Extracting Temporal Information from Open Domain Text: A Comparative Exploration

David Ahn Sisay Fissaha Adafre Maarten de Rijke

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

ABSTRACT

The utility of data-driven techniques in the end-to-end problem of temporal information extraction is unclear. *Recognition* of temporal expressions yields readily to machine learning, but *normalization* seems to call for a rule-based approach. We explore two aspects of the (potential) utility of data-driven methods in the temporal information extraction task. First, we look at whether improving recognition beyond the rule base used by a normalizer has an effect on normalization performance, comparing normalizer performance when fed by several recognition systems. We also perform an error analysis of our normalizer's performance to uncover aspects of the normalization task that might be amenable to data-driven techniques.

Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: Linguistic processing

General Terms

Temporal information extraction

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 (often unwilling) user with a relatively large amount of text to consume. There is a need for tools that reduce the amount of text that has to be read to obtain the desired information. To address this need, the IR research community is currently exploring ways of pinpointing highly relevant information. We view information extraction [4] as one of the core technologies to help facilitate highly focused IR. Indeed, recognizing entities and semantically meaningful relations between those entities is key to providing focused information access.

Temporal information extraction provides a particularly interesting task in this respect. Temporal expressions (from now on, *timexes*) are natural language phrases that refer directly to time points or intervals. They not only convey temporal information on their own but also serve as anchors for locating events referred to in text. For reasons to be explained below, *recognizing* temporal expressions has become a "do-able" task, even without tremendous knowledge engineering efforts. Moreover, in recent years, the task of *interpreting* temporal expressions has begun to receive attention [17, 21]. And finally, from a user's perspective, temporal aspects of events and entities, and of text snippets and documents, provide a very natural mechanism for organizing information. Specifically, Question Answering (QA) is an area where accurate analysis (both recognition and normalization) of temporal expressions plays an important role. For example, 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).

The importance of the proper treatment of timexes is reflected by the relatively large number of NLP evaluation efforts where they play a role. Recognizing timexes, for example, is an intergral part of many information extraction tasks (e.g., MUC-6 and 7 Named Entity Recognition tasks, ACE-2004 Event Recognition task). There are a number of annotation guidelines for timexes [10, 22, 20]. And recently, a timex annotated corpus has been released with the aim of improving the tasks related to timexes [24].

The type of timexes considered in a typical information extraction task are limited to date and time values [19, 7]. In the 2004 Temporal Expression Recognition and Normalization (TERN) task, however, a wide variety of timexes are considered—which makes the task of recognition and normalization more challenging and much more interesting.

This is an exploratory paper, in which our main interest is in the end-to-end timex-recognition-plus-interpretation task. Specifically, we are interested in identifying opportunities for the use of data-driven methods in the interpretation part of the task. On the recognition side, rule-based systems can provide very high precision but require significant human effort in rule development to achieve reasonable recall. As we will see, machine learning can provide excellent results on the recognition task with minimal human intervention, given a tagged corpus. Furthermore, such a data-driven approach may be preferable because of its portability and robustness.

For the interpretation task, however, machine-learning

DIR 2005, 5th Dutch-Belgian Information Retrieval Workshop.

methods based on sequence labeling seem less appropriate. For one thing, there are a potentially unlimited number of classes (i.e., temporal values). Another problem is that a significant proportion of timexes require non-local context for interpretation (i.e., they are anaphoric or deictic). Even more problematic is the fact that many timexes require significant temporal computation with respect to contextually given information—the connection between form and content is mediated by both context and world knowledge.

Nevertheless, we aim to understand what the opportunities are for using robust, "shrink-wrapped" machine-learning tools for the interpretation task. We are interested in determining whether the use of machine learning for recognition affects normalization performance downstream, as well as where within the normalization task a data-driven approach might provide some leverage. To these ends, we present several experiments comparing recognition performance of machine-learning and rule-based systems and also comparing the performance of a rule-based normalizer on the output of various recognition systems. We analyze the performance of the rule-based normalizer, looking for places to apply machine learning.

In §2, we outline the TERN evaluation tasks of identification and normalization. In §3, we describe our identification experiments, comparing two rule-based systems and two runs of a machine learning system with different feature sets. Our normalization system is described in §4. Finally, we conclude in §5.

2. TASK DESCRIPTION

We participated in the 2004 edition of TERN, the Temporal Expression Recognition and Normalization Evaluation (`http://timex2.mitre.org/tern.html`). The TERN evaluation is organized under the auspices of the Automatic Content Extraction (ACE, `http://www.nist.gov/speech/tests/ace/`) program, whose objective is to develop natural language processing technology to support automatic understanding of textual data. We submitted runs from one of the rule-based recognition systems and from the rule-based normalization system described below. The TERN evaluation provided specific guidelines for the identification and normalization of timexes, as well as tagged corpora for training and testing and evaluation software. We followed these guidelines and used these resources for the experiments described below.

The TERN evaluation consisted of two distinct 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 [10].

The TIDES guidelines introduce a type of SGML or XML element, TIMEX2, to mark timexes in text [10]. TIMEX2 elements may contain one or more of the following attributes: VAL, ANCHOR_DIR, ANCHOR_VAL, MOD, SET, NON_SPECIFIC. The VAL attribute indicates the reference of the TIMEX2; its range of values are an extension of the ISO 8601 standard for representing time [13]. The ANCHOR_DIR and ANCHOR_VAL attributes are for temporal expressions (primarily durations and fuzzy references) that are anchored to a reference time. At present, the MOD, SET, and NON_SPECIFIC attributes are normalization attributes that our system does not handle.

The recognition and normalization tasks are performed with respect to corpora of transcribed broadcast news speech and news wire texts from ACE 2002, ACE 2003, and ACE 2004, marked up in SGML format and, for the training set, hand-annotated for TIMEX2s.

An official scorer that evaluates both recognition and normalization performance is provided as part of the TERN evaluation. For recognition results, the scorer computes precision, recall, and F-measure both for TIMEX2 tags (in other words, for *overlap* with a gold standard TIMEX2 element) and for *extent* of TIMEX2 elements (in other words, exact match of entire timexes). For normalization, the scorer computes precision, recall, and F-measure for each of the normalization attributes listed above. Recall (and thus, F-measure), however, is computed not with respect to all possible TIMEX2 elements in the gold standard but only with respect to the TIMEX2 elements tagged by the system. Because we are interested in the end-to-end task, we report our results in this paper with recall (and F-measure) computed with respect to all possible TIMEX2s.

3. RECOGNITION

The recognition task is to identify phrases that refer to time points. The TIDES guidelines limit the set of markable timexes (which they indicate 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*

Recognition can be thought of either as a partial parsing task or as a labeling task, and rule-based methods or machine-learning systems can be deployed, according to the perspective taken. In §3.1, we describe a machine-learning approach to timex identification, which is based on Conditional Random Fields, and then in §3.2, we present our rule-based approach.

3.1 Machine learning methods

Timexes are a type of entity commonly dealt with in named entity recognition tasks in which machine learning techniques have been shown to provide good results. As mentioned before, though, the type of timexes considered in TERN are quite a bit more diverse than the ones considered in typical information extraction tasks. Nevertheless, these timexes still have some desirable properties from the perspective of machine learning. In particular, the core vocabulary used for building them is restricted, so the use of lexical features does not significantly increase the feature set used for machine learning.

A machine learning technique that has recently been introduced to tackle the problem of labeling and segmenting sequence data is Conditional Random Fields (CRFs, [16]). Unlike Hidden Markov Models [16], 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. This enables the incorporation of overlapping and interacting features into the model. CRFs have been shown to perform well in a number of natural language processing applications, such as POS tagging [16], shallow parsing or noun chunking [23], and named entity recognition [18]. Their characteristics make CRFs ideally suited for the specific task of recognizing timexes as they provide us with a framework for combining evidence from different sources to maximize performance.

We used the implementation of CRFs from the minorThird toolkit for extracting timexes from text [8]. The training material is a tagged corpus in which the timexes are marked by XML tags. The task, then, is to learn from these training instances of timexes rules or patterns to recognize new instances. In this framework, phrase identification tasks are reduced to word labeling tasks by assigning each word one of the labels (B)egin, (I)nside, (O)utside, according to whether it begins a phrase of interest (in our case, a timex), continues such a phrase, or is not part of such a phrase [14].

3.1.1 Experiments

The training data consists of 511 files from ACE 2002, ACE 2003 and ACE 2004; the test data consists of 192 files. As mentioned earlier, the temporal expressions in the training files, are marked with XML tags. The minorThird system automatically converts from XML format to B-I-O format. A temporal expression enclosed by `<TIMEX2>` tags constitutes a span. The features in the training instances are generated by looking at the surface forms of the tokens in the spans and their surrounding contexts.

In order to simplify our implementation and achieve reasonably high generalization, we use simple lexical and character features which can easily be derived from the surface forms of words or phrases. We used a context window of three words to the left and right. One set of features is the lowercase form of all the tokens in the span, with each token contributing a separate feature. The tokens in the context window constitute another set of features. These feature sets capture the lexical content and context of timexes. Additionally, character types and character type patterns of tokens in the spans are used to capture the character patterns exhibited by some of the tokens in temporal expressions. The patterns are defined using the symbols, A, a, X, x, and 9. A and a are used to define character type features; X and x are used to express the character type pattern features; and 9 is used for representing numeric tokens. For example, the token *Monday* is represented by the character type 'Aaaaaa' and character type pattern 'Xx+'. Character features are extracted not only from tokens within the spans but also from the context windows. The first and the last tokens of a span and their corresponding character types and character type patterns are also part of the feature set. The number of tokens in a span is yet another feature.

3.1.2 Results

We train the system with the surface features listed previously. The recognition results for the CRF-based system, scored using the official TERN scorer, are given in Tables 1 and 2, row 1.

	TIMEX2		
	Precision	Recall	F
CRF	0.980	0.842	0.906
CRF + extra features	0.979	0.856	0.914
POS-only	0.979	0.633	0.769
POS and chunk	0.989	0.537	0.696
POS-only (w/years)	0.978	0.713	0.825
POS and chunk (w/years)	0.987	0.617	0.759

Table 1: Recognition results: TIMEX2 (overlap).

Table 1 indicates the system performance on partial-match identification of TIMEX2s, where credit is given for tagging any part of an actual TIMEX2. Table 2 indicates the system performance on exact-match identification, in which a system only gets credit for the identifying the exact extent of a TIMEX2. Unsurprisingly, the scores for the extent measure are lower than those for the TIMEX2 measure.

	TEXT		
	Precision	Recall	F
CRF	0.798	0.685	0.737
CRF + extra features	0.855	0.748	0.798
POS-only	0.795	0.514	0.625
POS and chunk	0.830	0.450	0.584
POS-only (w/years)	0.811	0.591	0.684
POS and chunk (w/years)	0.843	0.527	0.648

Table 2: Recognition results: TEXT (extent).

3.1.3 Error Analysis

An analysis of the output of the CRF system reveals errors which can be eliminated with additional features. One set of errors relates to spans that are too short or too long. To address this problem, we created a dictionary of core lexical items in timexes and restricted context information to these lexical items, which should improve detection of timex boundaries. The list was generated from the Penn TreeBank by extracting the most frequent lexical items occurring in the constituents labeled -TMP. Furthermore, the default character pattern for numeric tokens is the simple pattern 9+; it assigns the same form for all numeric values, which misses an important fact about the form of year expressions. A simple pattern which takes into account the form of a year expression enabled the system to extract year values more precisely. We also added a list of names for days of the week and months of the year as separate features. These simple additions to the default features resulted in significant performance improvements (Tables 1 and 2, row 2).

Although the use of additional features, such as lists of names of days, helps to improve the scores, there seems to be a limit to what can be achieved using such shallow methods given the fact that the training material is fixed. A deeper level of linguistic analysis needs to be made to achieve significant improvement. Analysis of some of the errors suggest that some knowledge of constituency, e.g., noun chunking or dependency parsing, might help in identifying timex boundaries. In general, the CRF-based recognizer has a tendency to limit extents to common temporal elements, such as, for example, *Orthodox Christmas Eve* → *Christmas*.

3.2 Rule-based methods

Finite-state automata have been employed extensively in partial parsing, or chunking, as well as in morphological analysis and other natural language processing tasks [2, 15, 5]. In partial parsing, chunks form the basic level of constituency, the non-recursive "core" of a major phrase [1]. Chunks can be identified by regular expression patterns over part-of-speech tags, and, in turn, regular expression patterns over chunks can be used to form higher levels of constituency, such as simplex clauses.

One way of looking at the timex detection task is as a partial parsing task in which timexes form the level of interest. Apart from date and time expressions with fixed forms (which lend themselves readily to regular expression patterns), timexes are constrained to be natural language phrases headed by one of a small number of trigger words as discussed in §2. Chunks can be used to approximate these phrases and can, if needed, be joined by appropriate patterns to form timexes.

We experimented with using two levels of prior linguistic analysis to feed our rules: part-of-speech (POS) tags and chunks. In one experiment, we used patterns over text tagged only with POS tags; in the other, the patterns range over chunked and POS-tagged text.

Our development efforts concentrated on the patterns involving chunks and POS tags. We first built up patterns for word classes: deictics (*today*, *yesterday*, etc.), units (*hour*, *day*, etc.), days (*Monday*, *Tuesday*, etc.), months, temporal adjectives, temporal adverbs, and so on. We also built up complex expressions for times (e.g., 12:24:45.69GMT) and dates (e.g., 10/31/1999). Then, in principle, our identification patterns would simply look for chunks headed by words from the appropriate classes. We followed this principle for units and deictics, but for other classes, we wrote patterns that look at more of the relevant internal structure of chunks, expecting that this might help normalization. For example, one of our complex date patterns is:

```
<NC>[ Month Date Comma? Year_full ]
```

where `Month` refers to the month class, `Date` refers to the date class (i.e., 1–30), and `Year_full` refers to years from 1900–2099. In developing rules for normalization, as described in §4, this pattern can be interpreted directly.

For our experiment without chunks, we used the same word classes and complex expressions for times and dates, as well as the patterns that looked at the internal structure of chunks. We then wrote simple noun and preposition chunk patterns to allow for head-based chunk patterns. Thus, for noun chunks headed by units, we used the following pattern:

```
(DT|CD|PP\$)? \
  (CD|JJ[RS]?|RB[RS]?|VVG|VVN)* NN? Unit
```

where `(DT|CD|PP$)` matches specifiers (determiners, cardinal numbers, and possessive pronouns), `(CD|JJ[RS]?|...)` matches modifiers, including adjectives, adverbs, and participles, `NN` matches prenominals, and `Unit` matches words from the unit class.

3.2.1 Experiments

We built two rule-based systems for timex identification: the first uses only POS tags while the second makes use of chunks and POS tags. The first system is based on hand-built patterns over POS tags and lexical items. For POS-tagging, we use TreeTagger [25] and convert the output into XML. The intuition behind the use of POS tags is that they provide some level of generalization for pattern writing, as well as resolving some word sense ambiguities (e.g., the ordinal number sense of *second* can be distinguished from other senses (including the time unit sense) simply by virtue of being tagged as an adjective[1]). While the generalization provided by POS-tagging is merely a convenience for closed-class categories, such as determiners, for which patterns could exhaustively list *a*, *the*, *this*, etc., it is absolutely necessary for writing patterns over open-class categories, such as adjectives and nouns.

To compile our regular expression patterns into finite-state automata, we use flex [11], a standard scanner generator that does character-level regular expression matching. The overall system is written in CDuce, a higher-order functional programming language that allows pattern matching over XML structures [6]. Since the TERN data is accompanied by a DTD indicating which document elements may contain TIMEX2s, a CDuce program whose type structure is derived directly from the DTD determines which elements need to be processed. It sends the text of these elements to be POS-tagged by TreeTagger and then TIMEX2-tagged by the flex-generated scanner. Finally, extraneous tags (i.e., TreeTagger output) are stripped off, and the output is aligned with the initial input to restore formatting.

The second system (which was essentially the recognition component of our submission to TERN) is based on the same hand-built patterns as our first rule-based system but takes chunks into account, as well. The intuition behind the use of chunks is that they provide ready-made approximations to phrases that just as crucially keep *out* words that do not belong as include words that do. Then, as we explain above, we can simply write patterns to look for chunks headed by timex trigger words and, if necessary, join a timex-headed chunk with a required complement chunk. For example, the timex *24 years after his death* consists of a noun chunk *24 years* and its complement preposition chunk *after his death*.

As before, we use TreeTagger, but here it serves double duty, POS-tagging and chunking the input; again, TreeTagger output is converted into XML. Now, regular expressions for matching sequences can be compiled into finite-state automata by a variety of tools (such as flex), but the patterns we use for matching the hierarchically structured TreeTagger output require compilation into tree automata. This is where CDuce comes in: it compiles regular expressions over hierarchical structures (in particular, XML elements) into tree automata, using pattern ordering and a greedy matching policy [12] to resolve conflicts. This system works exactly as the other one, except that instead of sending the TreeTag-

[1]Of course, the various nominal senses of *second* cannot be distinguished in this way.

ger output to a flex-generated scanner for TIMEX2-tagging, the CDuce program itself does the tagging.

3.2.2 Results

We ran each of the two systems described above on the TERN test corpus. The results, again scored using the official TERN scorer, are given in Tables 1 and 2, in the rows labeled *POS-only* and *POS and chunk*.

After the TERN evaluation, a preliminary error analysis revealed that both rule-based systems lack a rule to identify years standing alone. Adding a simple rule to identify years between 1900 and 2019 increases recall significantly for both tag and extent, with only a very small precision penalty. The results of adding this rule are shown in Tables 1 and 2 in rows *POS-only (w/years)* and *POS and chunk (w/years)*.

3.2.3 Error Analysis

Both rule-based systems very rarely misidentify something as a TIMEX2. Most of these spurious TIMEX2s are non-temporal occurrences of the words *past*, *present*, and *now*, which are anyhow only weakly normalizable. The real problem with the rule-based systems is in recall, which is largely a matter of insufficient patterns. For example, as previously mentioned, one pattern that was inexplicably left out of both rule-based systems is the simple year pattern.

Both rule-based systems do make a number of extent errors, though. The vast majority of these errors (90.5% for TERN, 93.1% for flex) result from the systems tagging too little of an expression, missing premodifiers and arguments. In some cases, it is clear that deeper syntactic information is needed to get the full extent—for example, chunking is not enough to distinguish *24 years after his death* from *Two years after the crash site*. The first phrase is a complete timex, since the phrase *after his death* is a PP, but the second expression is not—*after the crash site* is the beginning of a complement clause (*... was discovered*).

One caveat, though, regarding the use of deeper syntactic analysis: more complicated syntactic analysis is generally less reliable. Even the chunker, which performs relatively shallow analysis, introduces errors. Some of the extent errors peculiar to the chunk-based system include phrases such as *we're talking months*, *congress last week*, *Palestinian leader Yasser Arafat last week*, *secretly indicted nine months ago*. In each of these cases, the chunker included too much material in a noun chunk, which our patterns take as the boundary for timexes with unit triggers, such as *months*.

Overall, both rule-based systems exhibit high precision and low recall, a symptom of the development methodology and the amount of time devoted to actual rule development. The differences between the systems—greater precision for the chunk-based system and greater recall for the POS-tag-based system—are not particularly surprising. The chunk-based system requires more pieces of information to be in place for an expression to qualify as a TIMEX2 (e.g., a date expression needs not only to contain a Month and a Date, but it must also have been marked as a noun chunk), so false positives are less likely. But since its source for this additional information is less reliable (i.e., chunking is more difficult than tagging), it is more likely to reject well-formed expressions that have been POS-tagged correctly but chunked incorrectly. Interestingly, the use of chunks provides only a small boost to precision at a relatively high price for recall. Since chunking does not seem to be the right level of linguistic analysis to solve the extent problems, it is not clear that the additional processing they require is warranted, something that must be kept in mind when considering how to improve the CRF-based recognizers, as well.

3.3 Recognition: Upshot

To put these results in context, the top-performing system at the 2004 TERN evaluation—also a machine learning-based system—scored 0.981, 0.909, 0.944 (precision, recall, F-measure) for partial-match (TIMEX2) and 0.906, 0.840, 0.872 for exact-match (TEXT) [9]. The results of the CRF run with tuned features approaches this level of performance (and compares well to the other TERN systems). What is more, even the results of the CRF system using the default feature set are considerably better than either rule-based system. Recall is vastly improved, with only a very small precision penalty. Despite the differences in performance in the various systems, they shared several features: good precision, approaching good recall on tags, and problems with extent largely stemming from not having enough syntactic information to get arguments and modifiers right.

4. NORMALIZATION

Given the ease with which timex recognition can be performed by an off-the-shelf machine learning system with little human intervention, it is natural to question why one should bother with a rule-based system at all. One possible answer is normalization. Remember that recognition is only a part of the larger task of temporal information extraction. For the kind of downstream application we are interested in, it is important to interpret recognized expressions and render them in a standard format. The most natural way to approach this normalization problem is with a rule-based system,[2] and it seems that if effort is going to be invested in rule development for normalization, it is only natural to use those rules for recognition, as well. After all, for a downstream application that relies on normalized output, recall of unnormalizable timexes is superfluous.

The questions we would like to address, then, involve the proper role of machine learning in an end-to-end temporal information extraction system. Is it really the case that better recognition performance is wasted on rule-based normalization? Are there places within the normalization task itself where data-driven approaches can be applied?

We describe a rule-based normalization system we have built and report on several runs of the system on the output of different timex recognizers. A detailed error analysis of these runs reveals several points within the normalization pipeline where using a data-driven approach might improve performance. Furthermore, comparison of the results indicates that, at least for a rule-based system that relies heavily on imperfect upstream linguistic analysis components, such as ours, better recognition performance may, in fact, improve normalization.

4.1 Normalization method

Timex normalization is the problem of assigning a value to a recognized timex. As we discuss in §2, the TIDES guidelines distinguish several kinds of values; our normalization system focuses on time points, durations, and the

[2]Consider the TERN 2004 evaluation, where all recognition-only systems were machine learning-based, while all the systems participating in the full task were rule-based!

past, present, and future reference tokens. Time points are expressed by three different 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, however, must be interpreted relative to another temporal entity. Deictic timexes, such as *today*, *yesterday*, *three weeks ago*, *last Thursday*, *next month*, and so on, 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 some reference time—a salient time point previously evoked in the text that may shift as the text progresses. Moreover, some anaphoric timexes (those without an explicit direction indicator such as *next* or *previous*) also depend on the tense of the verb they modify: with past tense, they indicate the most recent time point prior to the reference time that matches the timex description; with present and future tense, they indicate the opposite.

For our normalization system, fully qualified and deictic timexes are straightforward to normalize, particularly since the documents it processes are time-stamped. Our system finesses anaphoric timexes in two ways. First, it simply treats them as deictics, instead of keeping track of introduced temporal entities and trying to determine their relative salience. Since the TERN corpus consists largely of short news stories focused on the immediate present, this heuristic seems reasonable. Secondly, instead of using the tense of the verb on which an anaphoric timex is dependent, we take the tense of the first (tensed) verb in the same sentence as an anaphoric timex. Determining dependency relations would require additional (more fragile) syntactic analysis, while finding tensed verbs is straightforward given the syntactic analysis already performed.

Durations are more difficult. 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 a point reading. Furthermore, many durations are anchored, either explicitly, as in *He will be barred from teaching for five years after that*, or implicitly. Our normalizer only normalizes fully qualified duration-like timexes, making no attempt to anchor them. Also, in the absence of an explicit directional indicator, such as *next* or *ago*, it assumes a duration reference.

Our normalizer also normalizes timexes that refer generally to the past, present, or future, such as *now*, *recently*, *future*. Such timexes receive a token value, PRESENT_REF, PAST_REF, or FUTURE_REF, as appropriate, as well as values for ANCHOR_DIR (AS_OF, BEFORE, or AFTER) and ANCHOR_VAL (the document timestamp).

4.2 Normalization system

In this section, we give some specific examples of normalization rules, as well as a technical overview of our system. Our normalization rules are mostly recognition rules augmented with pattern-matching variables to extract elements of the expression necessary to compute normalized values and with functions to perform the computation. For example, consider the pattern from §3.2 for matching fully qualified dates, augmented with pattern-matching variables:

```
<NC>[ (Month & m) (Date & d) Comma? (Year & y) ]
```

The function associated with this rule simply converts `m` to an integer and concatenates `y`, `m`, and `d` to produce the value.

For an example involving computation with respect to the reference time, consider:

```
<NC>[ (CD & cd) _* (Unit & un) ] <ADVC>[ Ago ]
```

where _ is a wildcard. The function associated with this recognition rule subtracts `cd un` units from the reference time and takes truncates resulting value to the appropriate granularity.

The most involved computation, requiring the examination of verb tense, is for some of the simplest expressions:

```
<NC>[ (Day & d) ]
```

where `Day` matches *Monday*, *Tuesday*, etc. The interpretation computation converts `d` into an integer; then, depending on the tense of the sentence, computes the date for the closest `d` in the past or in the future, and finally, produces a value from the year, month, and date of the result.

Finally, the rule for fully qualified durations is:

```
<NC>[ (CD & cd) _* (Unit & un) ]
```

where normalized versions of `cd` and `un` are concatenated (with a leading `P`) to produce a duration value. Note that this rule must be ordered *after* the *ago*-rule above and similar such rules in order to avoid spurious matching.

Much of the work done in rule development for identification paid off in development of the normalization system. Furthermore, on the assumption that for normalization, the ambiguity reduction and generalization benefits of POS tags are unnecessary, we were able to liberalize our normalization patterns so that they ignore POS tags. Our strategy of identifying duration timexes, however, using patterns matching noun chunks that simply have an appropriate head, meant that new patterns, such as the one above, had to be developed for interpreting such timexes.

The normalizer is written in Perl and takes as input the output of our TERN timex tagger, including not just newly added TIMEX2 tags, but also the XML elements indicating chunks and words. XPath queries are used to extract the document date stamp and sentences containing TIMEX2s and to determine the tense of these sentences (from the firstverb chunk). TIMEX2s are matched against regular expression patterns that extract the necessary information for normalization. Relative expressions are evaluated with respect to the document date stamp and the tense of the sentence; the Time::Piece Perl module is used to perform any necessary temporal arithmetic. The output of the normalizer is simply its input with the addition of the appropriate attributes for normalized TIMEX2s.[3]

4.3 Results

We ran our normalizer on the output of two different recognition systems (the chunk-based finite-state system, with the added year pattern, and the optimized CRF system from §3), as well as on the gold standard for recognition output, in order to see the effects of recognition performance on normalization performance. As we mention above, our initial expectation was that since the rules for normalization are more or less those of the rule-based recognizer, any

[3]We note that the normalization system we report on in this paper differs slightly from our TERN entry in that two simple bugs involving output format have been fixed.

	Corr	Incor	P	R	F
VAL	782	143	0.845	0.449	0.586
ANCHOR_DIR	63	11	0.851	0.150	0.255
ANCHOR_VAL	62	12	0.838	0.148	0.252

(a) Rule-based recognizer.

	Corr	Incor	P	R	F
VAL	885	231	0.793	0.501	0.614
ANCHOR_DIR	123	13	0.904	0.294	0.444
ANCHOR_VAL	121	15	0.890	0.289	0.436

(b) CRF recognizer.

	Corr	Incor	P	R	F
VAL	955	219	0.812	0.549	0.655
ANCHOR_DIR	137	14	0.901	0.327	0.480
ANCHOR_VAL	137	14	0.901	0.327	0.480

(c) Gold standard recognition.

Table 3: Normalization results

additional timexes identified by better recognition systems would not be normalized anyhow.

In Table 3(a), (b), and (c) we give the results of these three runs on the TERN test corpus. Remember that our recall and F-measure scores differ from the official TERN scores; they are computed with respect to all possible timexes rather than just those identified by the recognizer (1741 for VAL, and 419 for both ANCHOR_DIR and ANCHOR_VAL). According to the official TERN scorer, the recall scores (and correspondingly, the F-meastures) are higher for both the rule-based and CRF runs (0.699 and 0.583, respectively).

To put these results in context, the best performing system at TERN 2004 had scores of 0.875, 0.784, and 0.827 for VAL; 0.833, 0.585, 0.687 for ANCHOR_DIR, and 0.683, 0.649, 0.666 for ANCHOR_VAL (again, these scores are computed differently than the official TERN scores).

4.3.1 *Error analysis and discussion*

Looking at the results across runs, it seems that, in fact, better recognition actually does help normalization. We performed a preliminary comparison of the gold standard and rule-based recognizer runs to try to determine why this might be the case. Of course, the gold standard simply presents more actual timexes to the normalizer than the rule-based recognizer; the question is why, given that the normalizer patterns are the same as the rule-based recognizer patterns, the normalizer actually attempts to normalize any of these additional timexes.

What we found is that, for the most part, the upstream components are to blame. In order to produce input to the normalizer for the gold standard and CRF runs, we simply sent the recognized output through TreeTagger. As it turns out, though, the additional presence of TIMEX2 tags affects the tagging and chunking performance in two ways. First of all, it improves tokenization, especially of punctuation signs, allowing more normalization patterns to match. Chunking is also improved by the presence of TIMEX2 tags in the input. TreeTagger has an SGML mode in which it ignores, but does not respect, markup. Thus, it often produces chunks that cross TIMEX2 boundaries. We had to write a post-processor to close and re-open such chunks in the output of TreeTagger on the gold standard and CRF recognition output in order for our normalizer to run at all (it relies on well-formed XML). This post-process, however, creates "good" (matching) chunks in places where no such chunks exist in the output of TreeTagger on untagged input.

The other notable source of additional matches in the gold standard run is our liberalization of patterns in the normalization system to ignore POS tags. As it turns out, the lack of capitalization in the broadcast news documents results in a large number of mistagged day and month names which are thus missed by the rule-based recognizer.

We have not yet compared the CRF run with the rule-based run, but we expect that similar considerations will apply. We do note that the overall performance of recognition + normalization using the CRF recognizer is better than the strictly rule-based combination, so it looks as though any significant improvements in recognition performance, even if still imperfect, helps normalization.

We did perform a detailed error analysis of the run using the output of the rule-based recognizer and found a number of sources of (precision) error. The single largest source of error (42%) is an interesting property of the corpus—unlike in ordinary conversation, many of the documents in the corpus refer to the current day by name rather than as *today*.[4] Our finesse of anaphoric timexes is to blame for 17% of the errors—only one, though, actually requires interpretation with respect to a reference time distinct from the document time stamp; the remainder fail because of our strategy of choosing the first tensed verb in the sentence to decide directionality rather than looking more closely at the actual dependency relations. Errors of misclassification account for another 15% of the errors: the largest number of these are the result of normalizing a point reference as a duration; the other major misclassification error is with respect to uses of *today* to mean the present rather than the current day. A simple pattern-ordering bug results in 9% of the VAL errors and also explains all of the ANCHOR_DIR and ANCHOR_VAL errors. The remaining errors stem from problems with the recognizer or the tagger and chunker.

4.4 Normalization: Upshot

We take two lessons away from these experiments. The first is that, even from the perspective of the end-to-end task of temporal information extraction, it seems to be worthwhile to optimize recognition and normalization of timexes independently. The second is that data-driven approaches can help reduce the reliance on brittle upstream linguistic processing required by rule-based approaches. A caveat, though: as we point out in §3, it does seem that some sort of deeper syntactic analysis may be necessary to improve extent recognition performance, even for machine learning.

Additionally, our error analysis points to some aspects of the normalization task that may be amenable to a data-driven approach. Almost half of the errors our normalizer makes are the result of incorrectly guessing whether undirected anaphoric expressions such as *Tuesday* refer to the past, the present, or the future. This decision, though, can be easily stated as a classification problem for machine learning, with somewhat local relevant features (such as parent verb tense). Additionally, given appropriate data sources, even peculiarities of the data with regard to this decision, such as the use of *Tuesday* on a Tuesday to refer to the

[4]Most of these documents seem to be newswire items time-stamped late in the day, so the problem may be that they were intended for release on the following day.

current day rather than the previous Tuesday, can be automatically learned. The other major source of error is another sort of misclassification: confusing points, durations, and fuzzy reference. This decision, again, can be stated as a simple three-way classification problem, and, again, many of the relevant features (such as governing prepositions) are local. Thus, although it is not clear how to turn the general problem of normalization into a labeling task for machine learning, there do seem to be sub-problems that may be amenable to a data-driven approach.

5. CONCLUSION

Timex recognition is a task for which machine learning is well suited. An off-the-shelf system can produce respectable results straight out of the box, while a little bit of informed feature selection results in near state-of-the-art performance. There is room for improvement, though, particularly with respect to extent, where some sort of syntactic analysis seems to be required.

Rule-based systems are good at identifying timexes with high precision, while recall depends directly on effort invested. Our experiments indicate, though, that chunking is too shallow a level of syntactic analysis to help much with extent problems, and that, at the same time, it is inaccurate enough to introduce problems of its own.

With respect to normalization, we see, first of all, that better recognition performance does seem to improve rule-based normalization, even when the performance goes beyond that of a recognizer using the same set of rules as the normalizer. This suggests that independent optimization of recognition and normalization is a reasonable strategy for optimizing the end-to-end temporal information extraction task. Secondly, we have also uncovered some parts of the normalization task that may be amenable to a machine-learning approach. As we continue development of our normalizer, we proceed with ideas on how to use data-driven techniques, as well as expert linguistic knowledge, to improve performance.

6. 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.

7. REFERENCES

[1] S. Abney. Parsing by chunks. In *Principle-Based Parsing*. Kluwer Academic Publishers, 1991.

[2] S. Abney. Partial parsing via finite-state cascades. *Natural Language Engineering*, 1996.

[3] D. Ahn, V. Jijkoun, J. Kamps, G. Mishne, K. Müller, M. de Rijke, and S. Schlobach. The University of Amsterdam at TREC 2004. In *TREC 2004 Conference Notebook*, Gaithersburg, Maryland USA, 2004.

[4] D. Appelt and D. Israel. Introduction to information extraction technology: IJCAI-99 tutorial, 1999. `http://www.ai.sri.com/~appelt/ie-tutorial/`.

[5] K. Beesley and L. Karttunen. *Finite-State Morphology*. CSLI Publications, 2003.

[6] V. Benzaken, G. Castagna, and A. Frisch. CDuce: An XML-centric general-purpose language. In *Proceedings of the ACM International Conference on Functional Programming*, 2003.

[7] N. Chinchor. MUC-7 named entity task definition, September 1997. `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/ne_task.html`.

[8] W. Cohen. Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data, 2004. `http://minorthird.sourceforge.net`.

[9] L. Ferro. Annotating the tern corpus, 2004. `http://timex2.mitre.org/tern_2004/ferro2_TERN2004_annotation_sanitized.pdf`.

[10] L. Ferro, L. Gerber, I. Mani, and G. Wilson. *TIDES 2003 Standard for the Annotation of Temporal Expressions*. MITRE, April 2004.

[11] Flex. `http://www.gnu.org/software/flex`.

[12] A. Frisch and L. Cardelli. Greedy regular expression matching. In *Proceedings of ICALP*, 2004.

[13] ISO 8601: Information interchange – representation of dates and times, 1997.

[14] D. Jurafsky and J. Martin. *Speech and Language Processing*. Prentice-Hall, 2000.

[15] L. Karttunen, J.-P. Chanod, G. Grefenstette, and A. Schiller. Regular expressions for language engineering. *Natural Language Engineering*, 1997.

[16] 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.

[17] I. Mani and G. Wilson. Robust temporal processing of news. In *Proceedings of the 38th ACL*, 2000.

[18] 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.

[19] MUC-6 named entity task definition, May 1995. `http://www.cs.nyu.edu/cs/faculty/grishman/NEtask20.book_1.html`.

[20] 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.

[21] 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.

[22] A. Setzer and R. Gaizauskas. Annotating events and temporal information in newswire texts. In *Proceedings of LREC2000*, 2000.

[23] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology-NAACL*, 2003.

[24] TimeBank. `http://www.cs.brandeis.edu/~jamesp/arda/time/timebank.html`.

[25] TreeTagger. `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`.