# Towards Task-Based Temporal Extraction and Recognition

David Ahn, Sisay Fissaha Adafre and Maarten de Rijke

Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
(ahn|sfissaha|mdr)@science.uva.nl

**Abstract.** We seek to improve the robustness and portability of temporal information extraction systems by incorporating data-driven techniques. We present two sets of experiments pointing us in this direction. The first shows that machine-learning-based *recognition* of temporal expressions not only achieves high accuracy on its own but can also improve rule-based *normalization*. The second makes use of a staged normalization architecture to experiment with machine learned classifiers for certain disambiguation sub-tasks within the normalization task.

**Keywords.** Information extraction, natural language, temporal reasoning, text mining

## 1 Introduction

Current information retrieval (IR) systems allow us to locate documents that might contain pertinent information, but most of them leave it to the user to extract useful information from a ranked list. This leaves the user with a large amount of text to consume. *Information extraction* [2] (IE) is a core technology to help reduce the amount of text that has to be read to obtain the desired information. Indeed, recognizing entities and meaningful relations between them is key to providing focused information access. *Temporal* IE provides a particularly interesting task in this respect. Temporal expressions (*timexes*) are natural language phrases that refer directly to time points or intervals. They convey temporal information on their own and also serve as anchors for events referred to in text. From a user's perspective, temporal aspects of events and entities, and of text snippets, provide a natural mechanism for organizing information. An example of an area in which accurate analysis of temporal expressions plays an important role is Question Answering (QA). For instance, answering questions like *"When was Van Gogh born?"* requires accurate identification of the date of birth of the person under consideration (*recognition*) and rendering of the answer in some standard format (*normalization*). Recognizing temporal expressions is now a "do-able" task, even without tremendous knowledge engineering efforts. Moreover, in recent years, the task of automatically interpreting (or normalizing) temporal expressions has begun to receive attention [11,18].

Dagstuhl Seminar Proceedings 05151
Annotating, Extracting and Reasoning about Time and Events
http://drops.dagstuhl.de/opus/volltexte/2005/315

The importance of processing timexes is reflected by the large number of NLP evaluation efforts where they figure. Recognizing timexes is an intergral part of many IE tasks (e.g., MUC-6 and 7 Named Entity Recognition tasks, ACE-2004 Event Recognition task). There are various annotation guidelines for timexes [7,19,14]. And a timex annotated corpus has been released with the aim of improving the processing of timexes [22].

The type of timexes considered in a typical IE task are limited to date and time values [13,4]. In contrast, at the 2004 Temporal Expression Recognition and Normalization [21] evaluation, a wide variety of timexes are considered—which makes the task more interesting and much more challenging. The participants in the 2004 TERN evaluation evinced a notable split in their approaches to temporal IE: systems performing only recognition were all machine-learning-based, while systems performing the full recognition and normalization task were purely rule-based. The message from the recognition task was clear: given a tagged corpus, machine learning provides excellent recognition results with minimal human intervention. Furthermore, such a data-driven approach may be preferable because of its portability and robustness.

For the normalization task, it is not so obvious how to directly apply machine-learning methods based on sequence labeling. The classes involved (temporal values) are potentially unbounded, and a significant proportion of timexes require non-local context for interpretation. Additionally, many timexes require temporal computation with respect to contextually given information—the connection between form and content is mediated by both context and world knowledge.

What are the opportunities for using robust, "shrink-wrapped" machine-learning tools for temporal IE? Ultimately, we want to build a portable, maintainable system that can serve us well as a component for more high-level tasks. To this end, we want to make as much use of off-the-shelf machine-learning packages as possible. We would also like to limit the scope of rule application, in order to simplify rule-writing and maintenance. In this paper, we describe two sets of experiments that bring us closer to this goal. In the first set, we demonstrate that decoupling recognition from normalization—feeding a rule-based normalizer with an automatically learned recognizer—can improve overall performance. In the second set, we decompose the normalization task, allowing us to find opportunities for applying data-driven methods for disambiguation within normalization.

## 2 Background

### 2.1 The TERN setting

For the experiments we report on in this paper, we adopt the tasks, data, and evaluation methodology of the 2004 TERN evaluation [21]. The TERN evaluation is organized under the auspices of the Automatic Content Extraction program (ACE, http://www.nist.gov/speech/tests/ace/), whose objective is to develop natural language processing technology to support automatic understanding of textual data. TERN consists of two tasks: recognition

and normalization. Timex *recognition* involves correctly detecting and delimiting timexes in text. *Normalization* involves assigning recognized timexes a fully qualified temporal value. Both tasks are defined, for human annotators, in the TIDES TIMEX2 annotation guidelines [7]. These introduce an SGML element, TIMEX2, to mark timexes. TIMEX2 elements may contain a number of attributes; we focus on the VAL attribute, which indicates the actual reference of the TIMEX2. Its range of values are an extension of the ISO 8601 standard for representing time [8].

The recognition and normalization tasks are performed with respect to corpora of transcribed broadcast news speech and news wire texts from ACE 2002–4, marked up in SGML format and hand-annotated for TIMEX2s. The training and test sets for the TERN evaluation and for all of the experiments we describe in this paper consist of 511 and 192 documents with 5326 and 1828 TIMEX2s, respectively.

We use the TERN official scorer to evaluate our performance but add several metrics. The official scorer computes precision, recall, and F-measure for identification of TIMEX2s (overlap between a gold standard and a system TIMEX2) and exact-match of TIMEX2s (exact overlap) and for the attributes; since we focus on VALs, we report only VAL scores. For VALs, the scorer computes precision as the ratio of correct VALs to attempted VALs and recall as the ratio of correct VALs to possible VALs *in the TIMEX2s recognized by the system*—not all TIMEX2s in the gold standard. Since we are interested in the end-to-end task, we report results with recall (and F-measure) computed with respect to *all* possible TIMEX2s. We refer to the TERN versions as *relative recall (RR)* and *F (RF)*, and to our versions as *absolute recall (AR)* and *F (AF)*.

### 2.2 Recognition

The recognition task is to identify phrases that refer to time points. The TIDES guidelines limit the set of markable timexes (indicated with the TIMEX2 tag) to those phrases headed by a temporal trigger word. The latter seem to fall into several categories. Some refer to time units of definite duration (minute, afternoon, day, night, weekend, month, summer, season, quarter, year, decade, century, millennium, era, semester). Others refer to definite points in time (January, Monday, New Year's Eve, Washington's Birthday, yesterday, today, tomorrow, midnight). Still others indicate repetition with respect to a definite period (daily, monthly, biannual, semiannual, hourly, daily, monthly, ago). And some refer to temporal concepts that can at least be oriented on a timeline with respect to some definite time point (future, past, time, period, point, recent, former, current, ago, currently, lately).

Syntactically, TIMEX2s must be one of the following: noun, noun phrase, adjective, adverb, adjective phrase, or adverb phrase. All premodifiers and postmodifiers of the timex must be included in the extent of the TIMEX2 tag, e.g.,

– Premodifiers: *8* winters, *the past* week, *four bad* years, *about 15* minutes, *less than a* week

– Postmodifiers: *Nearly* three years *later*, the week *before last*, two years *ago*, three years *in prison*, Only days *after his father was assassinated*, months *of Israeli-Palestinian bloodshed and Israeli blockades*

Either rule-based systems or machine learning can be used for recognition, but as we saw in the 2004 TERN evaluation, all the recognition-only systems (which generally achieved better recognition scores than the full-task systems) were machine-learning-based. The reasons for this are clear: recognition can easily be cast as a sequence-labeling task, for which good machine-learning systems exist. Unsurprisingly, the recognition components of the full-task systems were all rule-based. Normalization is most straightforwardly conceived of as interpreting recognition rules. Thus, it is natural to develop a single set of rules to be used for recognition and paired with interpretation functions for normalization. However, the result is a monolithic rule-based system, which requires significant engineering efforts to build and maintain.

In this paper we explore the possibility of breaking down the task into smaller pieces and using the best methods possible for each sub-task in an effort to develop a more robust solution to the temporal IE task. The first step, described in §3, is to decouple recognition from normalization. Experimental results indicate that this decoupling (which both improves recognition and allows a liberalization of the normalization rules) improves overall, end-to-end performance.

### 2.3 Normalization

Timex normalization is the problem of assigning an ISO 8601 value to a recognized timex. The TIDES guidelines distinguish several kinds of values; our normalization system handles time points, durations, sets, and the past, present, and future tokens. Time points are expressed by three kinds of timexes: fully qualified, deictic, and anaphoric. Fully qualified timexes, such as *March 15, 2001*, can be normalized without reference to any other temporal entities. Deictic and anaphoric timexes, on the other hand, must be interpreted relative to another temporal entity. Deictic timexes, such as *today*, *yesterday*, *three weeks ago*, *last Thursday*, *next month*, are interpreted with respect to the time of utterance—for our corpus, the document time stamp. Anaphoric timexes, such as *March 15*, *the next week*, *Saturday*, are interpreted with respect to a reference time—a salient time point previously evoked in the text that may shift as the text progresses. Some anaphoric timexes (those without an explicit direction indicator such as *next* or *previous*) depend on other factors, such as the tense and aspect of the verb they modify, to determine whether the time point they refer to is before or after the reference time.

Durations are generally expressed by timexes headed by a unit (*day*, *months*, etc.). However, even fully qualified timexes expressing durations, i.e., those in which both the quantity and the unit are specified, such as *six months*, *800 years*, *three long days*, are systematically ambiguous between a duration and an anaphoric point reading. For example, in the sentence *The Texas Seven hid out there for three weeks*, the timex *three weeks* refers to a duration, whereas in the

sentence *California may run out of cash in three weeks*, the same timex refers to a point three weeks after the reference point.

Work on normalization with respect to TIMEX2-like guidelines goes back to [11], who use a rule-based system to identify and normalize timexes. Like our work in §4 (but unlike more recent work on normalization), they also use an automatically learned classifier within their system, but only for one task—distinguishing specific and generic uses of *today*. Unlike our work, however, they restrict normalization to date-valued expressions; the question of distinguishing points and durations, e.g., does not arise. Other rule-based normalization systems include [16,17], as well as the TERN full-task systems. Building on the approach in [11], we present in §4 a modular timex normalization system architecture that allows us to separate context-independent interpretation, for which we continue to use a rule-based approach, from context-dependent processing.

## 3 Decoupling recognition from normalization

Here, we describe a set of experiments (reported on in greater detail in [1]) in which we run the same rule-based timex normalizer on the output of several different timex recognizers. Our basic monolithic rule-based system is a two-pass system. In the first pass, a document is tokenized, POS tagged and chunked using TreeTagger [23], and then a series of regular expressions is used to find timexes. In the second pass, the document time stamp and all sentences containing TIMEX2s are extracted; the tense of each sentence (based on the first verb chunk) is also determined. Interpretation rules paired with the regular expressions for recognition are used to generate normalized values for each TIMEX2. Anaphoric and deictic expressions are evaluated with respect to the timestamp and tense information.

**Recognition with Conditional Random Fields** A recently-introduced machine learning technique for labeling and segmenting sequence data is Conditional Random Fields (CRFs, [9]). Unlike Hidden Markov Models, CRFs are based on exponential models in which probabilities are computed based on the values of a set of features induced from both the observation and label sequences. They have been used in POS tagging, shallow parsing [20], and named entity recognition [12]. We use the minorThird implementation of CRFs for extracting timexes from text [5]. Initial recognition results with the default features are: 98%/84%/91% (P/R/F) for identification and 80%/66%/74% (P/R/F) for exact-match. An error analysis suggested several additional features which resulted in substantial performance improvements: 98%/86%/91% (P/R/F) for identification and 86%/75%/80% (P/R/F) for exact-match.

**Decoupling experiments** We ran our normalizer on the output of two different recognition systems—the rule-based recognizer of our monolithic system and the optimized CRF recognizer—and on the gold standard, in order to see the effects of recognition performance on normalization performance. Our expectation was that since the normalization rules are basically identical to the rule-based recognizer, any additional recognized timexes would not be normalized anyhow.

| | Correct | Incorrect | P | RR | RF | AR | AF |
|---|---|---|---|---|---|---|---|
| Rule-based | 782 | 143 | 0.845 | 0.699 | 0.765 | 0.449 | 0.586 |
| CRF | 885 | 231 | 0.793 | 0.583 | 0.672 | 0.501 | 0.614 |
| Gold standard | 955 | 219 | 0.812 | 0.549 | 0.655 | 0.549 | 0.655 |

**Table 1.** Normalization results

Table 1 lists the results of these three runs on the TERN test corpus. For our purposes, AR and AF are more important than RR and RF. The results indicate that better recognition does help normalization. We analyzed the gold standard and rule-based recognizer runs to determine why. The former obviously presents more timexes to the normalizer than the latter; the question is why, given that the patterns for the normalizer and the rule-based recognizer are the same, the normalizer attempts these extra timexes. The two main reasons are the reliance of the rule-based recognizer on unreliable upstream components (tokenizer, tagger, and chunker) and our liberalization of the normalizer rules.

## 4 Zooming in on normalization

From the perspective of the end-to-end task of temporal IE, our experiments so far suggest that it is worthwhile to optimize recognition and normalization independently. By decoupling normalization from recognition, not only do we allow for independent optimization of recognition, but we also give ourselves the opportunity to conceive of normalization as an independent task—rules do not need to serve double-duty for both recognition and normalization. We now delve into normalization, exploring one way of decomposing normalization both to simplify the rule set and to bring in data-driven methods.

In the rest of this section, we lay out our analysis of the task and explain how we tackle the various sub-tasks. In §5, we focus on the machine-learning experiments we perform as part of normalization. We devote §6 to describing our end-to-end normalization experiments.

### 4.1 Decomposing the normalization task

We seek a robust, maintainable approach to the normalization task, one that allows for the use of data-driven techniques, where appropriate, and circumscribes the scope of rule application to simplify rule development. To that end, we decompose the task as described above into five discrete stages:

1. Lexical lookup: mapping names to numbers, units to ISO values, etc.
2. Context-independent composition: combining the values of the lexical tokens within a timex to produce a context-independent semantic representation.
3. Context-dependent classification: determining whether a timex is a point or duration, looks forward or backward, makes specific or generic reference, etc.
4. Reference time, or temporal focus, tracking: for anaphoric timexes, whose values must be computed with respect to a reference time.

5. Final computation: combining the results of all of these steps to produce a final normalized value.

This architecture clearly separates context-independent processing (stages 1 and 2), for which finite-state rules can be relatively easily developed, from context-dependent processing (stages 3 and 4), for which finite-state rule sets can quickly become unwieldy. It also distinguishes context-dependent classification tasks, which rely on primarily local context, from reference time tracking, which requires more global information. The final stage makes use only of meta-data produced by the earlier stages and does no linguistic processing.

In an error analysis of a purely rule-based normalization system, we have observed observed that classification of timexes (stage 3) is a significant source of error [1]. Now, classification of expressions into a limited set of classes using local context is exactly the kind of task at which machine-learning-based classifiers excel. Thus, this separation of tasks provides us with an opportunity to deploy an off-the-shelf machine learning package within the normalization task.

### 4.2 Addressing the interpretation stages

We use a rule-based system to handle stages 1 and 2 (which we refer to as *pre-normalization*). Timex lexicons and composition mechanisms may be learned, but our lexicon and composition rule set are relatively small and unambiguous, so writing and maintaining them is more or less straightforward. In addition to generating a context-independent representation for a timex, the composition mechanism also determines whether the timex is ambiguous in a way that can be resolved by one of the context-dependent classifiers.

Stage 3 is where we study the potential contribution of data-driven methods to timex normalization. We make use of a maximum entropy classifier to perform context-dependent classification. Based on the error analysis of [1], we have isolated three classification tasks that contribute to the errors their rule-based system makes and seem amenable to machine learning from relatively local surface features:

- The first task is distinguishing whether an ambiguous unit phrase refers to a point or a duration; we refer to this as the *point-duration problem*.
- The second task (*direction problem*) is determining whether an ambiguous anaphoric point-referring phrase refers to a point before, after, or the same as the reference time. Some of the instances of this class are generated by the first classifier.
- The third task (the *today problem*) is determining whether an occurrence of the word *today* refers specifically to the day of the article or broadcast or generically to the present.

Section 5 is devoted to a detailed description of the methods used to tackle these tasks.

As to stage 4, we experiment with two very simple models of temporal focus tracking. In the first, the system uses the document time stamp as the reference

time for all anaphoric expressions. This is an oversimplification of the problem of temporal focus, but it seems to be reasonable at least for day names in short news items. In the second temporal focus model, the system uses the most recent previous point-referring timex of suitable granularity as the reference time for an anaphoric expression. This, too, is a simplifying assumption, since it ignores the effects of discourse structure on focus tracking, among other things. (In both cases, deictic expressions, such as *tomorrow* and *three years ago*, are still computed with respect to the document time stamp).

Finally, we take a rule-based approach to stage 5. Temporal arithmetic is performed to derive a fully qualified temporal value from the context-independent value and the reference time of a timex, together with information from the context-dependent classifiers. Of the five stages of normalization, this stage is least obviously amenable to machine learning.

## 5 Developing the Classification Experiments

We now describe our machine-learning approach to the three classification tasks that make up stage 3 of our strategy for the overall normalization task. We use a maximum entropy classifier for our experiments [3]. Specifically, we use the minorthird implementation of maximum entropy classifiers, which uses the same underlying model as CRFs applied to sequence labeling [5].

### 5.1 Generating the Training Data

The training material is a tagged corpus consisting of plain text in which the timexes to be classified are marked by XML tags which encode the classes, e.g., `...<forward><dir_unknown>Tuesday</dir_un known></forward>...`. The inner XML tag marks a timex as an instance to be used for training while the outer XML tag assigns a class to the instance. The task is to learn from these example timex instances in the training corpus rules or patterns to classify new instances.

To generate the training data, we use the output of our pre-normalization stages, which tags ambiguous time unit phrases (for the point-duration task), ambiguous anaphoric timexes (for the direction task), and occurrences of *today* (for the *today* task). Not all unit phrases are ambiguous (e.g., *two years ago* is always point-referring), nor are all anaphoric timexes ambiguous with regards to direction (e.g., *a month later* is always forward-looking), so what counts as an instance for these problems is dependent on our pre-normalization.

Given that, generating training data for the point-duration and *today* tasks is straightforward, since the classes are reflected directly in the final normalized values. The final normalized value of a duration always begins with a `P` followed by a quantity indication, whereas that of a point begins with a digit. Similarly, occurrences of *today* that make generic reference are normalized as `PRESENT_REF`, while specifically referring occurrences have full date values.

Generating training data for the direction task, is non-trivial as it assumes a model of temporal focus tracking. We produced two training sets for the task,

based on the two focus models our normalization system can use: timestamp-based and recency-based (see §4.2). As both are simplistic models, the generated training data, in either case, is noisy.

### 5.2 The Point-Duration Problem

For the *point-duration task*, ambiguous time unit phrases need to be classified into three classes: point, duration and other. Using only lexical features, both in the timex itself and in the left and right context (window of 3 words), the system achieves an accuracy of 0.73 (frequency baseline: 0.40).

### 5.3 The Direction Problem

Confronted with the *direction problem* for day names in their date normalization system, Mani and Wilson [11] use hand-crafted rules which look at the tense of the closest verb in the same clause as a timex to determine the direction. The rule-based direction classifier we use in our experiments in §6 is based on this method, but here, we describe our machine learning approach. Tense alone cannot be used to identify direction accurately (see our error analysis in §6), so additional features, such as lexical items such as *last* and *earlier*, need to be used for the learning algorithm to produce a reasonable result. Using only such lexical features as these, with a context window of 3 words, the system, using the timestamp-based dataset, has an accuracy of 0.59 (frequency baseline: 0.44). Adding tense (derived from the POS tags of the closest finite verbs) as a feature improves the result by 0.02 (to 0.61). Using the same features with the recency-based data yields an accuracy of 0.57 (we discuss the difference in §6).

A closer look at the confusion matrix indicates that the classifier has problems in distinguishing between the *same* and *backward* classes. More than 30% of each class is assigned to the other class. These are the main sources of errors since the two classes constitute 89% of the total test instances.

### 5.4 The Generic-Specific Problem

Mani and Wilson [11] also report that ambiguity resulting from generic vs. specific meaning of timexes are a main source of error. They single out the timex *today*, which is most subject to this ambiguity, and automatically acquire a classifier for it. Their best system achieves an accuracy of 0.80. We have developed a similar classifier for our system using the features they propose. The resulting classifier achieves an accuracy of 0.85 in a 70/30 split experiment on the training data. Unfortunately, our data for this classifier is not only sparse, but heavily skewed (90% of the instances are specific). Thus, our system still underperforms the frequency baseline 0.89. In the experiments we describe in §6, we use the baseline classifier for this task.

### 5.5 Reflections

In general, all of these classification tasks are more difficult than the recognition task, where machine-learning approaches achieve very good results. Although two of the systems are better than the frequency baseline, there is a lot of room for improvement. Simple lexical features, which are successful in the recognition task, do not have the same impact on these classification tasks. This suggests a potential gain by providing the learner with more semantically motivated features. For now, though, we want to see how the results from relatively straightforward application of off-the-shelf machine learning affects our normalization task.

## 6 Experiments

We now describe the experiments we performed in trying to answer our main research question: whether data-driven methods can be successfully applied to the timex normalization task. We describe the approaches we compared, then the metrics used, the results, and, finally, an error analysis.

### 6.1 Experimental Setup

The system we used for our experiments consists of the following components. For the pre-normaliation stages (1 and 2), there are regular-expression grammars written in the JAPE formalism that run within the GATE system [6]. For stage 3, there are three kinds of classifiers for the point-duration and direction tasks: baseline classifiers (which assign the majority class), MaxEnt classifiers, described in §5 and rule-based classifiers, briefly described below. Also, there is the baseline classifier for the *today* task. For stages 4 and 5, there is a perl script (using the Time::Piece module) that traverses a document, tracking reference times and computing final normalized values.

The JAPE grammars include both simple (stage 1) grammars to normalize number expressions, month names, day names, and unit names (and identify likely years) and a larger (stage 2) grammar that takes these annotations as input and generates context-independent values for TIMEX2s. Many rules generate final normalized values, since many expressions (e.g., full dates, month/year expressions) do not require contextualizing. The remaining rules generate a context-independent value and, for ambiguous unit phrases, anaphoric points, and occurrences of *today*, a classification problem.

The rule-based classifiers are simple. The point-duration classifier has rules that match expressions that are likely point or duration indicators either within a unit phrase or immediately to the left of a unit phrase. The direction classifier uses the same heuristic as [11], relying on TreeTagger for part-of-speech tagging and chunking: it looks at the closest preceding verb chunk—if it is past or perfect tense, it labels the instance as "backward"; if it is present-progressive or a present tense copula, it labels the instance as "same"; and if it is any other present-tense (including modals, such as *will*, *shall*, *may*), it labels the timex as "forward."

We used the classifier module of the minorthird package for training a maximum entropy-based classifier. The tasks of transforming the instances into feature vectors and estimating parameters are done automatically by the system.

### 6.2 Ten Approaches

We ran the system over the TERN test corpus (192 documents, 1828 TIMEX2s, of which 1741 have a non-null VAL attribute), varying the classifiers and the reference tracking model used. Note that for all configurations, we used the baseline classifier for the today task, which assigns *specific* to every instance.

1. Baseline classifiers (assign majority class—point-duration: duration, direction: backward); timestamp-based
2. Baseline classifier for point-duration, rule-based classifier for direction; timestamp-based
3. Rule-based classifiers for both tasks; timestamp-based
4. Baseline classifier for p-d, MaxEnt classifier for dir; timestamp-based
5. MaxEnt classifiers for both tasks; timestamp-based
6. "Perfect" classifiers; timestamp-based
7. Baseline classifiers; recency-based
8. Baseline classifier for p-d, rule-based classifier for dir; recency-based
9. Rule-based classifiers for both tasks; recency-based
10. Baseline classifier for p-d, MaxEnt classifier for dir; recency-based
11. MaxEnt classifiers for both tasks; recency-based
12. "Perfect" classifiers; recency-based.

We include two "perfect" runs (6 and 12) to set a ceiling on the overall normalization performance of the classifiers.

### 6.3 Metrics

We use the official TERN scorer to evaluate our normalization performance. As we mentioned in §2.1, the scorer computes precision, recall, and F-measure for each normalization attribute; since we are focusing on the core VAL normalization, we report only VAL scores. For the TERN scorer, precision is the ratio of correctly normalized VALs to attempted VALs; recall, the ratio of correct VALs to possible VALs *in the recognized TIMEX2s*; and F-measure, $(2{*}P{*}R)/(P{+}R)$. Recall (and, F-measure) is computed not with respect to all possible TIMEX2s in the gold standard but only with respect to the TIMEX2s recognized by the system. Because we are interested in the end-to-end task, we also report results with recall (and F-measure) computed with respect to *all* possible TIMEX2s. We refer to the TERN versions as *relative recall* (RR) and *F-measure* (RF), and to our versions as *absolute recall* (AR) and *F-measure* (AF).

Since we use the same recognizer output for all of our runs, the difference between the relative and absolute measures is immaterial in comparing these results, but the absolute measures make comparison with other systems clearer.

| | Correct | P | RR | RF | AR | AF |
|---|---|---|---|---|---|---|
| 1 | 1063 | 0.733 | 0.694 | 0.713 | 0.611 | 0.666 |
| 2 | 1066 | 0.736 | 0.696 | 0.716 | 0.612 | 0.669 |
| 3 | 1063 | 0.734 | 0.694 | 0.714 | 0.611 | 0.667 |
| 4 | 1138 | 0.784 | 0.743 | 0.763 | 0.654 | 0.713 |
| 5 | 1124 | 0.775 | 0.734 | 0.754 | 0.646 | 0.704 |
| 6 | 1230 | 0.848 | 0.803 | 0.825 | 0.706 | 0.771 |
| 7 | 1060 | 0.734 | 0.692 | 0.713 | 0.609 | 0.666 |
| 8 | 1059 | 0.735 | 0.692 | 0.713 | 0.608 | 0.666 |
| 9 | 1058 | 0.734 | 0.691 | 0.712 | 0.608 | 0.665 |
| 10 | 1117 | 0.774 | 0.730 | 0.751 | 0.643 | 0.701 |
| 11 | 1105 | 0.765 | 0.722 | 0.743 | 0.635 | 0.694 |
| 12 | 1214 | 0.841 | 0.793 | 0.816 | 0.697 | 0.762 |
| 13 | 1389 | 0.866 | 0.837 | 0.851 | 0.798 | 0.830 |

**Table 2.** Normalization experiments; row numbers 1–12 refer to the list of approaches in §6.2.

### 6.4 Results

Our results are given in Table 2, with the results from the best system at TERN for comparison (row 13). While our results are not yet up to that level, they are competitive with other systems; AF-scores for the TERN systems ranged from 0.439 to 0.830, with an average of 0.657.

Overall, the best (non-perfect) runs for each reference tracking model use the baseline p-d classifier and the MaxEnt dir classifier (AF-scores of 71.3% and 70.1% for timestamps (run 4) and recency-based (run 10, respectively). In each case, adding the MaxEnt p-d classifier reduces performance slightly (70.4% and 69.4% for runs 5 and 11), but in any case, the performance remains several points above that of any of the rule-based runs (F-scores between 66.5% and 66.9%). We comment on this fall-off in performance when adding point-based classifiers below. Comparing the two reference tracking models, it is clear that the timestamp-based model outperforms the recency-based model. We discuss the reasons for this below, as well.

There is clearly still room for improvement, both in the classifiers (as can be seen in comparisons with the "perfect" runs) and for the other components of the system. In addition to looking at classifier performance independently, we have performed an error analysis of the "perfect" runs to see where the other components go wrong. Again, see below.

Returning to our research questions, how can data-driven techniques be brought to bear on the normalization task? Our experiments show that data-driven methods outperform rule-based methods for crucial sub-tasks within the overall normalization task. Can we use "shrink-wrapped" data-driven language-processing solutions to make the semantic interpretation problem of timex normalization more robust, modular, and easily maintainable? The answer is clear. The relative ease of generating these runs by swapping out different classifiers

and temporal reference tracking modules suggests a clear "yes." Our staged architecture has done two things for us. (1) It has limited the set of absolutely required rules to both context-independent pre-normalization, where no rule has to make reference to any information outside of a given timex, and final computation, where rules make reference only to meta-data (the preliminary value generated by the pre-normalization, the classification values generated by the classifiers, and the reference time generated by the reference model). (2) It has identified a potential place for data-driven methods within the normalization process—namely, incorporating contextual information—and made it straightforward to experiment with both data-driven and rule-based methods to tackle this sub-task.

### 6.5 Where the Errors Originate

Our analysis of the final normalization performance of the perfect runs (6 and 12) allows us to pinpoint errors arising from the non-classifier stages of our system and allows us to limit the error analysis of the other runs to the classification results. Of the 216 incorrectly normalized TIMEX2s in run 6, 63 result from problems in the pre-normalization stages, including a single rule bug involving time zones that resulted in 36 errors. 26 errors arise from the recognizer extent errors; 18 errors, from the temporal reference model contributes 18 errors; and 6 errors, from temporal computation in stage 5. 46 errors are the result of a bug in the generation of the gold standard for the p-d classifier: durations of unspecified length are erroneously labeled as points. 46 more errors result from classes of timexes (sets and non-specific timexes) whose instances are ambiguous with point- and duration-referring timexes. For the 85 timexes that are recognized but not normalized, the failure is attributable to a recognition extent error or an omission in the pre-normalization rule base.

To see the effect of varying temporal reference models, we turn to the other "perfect" run, run 12, and look at anaphoric timexes. Even though at least 18 errors in the timestamp-based perfect run result from poor temporal reference tracking, the recency-based model does no better. Why? In one document, 4 of the timestamp-based errors occur in a chain, and because of a recognition extent error, the timex immediately preceding the chain that establishes the anchor point is not normalized at all, so the recency-based model also fails on all four timexes. For several other timexes, the recency-based model chooses the correct reference time, but because of errors in stage 5, the wrong final value is computed. Finally, in the remaining cases, neither the timestamp-based nor the recency-based model is sufficient to account for phenomena that are known to be hard, such as anchoring to an event, discourse effects, or shifts in granularity that allow for non-specific reference. Additionally, the recency-based model introduces several errors of its own.

Turning to the ambiguous point-referring timexes that are sent by stage 2 to the p-d classifier, we see why there is an across-the-board fall-off in end-to-end performance when adding a rule-based or MaxEnt classifier even though their classification accuracy is much better than the baseline (see Table 3). 73 of the 83

| | Baseline | Rule-based | MaxEnt |
|---|---|---|---|
| Point-duration | 40% | 54% | 73% |
| Direction (timestamp) | 46% | 39% | 62% |
| Direction (recency) | 43% | 38% | 57% |

**Table 3.** Classification accuracy results.

ambiguous point-referring timexes are ones that even the perfect runs normalize incorrectly (for various reasons, all among those outlined above). Furthermore, none of the 102 "other" timexes (with respect to the p-d task) are correctly normalized by the perfect runs (they refer to non-specific or quantified entities, which have limited support in the final normalizer). Meanwhile, almost all of the 86 durations are normalized correctly. Since the baseline labels all instances as durations, it only misses 10 points that might have been correctly normalized in the end. The other classifiers have to have perfect accuracy on durations and recognize some of those 10 points as points to beat that, and even then, the difference is not very large.

Where the MaxEnt classifiers do make a difference is in the dir task. Here, even though the improvement in classification accuracy over the baseline is smaller than for the p-d task (see Table 3), using the MaxEnt dir classifier with either temporal reference model results in F-score improvements of 3 to 4 percentage points over both the baseline and the rule-based dir classifier. Of course, in terms of classification accuracy, the rule-based classifier, while doing better than chance, actually does worse than the baseline. While tense may be an important feature in deciding direction, it is far from the only feature.

Comparing the two MaxEnt classifiers, we see that the classifier trained and tested on data generated using the timestamp reference model outperforms the one with the recency-based model. The classifiers have a broadly similar distribution of misclassified instances: they have difficulty discriminating between the "backward" and "same" classes; the main difference in accuracy is due to recall of the "same" class. One possible explanation for the difference between the classifiers is that the stage 2 process is mistakenly labeling some deictic timexes as anaphoric ones, and a cursory examination of the data seems to bear this out: while deictic timexes such as *today* and *this week* are correctly flagged as deictic in stage 2, others, such as *this afternoon* are not.

## 7 Conclusions

In performing IE tasks, we would like to make as much use as possible of off-the-shelf, shrink-wrapped machine learning packages rather than going through the time-consuming process of manually developing rule-based systems. To that end, we have presented a staged temporal IE architecture that allows for experimentation with different approaches to different parts of the normalization task. Within this architecture, we have identified sub-tasks that seem amenable

to a machine-learning approach, and we have performed several experiments comparing machine-learning and rule-based approaches to these sub-tasks.

Overall, we find that data-driven methods can be applied at several points within a temporal IE system. In fact, by dividing the task into stages, we find that the tasks that require the most complex rule systems—incorporating a variety of information to make classification decisions—are precisely the ones that are well-suited for machine learning. Thus, by confining rule-based components to context-independent stages, we can vastly simplify their development and, at the same time, explore the use of data-driven methods in semantic normalization.

A major obstacle in using machine learning to acquire classifiers is generating reliable training data. We are reconsidering the ordering of the stages in our architecture—rather than have the rule-based component of stage 2 pick out, for instance, the ambiguous unit phrases for stage 3 classification as points, durations, sets, etc., perhaps all unit phrases could be classified prior to stage 2. This would result in much more training data for learning this classifier and would further simplify the rules required for context-independent interpretation.

We plan to develop more accurate and robust of temporal reference models. The main problem here is the sparsity of explicit timexes and the interaction of temporal reference with event reference. There is a lot of theoretical work on temporal reference, and the interaction between events and times. Recent computational work [10] using machine learning to assign reference times to sequences of clauses is also relevant, as are the TimeML guidelines and TimeBank corpus, which mark up not just timexes but also events and links [15,22].

Ultimately, we want our temporal IE work to end up as a component within a broader end-user task, such as temporal question answering or text mining, both to motivate and inform future annotation and architectural choices, and because we believe this will uncover new aspects of temporal IE. We encourage the TERN organizers to consider task-based evaluation efforts.

**Acknowledgments** David Ahn was supported by the Netherlands Organization for Scientific Research (NWO) under project number 612.066.302. Sisay Fissaha Adafre was supported by NWO under project number 220-80-001. Maarten de Rijke was supported by grants from NWO, under project numbers 365-20-005, 612.069.006, 220-80-001, 612.000.106, 612.000.207, 612.066.302, 264-70-050, and 017.001.190.

## References

1. D. Ahn, S. Fissaha Adafre, and M. de Rijke. Extracting Temporal Information from Open Domain Text: A Comparative Exploration. *Journal of Digital Information Management*, 3(1):14–20, 2005.
2. D. Appelt and D. Israel. Introduction to information extraction technology: IJCAI-99 tutorial, 1999. URL: http://www.ai.sri.com/~appelt/ie-tutorial/.
3. A. Berger, S. Della Pietra, and V. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

4. N. Chinchor. MUC-7 named entity task definition, September 1997. URL: http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/ne_task.html.
5. W. Cohen. Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data, 2004. URL: http://minorthird.sourceforge.net.
6. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.
7. L. Ferro, L. Gerber, I. Mani, and G. Wilson. *TIDES 2003 Standard for the Annotation of Temporal Expressions*. MITRE, April 2004.
8. ISO 8601: Information interchange – representation of dates and times, 1997.
9. J. Lafferty, F. Pereira, and A. McCallum. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.
10. I. Mani and B. Schiffman. Temporally anchoring and ordering events in news. In J. Pustejovsky and R. Gaizauskas, editors, *Time and Event Recognition in Natural Language*. John Benjamins, to appear.
11. I. Mani and G. Wilson. Robust temporal processing of news. In *Proceedings of the 38th ACL*, 2000.
12. A. McCallum and W. Li. Early results for Named Entity Recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the 7th CoNLL*, 2003.
13. MUC-6. Named entity task definition, May 1995. URL: http://www.cs.nyu.edu/cs/faculty/grishman/NEtask20.book_1.html.
14. J. Pustejovsky, J. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, and G. Katz. TimeML: Robust specification of event and temporal expressions in text. In *Proceedings of the AAAI Spring Symposium*, 2003.
15. J. Pustejovsky, R. Sauri, A. Setzer, R. Gaizauskas, and B. Ingria. *TimeML Annotation Guidelines*, 2002.
16. E. Saquete, P. Martinez-Barco, and R. Munoz. Recognizing and tagging temporal expressions in Spanish. In *Workshop on Annotation Standards for Temporal Information in Natural Language, LREC 2002 (Third International Conference on Language Resources and Evaluation)*, pages 44–51, 2002.
17. F. Schilder. Extracting meaning from temporal nouns and temporal prepositions. *ACM Transactions on Asian Language and Information Processing*, 2004.
18. F. Schilder and C. Habel. From temporal expressions to temporal information: Semantic tagging of news messages. In *Proceedings of the ACL-2001 Workshop on Temporal and Spatial Information Processing*, 2001.
19. A. Setzer and R. Gaizauskas. Annotating events and temporal information in newswire texts. In *Proceedings of LREC2000*, 2000.
20. F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology-NAACL*, 2003.
21. TERN. Temporal Expression Recognition and Normalization, 2004. URL: http://timex2.mitre.org/tern.html.
22. TimeBank. Annotated corpus, 2004. URL: http://www.cs.brandeis.edu/~jamesp/arda/time/timebank.html.
23. TreeTagger. A language independent part-of-speech tagger, 2004. URL: http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/.