Feature Engineering and Post-Processing for Temporal Expression Recognition Using Conditional Random Fields

Sisay Fissaha Adafre **Maarten de Rijke**
Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
sfissaha,mdr@science.uva.nl

Abstract

We present the results of feature engineering and post-processing experiments conducted on a temporal expression recognition task. The former explores the use of different kinds of tagging schemes and of exploiting a list of core temporal expressions during training. The latter is concerned with the use of this list for post-processing the output of a system based on conditional random fields.

We find that the incorporation of knowledge sources both for training and post-processing improves recall, while the use of extended tagging schemes may help to offset the (mildly) negative impact on precision. Each of these approaches addresses a different aspect of the overall recognition performance. Taken separately, the impact on the overall performance is low, but by combining the approaches we achieve both high precision and high recall scores.

1 Introduction

Temporal expressions (*timex*es) 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 a text. Timex recognition is a named entity recognition (NER) task to which a variety of natural language processing and machine learning techniques have been applied. As with other NER tasks, timex recognition is naturally viewed as a sequence labeling task, easily lending itself to machine learning techniques such as conditional random fields (CRFs) (Lafferty et al., 2001).

A preliminary experiment showed that, using CRFs, a respectable recognition performance can easily be achieved with a straightforward baseline system that is based on a simple tagging scheme and requires very little tuning, yielding F-scores around 0.78 (exact match) or even 0.90 (partial match). Interestingly, these high scores are mainly due to high or even very high precision scores, while recall leaves much to be improved.

The main focus of this paper is on boosting recall while maintaining precision at an acceptable (i.e., high) level. We report on two types of experiments aimed at achieving this goal. One type concerns feature engineering and the other concerns post-processing the output of a machine learner. While we do exploit the special nature of timexes, for portability reasons we avoid using task-specific and richer linguistic features (POS, chunks, etc.). Instead, we focus on features and techniques that can readily be applied to other NER tasks.

Specifically, our feature engineering experiments have two facets. The first concerns identification of a set of simple features that results in high generalization ability (accuracy). Here, particular emphasis will be placed on the use of a list of core timexes as a feature. The assumption is that the performance of data-driven approaches for timex recognition can be improved by taking into account the peculiar properties of timexes. Timexes exhibit various patterns, ranging from regular patterns that can easily be captured using simple regular expressions to complex linguistic forms (phrases). While timexes are real-

Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in NLP, pages 9–16,
Ann Arbor, June 2005.

ized in different phrase types, the core lexical items of timexes are restricted. This suggests that a list of core timexes can easily be compiled and used in machine learning-based timex recognition. One approach of integrating such a list is using them to generate features, but the availability of such a list also opens up other possibilities in feature design that we present in later sections.

The second aspect concerns the tagging scheme. As in most NER experiments, the task of recognizing timexes is reduced to tagging. Commonly used tagging schemes are Inside-Outside (IO) and Begin-Continue-End-Unique-Negative (BCEUN) (Borthwick et al., 1998). The IO tagging scheme, which we use as a baseline, assigns the tag I to a token if it is part of a timex and O otherwise. The richer BCEUN scheme assigns the five tags B, C, E, U, and N to tokens depending on whether the token is single-token timex (U), a non-timex (N), appears at the beginning (B), at the end (E) or inside a timex boundary (C). In this paper, we compare the IO, BCEUN and an extended form of the BCEUN tagging scheme. The extended scheme adds two tags, PRE and POST, to the BCEUN scheme, which correspond to tokens appearing to the left and to the right of a timex.

In contrast, our post-processing experiments investigate the application of the list of core timexes for filtering the output of a machine learner. The incorporation into the recognition process of explicit knowledge in the form of a list for post-processing requires a carefully designed strategy to ensure that the important properties of the trained model are kept intact as much as possible while at the same-time improving overall results. We present an approach for using a list for post-processing that exploits the knowledge embodied in the trained model.

The paper is organized as follows. In Section 2 we provide background material, both on the timex extraction task (§2.1) and on the machine learning techniques on which we build in this paper, conditional random fields (§2.2). Our ideas on engineering feature sets and tagging schemes are presented in Section 3, while we describe our method for exploiting the explicit knowledge contained in a list in Section 4. In Section 5, we describe the experimental setup and present the results of our experiments. Related work is briefly reviewed in Section 6, and we conclude in Section 7.

2 Background

2.1 Task Description

In recent years, temporal aspects of information access have received increasing amounts of attention, especially as it relates to news documents. In addition to factual content, news documents have a temporal context, reporting events that happened, are happening, or will happen in relation to the publication date. Temporal document *retrieval* concerns the inclusion of both the document publication date and the in-text temporal expressions in the retrieval model (Kalczynski and Chou, 2005). The task in which we are interested in this paper is identifying the latter type of expressions, i.e., *extraction* of temporal expressions. TERN, the Temporal Expression Recognition and Normalization Evaluation, is organized under the auspices of the Automatic Content Extraction program (ACE, `http://www.nist.gov/speech/tests/ace/`). The TERN evaluation provides specific guidelines for the identification and normalization of timexes, as well as tagged corpora for training and testing and evaluation software. These guidelines and resources were used 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. Our focus in this paper is on the recognition task; it is defined, for human annotators, in the TIDES TIMEX2 annotation guidelines (Ferro et al., 2004). The recognition task is 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 the recognition performance is provided as part of the TERN evaluation. It computes precision, recall, and F-measure both for TIMEX2 tags (i.e., for *overlap* with a gold standard TIMEX2 element) and for *extent* of TIMEX2 elements (i.e., exact match of entire timexes).

2.2 Conditional Random Fields

We view the recognition of timexes task as a sequence labeling task in which each token in the text

is classified as being either a timex or not. One machine learning technique that has recently been introduced to tackle the problem of labeling and segmenting sequence data is conditional random fields (CRFs, (Lafferty et al., 2001)). CRFs are conditional probability distributions that take the form of exponential models. The special case of linear chain CRFs, which takes the following form, has been widely used for sequence labeling tasks:

$$P(y \mid x) = \frac{1}{Z(x)} \exp\left(\sum_{t=1}\sum_{k} \lambda_k f_k(t, y_{t-1}, y_t, x)\right),$$

where $Z(x)$ is the normalization factor, $X = \{x_1, \ldots, x_n\}$ is the observation sequence, $Y = \{y_1, \ldots, y_T\}$ is the label sequences, f_k and λ_k are the feature functions and their weights respectively. An important property of these models is that probabilities are computed based on a set of feature functions, i.e., f_k (usually binary valued), which are defined on both the observation X and label sequences Y. These feature functions describe different aspect of the data and may overlap, providing a flexible way of describing the task.

CRFs have been shown to perform well in a number of natural language processing applications, such as POS tagging (Lafferty et al., 2001), shallow parsing or NP chunking (Sha and Pereira, 2003), and named entity recognition (McCallum and Li, 2003). In this paper, CRFs are applied to the recognition of timexes; in our experiments we used the minorThird implementation of CRFs (Cohen, 2004).

3 Feature Engineering

The success of applying CRFs depends on the quality of the set of features used and the tagging scheme chosen. Below, we discuss these two aspects in greater detail.

3.1 Feature sets

Our baseline feature set consists of simple lexical and character features. These features are derived from a context window of two words (left and right). Specifically, the features are the lowercase form of all the tokens in the span, with each token contributing a separate feature, and the tokens in the left and right context window constitute another set of features. These feature sets capture the lexical content and context of timexes. Additionally, character type pattern features (such as capitalization, digit sequence) of tokens in the timexes are used to capture the character patterns exhibited by some of the tokens in temporal expressions. These features constitute the *basic feature* set.

Another important feature is the list of core timexes. The list is obtained by first extracting the phrases with -TMP function tags from the PennTree bank, and taking the words in these phrases (Marcus et al., 1993). The resulting list is filtered for stopwords. Among others, the list of core timexes consists of the names of days of the week and months, temporal units 'day,' 'month,' 'year,' etc. This list is used to generate binary features. In addition, the list is used to guide the design of other complex features that may involve one or more of token-tag pairs in the context of the current token. One way of using the list for this purpose is to generate a feature that involves bi-grams tokens. In certain cases, information extracted from bi-grams, e.g. +Xx 99 (May 20), can be more informative than information generated from individual tokens. We refer to these features as the *list* feature set.

3.2 Tagging schemes

A second aspect of feature engineering that we consider in this paper concerns different tagging schemes. As mentioned previously, the task of recognizing timexes is reduced to a sequence-labeling task. We compare three tagging schemes, IO (our baseline), BCEUN, and BCEUN+PRE&POST. While the first two are relatively standard, the last one is an extension of the BCEUN scheme. The intuition underlying this tagging scheme is that the most relevant features for timex recognition are extracted from the immediate context of the timex, e.g., the word 'During' in (1) below.

(1) During <TIMEX2>the past week</TIMEX2>, the storm has pounded the city.
During-PRE the-B past-C week-E ,-POST the storm has pounded the city.

Therefore, instead of treating these elements uniformly as outside (N), which ignores their relative importance, we conjecture that it is worthwhile to

assign them a special category, like PRE and POST corresponding to the tokens immediately preceding and following a timex, and that this leads to improved results.

4 Post-processing Using a List

In this section, we describe the proposed method for incorporating a list of core lexical timexes for post-processing the output of a machine learner. As we will see below, although the baseline system (with the IO tagging scheme and the basic feature set) achieves a high accuracy, the recall scores leave much to be desired. One important problem that we have identified is that timexes headed by core lexical items on the list may be missed. This is either due to the fact that some of these lexical items are semantically ambiguous and appear in a non-temporal sense, or the training material does not cover the particular context. In such cases, a reliable list of core timexes can be used to identify the missing timexes. For the purposes of this paper, we have created a list containing mainly headwords of timexes. These words are called *trigger words* since they are good indicators of the presence of temporal expressions.

How can we use trigger words? Before describing our method in some detail, we briefly describe a more naive (and problematic) approach. Observe that trigger words usually appear in a text along with their complements or adjuncts. As a result, picking only these words will usually contribute to token recall but span precision is likely to drop. Furthermore, there is no principled way of deciding which one to pick (semantically ambiguous elements will also be picked). Let's make this more precise. The aim is to take into account the knowledge acquired by the trained model and to search for the next optimal sequence of tags, which assigns the missed timex a non-negative tag. However, searching for this sequence by taking the whole word sequence is impractical since the number of possible tag sequences (number of all possible paths in a viterbi search) is very large. But if one limits the search to a window of size n ($n < 6$), sequential search will be feasible. The method, then, works on the output of the system. We illustrate the method by using the example given in (2) below.

(2) The chairman arrived in the city yesterday, and will leave next week. The press conference will be held tomorrow afternoon.

Now, assume that (2) is a test instance (a two-sentence document), and that the system returns the following best sequence (3). For readability, the tag N is not shown on the words that are assigned negative tags in all the examples below.

(3) The chairman arrived in the city yesterday-U , and will leave next week . The press conference will be held tomorrow-B afternoon-E .

According to (3), the system recognizes only 'yesterday' and 'tomorrow afternoon' but misses 'next week'. Assuming our list of timexes contains the word 'week', it tells us that there is a missing temporal expression, headed by 'week.' The naive method is to go through the above output sequence and change the token-tag pair 'week-N' to 'week-U'. This procedure recognizes the token 'week' as a valid temporal expression, but this is not correct: the valid temporal expression is 'next week'.

We now describe a second approach to incorporating the knowledge contained in a list of core lexical timexes as a post-processing device. To illustrate our ideas, take the complete sequence in (3) and extract the following segment, which is a window of 7 tokens centered at 'week'.

(4) ...[will leave next week . The press] ...

We *reclassify* the tokens in (4) assuming the history contains the token 'and' (the token which appears to the left of this segment in the original sequence) and the associated parameters. Of course, the best sequence will still assign both 'next' and 'week' the N tag since the underlying parameters (feature sets and the associated weights) are the same as the ones in the system. However, since the word sequence in (4) is now short (contains only 7 words) we can maintain a list of all possible tag sequences for it and perform a sequential search for the next best sequence, which assigns the 'week' token a non-negative tag. Assume the new tag sequence looks as follows:

(5) ...[will leave next-B week-E . The press] ...

This tag sequence will then be placed back into the original sequence resulting in (6):

(6) The chairman arrived in the city yesterday-U , and will leave next-B week-E . The press conference will be held tomorrow-B afternoon-E .

In this case, all the temporal expressions will be extracted since the token sequence 'next week' is properly tagged. Of course, the above procedure can also return other, invalid sequences as in (7):

(7) a. . . . will leave next-B week-C . The press . . .
b. . . . will leave next week-C . The press . . .
c. . . . will leave next week-C .-E The press . . .

The final extraction step will not return any timex since none of the candidate sequences in (7) contains a valid tag sequence. The assumption here is that of all the tag sequences, which assign the token 'week' a non-negative tag, those tag sequences which contain the segment 'next-B week-E' are likely to receive a higher weight since the underlying system is trained to recognize temporal expressions and the phrase 'next week' is a likely temporal expression.

This way, we hypothesize, it is possible to exploit the knowledge embodied in the trained model. As pointed out previously, simply going through the list and picking only head words like 'week' will not guarantee that the extracted tokens form a valid temporal expression. On the other hand, the above heuristics, which relies on the trained model, is likely to pick the adjunct 'next'.

The post-processing method we have just outlined boils down to reclassifying a small segment of a complete sequence using the same parameters (feature sets and associated weights) as the original model, and keeping all possible candidate sequences and searching through them to find a valid sequence.

5 Experimental Evaluation

In this section we provide an experimental assessment of the feature engineering and post-processing methods introduced in Sections 3 and 4. Specifically, we want to determine what their impact is on the precision and recall scores of the baseline system, and how they can be combined to boost recall while keeping precision at an acceptable level.

5.1 Experimental data

The training data consists of 511 files, and the test data consists of 192 files; these files were made available in the 2004 Temporal Expression Recognition and Normalization Evaluation. The temporal expressions in the training files are marked with XML tags. The minorThird system takes care of automatically converting from XML format to the corresponding tagging schemes. 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.

5.2 Experimental results

Richer feature sets Table 1 lists the results of the first part of our experiments. Specifically, for every tagging scheme, there are two sets of features, *basic* and *list*. The results are based on both exact-match and partial match between the spans in the gold standard and the spans in the output of the systems, as explained in Subsection 2.1. In both the exact and partial match criteria, the addition of the list features leads to an improvement in recall, and no change or a decrease in precision.

In sum, the feature addition helps recall more than it hurts precision, as the F score goes up nearly everywhere, except for the exact-match/baseline pair.

Tagging schemes In Table 1 we also list the extraction scores for the tagging schemes we consider, IO, BCEUN, and BCEUN+PRE&POST, as described in Section 3.2.

Let us first look at the impact of the different tagging schemes in combination with the basic feature set (rows 3, 5, 7). As we go from the baseline tagging scheme IO to the more complex BCEUN and BCEUN+PRE&POS, precision increases on the exact-match criterion but remains almost the same on the partial match criterion. Recall, on the other hand, does not show the same trend. BCEUN has the highest recall values followed by BCEUN+PRE&POST and finally IO. In general, IO based tagging seems to perform worse whereas BCEUN based tagging scores slightly above its extended tagging scheme BCEUN+PRE&POST.

Next, considering the combination of extending the feature set and moving to a richer tagging scheme (rows 4, 6, 8), we have very much the same pattern. In both the exact match and the partial match setting, BCEUN tops (or almost tops) the two

		Exact Match			Partial Match		
Tagging scheme	Features	Prec.	Rec.	F	Prec.	Rec.	F
IO (baseline)	basic	0.846	0.723	0.780	0.973	0.832	0.897
	basic + list	0.822	0.736	0.776	0.963	0.862	0.910
BCEUN	basic	0.874	0.768	0.817	0.974	0.856	0.911
	basic + list	0.872	**0.794**	**0.831**	0.974	**0.887**	**0.928**
BCEUN+PRE&POS	basic	**0.882**	0.749	0.810	**0.979**	0.831	0.899
	basic + list	0.869	0.785	0.825	0.975	0.881	0.925

Table 1: Timex: Results of training on basic and list features, and different tagging schemes. Highest scores (Precision, Recall, F-measure) are in bold face.

other schemes in both precision and recall.

In sum, the richer tagging schemes function as precision enhancing devices. The effect is clearly visible for the exact-match setting, but less so for partial matching. It is not the case that the learner trained on the richest tagging scheme outperforms all learners trained with poorer schemes.

Post-processing Table 2 shows the results of applying the post-processing method described in Section 4. One general pattern we observe in Table 2 is that the addition of the list features improves precision for IO and BCEUN tagging scheme and shows a minor reduction in precision for BCEUN+PRE&POS tagging scheme in both matching criteria. Similarly, in the presence of post-processing, the use of a more complex tagging scheme results in a better precision. On the other hand, recall shows a different pattern. The addition of list features improves recall both for BCEUN and BCEUN+PRE&POS, but hurts recall for the IO scheme for both matching criteria.

Comparing the results in Table 1 and Table 2, we see that post-processing is a recall enhancing device since all the recall values in Table 2 are higher than the recall values in Table 1. Precision values in Table 2, on the other hand, are lower than those of Table 1. Importantly, the use of a more complex tagging scheme such as BCEUN+PRE&POS, allows us to minimize the drop in precision. In general, the best result (on partial match) in Table 1 is achieved through the combination of BCEUN and basic&list features whereas the best result in Table 2 is achieved by the combination of BCEUN+PRE&POS and basic &list features. Although both have the same overall scores on the exact match criteria, the latter performs better on partial match criteria. This, in turn, shows that the combination of post-processing, and BCEUN+PRE&POS achieves better results.

Stepping back We have seen that the extended tagging scheme and the post-processing methods improve on different aspects of the overall performance. As mentioned previously, the extended tagging scheme is both recall and precision-oriented, while the post-processing method is primarily recall-oriented. Combining these two methods results in a system which maintains both these properties and achieves a better overall result. In order to see how these two methods complement each other it is sufficient to look at the highest scores for both precision and recall. The extended tagging scheme with basic features achieves the highest precision but has relatively low recall. On the other hand, the simplest form, the IO tagging scheme and basic features with post-processing, achieves the highest recall and the lowest precision in partial match. This shows that the IO tagging scheme with basic features imposes a minimal amount of constraints, which allows for most of the timexes in the list to be extracted. Put differently, it does not discriminate well between the valid vs invalid occurrences of timexes from the list in the text. At the other extreme, the extended tagging scheme with 7 tags imposes strict criteria on the type of words that constitute a timex, thereby restricting which occurrences of the timex in the list count as valid timexes.

In general, although the overall gain in score is limited, our feature engineering and post-processing efforts reveal some interesting facts. First, they show one possible way of using a list for post-processing.

		Exact Match			Partial Match		
Tagging scheme	Features	Prec.	Rec.	F	Prec.	Rec.	F
IO	basic (baseline)	0.846	0.723	0.780	**0.973**	0.832	0.897
	basic	0.756	0.780	0.768	0.902	**0.931**	0.916
	basic + list	0.772	0.752	0.762	0.930	0.906	0.918
BCEUN	basic	0.827	0.789	0.808	0.945	0.901	0.922
	basic + list	0.847	0.801	0.823	0.958	0.906	0.931
BCEUN+PRE&POS	basic	0.863	0.765	0.811	**0.973**	0.863	0.915
	basic + list	**0.861**	**0.804**	**0.831**	0.970	0.906	**0.937**

Table 2: Timex: Results of applying post-processing on the systems in Table 1. The baseline (from Table 1) is repeated for ease of reference; it does not use post-processing. Highest scores (Precision, Recall, F-measure) are in bold face.

This method is especially appropriate for situations where better recall is important. It offers a means of controlling the loss in precision (gain in recall) by allowing a systematic process of recovering missing expressions that exploits the knowledge already embodied in a probabilistically trained model, thereby reducing the extent to which we have to make random decisions. The method is particularly sensitive to the criterion (the quality of the list in the current experiment) used for post-processing.

6 Related Work

A large number of publications deals with extraction of temporal expressions; the task is often treated as part of a more involved task combining recognition and normalization of timexes. As a result, many timex interpretation systems are a mixture of both rule-based and machine learning approaches (Mani and Wilson, 2000). This is partly due to the fact that timex recognition is more amenable to data-driven methods whereas normalization is best handled using primarily rule-based methods. We focused on machine learning methods for the timex recognition task only. See (Katz et al., 2005) for an overview of methods used for addressing the TERN 2004 task.

In many machine learning-based named-entity recognition tasks dictionaries are used for improving results. They are commonly used to generate binary features. Sarawagi and Cohen (2004) showed that semi-CRFs models for NE recognition perform better than conventional CRFs. One advantage of semi-CRFs models is that the units that will be tagged are segments which may contain one or more tokens, rather than single tokens as is done in conventional CRFs. This in turn allows one to incorporate segment based-features, e.g., segment length, and also facilitates integration of external dictionaries since segments are more likely to match the entries of an external dictionary than tokens. In this paper, we stuck to conventional CRFs, which are computationally less expensive, and introduced post-processing techniques, which takes into account knowledge embodied in the trained model.

Kristjannson et al. (2004) introduced constrained CRFs (CCRFs), a model which returns an optimal label sequence that fulfills a set of constraints imposed by the user. The model is meant to be used in an interactive information extraction environment, in which the system extracts structured information (fields) from a text and presents it to the user, and the user makes the necessary correction and submits it back to the system. These corrections constitute an additional set of constraints for CCRFs. CCRFs re-computes the optimal sequence by taking these constraints into account. The method is shown to reduce the number of user interactions required in validating the extracted information. In a very limited sense our approach is similar to this work. The list of core lexical timexes that we use represents the set of constraints on the output of the underlying system. However, our method differs in the way in which the constraints are implemented. In our case, we take a segment of the whole sequence that contains a missing timex, and reclassify the words in this segment while keeping all possible tag sequences sorted based on their weights. We then

search for the next optimal sequence that assigns the missing timex a non-negative tag sequentially. On the other hand, Kristjannson et al. (2004) take the whole sequence and recompute an optimal sequence that satisfies the given constraints. The constraints are a set of states which the resulting optimal sequence should include.

7 Conclusion

In this paper we presented different feature engineering and post-processing approaches for improving the results of timex recognition task. The first approach explores the different set of features that can be used for training a CRF-based timex recognition system. The second investigates the effect of the different tagging scheme for timex recognition task. The final approach we considered applies a list of core timexes for post-processing the output of a CRF system. Each of these approaches addresses different aspects of the overall performance. The use of a list of timexes both during training and for post-processing resulted in improved recall whereas the use of a more complex tagging scheme results in better precision. Their individual overall contribution to the recognition performances is limited or even negative whereas their combination resulted in substantial improvements over the baseline.

While we exploited the special nature of timexes, we did avoid using linguistic features (POS, chunks, etc.), and we did so for portability reasons. We focused exclusively on features and techniques that can readily be applied to other named entity recognition tasks. For instance, the basic and list features can also be used in NER tasks such as PERSON, LOCATION, etc. Moreover, the way that we have used a list of core expressions for post-processing is also task-independent, and it can easily be applied for other NER tasks.

Acknowledgments

Sisay Fissaha Adafre was supported by the Netherlands Organization for Scientific Research (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

[Borthwick et al.1998] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Workshop on Very Large Corpora, ACL*.

[Cohen2004] W. Cohen. 2004. Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. `http://minorthird.sourceforge.net`.

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

[Kalczynski and Chou2005] P.J. Kalczynski and A. Chou. 2005. Temporal document retrieval model for business news archives. *Information Processing and Management*, 41:635–650.

[Katz et al.2005] G. Katz, J. Pustejovsky, and F. Schilder, editors. 2005. *Proceedings Dagstuhl Workshop on Annotating, Extracting, and Reasoning about Time and Events*.

[Kristjannson et al.2004] T. Kristjannson, A. Culotta, P. Viola, and A. McCallum. 2004. Interactive information extraction with constrained conditional random fields. In *Nineteenth National Conference on Artificial Intelligence, AAAI*.

[Lafferty et al.2001] J. Lafferty, F. Pereira, and A. McCallum. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*.

[Mani and Wilson2000] I. Mani and G. Wilson. 2000. Robust temporal processing of news. In *Proceedings of the 38th ACL*.

[Marcus et al.1993] M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330.

[McCallum and Li2003] A. McCallum and W. Li. 2003. Early results for Named Entity Recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the 7th CoNLL*.

[Sarawagi and Cohen2004] S. Sarawagi and W.W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In *NIPs (to appear)*.

[Sha and Pereira2003] F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology-NAACL*.