Language-Independent Discriminative Parsing of Temporal Expressions

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

Jakob Uszkoreit
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94303
uszkoreit@google.com

Abstract

Temporal resolution systems are traditionally tuned to a particular language, requiring significant human effort to translate them to new languages. We present a language independent semantic parser for learning the interpretation of temporal phrases given only a corpus of utterances and the times they reference. We make use of a latent parse that encodes a language-flexible representation of time, and extract rich features over both the parse and associated temporal semantics. The parameters of the model are learned using a weakly supervised bootstrapping approach, without the need for manually tuned parameters or any other language expertise. We achieve state-of-the-art accuracy on all languages in the TempEval-2 temporal normalization task, reporting a 4% improvement in both English and Spanish accuracy, and to our knowledge the first results for four other languages.

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*[1] are often more useful in their grounded form – e.g., `August 4 - August 11`.

Many approaches to this problem make use of rule-based methods, combining regular-expression matching and hand-written interpretation functions. In contrast, we would like to learn the interpretation of a temporal expression probabilistically. This allows propagation of uncertainty to higher-level components, and the potential to dynamically back off to a rule-based system in the case of low confidence parses. In addition, we would like to use a representation of time which is broadly applicable to multiple languages, without the need for language-specific rules or manually tuned parameters.

Our system requires annotated data consisting only of an input phrase and an associated *grounded* time, relative to some reference time; the language-flexible parse is entirely latent. Training data of this weakly-supervised form is generally easier to collect than the alternative of manually creating and tuning potentially complex interpretation rules.

A large number of languages conceptualize time as lying on a one dimensional line. Although the surface forms of temporal expressions differ, the basic operations many languages use can be mapped to operations on this time line (see Section 3). Furthermore, many common languages share temporal units (hours, weekdays, etc.). By structuring a latent parse to reflect these semantics, we can define a single model which performs well on multiple languages.

A discriminative parsing model allows us to define sparse features over not only lexical cues but also the temporal value of our prediction. For example, it allows us to learn that we are much more likely to express *March 14th* than *2pm in March* – despite the fact that both interpretations are composed of similar types of components. Furthermore, it allows us to define both sparse n-gram and denser but less informative bag-of-words features over multi-word phrases, and allows us to handle numbers in a flexible way.

We briefly describe our temporal representation and grammar, followed by a description of the learning algorithm; we conclude with experimental results on the six languages of the TempEval-2 A task.

[1] Spoken on, for instance, August 20.

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 83–92,
Sofia, Bulgaria, August 4-9 2013. ©2013 Association for Computational Linguistics

2 Related Work

Our approach follows the work of Angeli et al. (2012), both in the bootstrapping training methodology and the temporal grammar. Our foremost contributions over this prior work are: (i) the utilization of a discriminative parser trained with rich features; (ii) simplifications to the temporal grammar which nonetheless maintain high accuracy; and (iii) experimental results on 6 different languages, with state-of-the-art performance on both datasets on which we know of prior work.

As in this previous work, our approach draws inspiration from work on semantic parsing. The latent parse parallels the formal semantics in previous work. Supervised approaches to semantic 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. Importantly, the logical form of these parses contain all of the predicates and entities used in the parse – unlike the label provided in our case, where a grounded time can correspond to any of a number of latent parses. Along this line, recent work by Clarke et al. (2010) and Liang et al. (2011) relax supervision to require only annotated answers rather than full logical forms.

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 a strong comparison in English.

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.

In addition, there has been work on parsing Spanish expressions; UC3M (Vicente-Díez et al., 2010) produce the strongest results on the TempEval-2 corpus. Of the systems entered in the original task, TIPSem (Llorens et al., 2010) was the only system to perform bilingual interpretation for English and Spanish. Both of the above systems rely primarily on hand-built rules.

3 Temporal Representation

We define a compositional representation of time, similar to Angeli et al. (2012), but with a greater focus on efficiency and simplicity. The representation makes use of a notion of temporal *types* and their associated semantic *values*; a grammar is constructed over these types, and is grounded by appealing to the associated values.

A summary of the temporal type system is provided in Section 3.1; the grammar is described in Section 3.2; key modifications from previous work are highlighted in Section 3.3.

3.1 Temporal Types

Temporal expressions are represented either as a Range, Sequence, or Duration. The root of a parse tree should be one of these types. In addition, phrases can be tagged as a Function; or, as a special Nil type corresponding to segments without a direct temporal interpretation. Lastly, a type is allocated for numbers. We describe each of these briefly below.

Range [and Instant] A period between two dates (or times), as per an interval-based theory of time (Allen, 1981). This includes entities such as `Today`, `1987`, or `Now`.

Sequence A sequence of Ranges, occurring at regular but not necessarily constant intervals. This includes entities such as `Friday`, `November 27`th, or `last Friday`. A Sequence is defined in terms of a partial completion of calendar fields. For example, `November 27`th would define a Sequence whose year is unspecified, month is November, and day is the 27th; spanning the entire range of the lower order fields (in this case, a day). This example is illustrated in Figure 1. Note that a Sequence implicitly selects a possibly infinite number of possible Ranges.

To select a particular grounded time for a Sequence, we appeal to a notion of a *reference time* (Reichenbach, 1947). 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, and comes at the expense of certain mistakes (see Section 5.3), it is nonetheless useful in practice.

To a first approximation, grounding a sequence given a reference time corresponds to filling in the unspecified fields of the sequence with the fully-specified fields of the reference time. This pro-

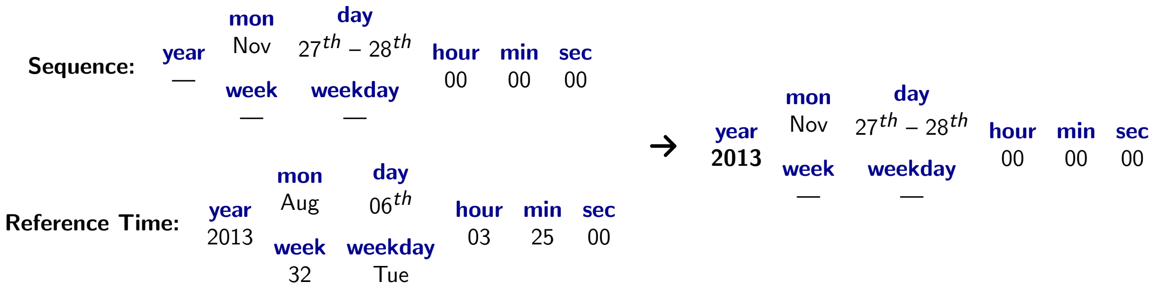


Figure 1: An illustration of grounding a Sequence. When grounding the Sequence `November 27`th with a reference time `2013-08-06 03:25:00`, we complete the missing fields in the Sequence (the year) with the corresponding field in the reference time (2013).

cess has a number of special cases not enumerated here,[2] but the complexity remains constant time.

Duration A period of time. This includes entities like `Week`, `Month`, and `7 days`. A special case of the Duration type is defined to represent *approximate* durations, 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 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 are enumerated in Table 2.

Nil A special Nil type denotes 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.

Number Lastly, a special Number type is defined for tagging numeric expressions.

3.2 Temporal Grammar

Our approach assumes that natural language descriptions of time are compositional in nature; that is, each word attached to a temporal phrase is compositionally modifying the meaning of the phrase. We define a grammar jointly over temporal *types* and *values*. The types serve to constrain the parse and allow for coarse features; the values encode specific semantics, and allow for finer features. At the root of a parse tree, we recursively apply the functions in the tree to obtain a final temporal value.

This approach can be presented as a rule-to-rule translation (Bach, 1976; Allen, 1995, p. 263), or a constrained Synchronous PCFG (Yamada and Knight, 2001).

Formally, we define our grammar as $G = (\Sigma, S, \mathcal{V}, T, \mathcal{R})$. 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. For each $v \in \mathcal{V}$ we define an (infinite) set T_v corresponding to the possible instances of type v. Each node in the tree defines a pair (v, τ) such that $\tau \in T_v$. A rule $R \in \mathcal{R}$ is defined as a pair $R = \big(v_i \rightarrow v_j v_k, f : (T_{v_j}, T_{v_k}) \rightarrow T_{v_i}\big)$. This definition is trivially adapted for the case of unary rules.

The form of our rules reveals the synchronous aspect of our grammar. The structure of the tree is bound by the first part over types v – these types are used to populate the chart, and allow for efficient inference. The second part is used to evaluate the semantics of the parse, $\tau \in T_{v_i}$, and allows partial derivations to be discriminated based on richer information than the coarse types.

We adopt the preterminals of Angeli et al. (2012). Each preterminal consists of a *type* and a *value*; neither which are lexically informed. That is, the word *week* and preterminal (`Week`, Duration) are not tied in any way. A total of 62 preterminals are defined corresponding to instances of Ranges, Sequences, and Durations; these are summarized in Table 1.

In addition, 10 functions are defined for manipulating temporal expressions (see Table 2). The majority of these mirror generic operations on intervals on a timeline, or manipulations of a sequence. Notably, like intervals, times can be

[2] Some of these special cases are caused by variable days of the month, daylight savings time, etc. Another class arises from pragmatically peculiar utterances; e.g., *the next Monday in August* uttered in the last week of August should ground to August of next year (rather than the reference time's year).

Type	Example 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`

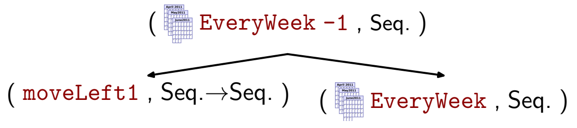
Table 1: The content-bearing preterminals of the grammar, arranged by their types. Note that the Sequence type contains more elements than enumerated here; however, only a few of each characteristic type are shown here for brevity.

Function	Description
shiftLeft	Shift a Range left by a Duration
shiftRight	Shift a Range right by a Duration
shrinkBegin	Take the first Duration of a Range
shrinkEnd	Take the last Duration of a Range
catLeft	Take the Duration after a Range
catRight	Take the Duration before a Range
moveLeft1	Shift a Sequence left by 1
moveRight1	Shift a Sequence right by 1
n^{th} x of y	Take the n^{th} element in y
approximate	Make a Duration approximate

Table 2: The functional preterminals of the grammar. The name and a brief description of the function are given; the functions are most easily interpreted as operations on either an interval or sequence. All operations on Ranges can equivalently be applied to Sequences.

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*).

Combination rules in the grammar mirror type-checked curried function application. For instance, the function `moveLeft1` applied to `week` (as in *last week*) yields a grammar rule:



In more generality, we create grammar rules for applying a function on either the left or the right, for all possible type signatures of f: $f(x, y) \odot x$ or $x \odot f(x, y)$.

Additionally, a grammar rule is created for intersecting two Ranges or Sequences, for multiplying a duration by a number, and for absorbing a Nil span. Each of these can be though of as an implicit function application (in the last case, the identity function).

3.3 Differences From Previous Work

While the grammar formalism is strongly inspired by Angeli et al. (2012), a number of key differences are implemented to both simplify the framework, and make inference more efficient.

Sequence Grounding The most time-consuming and conceptually nuanced aspect of temporal inference in Angeli et al. (2012) is intersecting Sequences. In particular, there are two modes of expressing dates which resist intersection: a day-of-month-based mode and a week-based mode. Properly grounding a sequence which defines both a day of the month and a day of the week (or week of the year) requires backing off to an expensive search problem.

To illustrate, consider the example: *Friday the 13th*. Although both a Friday and a 13th of the month are easily found, the intersection of the two requires iterating through elements of one until it overlaps with an element of the other. At training time, a number of candidate parses are generated for each phrase. When considering that these parses can become both complex and pragmatically unreasonable, this can result in a noticeable efficiency hit; e.g., during training a sentence could have a [likely incorrect] candidate interpretation of: *nineteen ninety-six Friday the 13ths from now*.

In our Sequence representation, such intersections are disallowed, in the same fashion as February 30th would be.

Sequence Pragmatics For the sake of simplicity the pragmatic distribution over possible groundings of a sequence is replaced with the single most likely offset, as learned empirically from the English TempEval-2 corpus by Angeli et al. (2012).

No Tag Splitting The Number and Nil types are no longer split according to their ordinality/magnitude and subsumed phrase, respectively.

More precisely, there is a single nonterminal (Nil), rather than a nonterminal symbol characterizing the phrase it is subsuming (Nil-*the*, Nil-*a*, etc.). This information is encoded more elegantly as features.

4 Learning

The system is trained using a discriminative k-best parser, which is able to incorporate arbitrary features over partial derivations. We describe the parser below, followed by the features implemented.

4.1 Parser

Inference A discriminative k-best parser was used to allow for arbitrary features in the parse tree. In the first stage, spans of the input sentence are tagged as either text or numbers. A rule-based number recognizer was used for each language to recognize and ground numeric expressions, including information on whether the number was an ordinal (e.g., *two* versus *second*). Note that unlike conventional parsing, a tag can span multiple words. Numeric expressions are treated as if the numeric value replaced the expression.

Each rule of the parse derivation was assigned a score according to a log-linear factor. Specifically, each rule $R = (v_i \rightarrow v_j v_k, f)$ with features over the rule and derivation so far $\phi(R)$, subject to parameters θ, is given a probability:

$$P(v_i \mid v_j, v_k, f; \theta) \propto e^{\theta^T \phi(R)} \tag{1}$$

Naïvely, this parsing algorithm gives us a complexity of $O(n^3 k^2)$, where n is the length of the sentence, and k is the size of the beam. However, we can approximate the algorithm in $O(n^3 k \log k)$ time with cube pruning (Chiang, 2007). With features which are not context-free, we are not guaranteed an optimal beam with this approach; however, empirically the approximation yields a significant efficiency improvement without noticeable loss in performance.

Training We adopt an EM-style bootstrapping approach similar to Angeli et al. (2012), in order to handle the task of parsing the temporal expression without annotations for the latent parses. Each training instance is a tuple consisting of the words in the temporal phrase, the annotated grounded time τ^*, and the reference time.

Given an input sentence, our parser will output k possible parses; when grounded to the reference time these correspond to k candidate times: $\tau_1 \dots \tau_k$, each with a normalized probability $P(\tau_i)$. This corresponds to an approximate E step in the EM algorithm, where the distribution over latent parses is approximated by a beam of size k. Although for long sentences the number of parses is far greater than the beam size, as the parameters improve, increasingly longer sentences will have correct derivations in the beam. In this way, a progressively larger percentage of the data is available to be learned from at each iteration.

To approximate the M step, we define a multiclass hinge loss $l(\theta)$ over the beam, and optimize using Stochastic Gradient Descent with AdaGrad (Duchi et al., 2010):

$$l(\theta) = \max_{0 \le i < k} \mathbb{1}[\tau_i \neq \tau^*] + P_\theta(\tau_i) - P_\theta(\tau^*) \tag{2}$$

We proceed to describe our features.

4.2 Features

Our framework allows us to define arbitrary features over partial derivations. Importantly, this allows us to condition not only on the PCFG probabilities over *types* but also the partial semantics of the derivation. We describe the features used below; a summary of these features for a short phrase is illustrated in Figure 2.

Bracketing Features A feature is defined over every nonterminal combination, consisting of the pair of children being combined in that rule. In particular, let us consider a rule $R = (v_i \rightarrow v_j v_k, f)$ corresponding to a CFG rule $v_i \rightarrow v_j v_k$ over *types*, and a function f over the semantic values corresponding to v_j and v_k: τ_j and τ_k. Two classes of bracketing features are extracted: features are extracted over the types of nonterminals being combined (v_j and v_k), and over the top-level semantic derivation of the nonterminals (f, τ_j, and τ_k).

Unlike syntactic parsing, child types of a parse tree uniquely define the parent type of the rule; this is a direct consequence of our combination rules being functions with domains defined in terms of the temporal types, and therefore necessarily projecting their inputs into a single output type. Therefore, the first class of bracketing features – over types – reduce to have the exact same expressive power as the nonterminal CFG rules of Angeli et al. (2012). Examples of features in this class are features 13 and 15 in Figure 2 (b).

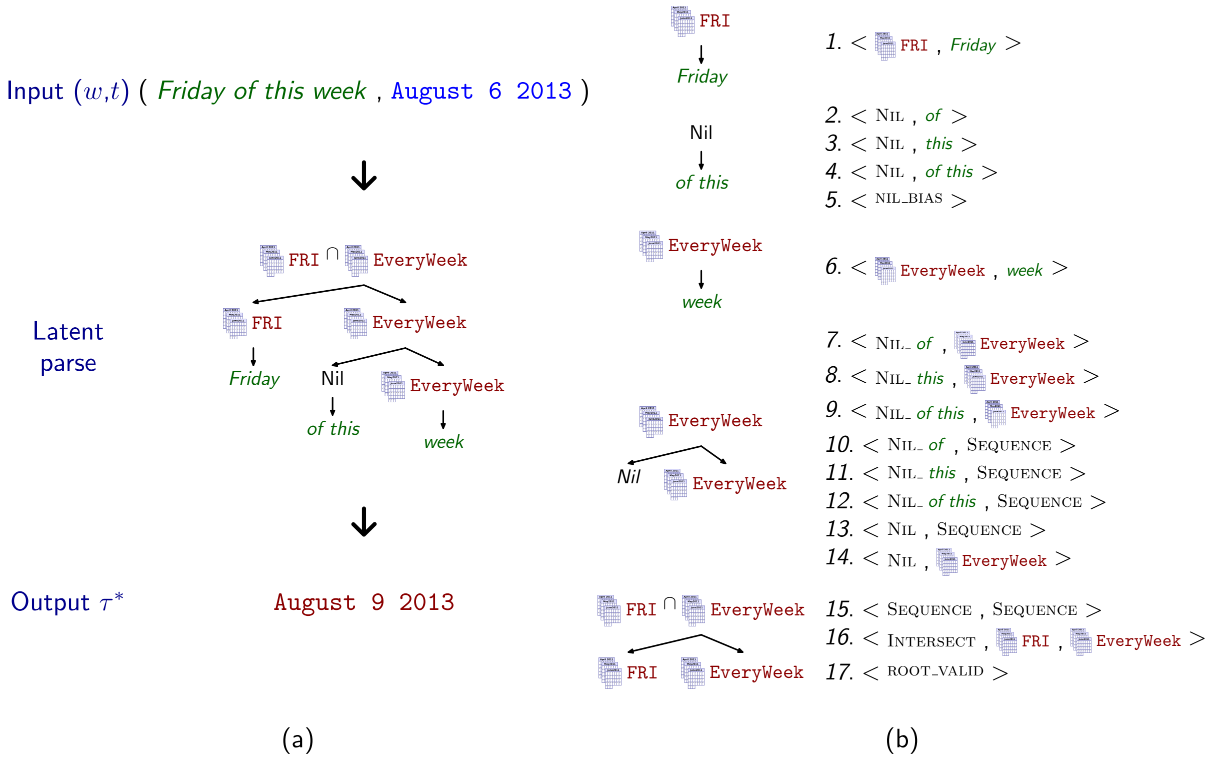


Figure 2: An example parse of *Friday of this week*, along with the features extracted from the parse. A summary of the input, latent parse, and output for a particular example is given in (a). The features extracted for each fragment of the parse are given in (b), and described in detail in Section 4.2.

We now also have the flexibility to extract a second class of features from the semantics of the derivation. We define a feature bracketing the most recent semantic function applied to each of the two child derivations; along with the function being applied in the rule application. If the child is a preterminal, the semantics of the preterminal are used; otherwise, the outermost (most recent) function to be applied to the derivation is used. To illustrate, a tree fragment combining `August` and `2013` into `August 2013` would yield the feature <INTERSECT, AUGUST, 2013>. This can be read as a feature for the rule applying the intersect function to August and 2013. Furthermore, intersecting `August 2013` with the 12*th* of the month would yield a feature <INTERSECT, INTERSECT, 12*th*>. This can be read as applying the intersect function to a subtree which is the intersection of two terms, and to the 12*th* of the month. Features 14 and 16 in Figure 2 (b) are examples of such features.

Lexical Features The second large class of features extracted are lexicalized features. These are primarily used for tagging phrases with preterminals; however, they are also relevant in incorporating cues from the yield of Nil spans. To illustrate, *a week* and *the week* have very different meanings, despite differing by only their Nil tagged tokens.

In the first case, a feature is extracted over the *value* of the preterminal being extracted, and the phrase it is subsuming (e.g., features 1–4 and 6 in Figure 2 (b)). As the type of the preterminal is deterministic from the value, encoding a feature on the type of the preterminal would be a coarser encoding of the same information, and is empirically not useful in this case. Since a multi-word expression can parse to a single nonterminal, a feature is extracted for the entire n-gram in addition to features for each of the individual words. For example, the phrase *of this* – of type Nil – would have features extracted: <NIL, *of*>, <NIL, *this*>, and <NIL, *of this*>.

In the second case – absorbing Nil-tagged spans – we extract features over the words under the Nil span joined with the type and value of the other derivation (e.g., features 7–12 in Figure 2 (b)). As above, features are extracted for both n-grams and for each word in the phrase. For example, combining *of this* and *week* would yield features

	Train		Test	
System	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
ParsingTime	0.90	0.72	0.88	0.72
OurSystem	**0.94**	**0.81**	0.91	**0.76**

Table 3: English results for TempEval-2 attribute scores for our system and four previous systems. The scores are calculated using gold extents, forcing an interpretation for each parse.

	Train		Test	
System	Type	Value	Type	Value
UC3M	—	—	0.79	0.72
OurSystem	0.90	0.84	**0.92**	**0.76**

Table 4: Spanish results for TempEval-2 attribute scores for our system and the best known previous system. The scores are calculated using gold extents, forcing an interpretation for each parse.

<*of*, EVERYWEEK>, <*this*, EVERYWEEK>, and <*of this*, EVERYWEEK>.

In both cases, numbers are featurized according to their order of magnitude, and whether they are ordinal. Thus, the number tagged from *thirty-first* would be featurized as an ordinal number of magnitude 2.

Semantic Validity Although some constraints can be imposed to help ensure that a top-level parse will be valid, absolute guarantees are difficult. For instance, February 30 is never a valid date; but, it would be difficult to disallow any local rule in its derivation. To mediate this, an indicator feature is extracted denoting whether the grounded semantics of the derivation is valid. This is illustrated in Figure 2 (b) by feature 17.

Nil Bias Lastly, an indicator feature is extracted for each Nil span tagged (feature 5 in Figure 2 (b)). In part, this discourages over-generation of the type; in another part, it encourages Nil spans to absorb as many adjacent words as possible.

We proceed to describe our experimental setup and results.

5 Evaluation

We evaluate our model on all six languages in the TempEval-2 Task A dataset (Verhagen et al., 2010), comparing against state-of-the-art systems for English and Spanish. New results are reported on smaller datasets from the four other languages. To our knowledge, there has not been any prior work on these corpora.

We describe the TempEval-2 datasets in Section 5.1, present experimental results in Section 5.2, and discuss system errors in Section 5.3.

5.1 TempEval-2 Datasets

TempEval-2, from SemEval 2010, focused on retrieving and reasoning about temporal information from newswire. Our system evaluates against Task A – detecting and resolving temporal expressions. Since we perform only the second of these, we evaluate our system assuming gold detection.

The dataset annotates six languages: English, Spanish, Italian, French, Chinese, and Korean; of these, English and Spanish are the most mature. We describe each of these languages, along with relevant quirks, below:

English The English dataset consists of 1052 training examples, and 156 test examples. Evaluation was done using the official evaluation script, which checks for exact match between `TIMEX3` tags. Note that this is stricter than our training objective; for instance, *24 hours* and *a day* have the same interpretation, but have different `TIMEX3` strings. System output was heuristically converted to the `TIMEX3` format; where ambiguities arose, the convention which maximized training accuracy was chosen.

Spanish The Spanish dataset consists of 1092 training examples, and 198 test examples. Evaluation was identical to the English, with the heuristic `TIMEX3` conversion adapted somewhat.

Italian The Italian dataset consists of 523 training examples, and 126 test examples. Evaluation was identical to English and Spanish.

Chinese The Chinese dataset consists of 744 training examples, and 190 test examples. Of these, only 659 training and 143 test examples had a temporal value marked; the remaining examples had a type but no value, and are therefore impossible to predict. Results are also reported on a clean corpus with these impossible examples omitted.

The Chinese, Korean, and French corpora had noticeable inconsistencies in the `TIMEX3` annotation. Thus, evaluations are reported according

	Train			Test		
Language	# examples	Type	Value	# examples	Type	Value
English	1052	0.94	0.81	156	0.91	0.76
Spanish	1092	0.90	0.84	198	0.92	0.76
Italian	523	0.89	0.85	126	0.84	0.38
Chinese†	744	0.95	0.65	190	0.87	0.48
Chinese (clean)†	659	0.97	0.73	143	0.97	0.60
Korean†	247	0.83	0.67	91	0.82	0.42
French†	206	0.78	0.76	83	0.78	0.35

Table 5: Our system's accuracy on all 6 languages of the TempEval-2 corpus. Chinese is divided into two results: one for the entire corpus, and one which considers only examples for which a temporal value is annotated. Languages with a dagger (†) were evaluated based on semantic rather than string-match correctness.

to the training objective: if two TIMEX3 values ground to the same grounded time, they are considered equal. For example, in the example above, *24 hours* and *a day* would be marked identical despite having different TIMEX3 strings.

Most TIMEX3 values convert naturally to a grounded representation; values with wildcards representing Sequences (e.g., 1998-QX or 1998-XX-12) ground to the same value as the Sequence encoding that value would. For instance, 1998-QX is parsed as *every quarter in 1998*.

Korean The Korean dataset consists of 287 training examples, and 91 test examples. 40 of the training examples encoded dates as a long integer For example: 003000000200001131951006 grounds to January 13, 2000 at the time 19:51. These were removed from the training set, yielding 247 examples; however, all three such examples were left in the test set. Evaluation was done identically to the Chinese data.

French Lastly, a dataset for French temporal expressions was compiled from the TempEval-2 data. Unlike the other 5 languages, the French data included only the raw TIMEX3 annotated newswire documents, encoded as XML. These documents were scraped to recover 206 training examples and 83 test examples. Evaluation was done identically to the Chinese and Korean data.

We proceed to describe our experimental results on these datasets.

5.2 Results

We compare our system with state-of-the-art systems for both English and Spanish. To the best of our knowledge, no prior work exists for the other four languages.

We evaluate in the same framework as Angeli et al. (2012). 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 for those systems, as the system is not allowed to abstain on expressions it does not recognize.

The systems compared against are:

- GUTime (Mani and Wilson, 2000), a widely used, older rule-based system.
- HeidelTime (Strötgen and Gertz, 2010), the top system at the TempEval-2 task for English.
- SUTime (Chang and Manning, 2012), a more recent rule-based system for English.
- ParsingTime (Angeli et al., 2012), a semantic parser for temporal expressions, similar to this system (see Section 2).
- UC3M (Vicente-Díez et al., 2010), a rule-based system for Spanish.

Results for the English corpus are shown in Table 3. Results for Spanish are shown in Table 4. Lastly, a summary of results in all six languages is shown in Table 5.

A salient trend emerges from the results – while training accuracy is consistently high, test accuracy drops sharply for the data-impoverished languages. This is consistent with what would be expected from a discriminatively trained model in data-impoverished settings; however, the consistent training accuracy suggests that the model nonetheless captures the phenomena it sees in

Error Class	English	Spanish
Pragmatics	29%	23%
Type error	16%	5%
Incorrect number	10%	5%
Relative Range	7%	2%
Incorrect parse	19%	36%
Missing context	16%	23%
Bad reference time	3%	6%

Table 6: A summary of errors of our system, by percentage of incorrect examples for the English and Spanish datasets. The top section describes errors which could be handled in our framework, while the bottom section describes examples which are either ambiguous (missing context), or are annotated inconsistently relative the reference time.

training. This suggests the possibility for improving accuracy further by making use of more data during training.

5.3 Discussion

We characterize the examples our system parses incorrectly on the English and Spanish datasets in Table 6, expanding on each class of error below.

Pragmatics This class of errors is a result of pragmatic ambiguity over possible groundings of a sequence – for instance, it is often ambiguous whether *next weekend* refers to the coming or subsequent weekend. These errors manifest in either dropping a function (*next*, *last*), or imagining one that is not supported by the text (e.g., *this week* parsed as next week).

Type error Another large class of errors – particularly in the English dataset – arise from not matching the annotation's type, but otherwise producing a reasonable response. For instance, the system may mistake a day on the calendar (a Range), with a day, the period of time.

Incorrect number A class of mistakes arises from either omitting numbers from the parse, or incorrectly parsing numbers – the second case is particularly prevalent for written years, such as *seventeen seventy-six*.

Relative Range These errors arise from attempting to parse a grounded Range by applying functions to the reference time. For example, from a reference time of August 8th, it is possible to "correctly" parse the phrase *August 1* as a week ago; but, naturally, this parse does not generalize well. This class of errors, although relatively small, merits special designation as it suggests a class of correct responses which are correct for the wrong reasons. Future work could explore mitigating these errors for domains where the text is further removed from the events it is describing than most news stories are.

Incorrect parse Errors in this class are a result of failing to find the correct parse, for a number of reasons not individually identified. A small subset of these errors (notably, 6% on the Spanish dataset) are a result of the grammar being insufficiently expressive with the preterminals we defined. For instance, we cannot capture fractional units, such as in *half an hour*.

Missing context A fairly large percentage of our errors arise from failing to classify inputs which express ambiguous or poorly defined times. For example, *from time to time* (annotated as the future), or *that time* (annotated as 5 years). Many of these require either some sort of inference or a broader understanding of the context in which the temporal phrase is uttered, which our system does not attempt to capture.

Bad reference time The last class of errors cover cases where the temporal phrase is clear, but annotation differs from our judgment of what would be reasonable. These are a result of assuming that the reference time of an utterance is the publication time of the article.

6 Conclusion

We have presented a discriminative, multilingual approach to resolving temporal expressions, using a language-flexible latent parse and rich features on both the *types* and *values* of partial derivations in the parse. We showed state-of-the-art results on both languages in TempEval-2 with prior work, and presented results on four additional languages.

Acknowledgments Work was done in the summer of 2012 while the first author was an intern at Google. We would like to thank Chris Manning, and our co-workers at Google for their insight and help.

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.

Gabor Angeli, Christopher D. Manning, and Daniel Jurafsky. 2012. Parsing time: Learning to interpret time expressions. In *NAACL-HLT*.

E. Bach. 1976. An extension of classical transformational grammar. In *Problems of Linguistic Metatheory (Proceedings of the 1976 Conference)*, Michigan State University.

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

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

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.

John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

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.

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.

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.

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

Hector Llorens, Estela Saquete, and Borja Navarro. 2010. Tipsem (english and spanish): Evaluating crfs and semantic roles in tempeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 284–291.

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

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.

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.

María Teresa Vicente-Díez, Julián Moreno Schneider, and Paloma Martínez. 2010. Uc3m system: Determining the extent, type and value of time expressions in tempeval-2. In *proceedings of the Semantic Evaluation–2 (Semeval 2010), ACL Conference, 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.