Extracting Time Expressions from Clinical Text

Timothy A. Miller[1], **Steven Bethard**[2], **Dmitriy Dligach**[1], **Chen Lin**[1], and **Guergana K. Savova**[1]
[1] Boston Children's Hospital Informatics Program, Harvard Medical School
{firstname.lastname}@childrens.harvard.edu
[2] Department of Computer and Information Sciences, University of Alabama at Birmingham
steven.bethard@colorado.edu

Abstract

Temporal information extraction is important to understanding text in clinical documents. Temporal expression extraction provides explicit grounding of events in a narrative. In this work we provide a direct comparison of various ways of extracting temporal expressions, using similar features as much as possible to explore the advantages of the methods themselves. We evaluate these systems on both the THYME (Temporal History of Your Medical Events) and i2b2 Challenge corpora. Our main findings are that simple sequence taggers outperform conditional random fields on the new data, and higher-level syntactic features do not seem to improve performance.

1 Introduction

Temporal information is ubiquitous in clinical narratives, and accurately extracting temporal information has recently been the focus of a great deal of work in clinical natural language processing (NLP) (Raghavan et al., 2012; Miller et al., 2013; Sun et al., 2013). Relevant temporal information includes events, time expressions, and temporal relations between pairs of events and/or times. The accurate extraction of temporal information would be enabling technology for sophisticated downstream processing that requires temporal awareness of patient status. One promising application is question answering, where a physician can directly ask questions about a patient's medical record. Many question types of interest are explicitly temporal (*When was the patient's last colonoscopy?*), but almost all are implicitly temporal in the sense that every question needs to be understood relative to some time frame (*What drugs is the patient on?* cannot simply return all drugs in the record but has to understand the question itself is anchored in the present).

Time Class	Example
Date	February 2 2010, Friday morning
Time	5:30 PM, 20 minutes ago
Duration	For the next 24 hours, nearly 2 weeks
Quantifier	twice, three times
Prepostexp	postoperatively, post-surgery
Set	twice daily, weekly

Table 1: Time expression classes and two examples of each class.

This work focuses on the automatic identification of time expressions in clinical text. Time expressions are words and phrases that correspond to points or spans on a timeline, such as dates or times. Other temporal expression types include *Durations*, *Quantifiers*, *Sets*, and *Prepostexps*. Table 1 shows the time expression classes used in this work, with examples given of each class. The significant deviation from general domain methods is the *Prepostexp* type, which is specific to the clinical domain. Exemplified by terms like *postoperatively*, this type represents time spans relative to some event, often an operation.

Temporal information extraction has been a topic of a great deal of work both in the clinical and general NLP domains. In the general NLP domain, the TimeBank (Pustejovsky et al., 2003) spurred much of the early research by providing a manually annotated corpus of events, times and temporal relations. Shared tasks such as TERN[1], which focused on time expressions, and TempEval (Verhagen et al., 2007; Verhagen et al., 2010; UzZaman et al., 2013), which included events and temporal relations as well, helped build a com-

[1] http://www.itl.nist.gov/iad/mig/tests/ace/2004/

Proceedings of the 2015 Workshop on Biomedical Natural Language Processing (BioNLP 2015), pages 81–91,
Beijing, China, July 30, 2015.

munity around temporal information extraction. The community explored a wide variety of approaches, the best of which used either manually engineered databases of regular expression rules (Strötgen et al., 2013) or a supervised learning word classification paradigm (Bethard, 2013), and achieved precisions and recalls above 80% in the shared tasks.

In the clinical domain, temporal information extraction has seen a great deal of recent interest, with the i2b2 (Informatics for Integrating Biology and the Bedside) shared task on temporal information extraction (Sun et al., 2013) and the recent release of the THYME (Temporal History of Your Medical Events) corpus of clinical annotations (Styler IV et al., 2014). The i2b2 shared task contained a track explicitly focusing on extraction of temporal expressions. In that task, a variety of approaches were used for time expression extraction. The best performing system (Xu et al., 2013) used machine learning, with a conditional random field classifier (CRF) for finding spans and a support vector machine classifier for classifying attributes. Other top approaches used adapted regular expressions (Sohn et al., 2013) on top of the off the shelf Heideltime system (a general-domain NLP system for parsing time expressions) (Strötgen and Gertz, 2010). Another approach used a hybrid system where the output from a CRF-based system was combined with the output of a rule-based system (Kovačević et al., 2013).

In this work, we develop and evaluate several machine learning methods for extracting time expressions from clinical text. These methods include simple sequential classifiers, a sequential model (conditional random field), a constituency parser-based method, and an ensemble sequence method that attempts to leverage the differing performance of all the other models. The contributions of this work are the comparison and analysis of a large number of different machine learning models for this task, the first use of deep syntactic features for this task, and an evaluation on two different corpora, including the first evaluation of these methods on the THYME corpus.

	THYME (TempEval)	i2b2
Date	1271	1639
Time	54	69
Duration	195	406
Quantifier	61	n/a
Set	83	n/a
Prepostexp	149	n/a
Frequency	n/a	249

Table 2: Descriptive statistics of THYME and i2b2 corpora. Frequency in i2b2 is roughly the union of set and quantifier in THYME.

2 Materials and Methods

2.1 Corpora

We use two corpora for training and evaluating the methods described above. The first is the THYME corpus (Styler IV et al., 2014), which consists of clinical and pathology notes of patients with colon cancer from Mayo Clinic. The THYME corpus is split into training, development, and test sets based on patient number, with 50% in training and 25% each in development and test sets. For our experiments we use the same patient set as the upcoming TempEval 2015[2], patients 28-127. The training data contains 1874 time expressions, the development contains 1119, and the test set contains 1047. We used the development set for optimizing learning parameters, then combined it with the training set to build the system used for reporting results in Section 3.

The second corpus we use is the i2b2 2012 Challenge dataset (Sun et al., 2013). The i2b2 dataset contains discharge summaries from Partners Healthcare and Beth Israel Deaconess Medical Center. This data is split into a training and test set, with no predefined development set. We arbitrarily set aside filenames above 600 from the training set as a development set and for tuning parameters. Under this configuration, the i2b2 dataset contained 1856 training examples, 507 development examples, and 1820 test examples. Again, training and development examples were combined to build the system that is evaluated in Section 3.

Table 2 shows the distribution of the different time classes in the THYME and i2b2

[2]http://alt.qcri.org/semeval2015/task6/

corpora. While distribution is broadly similar, i2b2 had a higher percentage of *duration* expressions while THYME had many *prepost-exp* expressions, which in i2b2 were annotated as the *date* category.

2.2 Systems

We implemented a variety of systems in an attempt to empirically evaluate the best way to model the time span classification task. For all systems, the temporal expression extractor is implemented within Apache cTAKES[3] (clinical Text Analysis and Knowledge Extraction System) (Savova et al., 2011), making use of its components for feature generation as well as its interface to the source general-domain NLP system ClearTK (Bethard et al., 2014) which in turn interfaces with different machine learning libraries, including LibSVM (Chang and Lin, 2011) and CRFSuite (Okazaki, 2007).

2.2.1 Sequence Models

We developed three sequence-based models for this task, each with different perceived strengths. The first system is perhaps the simplest, a standard BIO (Begin-Inside-Outside) tagger using an off the shelf support vector machine (SVM) classifier (Cortes and Vapnik, 1995). BIO taggers work by labeling every token in a sentence as the beginning (B), inside (I), or outside (O) of some subsequence in the data (in this case a temporal expression). The tagger progresses left to right through a sentence, making a classification decision at each word, with features based on any information that would be available to a system at run time. After processing a sentence, tag sequences are converted to time expression spans and evaluated in the span format. The main benefit of this system is its efficiency, as it operates in a "greedy" fashion, getting a locally optimal labeling.

The second sequence system is a backwards BIO tagger. This system works just like the BIO tagger described above, except it starts at the end of the sentence and works its way forward. As mentioned above, this family of models is not globally optimal. In preliminary work, we found that the BIO tagger frequently left off the first word of a time expression, especially if it was a common word like 'the' or 'this.' Additionally, time expressions are often noun phrases, which typically carry a lot of meaning in the right-most word, so starting from the right has that advantage as well. For evaluation purposes, this model and the forward BIO tagger can be given exactly the same features, so there is a very clear evaluation of just the single difference in model strategy, going forwards or backwards.

The third system is a conditional random field (CRF) sequence labeler. Conditional random fields (Lafferty et al., 2001) are discriminatively-trained undirected graphical models that find the globally optimal labeling for a given configuration of random variables. We use a standard CRF architecture, the linear-chain CRF, where the random variables for sequence labels have only dependencies between the previous and next label, and random variables for arbitrary features of the observed evidence. Like the sequential taggers above, the CRF tagger assigns BIO tags to every word in a sequence, and time expressions are deterministically extracted from those assignments. The CRF tagger processes one sentence at a time, assigning labels to all tokens within that sentence simultaneously.

2.2.2 Constituency Model

The other system we developed is based on a constituency parse representation. Constituency trees represent the phrasal structure of a sentence, building up structure from the word level to a single tree which encloses the whole sentence. Our time expression classification model starts at the root of a tree and traverses it depth-first for a given sentence, and at each node in the tree classifies the enclosed span of words as a time expression or not a time expression, using a support vector machine classifier. During the depth-first traversal, further downward traversal is terminated with a positive classification (a time expression is found) or with a constituency spanning a single word. Figure 1 shows an example constituency tree with a time expression.

During training, the depth-first search will compare the span of every constituent in the search path with the spans of the gold standard, and any matching constituents are positive instances. Any non-matching spans are negative training instances. Features for each

[3]http://ctakes.apache.org

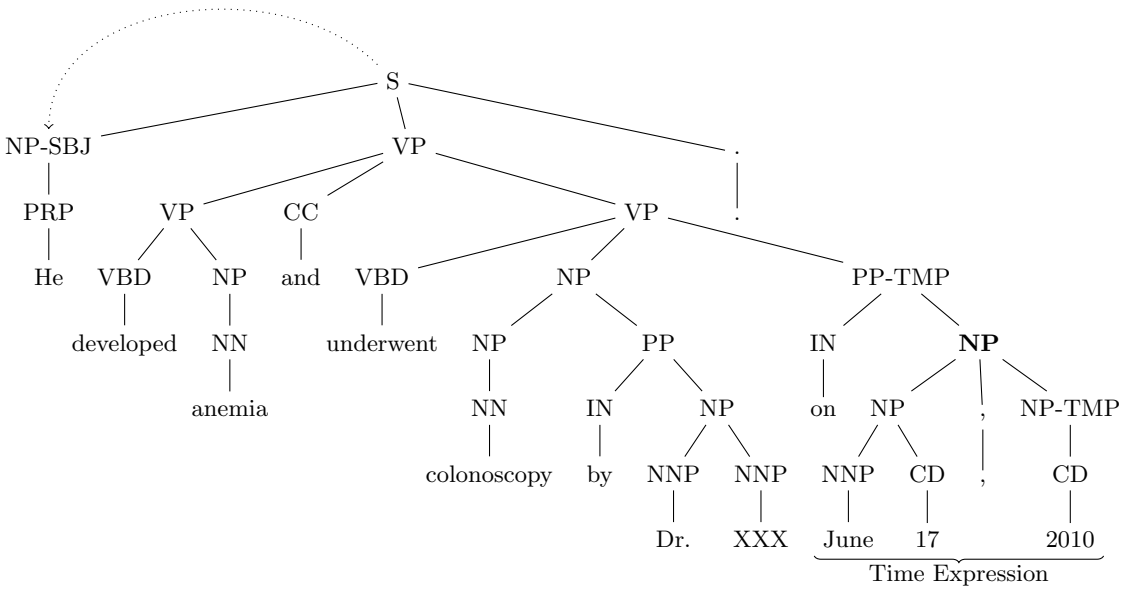

Figure 1: **Example constituency tree containing a time expression.** This sentence contains a single time-expression (*June 17, 2010*), spanned by the bolded NP in the figure. That NP is a single positive training instance, while all other constituents will be negative training instances.

positive or negative instance (described in detail below) can be extracted arbitrarily based on the position of the instance in the tree, but this representation obviously lends itself more to hierarchical, syntax-based features and makes sequence-based features more difficult (though not impossible) to represent.

The appeal of this approach is that time expressions will nearly always be constituents, so the classifier is constrained to select only constituent sequences. This also seems to combine advantages of the systems above, as it gets to consider whole spans at once (like the global optimization of the CRF), while using a simple binary classifier (like the SVM-based BIO taggers). One potential drawback is that it requires high accuracy parsing, at least for constituents composed of temporal expressions.

2.2.3 Ensemble Model

The final model we developed is an ensemble sequence model that is trained on features encapsulating the outputs of the four above systems and making predictions based on those features. The rationale for this model is that our other models differ enough to have varying strengths and weaknesses, and an ensemble system may be able to learn when to select which system. The features at each word in the sequence are the outputs of the component systems at a window of width n around the word. So, for systems i the features at position j in the document are the following set:

$$feats_j = \cup_i \cup_{j'=j-n}^{j+n} \{out_{j'}^i\} \quad (1)$$

where $out_i^j \in \{B, I, O\}$, indicating the output label of system i at position j.

We use a CRF-based tagger for this model – with a much smaller (and thus more learnable) feature space, our intuition is that a globally optimal model should have even more of an advantage over word-by-word discriminative taggers. In preliminary work we found that a window of $n = 1$ gave the best performance on the development set, so the final model was trained using that value.

2.3 Features

To make the comparison fair, we made an effort to have feature parity between systems as much as possible. For the three sequence-based models this was largely accomplished. For the constituency parser-based model the approach is so different that the features do not align perfectly with the other systems, but roughly the same information is present.

Feature Type	Features
Tokens	Word=June,Word=17,Word=COMMA,Word=2010
POS tags	POS=NNP,POS=CD,POS=COMMA,POS=CD
Character classes	Char=LuLlLlLl,CharCollapsed=LuLl,Char=NdNd,CharCollapsed=Nd, Char=Pc,CharCollapsed=Pc,Char=NdNdNdNd,CharCollapsed=Nd
Gazetteer	MonthOfYear,Number,Year
Parse	node=NP,parent=PP,prod=NP→NP-COMMA-NP,root=false,leaf=false

Table 3: Table representing features extracted from the time expression in Figure 1, organized by feature type. The comma character is represented as COMMA so it is not confused with the commas used to separate features. Character classes are explained in the main text.

2.3.1 Sequence Features

Sequence features include lexical features, gazetteer features, syntactic parse features, and section features. Lexical and gazetteer features are both token-based, and are defined for both the current token under consideration (i.e., the one the classifier is currently trying to label) and the three tokens on either side of the token under consideration. These features include token part of speech (POS) tags and two character based features. The part of speech tags are obtained from the cTAKES POS tagger (a clinical data-trained wrapper for the Apache OpenNLP[4] POS tagger). The character-based features map every character in the token to a unicode character category[5], for example, uppercase letter (*Lu*), lowercase letter (*Ll*), decimal digit (*Nd*), etc. This character-mapped token representation is then turned into two features, one in the unmodified format and one where repeats are collapsed. For example, the token "2004" would map to two features: one where its represented as four digit characters (*NdNdNdNd*) and one where the repeats are collapsed (*Nd*).

Gazetteer features rely on a lookup table that contains information about lexical items that are very likely to generalize. We define a small set of temporal word classes and created a gazetteer that maps lexical items to those classes. The set of classes with representative examples is: {Number (numbers up to 200), Year (four-digit numbers that could reasonably appear in current notes), Unit (second, minute), PartOfDay (morning), DayOfWeek (Monday), WeekendOfWeek (weekend), MonthOfYear (January, jan), SeasonOfYear (Summer), DecadeOfCentury (nineties), Time (noon), Age (teenager), TimeReference (previously), Frequency (monthly), Adjuster (next), Modifier (nearly), PrePost (postoperative), TimeSeparator (:)}. The full list of items is too long to list here but will be part of the open source release of this system.

Parse features for the sequence model are not as natural a fit as with a constituency node-based model, but some features can be derived based on spans. With the BIO tagger models (forward and backward) we define a *candidate span* to consider, defined in terms of the forward tagger but easily extendable to the backwards tagger. A candidate span for the current token we are classifying has as its rightmost token the current token, and its leftmost token as the start of the sequence that the current token would be a part of if it is classified as part of a temporal expression. This is simple to find in practice: if the previous token is O (not part of a temporal expression) then the current candidate span is only the current token; otherwise the candidate span starts at the most recent token labeled B (the start of a temporal expression). For the CRF sequence tagger, classification decisions have not yet been made, so the candidate span always covers only the current token.

Given this definition of a candidate span, we define several features. We have one feature for the category of the lowest constituent that dominates the current span, a feature for the parent category of the dominating node, a feature that indicates whether the dominating node is a leaf (preterminal) node, and a feature to indicate whether the dominating node matches the current span exactly.

[4] `http://opennlp.apache.org`

[5] `http://www.unicode.org/reports/tr44/#General_Category_Values`

We then have production-rule associated features. First, we simply represent the production rule of the dominating node as a string (e.g., "NP -> DT NN"). We also use "bag of children" features which represent each of the elements of the right hand side of the production rule, ignoring ordering.

The next feature type is based on surrounding classifications. Here the BIO taggers have access to the previous classification decision (B, I, or O). The CRF in the linear chain configuration can use labels on either side of the current word. While this represents a difference in features available to the systems, it is one that is inherent to the methodology (something that is only possible with CRFs and not with BIO taggers) so we consider this to not violate our goal of feature parity.

The last feature type is specific to the THYME corpus, as it is based on identifiers in the section headers of Mayo Clinic notes. These identifiers are easily extracted with regular expressions and are codes that indicate the purpose of a section (e.g., medications, allergies, etc.). For this feature we simply use the string representing the code for the section that encloses the token under consideration. These are intended to capture the fact that some sections may contain condensed narrative, and are likely to contain time expressions, while others have expressions that resemble time expressions but are not (*5/9* to mean five out of nine).

2.3.2 Constituency Features

The constituency parse-based system attempts to use similar features where possible – we will refer to the features above when possible and point out implementation differences.

First, the features for character class and part of speech for tokens are replicated, by applying them to all the tokens within the span of the current tree node being classified. Gazetteer features are replicated similarly – each word covered by the current tree node is mapped to its time class, if it exists. This is done without reference to ordering.

From the tree itself, we use several features similar to those above, but explicitly based on the tree rather than having to be mapped to the tree. For the current node and its parent, we have features for node category (e.g., NP). For the current node alone we use boolean features for whether it is the root node of the sentence and whether it is a leaf node. We have string features for the bag of children, as well as a feature representing the production rule. Table 3 shows the features that would be extracted for this classifier for the time expression in Figure 1.

2.4 Evaluation

Our evaluation looks at three variables – different machine learning methods, the usefulness of automatic parses at deriving syntactic features, and the domain of the data. For scoring the evaluation, we primarily use a simple scorer built into ClearTK that requires *exact* span matching. We also track partially overlapping spans and count them as correct for *overlapping* span matching. For comparability, we also use the 2012 i2b2 Challenge scoring tool for i2b2 data, which allows both exact and overlapping matching. We use exact span matching as our primary scoring method to conservatively estimate performance, in part because the output of these systems will typically be passed to a time normalization system, which may not be able to handle the variations in input. The metrics we use are precision $\left(\frac{\#correcttimespans}{\#predictedtimespans}\right)$, recall $\left(\frac{\#correcttimespans}{\#goldtimespans}\right)$, and F1 $\left(\frac{2*p*r}{p+r}\right)$.

We look first at multiple methods on two different corpora. In this experiment we are looking to see whether there is any method which is clearly superior to the others, especially across corpora. This experiment is important because methods like the CRF and the CRF-based ensemble have some nice theoretical properties (finding the globally optimal sequence), but as a result have slower run time, and to understand this tradeoff we need to measure performance differences. For the first four systems (the non-ensemble systems), we simply train each method on the combined training and development sets for each corpus, and test on the test set for that corpus.

For the ensemble system, we note that since it is trained on the outputs of other systems, we must do an internal cross-validation of the component systems before performing the tests, to ensure that the labels provided to the ensemble method are representative of what it

	THYME						i2b2 2012 Challenge					
	Exact			Overlapping			Exact			Overlapping		
	P	R	F	P	R	F	P	R	F			
BIO	0.784	0.676	**0.726**	0.948	0.836	0.888	0.775	0.718	0.745	0.921	0.853	0.886
Backwards	0.770	**0.687**	**0.726**	0.948	**0.846**	0.894	0.786	**0.740**	**0.762**	0.917	**0.862**	**0.889**
CRF	**0.788**	0.584	0.671	0.961	0.712	0.818	**0.814**	0.617	0.702	**0.960**	0.728	0.828
Constituency	0.715	0.563	0.630	**0.989**	0.799	0.884	0.657	0.545	0.596	0.920	0.762	0.834
Ensemble	0.784	0.669	0.722	0.962	0.841	**0.897**	0.809	0.706	0.754	0.948	0.828	0.884
Xu et al.(2013)										*0.881*	*0.950*	*0.914*

Table 4: Precision (P), Recall (R), and F1-Score (F) for different systems and corpora. The highest score in each column is in bold. BIO=Begin-Inside-Outside tagger, Backwards=Reverse BIO tagger,CRF=Conditional Random Field tagger,Constituency=Constituency parser-based classifier, Ensemble=CRF-based ensemble classifier. Italicized results from Xu et al. indicated reported, not replicated, results.

will see on test data. We first perform a 5-fold cross validation on the training set, for each fold training the component on four folds and running the trained component on the fifth. The output on that fifth fold forms the training data that the ensemble method will see. By repeating that for each fold, the ensemble method obtains proper system-generated labels from the component system for the entire training set to use as its training data.

The second experiment looks at the importance of accurate syntactic parsing for generating features. For the syntax-focused experiments, we use only the THYME corpus, since it has a layer of gold standard treebank annotations. The tagger we evaluate is the best performing system on the first experiment, the Backwards BIO tagger. In this case we examine three different conditions: First, using gold standard treebank for feature extraction; second, using automatic parses from a THYME-trained parser; and finally, without any syntactic features at all.

The final experiment examines the domain-specificity of the systems and corpora. In this experiment we train the best performing system (Backwards BIO tagger) on THYME data and then test on i2b2 data, and vice versa.

3 Results

Results are shown in Tables 4-6. Table 4 shows the results of the primary experiment – performance of the various systems on both THYME and i2b2 corpora. In most conditions, the Backwards BIO Tagger obtains the highest or tied for the highest F-score, while the regular BIO tagger and ensemble method

	THYME		
	Exact		
	P	R	F
Gold	0.771	0.699	0.733
Automatic	*0.770*	*0.687*	*0.726*
No Syntax	0.773	0.690	0.729

Table 5: Precision (P), recall (R), and F1-Score (F) for different syntactic configurations of the *Backwards* BIO tagger system. Gold - Manually annotated trees from Treebank used for features. Automatic – parser trained on clinical text from THYME Treebank, italicized to denote that it is copied from Table 4 above. No Syntax – Backwards BIO tagger system with no syntactic features.

obtain very competitive F-scores. The Backwards BIO tagger tends to have the best recall of all systems, while preserving precision at a relatively high level. The CRF, despite being theoretically globally optimal, is not competitive in terms of F-score with the SVM-based taggers. The ensemble CRF nominally obtains the best performance in the *Overlapping* metric on the THYME corpus, but the improvement is marginal.

The backwards BIO tagger achieved an F-score of 0.889 on the i2b2 Challenge data allowing for partial matches (the *Overlapping* column). The best performing system in the i2b2 Challenge (Xu et al., 2013) is shown in the last row, with an F1 score of 0.914, with an advantage on recall. Our best system performance would tie for 4th in the span matching part of that challenge, without tuning for that dataset. While we incorporated features based on the best-performing similar system (Xu et al., 2013), including punctuation information,

prepositions, and chunk information, these did not improve performance. Their paper described a larger system and did not contain enough detail on time expression extraction to replicate exactly (e.g., the *Other Keywords* section in the online supplement is not exhaustive and probably is important to their result).

Table 5 shows the results of experiments examining the role of syntactic features on our best performing system, the Backwards BIO tagger. This experiment suggests syntactic features are not valuable for the test set. Neither using gold standard trees for extracting features, nor removing syntactic features altogether, changed performance meaningfully.

Finally, Table 6 shows results of cross-domain experiments, using the best-performing Backwards BIO tagger. Performance falls quite a bit relative to the in-domain trained experiments, even in the relaxed *Overlapping* condition. Training on THYME and testing on i2b2 results in the worst performance, with an exact span matching F1 Score of 0.422.

4 Discussion

While the results are competitive with the best systems at the i2b2 Challenge, they raise many interesting questions.

First, it is very interesting that the best performing systems are the simplest and fastest. Despite the theoretical advantages of the conditional random field's global sequence optimization, the BIO approaches using local classifiers typically obtain the best performance. This is also in contrast with results from the i2b2 Challenge, where the best performing system used a CRF approach. We extensively explored the parameter space for CRFs on development data and our sense was that throughout this entire space performance lagged SVM-based tagging systems.

Next, it is unfortunate but interesting that the ensemble method does not improve performance over the component systems. Error analysis for this system showed both examples where the first word was missed and examples where the last word was missed. The forwards and backwards BIO taggers should be obtaining complementary errors of these types. Thus it is not clear why the ensemble method is unable to take advantage of the information from multiple systems to improve performance.

The syntax-based system shows the biggest gain when switching from the scorer that considers exact spans to one that considers overlapping spans. In preliminary work using gold standard parses, the exact span scores were significantly higher. These two facts suggest that the primary reason for the low accuracy of this model on exact spans is parsing errors. This was, in fact, one motivation for incorporating parser features – if the parser cannot reliably find exact spans, perhaps it is still possible to use its output at word levels to find patterns that a sequence-based model could use.

The lack of improvements with syntactic features in these experiments is therefore somewhat confusing, as using a totally syntax-based system is able to obtain decent performance. One hypothesis for their lack of impact is that annotation consistency plays a role. We noticed that annotations of time expressions around prepositional phrases are inconsistent. For example, in the prepositional phrase *since Tuesday*, the time expression is only *Tuesday*, but in some cases the whole PP is annotated in the gold standard. This may help explain the large jump in performance when partially overlapping spans are included, as there are many errors that are off by only an added or dropped preposition at the start of the time expression. (Note that this explanation may also apply to the constituency parser model.)

The cross-corpus performance (training on THYME and testing on i2b2 and vice versa) is surprisingly low. While the annotation guidelines are similar, one major difference is the addition of the *prepostexp* class to THYME, for expressions like *postoperative*. Meanwhile, i2b2 challenge data annotates expressions like *postoperative day 5*, which do not occur in THYME data, as the *date* class. This affects both recall and precision on the THYME to i2b2 evaluation as expressions like *postoperative day 5* cause both recall errors (not getting the whole expression) and precision errors (predicting the first word of the expression). Additional errors in this direction are caused by unseen abbreviations in THYME that are common in i2b2 (*POD* for postoperative day,

		Exact			Overlapping		
Train Corpus	Test Corpus	P	R	F	P	R	F
THYME	THYME	*0.770*	*0.687*	*0.726*	*0.948*	*0.846*	*0.894*
i2b2	i2b2	*0.786*	*0.740*	*0.762*	*0.917*	*0.862*	*0.889*
THYME	i2b2	0.436	0.410	0.422	0.722	0.679	0.700
i2b2	THYME	0.589	0.432	0.498	0.860	0.629	0.727

Table 6: Precision (P), recall (R), and F1-Score (F) for cross-domain experiments. We use the best-performing system for each experiment (Backwards BIO), with automatic parse features from a THYME-trained parser. Italicized rows are copied from Table 4 above for ease of comparison.

x 2 for twice a day). In the other direction (train on i2b2, test on THYME), recall errors are worst because it does not correctly identify any of the prepostop expressions. Surprisingly, in both directions there are relatively simple date formats missed due to slight differences in convention – THYME data often uses month names (e.g. Jan 5, 2014) while i2b2 typically does not, while i2b2 uses MM-YY format (e.g., March 7 represented as 05-07) while THYME does not. This suggests that a better system could be obtained by training on both corpora, although this will require some reconciliation of the differences in time classes, primarily what THYME calls *prepostexp* time expressions.

Errors on the best-performing system are primarily those where the start or end of the time expression is off by one. As above, these may be partially due to inconsistent prepositional phrase annotation, and the effect of fixing this is roughly seen in the overlapping scoring criterion. The remaining errors probably represent the most opportunity for system improvement, so we focus on that briefly.

One common issue occurs with coordination – phrases like *2003 or 2004*. While these are annotated as a single span, the system will get the two individual years, resulting in one recall error but two precision errors. This type of error might be fixed by a second pass that joins together time expressions connected by coordinators. A modified syntactic approach that operates bottom-up instead of top-down might also correctly recognize such expressions. Another source of error is in expressions that are unusually expressed in a few instances, such as *times three* to mean something happened three times. While this is *in* the training data, it is not the primary way of indicating this meaning, and there are not enough instances to learn this modification. Similarly, sometimes punctuation is inserted or modified into an expression that slightly changes its representation to the classification algorithm (e.g., *one-year* with a dash rather than *one year*). Fixing this issue in a general way is a tricky problem, as it is related to the larger issue of there being many ways to instantiate any given concept (*half past 7*, *half of 8*, *7:30*, etc.). In the clinical domain one hopes that usage is a bit more constrained and that one might be able to get away with a simpler approach such as just ignoring punctuation.

In conclusion, we have presented and evaluated multiple machine learning methods for temporal expression extraction. Our results suggest that simpler and faster BIO sequence tagger methods are as good as more complex models or ensemble methods. We also show that deep syntax does not seem beneficial to this task. Finally, we show that there is significance performance degradation when applying to new corpora, despite similar annotation guidelines and domains.

5 Acknowledgments

The project described was supported by R01LM010090 (THYME) from the National Library Of Medicine. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

Steven Bethard, Philip Ogren, and Lee Becker. 2014. Cleartk 2.0: Design patterns for machine learning in uima. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International*

Conference on Language Resources and Evaluation (LREC'14), pages 3289–3293, Reykjavik, Iceland, May. European Language Resources Association (ELRA).

Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 10–14, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Aleksandar Kovačević, Azad Dehghan, Michele Filannino, John A Keane, and Goran Nenadic. 2013. Combining rules and machine learning for extraction of temporal expressions and events from clinical narratives. *Journal of the American Medical Informatics Association*, 20(5):859–866.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Timothy Miller, Steven Bethard, Dmitriy Dligach, Sameer Pradhan, Chen Lin, and Guergana Savova. 2013. Discovering temporal narrative containers in clinical text. In *Proceedings of the 2013 Workshop on Biomedical Natural Language Processing*, pages 18–26, Sofia, Bulgaria, August. Association for Computational Linguistics.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (CRFs). `http://www.chokkan.org/software/crfsuite/`.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*, volume 2003, page 40.

Preethi Raghavan, Eric Fosler-Lussier, and Albert M Lai. 2012. Temporal classification of medical events. In *Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, pages 29–37. Association for Computational Linguistics.

Guergana K. Savova, Wendy W. Chapman, Jiaping Zheng, and Rebecca S. Crowley. 2011. Anaphoric relations in the clinical narrative: corpus creation. *J Am Med Inform Assoc*, 18:459–465.

Sunghwan Sohn, Kavishwar B Wagholikar, Dingcheng Li, Siddhartha R Jonnalagadda, Cui Tao, Ravikumar Komandur Elayavilli, and Hongfang Liu. 2013. Comprehensive temporal information detection from clinical text: medical events, time, and tlink identification. *Journal of the American Medical Informatics Association*, pages amiajnl–2013.

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

Jannik Strötgen, Julian Zell, and Michael Gertz. 2013. Heideltime: Tuning english and developing spanish resources for tempeval-3. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 15–19, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

William F. Styler IV, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C de Groen, Brad Erickson, Timothy Miller, Chen Lin, Guergana Savova, and James Pustejovsky. 2014. Temporal annotation in the clinical domain. *Transactions of the Association for Computational Linguistics*, 2:143–154.

Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. 2013. Evaluating temporal relations in clinical text: 2012 i2b2 challenge. *Journal of the American Medical Informatics Association*, 20(5):806–813.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 75–80, Prague, Czech Republic,

June. Association for Computational Linguistics.

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

Yan Xu, Yining Wang, Tianren Liu, Junichi Tsujii, and Eric I-Chao Chang. 2013. An end-to-end system to identify temporal relation in discharge summaries: 2012 i2b2 challenge. *Journal of the American Medical Informatics Association : JAMIA*.