Parsing Time: Learning to Interpret Time Expressions

Gabor Angeli
Stanford University
Stanford, CA 94305
angeli@stanford.edu

Christopher D. Manning
Stanford University
Stanford, CA 94305
manning@stanford.edu

Daniel Jurafsky
Stanford University
Stanford, CA 94305
jurafsky@stanford.edu

Abstract

We present a probabilistic approach for learning to interpret temporal phrases given only a corpus of utterances and the times they reference. While most approaches to the task have used regular expressions and similar linear pattern interpretation rules, the possibility of phrasal embedding and modification in time expressions motivates our use of a compositional *grammar of time expressions*. This grammar is used to construct a *latent parse* which evaluates to the time the phrase would represent, as a logical parse might evaluate to a concrete entity. In this way, we can employ a loosely supervised EM-style bootstrapping approach to learn these latent parses while capturing both syntactic uncertainty and pragmatic ambiguity in a probabilistic framework. We achieve an accuracy of 72% on an adapted TempEval-2 task – comparable to state of the art systems.

1 Introduction

Temporal resolution is the task of mapping from a textual phrase describing a potentially complex time, date, or duration to a normalized (*grounded*) temporal representation. For example, possibly complex phrases such as *the week before last* are often more useful in their grounded form – e.g., `January 1 - January 7`.

The dominant approach to this problem in previous work has been to use rule-based methods, generally a combination of regular-expression matching followed by hand-written interpretation functions.

In general, it is appealing to learn the interpretation of temporal expressions, rather than hand-building systems. Moreover, complex hierarchical temporal expressions, such as *the Tuesday before last* or *the third Wednesday of each month*, and ambiguous expressions, such as *last Friday*, are difficult to handle using deterministic rules and would benefit from a recursive and probabilistic phrase structure representation. Therefore, we attempt to learn a temporal interpretation system where temporal phrases are parsed by a grammar, but this grammar and its semantic interpretation rules are latent, with only the input phrase and its grounded interpretation given to the learning system.

Employing probabilistic techniques allows us to capture ambiguity in temporal phrases in two important respects. In part, it captures syntactic ambiguity – e.g., *last Friday the 13th* bracketing as either *[last Friday] [the 13th]*, or *last [Friday the 13th]*. This also includes examples of lexical ambiguity – e.g., two meanings of *last* in *last week of November* versus *last week*. In addition, temporal expressions often carry a pragmatic ambiguity. For instance, a speaker may refer to either the next or previous Friday when he utters *Friday* on a Sunday. Similarly, *next week* can refer to either the coming week or the week thereafter.

Probabilistic systems furthermore allow propagation of uncertainty to higher-level components – for example recognizing that *May* could have a number of non-temporal meanings and allowing a system with a broader contextual scope to make the final judgment. We implement a CRF to detect temporal expressions, and show our model's ability to act as a component in such a system.

We describe our temporal representation, followed by the learning algorithm; we conclude with experimental results showing our approach to be competitive with state of the art systems.

2 Related Work

Our approach draws inspiration from a large body of work on parsing expressions into a logical form. The latent parse parallels the formal semantics in previous work, e.g., Montague semantics. Like these representations, a parse – in conjunction with the reference time – defines a set of matching entities, in this case the grounded time. The matching times can be thought of as analogous to the entities in a logical model which satisfy a given expression.

Supervised approaches to logical parsing prominently include Zelle and Mooney (1996), Zettlemoyer and Collins (2005), Kate et al. (2005), Zettlemoyer and Collins (2007), *inter alia*. For example, Zettlemoyer and Collins (2007) learn a mapping from textual queries to a logical form. This logical form importantly contains all the predicates and entities used in their parse. We loosen the supervision required in these systems by allowing the parse to be entirely latent; the annotation of the grounded time neither defines, nor gives any direct cues about the elements of the parse, since many parses evaluate to the same grounding. To demonstrate, the grounding for a week ago could be described by specifying a month and day, or as a *week ago*, or as *last* x – substituting today's day of the week for x. Each of these correspond to a completely different parse.

Recent work by Clarke et al. (2010) and Liang et al. (2011) similarly relax supervision to require only annotated answers rather than full logical forms. For example, Liang et al. (2011) constructs a latent parse similar in structure to a dependency grammar, but representing a logical form. Our proposed lexical entries and grammar combination rules can be thought of as paralleling the lexical entries and predicates, and the implicit combination rules respectively in this framework. Rather than querying from a finite database, however, our system must compare temporal expression within an infinite timeline. Furthermore, our system is run using neither lexical cues nor intelligent initialization.

Related work on interpreting temporal expressions has focused on constructing hand-crafted interpretation rules (Mani and Wilson, 2000; Saquete et al., 2003; Puscasu, 2004; Grover et al., 2010). Of these, HeidelTime (Strötgen and Gertz, 2010) and SUTime (Chang and Manning, 2012) provide particularly strong competition.

Recent probabilistic approaches to temporal resolution include UzZaman and Allen (2010), who employ a parser to produce deep logical forms, in conjunction with a CRF classifier. In a similar vein, Kolomiyets and Moens (2010) employ a maximum entropy classifier to detect the location and temporal type of expressions; the grounding is then done via deterministic rules.

3 Representation

We define a compositional representation of time; a type system is described in Section 3.1 while the grammar is outlined in Section 3.2 and described in detail in Sections 3.3 and 3.4.

3.1 Temporal Expression Types

We represent temporal expressions as either a Range, Sequence, or Duration. We describe these, the Function type, and the miscellaneous Number and Nil types below:

Range [and Instant] A period between two dates (or times). This includes entities such as `Today`, `1987`, or `Now`. We denote a range by the variable r. We maintain a consistent interval-based theory of time (Allen, 1981) and represent instants as intervals with zero span.

Sequence A sequence of Ranges, not necessarily occurring at regular intervals. This includes entities such as `Friday`, `November 27`th, or `last Friday`. A Sequence is a tuple of three elements $s = (r_s, \Delta_s, \rho_s)$:

1. $r_s(i)$: The ith element of a sequence, of type Range. In the case of the sequence `Friday`, $r_s(0)$ corresponds to *the Friday in the current week*; $r_s(1)$ is the Friday in the following week, etc.
2. Δ_s: The *distance* between two elements in the sequence – approximated if this distance is not constant. In the case of `Friday`, this distance would be a week.
3. ρ_s: The *containing unit* of an element of a sequence. For example, $\rho_{\texttt{Friday}}$ would be the Range corresponding to *the current week*. The sequence index $i \in \mathbb{Z}$ in $r_s(i)$ is defined relative

to $r_s(0)$ – the element in the same containing unit as the reference time.

We define the *reference time* t (Reichenbach, 1947) to be the instant relative to which times are evaluated. For the TempEval-2 corpus, we approximate this as the publication time of the article. While this is conflating Reichenbach's reference time with speech time, it is a useful approximation.

To contrast with Ranges, a Sequence can represent a number of grounded times. Nonetheless, pragmatically, not all of these are given equal weight – an utterance of *last Friday* may mean either of the previous two Fridays, but is unlikely to ground to anything else. We represent this ambiguity by defining a distribution over the elements of the Sequence. While this could be any distribution, we chose to approximate it as a Gaussian.

In order to allow sharing parameters between any sequence, we define the domain in terms of the *index* of the sequence rather than of a constant unit of time (e.g., seconds). To illustrate, the distribution over `April` would have a much larger variance than the distribution over `Sunday`, were the domains fixed. The probability of the ith element of a sequence thus depends on the beginning of the range $r_s(i)$, the reference time t, and the distance between elements of the sequence Δ_s. We summarize this in the equation below, with learned parameters μ and σ:

$$P_t(i) = \int_{\delta=-0.5}^{0.5} \mathcal{N}_{\mu,\sigma}\left(\frac{r_s(i) - t}{\Delta_s} + \delta\right) \quad (1)$$

Figure 1 shows an example of such a distribution; importantly, note that moving the reference time between two elements dynamically changes the probability assigned to each.

Duration A period of time. This includes entities like `Week`, `Month`, and `7 days`. We denote a duration with the variable d.

We define a special case of the Duration type to represent *approximate* durations, identified by their canonical unit (`week`, `month`, etc). These are used to represent expressions such as *a few years* or *some days*.

Function A function of arity less than or equal to two representing some general modification to one

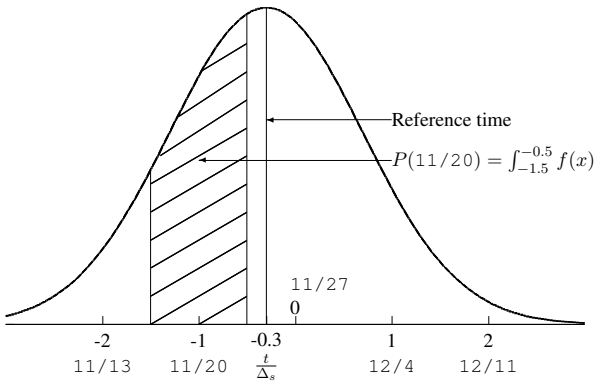

Figure 1: An illustration of a temporal distribution, e.g., `Sunday`. The reference time is labeled as time t between `Nov 20` and `Nov 27`; the probability that this sequence is referring to `Nov 20` is the integral of the marked area. The domain of the graph are the indices of the sequence; the distribution is overlaid with mean at the (normalized) reference time t/Δ_s; in our case Δ_s is a week. Note that the probability of an index changes depending on the exact location of the reference time.

of the above types. This captures semantic entities such as those implied in *last* x, *the third* x *[of* y*]*, or x *days ago*. The particular functions and their application are enumerated in Table 2.

Other Types Two other types bear auxiliary roles in representing temporal expressions, though they are not directly temporal concepts. In the grammar, these appear as preterminals only.

The first of these types is Number – denoting a number without any temporal meaning attached. This comes into play representing expressions such as *2 weeks*. The other is the Nil type – denoting terms which are not directly contributing to the semantic meaning of the expression. This is intended for words such as *a* or *the*, which serve as cues without bearing temporal content themselves. The Nil type is lexicalized with the word it generates.

Omitted Phenomena The representation described is a simplification of the complexities of time. Notably, a body of work has focused on reasoning about events or states relative to temporal expressions. Moens and Steedman (1988) describes temporal expressions relating to changes of state; Condoravdi (2010) explores NPI licensing in temporal expressions. Broader context is also not

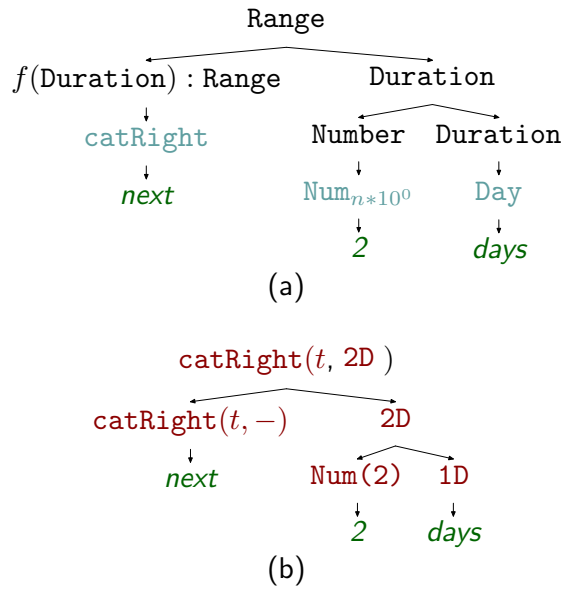

Figure 2: The grammar – (a) describes the CFG parse of the temporal *types*. Words are tagged with their nonterminal entry, above which only the types of the expressions are maintained; (b) describes the corresponding combination of the temporal *instances*. The parse in (b) is deterministic given the grammar combination rules in (a).

directly modeled, but rather left to systems in which the model would be embedded. Furthermore, vague times (e.g., *in the 90's*) represent a notable chunk of temporal expressions uttered. In contrast, NLP evaluations have generally not handled such vague time expressions.

3.2 Grammar Formalism

Our approach builds on the assumption that natural language descriptions of time are compositional in nature. Each word attached to a temporal phrase is usually compositionally modifying the meaning of the phrase. To demonstrate, we consider the expression *the week before last week*. We can construct a meaning by applying the modifier *last* to *week* – creating the previous week; and then applying *before* to *week* and *last week*.

We construct a paradigm for parsing temporal phrases consisting of a standard PCFG over temporal *types* with each parse rule defining a function to apply to the child nodes, or the word being generated. At the root of the tree, we recursively apply the functions in the parse tree to obtain a final temporal *value*. One can view this formalism as a rule-to-rule translation (Bach, 1976; Allen, 1995, p. 263), or a constrained Synchronous PCFG (Yamada and Knight, 2001).

Our approach contrasts with common approaches, such as CCG grammars (Steedman, 2000; Bos et al., 2004; Kwiatkowski et al., 2011), giving us more flexibility in the composition rules. Figure 2 shows an example of the grammar.

Formally, we define our temporal grammar $G = (\Sigma, S, \mathcal{V}, \mathcal{W}, \mathcal{R}, \theta)$. The alphabet Σ and start symbol S retain their usual interpretations. We define a set $\mathcal{V}$ to be the set of types, as described in Section 3.1 – these act as our nonterminals. For each $v \in \mathcal{V}$ we define an (infinite) set W_v corresponding to the possible instances of type v. Concretely, if $v = \textsf{Sequence}$, our set $W_v \in \mathcal{W}$ could contain elements corresponding to *Friday*, *last Friday*, *Nov. 27th*, etc. Each node in the tree defines a pair (v, w) such that $w \in W_v$, with combination rules defined over v and function applications performed on w.

A rule $R \in \mathcal{R}$ is defined as a pair $R = \left(v_i \rightarrow v_j v_k, f : (W_{v_j}, W_{v_k}) \rightarrow W_{v_i}\right)$. The first term is our conventional PCFG rule over the types $\mathcal{V}$. The second term defines the function to apply to the values returned recursively by the child nodes. Note that this definition is trivially adapted for the case of unary rules.

The last term in our grammar formalism denotes the rule probabilities θ. In line with the usual interpretation, this defines a probability of applying a particular rule $r \in R$. Importantly, note that the distribution over possible groundings of a temporal expression are not included in the grammar formalism. The learning of these probabilities is detailed in Section 4.

3.3 Preterminals

We define a set of preterminals, specifying their eventual *type*, as well as the temporal *instance* it produces when its function is evaluated on the word it generates (e.g., $f(\textit{day}) = \texttt{Day}$). A distinction is made in our description between entities with content roles versus entities with a functional role.

The first – consisting of Ranges, Sequences, and Durations – are listed in Table 1. A total of 62 such preterminals are defined in the implemented system, corresponding to primitive entities often appearing in newswire, although this list is easily adaptable to

Function	Description	Signature(s)
`shiftLeft`	Shift a Range or Sequence left by a Duration	$f : \mathsf{S}, \mathsf{D} \rightarrow \mathsf{S}$; $f : \mathsf{R}, \mathsf{D} \rightarrow \mathsf{R}$
`shiftRight`	Shift a Range or Sequence right by a Duration	$f : \mathsf{S}, \mathsf{D} \rightarrow \mathsf{S}$; $f : \mathsf{R}, \mathsf{D} \rightarrow \mathsf{R}$
`shrinkBegin`	Take the first Duration of a Range/Sequence	$f : \mathsf{S}, \mathsf{D} \rightarrow \mathsf{S}$; $f : \mathsf{R}, \mathsf{D} \rightarrow \mathsf{R}$
`shrinkEnd`	Take the last Duration of a Range/Sequence	$f : \mathsf{S}, \mathsf{D} \rightarrow \mathsf{S}$; $f : \mathsf{R}, \mathsf{D} \rightarrow \mathsf{R}$
`catLeft`	Take Duration units after the end of a Range	$f : \mathsf{R}, \mathsf{D} \rightarrow \mathsf{R}$
`catRight`	Take Duration units before the start of a Range	$f : \mathsf{R}, \mathsf{D} \rightarrow \mathsf{R}$
`moveLeft1`	Move the origin of a sequence left by 1	$f : \mathsf{S} \rightarrow \mathsf{S}$
`moveRight1`	Move the origin of a sequence right by 1	$f : \mathsf{S} \rightarrow \mathsf{S}$
n^{th} x `of` y	Take the n^{th} Sequence in y (Day of Week, *etc*)	$f : \mathsf{Number} \rightarrow \mathsf{S}$
`approximate`	Make a Duration approximate	$f : \mathsf{D} \rightarrow \mathsf{D}$

Table 2: The functional preterminals of the grammar; R, S, and D denote Ranges Sequences and Durations respectively. The name, a brief description, and the type signature of the function (as used in parsing) are given. Described in more detail in Section 3.4, the functions are most easily interpreted as operations on either an interval or sequence.

Type	Instances
Range	`Past`, `Future`, `Yesterday`, `Tomorrow`, `Today`, `Reference`, `Year(`n`)`, `Century(`n`)`
Sequence	`Friday`, `January`, ... `DayOfMonth`, `DayOfWeek`, ... `EveryDay`, `EveryWeek`, ...
Duration	`Second`, `Minute`, `Hour`, `Day`, `Week`, `Month`, `Quarter`, `Year`, `Decade`, `Century`

Table 1: The content-bearing preterminals of the grammar, arranged by their types. Note that the Sequence type contains more elements than enumerated here; however, only a few of each characteristic type are shown here for brevity.

fit other domains. It should be noted that the expressions, represented in `Typewriter`, have no a priori association with words, denoted by *italics*; this correspondence must be learned. Furthermore, entities which are subject to interpretation – for example `Quarter` or `Season` – are given a concrete interpretation. The n^{th} quarter is defined by evenly splitting a year into four; the seasons are defined in the same way but with winter beginning in December.

The functional entities are described in Table 2, and correspond to the Function type. The majority of these mirror generic operations on intervals on a timeline, or manipulations of a sequence. Notably, like intervals, times can be moved (*3 weeks ago*) or their size changed (*the first two days of the month*), or a new interval can be started from one of the endpoints (*the last 2 days*). Additionally, a sequence can be modified by shifting its origin (*last Friday*), or taking the n^{th} element of the sequence within some bound (*fourth Sunday in November*).

The lexical entry for the Nil type is tagged with the word it generates, producing entries such as Nil(a), Nil(November), etc. The lexical entry for the Number type is parameterized by the order of magnitude and ordinality of the number; e.g., 27^{th} becomes Number(10^1,ordinal).

3.4 Combination Rules

As mentioned earlier, our grammar defines both combination rules over types (in $\mathcal{V}$) as well as a method for combining temporal instances (in $W_v \in \mathcal{W}$). This method is either a function application of one of the functions in Table 2, a function which is implicit in the text (intersection and multiplication), or an identity operation (for Nils). These cases are detailed below:

- Function application, e.g., *last week*. We apply (or partially apply) a function to an argument on either the left or the right: $f(x, y) \odot x$ or $x \odot f(x, y)$. Furthermore, for functions of arity 2 taking a Range as an argument, we define a rule treating it as a unary function with the reference time taking the place of the second argument.

- Intersecting two ranges or sequences, e.g.,

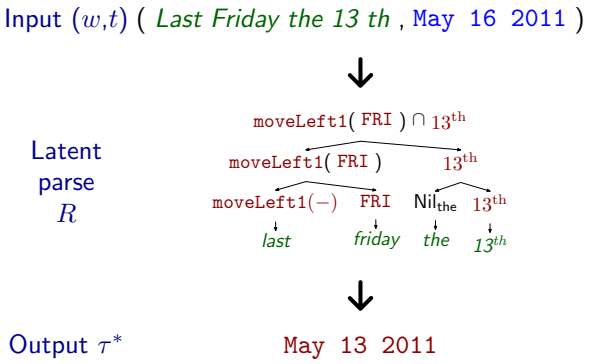

Figure 3: An overview of the system architecture. Note that the parse is latent – that is, it is not annotated in the training data.

November 27th. The intersect function treats both arguments as intervals, and will return an interval (Range or Sequence) corresponding to the overlap between the two.[1]

- Multiplying a Number with a Duration, e.g., *5 weeks*.
- Combining a non-Nil and Nil element with no change to the temporal expression, e.g., *a week*. The lexicalization of the Nil type allows the algorithm to take hints from these supporting words.

We proceed to describe learning the parameters of this grammar.

4 Learning

We present a system architecture, described in Figure 3. We detail the inference procedure in Section 4.1 and training in Section 4.2.

4.1 Inference

To provide a list of candidate expressions with their associated probabilities, we employ a k-best CKY parser. Specifically, we implement Algorithm 3 described in Huang and Chiang (2005), providing an $O(Gn^3k \log k)$ algorithm with respect to the grammar size G, phrase length n, and beam size k. We set the beam size to 2000.

[1] In the case of complex sequences (e.g., *Friday the 13th*) an A* search is performed to find overlapping ranges in the two sequences; the origin $r_s(0)$ is updated to refer to the closest such match to the reference time.

Revisiting the notion of pragmatic ambiguity, in a sense the most semantically complete output of the system would be a distribution – an utterance of *Friday* would give a distribution over Fridays rather than a best guess of its grounding. However, it is often advantageous to ground to a concrete expression with a corresponding probability. The CKY k-best beam and the temporal distribution – capturing syntactic and pragmatic ambiguity – can be combined to provide a *Viterbi* decoding, as well as its associated probability.

We define the probability of a syntactic parse y making use of rules $R \subseteq \mathcal{R}$ as $P(y) = P(w_1, \ldots w_n; R) = \prod_{i \to j,k \in R} P(j, k \mid i)$. As described in Section 3.1, we define the probability of a grounding relative to reference time t and a particular syntactic interpretation $P_t(i|y)$. The product of these two terms provides the probability of a grounded temporal interpretation; we can obtain a Viterbi decoding by maximizing this joint probability:

$$P_t(i, y) = P(y) \times P_t(i|y) \quad (2)$$

This provides us with a framework for obtaining grounded times from a temporal phrase – in line with the annotations provided during training time.

4.2 Training

We present an EM-style bootstrapping approach to training the parameters of our grammar jointly with the parameters of our Gaussian temporal distribution.

Our TimEM algorithm for learning the parameters for the grammar (θ), jointly with the temporal distribution (μ and σ) is given in Algorithm 1. The inputs to the algorithm are the initial parameters θ, μ, and σ, and a set of training instances $\mathcal{D}$. Furthermore, the algorithm makes use of a Dirichlet prior α on the grammar parameters θ, as well as a Gaussian prior $\mathcal{N}$ on the mean of the temporal distribution μ. The algorithm outputs the final parameters θ^*, μ^* and σ^*.

Each training instance is a tuple consisting of the words in the temporal phrase w, the annotated grounded time τ^*, and the reference time of the utterance t. The input phrase is tokenized according to Penn Treebank guidelines, except we additionally

Algorithm 1: TimEM

Input: Initial parameters θ, μ, σ; data $\mathcal{D} = \{(w, \tau^*, t)\}$; Dirichlet prior α, Gaussian prior $\mathcal{N}$

Output: Optimal parameters θ^*, μ^*, σ^*

1 **while** *not converged* **do**
2 $\quad (\bar{M}_\theta, \bar{M}_{\mu,\sigma})$:= `E-Step` $(\mathcal{D},\theta,\mu,\sigma)$
3 $\quad (\theta, \mu, \sigma)$:= `M-Step` $(\bar{M}_\theta, \bar{M}_{\mu,\sigma})$
4 **end**
5 **return** (θ_s, μ, σ)

6 **begin** `E-Step` $(\mathcal{D},\theta,\mu,\sigma)$
7 $\quad \bar{M}_\theta = []$; $\bar{M}_{\mu,\sigma} = []$
8 $\quad$ **for** $(w, \tau^*, t) \in \mathcal{D}$ **do**
9 $\quad\quad \bar{m}_\theta$ = []; $\bar{m}_{\mu,\sigma}$ = []
10 $\quad\quad$ **for** $y \in$ *k-bestCKY*(w, θ) **do**
11 $\quad\quad\quad$ **if** $p = P_{\mu,\sigma}(\tau^* \mid y, t) > 0$ **then**
12 $\quad\quad\quad\quad \bar{m}_\theta$ += (y, p); $\bar{m}_{\mu,\sigma}$ += (i, p)
13 $\quad\quad\quad$ **end**
14 $\quad\quad$ **end**
15 $\quad\quad \bar{M}$ += normalize$(\bar{m}_\theta)$
16 $\quad\quad \bar{M}_{\mu,\sigma}$ += normalize$(\bar{m}_{\mu,\sigma})$
17 $\quad$ **end**
18 $\quad$ **return** $\bar{M}$
19 **end**

20 **begin** `M-Step` $(\bar{M}_\theta,\bar{M}_{\mu,\sigma})$
21 $\quad \theta'$:= bayesianPosterior$(\bar{M}_\theta, \alpha)$
22 $\quad \sigma'$:= mlePosterior$(\bar{M}_{\mu,\sigma})$
23 $\quad \mu'$:= bayesianPosterior$(\bar{M}_{\mu,\sigma}, \sigma', \mathcal{N})$
24 $\quad$ **return** (θ', μ', σ')
25 **end**

split on the characters '-' and '/,' which often delimit a boundary between temporal entities. Beyond this preprocessing, no language-specific information about the meanings of the words are introduced, including syntactic parses, POS tags, etc.

The algorithm operates similarly to the EM algorithms used for grammar induction (Klein and Manning, 2004; Carroll and Charniak, 1992). However, unlike grammar induction, we are allowed a certain amount of supervision by requiring that the predicted temporal expression match the annotation. Our *expected statistics* are therefore more accurately our normalized expected counts of *valid* parses.

Note that in conventional grammar induction, the expected sufficient statistics can be gathered analytically from reading off the chart scores of a parse. This does not work in our case for two reasons. In part, we would like to incorporate the probability of the temporal grounding in our feedback probability. Additionally, we are only using parses which are valid candidates – that is, the parses which ground to the correct time $\tau*$ – which we cannot establish until the entire expression is parsed. The expected statistics are thus computed non-analytically via a beam on both the possible parses (line 10) and the possible temporal groundings of a given interpretation (line 11).

The particular EM updates are the standard updates for multinomial and Gaussian distributions given fully observed data. In the multinomial case, our (unnormalized) parameter updates, with Dirichlet prior α, are:

$$\theta'_{mn|l} = \alpha + \sum_{(y,p)\in\bar{M}_\theta} \sum_{v_{jk|i}\in y} \mathbb{1}\left(v_{jk|i} = v_{mn|l}\right) p \quad (3)$$

In the Gaussian case, the parameter update for σ is the maximum likelihood update; while the update for μ incorporates a Bayesian prior $\mathcal{N}(\mu_0, \sigma_0)$:

$$\sigma' = \sqrt{\frac{1}{\sum_{(i,p)\in\bar{M}_{\mu,\sigma}} p} \sum_{(i,p)\in\bar{M}_{\mu,\sigma}} (i-\mu')^2 \cdot p} \quad (4)$$

$$\mu' = \frac{\sigma'^2\mu_0 + \sigma_0^2 \sum_{(i,p)\in\bar{M}_{\mu,\sigma}} i \cdot p}{\sigma'^2 + \sigma_0^2 \sum_{(i,p)\in\bar{M}_{\mu,\sigma}} p} \quad (5)$$

As the parameters improve, the parser more efficiently prunes incorrect parses and the beam incorporates valid parses for longer and longer phrases. For instance, in the first iteration the model must learn the meaning of both words in *last Friday*; once the parser learns the meaning of one of them – e.g., *Friday* appears elsewhere in the corpus – subsequent iterations focus on proposing candidate meanings for *last*. In this way, a progressively larger percentage of the data is available to be learned from at each iteration.

5 Evaluation

We evaluate our model against current state-of-the art systems for temporal resolution on the English

System	Train		Test	
	Type	Value	Type	Value
GUTime	0.72	0.46	0.80	0.42
SUTime	0.85	0.69	**0.94**	0.71
HeidelTime	0.80	0.67	0.85	0.71
OurSystem	**0.90**	**0.72**	0.88	**0.72**

Table 3: TempEval-2 Attribute scores for our system and three previous systems. The scores are calculated using gold extents, forcing a guessed interpretation for each parse.

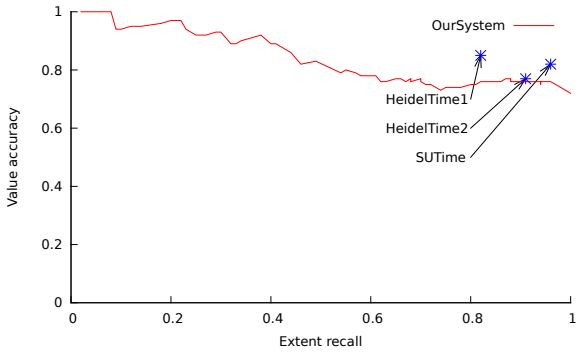

Figure 4: A precision-recall curve for our system, compared to prior work. The data points are obtained by setting a threshold minimum probability at which to guess a time creating different extent recall values. The curve falls below HeidelTime1 and SUTime in part from lack of context, and in part since our system was not trained to optimize this curve.

portion of the TempEval-2 Task A dataset (Verhagen et al., 2010).

5.1 Dataset

The TempEval-2 dataset is relatively small, containing 162 documents and 1052 temporal phrases in the training set and an additional 20 documents and 156 phrases in the evaluation set. Each temporal phrase was annotated as a `TIMEX3`[2] tag around an adverbial or prepositional phrase

5.2 Results

In the TempEval-2 A Task, system performance is evaluated on detection and resolution of expressions. Since we perform only the second of these, we evaluate our system assuming gold detection.

Similarly, the original TempEval-2 scoring scheme gave a precision and recall for detection, and an accuracy for only the temporal expressions attempted. Since our system is able to produce a guess for every expression, we produce a precision-recall curve on which competing systems are plotted (see Figure 4). Note that the downward slope of the curve indicates that the probabilities returned by the system are indicative of its confidence – the probability of a parse correlates with the probability of that parse being correct.

Additionally, and perhaps more accurately, we compare to previous system scores when constrained to make a prediction on every example; if no guess is made, the output is considered incorrect. This in general yields lower results, as the system is not allowed to abstain on expressions it does not recognize. Results are summarized in Table 3.

We compare to three previous rule-based systems. GUTime (Mani and Wilson, 2000) presents an older but widely used baseline.[3] More recently, SUTime (Chang and Manning, 2012) provides a much stronger comparison. We also compare to HeidelTime (Strötgen and Gertz, 2010), which represents the state-of-the-art system at the TempEval-2 task.

5.3 Detection

One of the advantages of our model is that it can provide candidate groundings for any expression. We explore this ability by building a detection model to find candidate temporal expressions, which we then ground. The detection model is implemented as a Conditional Random Field, with features over the morphology and context. Particularly, we define the following features:

- The word and lemma within 2 of the current word.
- The word shape[4] and part of speech of the current word.

[2]See `http://www.timeml.org` for details on the TimeML format and `TIMEX3` tag.

[3]Due to discrepancies in output formats, the output of GUTime was heuristically patched and manually checked to conform to the expected format.

[4]Word shape is calculated by mapping each character to one of uppercase, lowercase, number, or punctuation. The first four characters are mapped verbatim; subsequent sequences of similar characters are collapsed.

	Extent			Attribute	
System	P	R	F_1	Typ	Val
GUTime	0.89	0.79	0.84	0.95	0.68
SUTime	0.88	**0.96**	**0.92**	**0.96**	0.82
HeidelTime1	**0.90**	0.82	0.86	**0.96**	**0.85**
HeidelTime2	0.82	0.91	0.86	0.92	0.77
OurSystem	0.89	0.84	0.86	0.91	0.72

Table 4: TempEval-2 Extent scores for our system and three previous systems. Note that the attribute scores are now relatively low compared to previous work; unlike rule-based approaches, our model can guess a temporal interpretation for any phrase, meaning that a good proportion of the phrases not detected would have been interpreted correctly.

- Whether the current word is a number, along with its ordinality and order of magnitude
- Prefixes and suffixes up to length 5, along with their word shape.

We summarize our results in Table 4, noting that the performance indicates that the CRF and interpretation model find somewhat different phrases hard to detect and interpret respectively.

5.4 Discussion

Our system performs well above the GUTime baseline and is competitive with both of the more recent systems. In part, this is from more sophisticated modeling of syntactic ambiguity: e.g., *the past few weeks* has a clause *the past* – which, alone, should be parsed as `PAST` – yet the system correctly disprefers incorporating this interpretation and returns the approximate duration `1 week`. Furthermore, we often capture cases of pragmatic ambiguity – for example, empirically, *August* tends to refers to the previous August when mentioned in February.

Compared to rule-based systems, we attribute most errors the system makes to either data sparsity or missing lexical primitives. For example – illustrating sparsity – we have trouble recognizing *Nov.* as corresponding to `November` (e.g., *Nov. 13*), since the publication time of the articles happen to often be near November and we prefer tagging the word as `Nil` (analogous to *the 13th*). Missing lexical primitives, in turn, include tags for *1990s*, or *half* (in *minute and a half*); as well as missing functions, such as *or* (in *weeks or months*).

Remaining errors can be attributed to causes such as providing the wrong Viterbi grounding to the evaluation script (e.g., last rather than this Friday), differences in annotation (e.g., 24 hours is marked wrong against a day), or missing context (e.g., the publication time is not the true reference time), among others.

6 Conclusion

We present a new approach to resolving temporal expressions, based on synchronous parsing of a fixed grammar with learned parameters and a compositional representation of time. The system allows for output which captures uncertainty both with respect to the syntactic structure of the phrase and the pragmatic ambiguity of temporal utterances. We also note that the approach is theoretically better adapted for phrases more complex than those found in TempEval-2.

Furthermore, the system makes very few language-specific assumptions, and the algorithm could be adapted to domains beyond temporal resolution. We hope to improve detection and explore system performance on multilingual and complex datasets in future work.

Acknowledgements The authors would like to thank Valentin Spitkovsky, David McClosky, and Angel Chang for valuable discussion and insights. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, AFRL, or the US government.

References

James F. Allen. 1981. An interval-based representation of temporal knowledge. In *Proceedings of the 7th international joint conference on Artificial intelligence*, pages 221–226, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

James Allen. 1995. *Natural Language Understanding*. Benjamin/Cummings, Redwood City, CA.

E. Bach. 1976. An extension of classical transformational grammar. In *Problems of Linguistic Metatheory*

(Proceedings of the 1976 Conference), Michigan State University.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of Coling*, pages 1240–1246, Geneva, Switzerland. COLING.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. Technical report, Providence, RI, USA.

Angel Chang and Chris Manning. 2012. SUTIME: a library for recognizing and normalizing time expressions. In *Language Resources and Evaluation*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL*, pages 18–27, Uppsala, Sweden.

Cleo Condoravdi. 2010. NPI licensing in temporal clauses. *Natural Language and Linguistic Theory*, 28:877–910.

Claire Grover, Richard Tobin, Beatrice Alex, and Kate Byrne. 2010. Edinburgh-LTG: TempEval-2 system description. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, Sem-Eval, pages 333–336.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing, pages 53–64.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *AAAI*, pages 1062–1068, Pittsburgh, PA.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. In *ACL*.

Oleksandr Kolomiyets and Marie-Francine Moens. 2010. KUL: recognition and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, Sem-Eval '10, pages 325–328.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *EMNLP*, pages 1512–1523, Edinburgh, Scotland, UK.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *ACL*.

Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *ACL*, pages 69–76, Hong Kong.

Marc Moens and Mark Steedman. 1988. Temporal ontology and temporal reference. *Computational Linguistics*, 14:15–28.

G. Puscasu. 2004. A framework for temporal resolution. In *LREC*, pages 1901–1904.

Hans Reichenbach. 1947. *Elements of Symbolic Logic*. Macmillan, New York.

E. Saquete, R. Muoz, and P. Martnez-Barco. 2003. Terseo: Temporal expression resolution system applied to event ordering. In *Text, Speech and Dialogue*, pages 220–228.

Mark Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA, USA.

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

Naushad UzZaman and James F. Allen. 2010. TRIPS and TRIOS system for TempEval-2: Extracting temporal information from text. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, Sem-Eval, pages 276–283.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 57–62, Uppsala, Sweden.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *ACL*, pages 523–530.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055, Portland, OR.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pages 658–666. AUAI Press.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *EMNLP-CoNLL*, pages 678–687.