* corpora. Emails are of particular interest to us due to our work in project RADAR[14]: the project aims at building personal agents capable of scheduling meetings among different users. Understanding the meaning of temporal expressions is therefore a crucial step.

The email dataset used in our development and testing were collected from MBA students of Carnegie Mellon University over the year 1997 and 1998. The 277 students, organized in approximately 50 teams of 4 to 6 members, were participating in a 14-week course and running simulated companies in a variety of market scenarios [8]. For our study, 1,196 scheduling-related emails were manually selected from the 15,000+ dataset and were randomly divided into five sets (**email1** to **email5**). Only four of them are used in the results reported here: **email1** was used to establish our baseline, **email2** and **email5** were used for development, and part of **email4** was used for testing. The temporal expressions in all of the datasets were initially tagged using rules developed for MinorThird[15], and subsequently corrected manually by two of the authors. Table 3 shows some basic statistics of the datasets[16], and Fig. 7 shows a sample email from the datasets (edited).

It is worth noting that much of the previous work devoted on recognizing and normalizing temporal expressions have focused on *newswire texts*. Distribution-wise emails exhibit a very different nature: in [10] for example it was reported that the proportion of *explicit* expressions is about 25% in the the North American News Corpus. In contrast the same type of expressions accounts for only about 9.5% in the email datasets we use. Other characteristics of emails comparing to newswire include having a higher rate of human errors[17] and featuring more "creative" writing such as using tables, bullet lists, abbreviations, etc.

We first developed a prototype system and established our baseline over **email1** (50%). The system at that time did not have any focus tracking mechanism (i.e., it always used the time stamp as the focus), and it did not use any tense/aspect information. We then gradually developed TEA to its current form using **email1**, **email2** and **email5**. During the process we added the recency-based focus tracking mechanism, incorporated the tense/aspect information into each TCNL formula (via coordinate prefixes), and introduced several representational improvements. Finally we tested the system on the unseen dataset **email4**, and obtained the results shown in Table 4. Note that the percentages reported in the table are *accuracies*, i.e., the number of correctly anchored expressions over the total number of temporal expressions over a dataset, since we are assuming

	# of emails	# of tempex	explicit	deictic	relative	duration
email1	253	300	3 (1%)	139 (46.33%)	158 (52.67%)	N/A
email2	253	344	19 (5.5%)	112 (32.6%)	187 (54.4%)	27 (7.8%)
email4 (part.)	149	279	71 (25.4%)	77 (27.6%)	108 (38.7%)	22 (7.9%)
email5	126	213	14 (6.6%)	105 (49.3%)	92 (43.2%)	3 (1.4%)

Table 3. Basic statistics of the email datasets

Date: **Thu, 11 Sep 1997 00:14:36 -0500**

I have put an outline out in the n10f1 OpReview directory... *(omitted)*

We have very little time for this. Please call me **Thursday night** to get clarification. I will need graphs and prose in files **by Saturday Noon.**

– Mary

ps. Mark and John , I waited **until AFTER midnight** to send this .

Figure 7. A sample email (edited)

[13]This is not always possible as temporal references and functional terms might appear in a formula.

[14]Reflective Agent with Distributed Adaptive Reasoning. http://www.radar.cs.cmu.edu/external.asp

[15]`http://minorthird.sourceforge.net/`

[16]The percentages in some rows do not add up to 100% because some expressions like coordination can be classified into more than one type.

[17]This includes typos and use of incorrect expressions; e.g., using "tomorrow" in emails sent after midnight when "today" was intended.

	Accuracy	Parsing errors	Human errors	Anchoring errors
email1 (test)	50%	N/A	N/A	N/A
email2 (dev)	78.2%	10.47%	1.7%	9.63%
email5 (dev)	85.45%	5.16%	1%	8.39%
email4 (test)	76.34%	17.92%	$< 1\%$	5.74%

Table 4. Development and testing results

correct tagging of all of the expressions. Also note that the parsing errors referred to in Table 4 were brought by the incorrect/missing TCNL formulae produced by the Finite-state Parser. Our best result was achieved in the dev set **email5** (85.45%), and the accuracy over the test set **email4** was 76.34%. Overall the accuracy numbers are all compared favorably to the baseline. To put this performance in perspective, in [14] a similar task was performed over transcribed scheduling-related phone conversations. They reported an average accuracy 80.9% over the CMU test set and 68.9% over the NMSU test set. Although strictly speaking the two results cannot be compared due to differences in the nature of the corpora (transcription vs. typing), we nevertheless believe it represents a closer match compared to the other works done on the newswire genre. It should be noted that [14] also adopted a recency-based focus tracking method.

7 Conclusion and Future Work

In this paper we described a system capable of anchoring temporal expressions in English. System TEA features an extensible constraint-based calendar model and a compact representational language TCNL to capture the intensional meaning of temporal expressions. We also reported favorable results from our experiments of using TEA on several email datasets.

Looking into the future we would like to extend the implementation of TCNL to allow representing the meaning of *recurrence* expressions (e.g., "*every Wednesday at 6pm*"). Currently this is possible only for limited expressions such as "*6pm from Wednesday to Friday*" ([{$\texttt{18}_{\texttt{hour}}$}&[{wed}:{fri}]], which produces three coordinates when iterated). An extension is to introduce a pattern construct into an enumeration formula to denote recurrence (e.g., [{$\texttt{18}_{\texttt{hour}}$}&[{$*$wed}]] for "*every Wednesday at 6pm*").

We are also planning to use TEA on newswire texts in order to produce anchored event structures. Although the recency-based focus model so far has served us well in the email genre, we might need to devise a more elaborated tracking mechanism to account for the unique rhetorical structure often exhibited in newswire.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), or the Department of Interior-National Business Center (DOI-NBC).

References

[1] J. F. Allen. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23:123–154, 1984.

[2] I. Androutsopoulos. Temporal Meaning Representations in a Natural Language Front-end. In M. Gergatsoulis and P. Rondogiannis, editors, *Intensional Programming II (Proceedings of the 12th International Symposium on Languages for Intensional Programming*, Athens, Greece, 1999.

[3] C. Bettini, S. Jajodia, and S. X. Wang. *Time granularities in database, data mining, and temporal reasoning*. Springer-Verlag, Berlin, 2000.

[4] L. Ferro, L. Gerber, I. Mani, B. Sundheim, and G. Wilson. TIDES 2005 Standard for the Annotation of Temporal Expressions. Technical report, MITRE, April, 2005.

[5] B. Han and M. Kohlhase. A Time Calculus for Natural Language. In *The 4th Workshop on Inference in Computational Semantics*, Nancy, France, September 2003.

[6] B. Han and A. Lavie. A Framework for Resolution of Time in Natural Language. *TALIP Special Issue on Spatial and Temporal Information Processing*, 3(1):11–32, March 2004.

[7] J. R. Hobbs, G. Ferguson, J. Allen, P. Hayes, I. Niles, and A. Pease. A DAML ontology of time, Aug 23 2002.

[8] R. E. Kraut, S. R. Fussell, F. J. Lerch, and A. Espinosa. Coordination in teams: Evidence from a simulated management game. *Journal of Organizational Behavior*, to appear, 2004.

[9] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

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

[11] H. Ohlbach and D. Gabbay. Calendar logic. *Journal of Applied Non-classical Logics*, 8(4):291–324, 1998.

[12] Z. Ruttkay. Constraint Satisfaction - a Survey. Technical Report 11(2-3), CWI, 1998.

[13] R. Saurí, J. Littman, B. Knippen, R. Gaizauskas, A. Setzer, and J. Pustejovsky. *TimeML Annotation Guidelines, Version 1.2.1*, January 31 2006.

[14] J. M. Wiebe, T. P. O'Hara, T. Ohrstrom-Sandgren, and K. J. McKeever. An Empirical Approach to Temporal Reference Resolution. *Journal of Artificial Intelligence Research*, 9:247–293, 1998.