Context-dependent Semantic Parsing for Time Expressions

Kenton Lee†, Yoav Artzi†, Jesse Dodge‡∗, and Luke Zettlemoyer†

† Computer Science & Engineering, University of Washington, Seattle, WA
{kentonl, yoav, lsz}@cs.washington.edu

‡ Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA
jessed@cs.cmu.edu

Abstract

We present an approach for learning context-dependent semantic parsers to identify and interpret time expressions. We use a Combinatory Categorial Grammar to construct compositional meaning representations, while considering contextual cues, such as the document creation time and the tense of the governing verb, to compute the final time values. Experiments on benchmark datasets show that our approach outperforms previous state-of-the-art systems, with error reductions of 13% to 21% in end-to-end performance.

1 Introduction

Time expressions present a number of challenges for language understanding systems. They have rich, compositional structure (e.g., “2nd Friday of July”), can be easily confused with non-temporal phrases (e.g., the word “May” can be a month name or a verb), and can vary in meaning in different linguistic contexts (e.g., the word “Friday” refers to different dates in the sentences “We met on Friday” and “We will meet on Friday”). Recovering the meaning of time expressions is therefore challenging, but provides opportunities to study context-dependent language use. In this paper, we present the first context-dependent semantic parsing approach for learning to identify and interpret time expressions, addressing all three challenges.

Existing state-of-the-art methods use hand-engineered rules for reasoning about time expressions (Strötgen and Gertz, 2013). This includes both detection, identifying a phrase as a time expression, and resolution, mapping such a phrase into a standardized time value. While rule-based approaches provide a natural way to express expert knowledge, it is relatively difficult to encode preferences between similar competing hypotheses and provide prediction confidence. Recently, methods for learning probabilistic semantic parsers have been shown to address such limitations (Angeli et al., 2012; Angeli and Uszkoreit, 2013). However, these approaches do not account for any surrounding linguistic context and were mainly evaluated with gold standard mentions.

We propose to use a context-dependent semantic parser for both detection and resolution of time expressions. For both tasks, we make use of a hand-engineered Combinatory Categorial Grammar (CCG) to construct a set of meaning representations that identify the time being described. For example, this grammar maps the phrase “2nd Friday of July” to the meaning representation $\text{intersect}(\text{nth}(2, \text{friday}), \text{july})$, which encodes the set of all such days. Detection is then performed with a binary classifier to prune the set of text spans that can be parsed with the grammar (e.g., to tell that “born in 2000” has a time expression but “a 2000 piece puzzle” does not).

For resolution, we consider mentions sequentially and use a log-linear model to select the most likely meaning for each. This choice depends on contextual cues such as previous time expressions and the tense of the governing verb (e.g., as required to correctly resolve cases like “We should meet on the 2nd Friday of July”).

Such an approach provides a good balance between hand engineering and learning. For the relatively closed-class time expressions, we demonstrate that it is possible to engineer a high quality CCG lexicon. We take a data-driven approach for grammar design, preferring a grammar with high coverage even if it results in parsing ambiguities. We then learn a model to accurately select between competing parses and incorporate signals from the surrounding context, both more difficult to model with deterministic rules.

For both problems, we learn from TimeML an-
notations (Pustejovsky et al., 2005), which mark
mentions and the specific times they reference.
Training the detector is a supervised learning
problem, but resolution is more challenging, re-
quiring us to reason about latent parsing and
context-dependent decisions.

We evaluate performance in two domains: the
TempEval-3 corpus of newswire text (Uzzaman et
al., 2013) and the WikiWars corpus of Wikipedia
history articles (Mazar and Dale, 2010). On these
benchmark datasets, we present new state-of-the-
art results, with error reductions of up to 28% for
the detection task and 21% for the end-to-end task.

2 Formal Overview

Time Expressions We follow the TIMEX3 stan-
dard (Pustejovsky et al., 2005) for defining time
expressions within documents. Let a document
$D = \langle w_1, \ldots, w_n \rangle$ be a sequence of $n$
words $w_i$ and a mention $m = (i, j)$ indicate start
and end indices for a phrase $\langle w_i, \ldots, w_j \rangle$ in $D$. Define
a time expression $e = (t, v)$ to include both a
temporal type $t$ and value $v$. The temporal type
take one of four possible values, indicating if the expression
e is a date (e.g., “January 10, 2014”), time (e.g.,
“11:59 pm”), duration (e.g., “6 months”), or set (e.g.,
“every year”). The value $v$ is an extension
of the ISO 8601 standard, which encodes the time
that mention $m$ refers to in the context provided by
document $D$. For example, in a document written
on Tuesday, January 7, 2014, “Friday,” “three days
later,” and “January 10th” would all resolve to the
value 2014-01-10. The time values are similarly
defined for a wide range of expressions, such as
underspecified dates (e.g., XXXX-01-10 for “Ja-
nuary 10th” when the year is not inferable from
context) and durations (P2D for “two days”).

Tasks Our goal is to find all time expressions in
an input document. We divide the problem into
two parts: detection and resolution. The detection
problem is to take an input document $D$ and output
a mention set $M = \{ m_i \mid i = 1 \ldots n \}$ of phrases
in $D$ that describe time expressions. The resolution
problem (often also called normalization) is,
given a document $D$ and a set of mentions $M$, to
map each $m \in M$ to the referred time expression
e. This paper addresses both of these tasks.

Approach We learn separate, but related, mod-
els for detection and resolution. For both tasks, we
define the space of possible compositional mean-
ning representations $Z$, where each $z \in Z$ defines
a unique time expression $e$. We use a log-linear
CCG (Steedman, 1996; Clark and Curran, 2007)
to rank possible meanings $z \in Z$ for each men-
tion $m$ in a document $D$, as described in
Section 4. Both detection (Section 5) and resolution
(Section 6) rely on the semantic parser to identify
likely mentions and resolve them within context.
For learning we assume access to TimeML data
containing documents labeled with time expres-
sions. Each document $D$ has a set \{ $(m_i, e_i) \mid i = 1 \ldots n$ \}, where each mention $m_i$ marks a phrase
that resolves to the time expression $e_i$.

Evaluation We evaluate performance (Sec-
tion 8) for both newswire text and Wikipedia
articles. We compare to the state-of-the-art
systems for end-to-end resolution (Strögen and
Gertz, 2013) and resolution given gold men-
tions (Bethard, 2013b), both of which do not use
any machine learning techniques.

3 Representing Time

We use simply typed lambda calculus to represent
time expressions. Our representation draws heav-
ily from the representation proposed by Angeli et
al. (2012), who introduced semantic parsing for
this task. There are five primitive types: duration
$D$, sequence $S$, range $R$, approximate reference $A$
and numeral $N$, as described below. Table 1 lists
the available constants for each type.

Duration A period of time. Each duration is a
multiple of one of a closed set of possible base
durations (e.g., hour, day, and quarter), which
we refer to as its granularity. Table 1 includes the
complete set of base durations used.

Range A specific interval of time, following an
interval-based theory of time (Allen, 1981). The
interval length is one of the base durations, which
is the granularity of the range. Given two ranges
$R$ and $R'$, we say that $R \subseteq R'$ if the endpoints of
$R$ lie on or within $R'$.

Sequence A set of ranges with identical granu-
larity. The granularity of the sequence is that of
its members. For example, thursday, which has a
Day granularity, denotes the set of all day-granular ranges enclosing specific Thursdays. Given a range \( R \) and sequence \( S \), we say that \( R \in S \) if \( R \) is a member of \( S \). Given two sequences \( S \) and \( S' \) we say that \( S \subseteq S' \) if \( R \in S \) implies \( R \in S' \).

**Approximate Reference** An approximate time relative to the reference time. For example, *past* and *future*. To handle mentions such as “a while,” we add the constant *unknown*.

**Numerals** An integer, for example, 5 or 1990. Numerals are used to denote specific ranges, such as the year 2001, or to modify a duration’s length.

**Functions** We also allow for functional types, for example \( <s,r> \) is assigned to a function that maps from sequences to ranges. Table 2 lists all supported functions with example mentions.

**Context Dependent Constants** To mark places where context-dependent choices will need to be made during resolution, we use two placeholder constants. First, *ref_time* denotes the mention reference time, which is later set to either the document time or a previously resolved mention. Second, *temp_d* is used in the *shift* function to determine its return granularity, as described in Table 2, and is later replaced with the granularity of either the first or second argument of the enclosing *shift* function. Section 4.3 describes how these decisions are made.

### 4 Parsing Time Expressions

We define a three-step derivation to resolve mentions to their TIMEX3 value. First, we use a CCG to generate an initial logical form for the mention. Next, we apply a set of operations that modify the

| Type | Primitive Constants |
|------|---------------------|
| Duration | second, minute, hour, timeofday, day, month, season, quarter, weekend, week, year, decade, century, temp_d |
| Sequence | monday, tuesday, wednesday, thursday, friday, saturday, sunday, january, february, march, april, may, june, july, august, september, october, november, december, winter, spring, summer, fall, night, morning, afternoon, evening |
| Range | ref_time |
| Approximate reference | present, future, past, unknown |
| Numerals | 1, 2, 3, 1999, 2000, 2001, ... |

Table 1: The types and primitive logical constants supported by the logical language for time.

Initial logical form, as appropriate for its context. Finally, the logical form is resolved to a TIMEX3 value using a deterministic process.

#### 4.1 Combinatory Categorial Grammars

CCG is a linguistically motivated categorial formalism for modeling a wide range of language phenomena (Steedman, 1996; Steedman, 2000). A CCG is defined by a lexicon and a set of combinator terms. The lexicon pairs words with categories and the combinator terms define how to combine categories to create complete parse trees.

For example, Figure 1 shows a CCG parse tree for the phrase “one week ago.” The parse tree is read top to bottom, starting from assigning categories to words using the lexicon. The lexical entry *ago* \( \vdash NP\neg NP : \lambda x.\mathit{shift}(\mathit{ref}_{\mathit{time}}, -1 \times x, \mathit{temp}_{\mathit{d}}) \) for the word “ago” pairs it with a category that has syntactic type \( NP\neg NP \) and semantics \( \lambda x.\mathit{shift}(\mathit{ref}_{\mathit{time}}, -1 \times x, \mathit{temp}_{\mathit{d}}) \). Each intermediate parse node is then constructed by applying one of a small set of binary or unary operations (Steedman, 1996; Steedman, 2000), which modify both the syntax and semantics. We use backward (\(<\)) and forward (\(>\)) application and several unary type-shifting rules to handle number combinations. For example, in Figure 1 the category of the span “one week” is combined with the category of “ago” using backward application (\(<\)). Parsing concludes with a logical form representing the meaning of the complete mention.

**Hand Engineered Lexicon** To parse time expressions, we use a CCG lexicon that includes 287 manually designed entries, along with automatically generated entries such as numbers and common formats of dates and times. Figure 2 shows example entries from our lexicon.
### Table 2: Functional constants used to build logical expressions for representing time.

| Function | Description | Example |
|----------|-------------|---------|
| \( \times(n, (d, d)) \) | Given a duration \( D \) and a numeral \( N \), return a duration \( D' \) that is \( N \) times longer than \( D \). | “after three days of questioning” \( 3 \times \text{day} \) |
| \( \text{some} (e, d) \) | Given a duration \( D \), returns \( D' \), s.t. \( D' = D \) for some \( n > 1 \). | “he left for a few days” \( \text{some}(\text{day}) \) |
| \( \text{seq}(a, s) \) | Given a duration \( D \), generate a sequence \( S \), s.t. \( S \) includes all ranges of type \( D \). | previous(seq(year), ref_time) |
| \( \text{this}(x, (r, r)) \) | Given a sequence \( S \) and a range \( R \), returns the range \( R' \in S \), s.t. there exists \( R \subseteq R' \) and the length of \( R' \) is minimal. | “a meeting this friday” \( \text{this}(\text{friday}, \text{ref_time}) \) |
| \( \text{next}(x, (r, r)) \) | Given a sequence \( S \) and a range \( R \), returns the range \( R'' \in S' \) that is the one after/before \( \text{this}(S, R) \). | “arriving next month” \( \text{next}(\text{seq(month)}, \text{ref_time}) \) |
| \( \text{previous}(x, (r, r)) \) | Given a sequence \( S \) and a range \( R \), returns the range \( R'' \in S'' \) that is closest to \( R \) in the forward/backward direction. | “during the coming weekend” \( \text{nearest_forward(seq(weekend), \text{ref_time})} \) |
| \( \text{intersect}(x, (r, r)) \) | Given two sequences \( S, S' \), where the duration of entries in \( S \) is shorter than those in \( S' \), return a sequence \( S'' \subseteq S \), where for each \( R \in S'' \), there exists \( R' \in S' \) s.t. \( R \subseteq R' \). | “until the second quarter of the year” \( \text{nth}(2, \text{seq(quarter)}, \text{seq(year)}) \) |
| \( \text{shift}(x, (d, d)) \) | Given a range \( R \), a duration \( D \), and a duration \( G \), return the range \( R' \), s.t. the starting point of \( R' \) is moved by the length of \( D \). \( R' \) is converted to represent a range of granularity \( G \) by expanding if \( G \) has larger granularity, and is undefined if \( G \) has smaller granularity. | “a week ago, we went home” \( \text{shift}(\text{ref_time}, -1 \times 1 \times \text{week}, \text{temp_d}) \) |
| \( \times(n, (n, n)) \) | Given two numerals \( N' \) and \( N'' \), returns a numeral \( N''' \) representing their product \( N' \times N'' \). | “the battle lasted for one hundred days” \( 1 \times 100 \times \text{day} \) |
| \( + (n, n) \) | Given two numerals \( N' \) and \( N'' \), returns a numeral \( N''' \) representing their sum \( N' + N'' \). | “open twenty four hours” \( (20 + 4) \times \text{hour} \) |
| \( \overset{\text{every}}{x, s} \) | Given a sequence \( S \), returns a sequence with \( SE T \) temporal every(seq(day)) | “one dose each day” every(seq(day)) |
| \( \overset{\text{bc}}{x, s} \) | Convert a year to BC. | “during five hundred BC” \( \text{bc}(500, \text{seq(year)}) \) |

### Manually Designed Entries:

| Function | Description | Example |
|----------|-------------|---------|
| several | \( \lambda x.\text{some}(x) \) | \( N/P : \lambda x.\text{this}(x, \text{ref_time}) \) |
| this | \( \lambda x.\text{this}(x, \text{ref_time}) \) | \( N/P : \lambda x.\text{every}(x) \) |
| each | \( \lambda x.\text{every}(x) \) | \( N/P/\overset{\text{every}}{x, s} \) |
| before | \( \overset{\text{every}}{x, s} \) | \( N/P/\overset{\text{every}}{x, s} \) |
| year | \( N : \text{year} \) | \( N : \text{year} \) |
| wednesday | \( N : \text{wednesday} \) | \( \text{20s} : N : \text{nth}(192, \text{seq(week)} \) |
| yesterday | \( N : \text{shift}(\text{ref_time}, -1 \times \text{day}, \text{temp_d}) \) | \( \text{20s} : N : \text{nth}(192, \text{seq(week)} \) |

### Automatically Generated Entries:

| 1992 | \( N : \text{nth}(1992, \text{seq(year)} \) | \( N : \text{nth}(1992, \text{seq(year)} \) |
| nineteen ninety two | \( N : \text{nth}(992, \text{seq(year)} \) | \( 09:30 : N : \text{nth}(31, \text{seq(minute)}, \text{seq(hour)}) \) |
| 3rd | \( N : \text{nth}(3, \text{seq(day)}, \text{seq(month)}) \) | \( \text{3rd} : N : \text{nth}(3, \text{seq(day)}, \text{seq(month)}) \) |

| Figure 2: Example lexical entries. |

### 4.2 Context-dependent Operations

To correctly resolve mentions to TIMEX3 values, the system must account for contextual information from various sources, including previous mentions in the document, the document creation time, and the sentence containing the mention. We consider three types of context operations, each takes as input a logical form \( z' \), modifies it and returns a new logical form \( z \). Each context-dependent parse \( y \) specifies one operator of each type, which are applied to the logical form constructed by the CCG grammar, to produce the final, context-dependent logical form LF(\( y \)).

### Reference Time Resolution

The logical constant \( \text{ref_time} \) is replaced by either \( \text{dct} \), representing the document creation time, or \( \text{last_range} \), the last \( r \)-type mention resolved in the document. For example, consider the mention “the following year”, which is represented using the logical form \( \text{next}(\text{seq(year)}, \text{ref_time}) \). Within the sentence “1998 was colder than the following year”, the resolution of “the following year” depends on the previous mention “1998”. In contrast, in “The following year will be warmer”, its resolution depends on the document creation time.
Directionality Resolution If \( z' \) is s-typed we modify it to nearest forward(\( z', ref\text{ time} \)), nearest backward(\( z', ref\text{ time} \)), or \( z' \). For example, given the sentence “...will be launched in april”, the mention “april”, and its logical form \( april \), we would like to resolve it to the coming April, and therefore modify it to nearest forward(\( april, ref\text{ time} \)).

Shifting Granularity Every occurrence of the logical constant \( temp\_d \), which is used as an argument to the function shift (see Table 2), is replaced with the granularity of either the first argument, the origin of the shift, or the second argument, the delta of the shift. This determines the final granularity of the output. For example, if the reference time is 2002-01, the mention “two years earlier” would resolve to either a month (since the reference time is of month granularity) or a year (since the delta is of year granularity).

4.3 Resolving Logical Forms

For a context-dependent parse \( y \), we compute the TIMEX3 value \( TM(y) \) from the logical form \( z = LF(y) \) with a deterministic step that performs a single traversal of \( z \). Each primitive logical constant from Table 1 contributes to setting part of the TIMEX3 value (for example, specifying the day of the week) and the functional constants in Table 2 dictate transformations on the TIMEX3 values (for example, shifting forward or backward in time).\(^2\)

5 Detection

The detection problem is to take an input document \( D \) and output a mention set \( M = \{ m_i | i = 1, \ldots, n \} \), where each mention \( m_i \) indexes a specific phrase in \( D \) that delimits a time expression.

Algorithm The detection algorithm considers all phrases that our CCG grammar \( \Lambda \) (Section 4) can parse, uses a learned classifier to further filter this set, and finally resolves conflicts between any overlapping predictions. We use a CKY algorithm to efficiently determine which phrases the CCG grammar can parse and only allow logical forms for which there exists some context in which they would produce a valid time expression, e.g. ruling out \( intersect\text{ }\textit{monday}, \text{ }\textit{tuesday} \). Finally, we build the set \( M \) of non-overlapping mentions using a step similar to non-maximum suppression:

\[^2\]The full details are beyond the scope of this paper, but an implementation is available on the author’s website.

the mentions are sorted by length (longest first) and iteratively added to \( M \), as long as they do not overlap with any mention already in \( M \).

Filtering Model Given a mention \( m \), its document \( D \), a feature function \( \phi \), the CCG lexicon \( \Lambda \), and feature weights \( \theta \), we use a logistic regression model to define the probability distribution:

\[
P(t|m, D; \Lambda, \theta) = \frac{e^{\theta \cdot \phi(m, D, \Lambda)}}{1 + e^{\theta \cdot \phi(m, D, \Lambda)}}
\]

where \( t \) indicates whether \( m \) is a time expression.

Features We use three types of indicator features that test properties of the words in and around the potential mention \( m \).

Context tokens Indicate the presence of a set of manually specified tokens near the mention. These include quotations around the mention, the word “old” after the mention, and prepositions of time (such as “in”, “until”, and “during”) before.

Part of speech Indicators that pair each word with its part of speech, as assigned by the Stanford tagger (Toutanova et al., 2003).

Lexical group Each lexical entry belongs to one of thirteen manually defined lexical groups which cluster entries that contribute to the final time expression similarly. These groups include numbers, days of the week, months, seasons, etc. For each group, we include a feature indicating whether the parse includes a lexical entry from that group.

determiner dependency Indicates the presence of a determiner in the mention and whether its parent in the dependency tree (generated by the Stanford parser (de Marneffe et al., 2006)) also resides within the mention.

Learning Finally, we construct the training data by considering all spans that (1) the CCG temporal grammar can parse and (2) are not strict subsraps of an annotated mention. All spans that exactly matched the gold labels are used as positive examples and all others are negatives. Given this relaxed data, we learn the feature weights \( \theta \) with L1-regularization. We set the probability threshold for detecting a time expression by optimizing the F1 score over the training data.

6 Resolution

The resolution problem is to, given a document \( D \) and a set of mentions \( M \), map each \( m \in M \) to the correct time expression \( e \). Section 4 defined
the space of possible time expression that can be constructed for an input mention \( m \) in the context of a document \( D \). In general, there will be many different possible derivations, and we will learn a model for selecting the best one.

**Model** Let \( y \) be a context-dependent CCG parse, which includes a parse tree \( TR(y) \), a set of context operations \( CNTX(y) \) applied to the logical form at the root of the tree, a final context-dependent logical form \( LF(y) \) and a TIMEX3 value \( TM(y) \). Define \( \phi(m, D, y) \in \mathbb{R}^d \) to be a \( d \)-dimensional feature-vector representation and \( \theta \in \mathbb{R}^d \) to be a parameter vector. The probability of a parse \( y \) for mention \( m \) and document \( D \) is:

\[
P(y|m, D; \theta, \Lambda) = \frac{e^{\theta \cdot \phi(m, D, y)}}{\sum_{y'} e^{\theta \cdot \phi(m, D, y')}}
\]

The inference problem at test time requires finding the best resolution by solving \( y^*(m, D) = \arg \max_y P(y|m, D; \theta, \Lambda) \), where the final output TIMEX3 value is \( TM(y^*(m, D)) \).

**Inference** We find the best context-dependent parse \( y \) by enumeration, as follows. We first parse the input mention \( m \) with a CKY-style algorithm, following previous work (Zettlemoyer and Collins, 2005). Due to the short length of time expressions and the manually constructed lexicon, we can perform exact inference. Given a parse, we then enumerate all possible outcomes for the context resolution operators. In practice, there are never more than one hundred possibilities.

**Features** The resolution features test properties of the linguistic context surrounding the mention \( m \), relative to the context-dependent CCG parse \( y \).

**Governor verb** We define the governor verb to be the nearest ancestor verb in the dependency parse of any token in \( m \). We include features indicating the concatenation of the part-of-speech of the governor verb, its auxiliary verb if present, and the selected direction resolution operator (see Section 4.2). This feature helps to distinguish “They met on Friday” from “They will meet on Friday.”

**Temporal offset** If the final logical form \( LF(y) \) is a range, we define \( t \) to be the time difference between \( TM(y) \) and the reference time. For example, if the reference time is 2000-01-10 and the mention resolves to 2000-01-01, then \( t \) is -9 days. This feature indicates one of eleven bucketed values for \( t \), including same day, less than a week, less than a month, etc. It allows the model to encode the likely temporal progression of a narrative. This feature is ignored if the granularity of \( TM(y) \) or the reference time is greater than a year.

**Shift granularity** The logical constant \( shift \) (Table 2) takes three arguments: the origin (range), the delta (duration), and the output granularity (duration). This indicator feature is the concatenation of each argument’s granularity for every \( shift \) in \( LF(y) \). It allows the model to determine whether “a year ago” refers to a year or a day.

**Reference type** Let \( r \) denote whether the reference time is the document creation time \( dct \) or the last range \( last \_range \). Let \( g_l \) and \( g_r \) denote the granularities of \( LF(y) \) and the reference time, respectively. We include features indicating the concatenations: \( r+g_l, r+g_r, \) and \( r+g_l+g_r \). Additionally, we include features indicating the concatenation of \( r \) with each lexical entry used in the parse \( TR(y) \). These features allow the model to encode preferences in selecting the correct reference time.

**Fine-grained type** These features indicate the fine-grained type of \( TM(y) \), such as day of the month or week of the year. We also include a feature indicating the concatenation of each of these features with the direction resolution operator that was used. These features allow the model to represent, for example, that minutes of the year are less likely than days of the month.

**Intersections** These features indicate the concatenation of the granularities of any two sequences that appear as arguments to an \( intersect \) constant.

**Learning** To estimate the model parameters \( \theta \) we assume access to a set of training examples \( \{(m_i, d_i, c_i) : i = 1, \ldots, n\} \), where each mention \( m_i \) is paired with a document \( d_i \) and a TIMEX3 value \( c_i \). We use the AdaGrad algorithm (Duchi et al., 2011) to optimize the conditional, marginal log-likelihood of the data. For each mention, we marginalize over all possible context-dependent parses, using the predictions from the model on the previous gold mentions to fill in missing context, where necessary. After parameter estimation, we set a probability threshold for retaining a resolved time expression by optimizing value F1 (see Section 8) over the training data.
and Mooney (1996), Zettlemoyer and Collins (2005), and Wong and Mooney (2007). Recently, research in this area has focused on learning for various forms of relatively weak but easily gathered supervision. This includes learning from question-answer pairs (Clarke et al., 2010; Liang et al., 2011; Kwiatkowski et al., 2013), from conversational logs (Artzi and Zettlemoyer, 2011), with distant supervision (Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013), and from sentences paired with system behavior (Goldwasser and Roth, 2011; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013b). Recently, Angeli et al. introduced the idea of learning semantic parsers to resolve time expressions (Angeli et al., 2012) and showed that the approach can generalize to multiple languages (Angeli and Uszkoreit, 2013). Similarly, Bethard demonstrated that a hand-engineered semantic parser is also effective (Bethard, 2013b). However, these approaches did not use the semantic parser for detection and did not model linguistic context during resolution.

We build on a number of existing algorithmic ideas, including using CCGs to build meaning representations (Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2010; Kwiatkowski et al., 2011), building derivations to transform the output of the CCG parser based on context (Zettlemoyer and Collins, 2009), and using weakly supervised parameter updates (Artzi and Zettlemoyer, 2011; Artzi and Zettlemoyer, 2013b). However, we are the first to use a semantic parsing grammar within a mention detection algorithm, thereby avoiding the need to represent the meaning of complete sentences, and the first to develop a context-dependent model for semantic parsing of time expressions.

Time expressions have been extensively studied as part of the TimeEx task, including 9 teams who competed in the 2013 TempEval-3 competition (Uzzaman et al., 2013). This line of work builds on ideas from TimeBank (Pustejovsky et al., 2003) and a number of different formal models for temporal reasoning, e.g. Allen (1983), Moens and Steedman (1988). In 2013, HeidelTime (Strötgen and Gertz, 2013) was the top performing system. It used deterministic rules defined over regular expressions to perform both detection and resolution, and will provide a comparison system for our evaluation in Section 8.

### 8 Experimental Setup

#### Data

We evaluate performance on the TempEval-3 (Uzzaman et al., 2013) and WikiWars (Mazur and Dale, 2010) datasets. Figure 3 shows summary statistics for both datasets. For the TempEval-3 corpus, we use the given training and testing set splits. Since the training set has lower inter-annotator agreement than the testing set (Uzzaman et al., 2013), we manually corrected all of the mistakes we found in the training data. The original training set is denoted Dev* and the corrected Dev. We report (1) cross-validation development results on Dev*, (2) cross-validation development and ablation results for Dev, and (3) held-out test results after training with Dev. For WikiWars, we randomly assigned the data to include 17 training documents (2,228 time expressions) and 5 test documents (363 time expressions). We use cross-validation on the training data for development. All cross-validation experiments used 10 folds.

#### Implementation

Our system was implemented using the open source University of Washington Semantic Parsing Framework (Artzi and Zettlemoyer, 2013a). We used LIBLINEAR (Fan et al., 2008) to learn the detection model.

#### Parameter Settings

We use the same set of parameters for both datasets, chosen based on development experiments. For detection, we set the regularization parameter to 10 with a stopping crite-
Table: TempEval-3 development and test results, compared to the top systems in the shared task.

| System          | Dev* | Test |
|-----------------|------|------|
| This work       | 84.6 | 86.1 |
| HeidelTime      | 83.7 | 83.9 |
| NavyTime        | 78.7 | 79.7 |
| ClearTK         | 85.9 | 82.7 |

Table: WikiWars development and test results.

| System          | Dev* | Test |
|-----------------|------|------|
| This work       | 90.3 | 87.7 |
| HeidelTime      | 86.0 | 85.2 |

Figure 4: TempEval-3 development and test results, compared to the top systems in the shared task.

Figure 5: WikiWars development and test results.

Figure 6: TempEval-3 gold mention value accuracy.

Evaluation Metrics We use the official TempEval-3 scoring script and report the standard metrics. We report detection precision, recall and $F_1$ with relaxed and strict metrics; a gold mention is considered detected for the relaxed metric if any of the output candidates overlap with it and is detected for the strict metric if the extent of any output candidates matches exactly. For resolution, we report value accuracy, measuring correctness of time expressions detected according to the relaxed metric. We also report value precision, recall, and $F_1$, which score an expression as correct if it is both correctly detected (relaxed) and resolved. For end-to-end performance, value $F_1$ is the primary metric. Finally, we report accuracy and $F_1$ for temporal types, as defined in Section 2, for the TempEval dataset (WikiWars does not include type labels).

Comparison Systems We compare our system primarily to HeidelTime (Ströten and Gertz, 2013), which is state of the art in the end-to-end task. For the TempEval-3 dataset, we also compare to two other strong participants of the shared task. These include NavyTime (Chambers, 2013), which had the top relaxed detection score, and ClearTK (Bethard, 2013a), which had the top strict detection score and type $F_1$ score. We also include a comparison with Bethard’s synchronous context free grammar (SCFG) (Bethard, 2013b), which is state-of-the-art in the task of resolution with gold mention boundaries.

9 Results

End-to-end results Figure 4 shows development and test results for TempEval-3. Figure 5 shows these numbers for WikiWars. In both datasets, we achieve state-of-the-art test scores. For detection, we show up to 3-point improvements in strict and relaxed $F_1$ scores. These numbers outperform all systems participating in the shared task, which used a variety of techniques including hand-engineered rules, CRF tagging models, and SVMs. For resolution, we show up to 4-point improvements in the value $F_1$ score, also outperforming participating systems, all of which used hand-engineered rules for resolution.

Gold Mentions Figure 6 reports development and test results with gold mentions. Our approach outperforms the state of the art, SCFG (Bethard, 2013b), which also used a hand engineered grammar, but did not use machine learning techniques.

4 These numbers vary slightly from those reported; we did not count the document creation times as mentions.
Precision vs. Recall Our probabilistic model of time expression resolution allows us to easily tradeoff precision and recall for end-to-end performance by varying the resolution probability threshold. Figure 7 shows the precision vs. recall of the resolved values from 10-fold cross validation of TempEval-3 Dev and WikiWars Dev. We are able to achieve precision at or above 90% with reasonable recall, nearly 70% for WikiWars and over 85% for TempEval-3.

Ablation Study Figures 4-5 also show comparisons for our system with no context. We ablate the ability to refer to the context during resolution by removing contextual information from the resolution features and only allowing the document creation time to be the reference time.

We see an interesting asymmetry in the effect of modeling context across the two domains. We find that context is much more important in WikiWars (19 point difference) than in TempEval (2 point difference). This result reconfirms the difference in domains that Strötgen and Gertz (2012) noted during the development of HeidelTime: history articles have narrative structure that moves back and forth through time while newspaper text typically describes events happening near the document creation time. This difference helps us to understand why previous learning systems have been able to ignore context and perform well on newswire text.

Error Analysis To investigate the source of error, we compute oracle results for resolving gold mentions over the TempEval-3 Dev dataset. We found that our system produces a correct candidate derivation for 96% of the mentions.

We also manually categorized all resolution errors for end-to-end performance with 10-fold cross validation of the TempEval-3 Dev dataset, shown in Figure 8. The lexicon allows for effective parsing, contributing to only 2% of the overall errors. However, context is more challenging. The three largest categories, responsible for 64.7% of the errors, were incorrect use of the context operators. More expressive modeling will be required to fully capture the complex pragmatics involved in understanding time expressions.

10 Conclusion We presented the first context-dependent semantic parsing system to detect and resolve time expressions. Both models used a Combinatory Categorial Grammar (CCG) to construct a set of possible temporal meaning representations. This grammar defined the possible phrases for detection and the inputs to a context-dependent reasoning step that was used to construct the output time expression during resolution. Experiments demonstrated that our approach outperforms state-of-the-art systems.

In the future, we aim to develop joint models for reasoning about events and time expressions, including detection and resolution of temporal relations. We are also interested in testing coverage in new domains and investigating techniques for semi-supervised learning and learning with noisy data. We hypothesize that semantic parsing techniques could help in all of these settings, providing a unified mechanism for compositional analysis within temporal understanding problems.

Acknowledgments The research was supported in part by DARPA under the DEFT program through the AFRL (FA8750-13-2-0019) and the CSSG (N11AP20020), and the NSF (IIS-1115966, IIS-1252835). The authors thank Nicholas Fitzgerald, Tom Kwiatkowski, and Mark Yatskar for helpful discussions, and the anonymous reviewers for helpful comments.
References

James F. Allen. 1981. An interval-based representation of temporal knowledge. In Proceedings of the 7th International Joint Conference on Artificial Intelligence.

James F Allen. 1983. Maintaining knowledge about temporal intervals. Communications of the ACM, 26(11):832–843.

Gabor Angeli and Jakob Uszkoreit. 2013. Language-independent discriminative parsing of temporal expressions. In Proceedings of the Conference of the Association of Computational Linguistics.

Gabor Angeli, Christopher D Manning, and Daniel Jurafsky. 2012. Parsing time: Learning to interpret time expressions. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics.

Y. Artzi and L.S. Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.

Y. Artzi and L.S. Zettlemoyer. 2013a. UW SPF: The University of Washington Semantic Parsing Framework.

Y. Artzi and L.S. Zettlemoyer. 2013b. Weakly supervised learning of semantic parsers to actions. Transactions of the Association for Computational Linguistics, 1(1):49–62.

Steven Bethard. 2013a. Cleartk-timeml: A minimalist approach to tempeval 2013. In Second Joint Conference on Lexical and Computational Semantics.

Steven Bethard. 2013b. A synchronous context free grammar for time normalization. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.

Q. Cai and A. Yates. 2013. Semantic parsing freebase: Towards open-domain semantic parsing. In Joint Conference on Lexical and Computational Semantics: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity.

Nathanael Chambers. 2013. Navytime: Event and time ordering from raw text. In Second Joint Conference on Lexical and Computational Semantics.

Angel X Chang and Christopher Manning. 2012. Su-time: A library for recognizing and normalizing time expressions. In Proceedings of the 8th International Conference on Language Resources and Evaluation.

D.L. Chen and R.J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In Proceedings of the National Conference on Artificial Intelligence.

S. Clark and J. R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. Computational Linguistics, 33(4):493–552.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world’s response. In Proceedings of the Conference on Computational Natural Language Learning.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In Proceedings of the 5th International Conference on Language Resources and Evaluation.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12:2121–2159.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research, 9:1871–1874.

Michele Filannino, Gavin Brown, and Goran Nenadic. 2013. Mantine: Temporal expression identification and normalization in the tempeval-3 challenge. In Second Joint Conference on Lexical and Computational Semantics.

D. Goldwasser and D. Roth. 2011. Learning from natural instructions. In Proceedings of the International Joint Conference on Artificial Intelligence.

J. Krishnamurthy and T. Mitchell. 2012. Weakly supervised training of semantic parsers. In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.

T. Kwiatkowski, L.S. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.

T. Kwiatkowski, L.S. Zettlemoyer, S. Goldwater, and M. Steedman. 2011. Lexical Generalization in CCG Grammar Induction for Semantic Parsing. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.

P. Liang, M.I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In Proceedings of the Conference of the Association for Computational Linguistics.
Pawet Mazur and Robert Dale. 2010. Wikiwars: a new corpus for research on temporal expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Marc Moens and Mark Steedman. 1988. Temporal ontology and temporal reference. *Computational linguistics*, 14(2):15–28.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*.

James Pustejovsky, Bob Ingría, Roser Sauri, Jose Castano, Jessica Littman, Rob Gaizauskas, Andrea Setzer, Graham Katz, and Inderjeet Mani. 2005. The specification language timeml. *The language of time: A reader*, pages 545–557.

M. Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press.

M. Steedman. 2000. *The Syntactic Process*. The MIT Press.

Jannik Strötgen and Michael Gertz. 2012. Temporal tagging on different domains: Challenges, strategies, and gold standards. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*.

Jannik Strötgen and Michael Gertz. 2013. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology—Volume 1*.

N. Uzzaman, H. Llorens, L. Derczynski, M. Verhagen, J. Allen, and J. Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Proceedings of the International Workshop on Semantic Evaluation*.

Y.W. Wong and R.J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Conference of the Association for Computational Linguistics*.

J.M. Zelle and R.J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

L.S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

L.S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

L.S. Zettlemoyer and M. Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*. 