SVMs for the Temporal Expression Chunking Problem

Jordi Poveda Poveda
jpoveda@lsi.upc.edu

Mihai Surdeanu
surdeanu@lsi.upc.edu

LSI Department Technical Report
Ph.D. Programme on Artificial Intelligence (UPC)
November 2006

Contents

1 The Temporal Expression Chunking Problem

Chunking refers to any problem or task in NLP (natural language processing) which involves segmenting (i.e. identifying the boundaries of) the occurrences in a text of a certain type of entities. This gives rise to different types of chunking problems, for instance:

- Syntantic chunking, where the aim is to identify syntactic phrases (e.g. noun phrases, verb phrases, prepositional phrases, ...).
- Named entity recognition, where mentions in text of certain types of concepts that have a proper name designation (hence "named entity") are sought, for example: persons, locations, currency, organizations...
- Clause splitting (i.e. a group of words consisting of a subject and a predicate).

A related task, which may accompany chunking, is that of *classification* of the segmented chunks into appropriate types (e.g. PER for person, LOC for location, ORG for organization, etc., in the case of named entities).

Chunking of *temporal expressions* is a part of the more complex task called *temporal expression recognition and normalization* (TERN), in which the aim is to identify the mentions in a text of expressions that denote time, and to capture their meaning by writing them in a canonical representation. A common way to represent these temporal denoting expressions in text —as well as other entities of interest in information extraction, like named entities, events or relations— is to employ XML tags with convenient attributes to further qualify the entity. For example:

```
But even on <TIMEX2 VAL="1999-07-22">Thursday</TIMEX2>, there
were signs of potential battles <TIMEX2 VAL="FUTURE_REF"
ANCHOR_DIR="AFTER" ANCHOR_VAL="1999-07-22">ahead</TIMEX2>.
```

The TIMEX standard sets out guidelines for annotating and normalizing mentions of time expressions. These and a full account of the complex bestiary of temporal expressions can be found in [FER03].

There are two separate problems in the TERN task, one of identifying the *extension* of the mentions of temporal expressions in text (i.e. chunking) and the considerably more difficult problem of automatically normalizing the value of the temporal expression. Here we are only concerned with the first of these two problems. Contrary to what it may at first seem, temporal expressions come into a wide variety of forms ([WIL01]):

- Fully-specified time references: *16th June 2006*, *the twentieth century, Monday at 3pm.*
- Expressions that depend on a context: *last month*, *three days from today*, *February last year.*
- Anaphorical expressions and expressions relative to the time when the expression is written: *that day*, *yesterday*, *currently*, *then.*
- Durations or intervals: *a month*, *three days*, *some hours in the afternoon.*
- Frequencies or recurring times: *monthly*, *every other day, once a week*, *every first Sunday of a month.*
- Culturally dependent time denominations: *Easter*, *the month of Ramadan*, *St. Valentine.*
- Fuzzy or vaguely specified time references: *the future*, *some day*, *eventually*, *anytime you so desire.*

The chunking-related problems have traditionally been represented as the multiple classification problem of assigning *I/O/B tags* to each token in a sequence of text. The tags stand for Inside (I), Outside (O) and Begin (B), and are enough for delimiting non-overlapping, non-recursive chunks (i.e. chunks that neither appear inside the boundaries of the extension of another larger chunk, nor overlap with another chunk). If the task involves further discriminating the chunks into several classes, the I and B tags can be appended with a suffix to signify the type of the chunk (e.g B-PER, I-PER, B-LOC, I-LOC, B-ORG, I-ORG, O for the named entity recognition task). In the case of temporal expression chunking, the result looks like this:

```
In/O 2006/B ,/O thirty/B years/I after/O Mao/B Tse/I Tung/I
's/I death/I ,/O hordes/O of/O devote/O Chinese/O arrive/O
still/O daily/B to/O his/O memorial/O site/O ./O
```

For our experiments, we use the corpus of the ACE 2005 competition (Automatic Content Extraction) for training and testing. This corpus has the mentions of time expressions manually annotated using TIMEX tags. The composition of the corpus is 550 documents distributed in five categories (newswire, broadcast news, broadcast conversations, conversational telephone speech and weblogs), containing 257000 tokens and approx. 4650 time expression mentions.

Regarding evaluation, we are interested in three measures for quantifying the performance of the temporal expression recognizer:

Precision: The rate of returned temporal expressions that are correctly identified (i.e. correctly tagged divided by total tagged).

Recall: The rate of existing temporal expressions that are correctly identified (i.e. correctly tagged divided by those that should have been tagged).

F-1 Score: It is the harmonic mean of the two previous values, $F_1 = \frac{2 \times Precision \times Recall}{(Precision + Recall)}$.

Note that only temporal expression mentions whose boundaries are correctly identified count positively towards the precision and recall scores. Partial matches (a temporal expression that is identified partially, or misplaced) are not regarded as hits. We are not counting the rate of correct assignments of I/O/B tags at the token level: that one is a different measure and is termed *accuracy* —and normally, the attained score in this measure is also considerably higher.

2 Literature review

The TERN (Temporal Expression Recognition and Normalization) task has gathered attention only recently within the broader framework of Information Extraction (IE), and references to it in the research literature are still sparse. Temporal expression extraction has been most often approached as an intermediate step towards tackling the more difficult problems of event extraction and characterization, extraction of temporal relations (e.g. before, after, includes, ...), temporal ordering and timestamping of events, and temporal reasoning ([BOG05], [PER05]).

Grammars continue to be the strategy most commonly resorted to for time analysis in documents (see [SAQ08] for an example of a grammatical treatment of recognition and co-reference resolution of time expressions). Boguraev & Ando (2005) [BOG05] describes an hybrid approach to the extraction of time expressions, events and temporal relations, which combines grammatical parsing with machine learning. First, a cascade of finite-state grammars is used to target temporal expressions and to extract attributes for their normalization; separately, a combination of token, POS (part-of-speech), syntax and context, in addition to word profiling from unannotated corpora, are used as features in a robust risk minization classifier for event and temporal expression recognition.

Here we are rather interested in the more statistical treatment of chunking afforded by methods such as support vector machines (SVM). Perrig (2005)

[PER05] reports about the results of various experiments using SVMs for the recognition of time expressions, events, and temporal relations. The precison, recall and F-measure results are given for several configurations of features.

Isozaki & Kazawa (2002) [ISO02] propose an efficient classification method for the related problem of NE (Named Entity) recognition using SVMs. They remark that SVM classification of NEs is considerably slow as compared to other techniques, and identify the cause of this slowness in the high number of mathematical operations (dot products with a high number of support vectors) required for classification. They exploit the fact that the feature vector representation typically obtained in natural language processing applications —as a result of expanding categorial variables into sets of binary features (see section 3 for details) — is largely sparse, in order to propose a quicker calculation method based on expanding the quadratic kernel formula and grouping similar terms, along with extensions for feature selection and reduction of training time. Further in relation with NE recognition, Mayfield et al. (2005) [MAY05] describes an approach that makes use of SVMs to estimate transition probabilities in a lattice, where vertices are given by the possible assignments of type-qualified I/O/B tags to the words in a sentence (i.e. I-LOC, B-PER, ...), and arcs assign conditional probabilities to a certain I/O/B tag assignment for a word given the tag assignment of the previous word (much in the resemblance of a HMM). Thus, the problem of NE recognition is equated to finding the path of the highest-likehood tag assignment in the lattice.

One problem that appears in trying to associate a probabilistic interpretation to the output of a SVM classifier is that SVMs do not output probabilities, but margins (i.e. distances to a discriminating hyperplane). Platt (1999) [PLA99] discusses a method for associating posterior class probabilities to the outputs of SVMs by estimating the parameters of a sigmoid function, which gives the probability of an example being positive given its margin.

Lastly, other approaches that have been used in information extraction to recognize useful mentions in text (such as named entities, events, relations, etc.) involve some forms of symbolic learning. Turmo & Rodriguez (2002) [TUR02] describe EVIUS, a relational learner that employs inductive logic programming (ILP) to automatically extract rules for information extraction (IE rules) in multiple domains. In Freitag (1998) [FRE98], the performance of three different systems (based on Bayesian learning, grammatical inference and relational learning, respectively) in acquiring extraction patterns for IE is compared.

3 SVMs for Temporal Expression Chunking: the YamCha tagger

This section discusses our approach to the problem of temporal expression chunking using SVMs. Both training and classification involve a previous step of document preprocessing: tokenization and extraction of features for each token. For our experiments we have used the YamCha (Yet Another Multi-purpose

CHunk Annotator) toolkit[1], a free implementation of a multipurpose chunker that uses SVMs as the underlying algorithm for chunking and classification of chunks (more specifically, YamCha employs the free TinySVM library[2]).

The text is first tokenized (segmented into individual tokens) and then a number of features are computed for each token. Tokens correspond normally with individual words, but also include punctuation marks —which are separate tokens— and special constructs like the saxon genitive "'s". Features that we use for tokens include:

- Lexical: The token form itself, the token in lowercase, the token that results from removing all letters from the token (e.g. 3 for "3pm"), the token that results from removing all letters and numbers (e.g. - - - for 1995-07-12).
- Morphological: The POS (Part Of Speech) tag (for instance, NN for noun, JJ for adjective, CD for cardinal number, MD for modal verb, ...).
- Syntactic: The tag of the syntactic chunk that the token belongs to (e.g. I-NP for inside noun phrase, B-VP for beginning of verb phrase, ...).
- Format features: These are flags that indicate if the token has certain format characteristics (*isAllCaps*, *isAllCapsOrDots*, *isAllDigits*, *isAllDigitsOrDots* and *initialCap*).
- Features that indicate if the token belongs to certain specific classes of words: *isNumber* (e.g. *one*, *two*, *ten*, ...), *isMultiplier* (e.g. *hundred*, *thousands*, ...), *isDay* (e.g. *monday*, *mon, saturday*, *sat*, ...) and *isMonth* (e.g. *january, jan*, *june*, *jun.*, ...).
- Contextual features: All of the previous features, but in reference to the tokens occurring in a certain window to the left and right of the current token.
- The target classification: The correct I/O/B tag for a window of tokens in the left context (we call these dynamic features).

The former set of features has points in common with those described in [BOG05, ISO02, MAY05] for identification of events and temporal relations, and for named entity recognition. It is a subset of the features outlined in Surdeanu et al. (2005) [SUR05] for the task of NE recognition.

The part-of-speech tags and the syntactic tags are computed using a suite of basic NLP tools that provide the POS tagger and the parser. The rest of features are computed programmatically, and arranged into a tabular format which serves as input to the YamCha tagger. A sample of the input to YamCha looks as follows (features other than the token form, POS tag, syntactic chunk and correct classification are omitted for simplicity):

[1] http://chasen.org/~taku/software/yamcha/
[2] http://chasen.org/~taku/software/TinySVM/

```
          FORM          POS tag   SYNTAX     I/O/B tag
POS: -3   Philippines   NNP       B-NP       B-TIMEX
POS: -2   ,             ,         O          O
POS: -1   March         NNP       B-NP       B-TIMEX
POS:  0   4             CD        I-NP       I-TIMEX
POS: +1   (             (         I-NP       O
POS: +2   AFP           NNP       I-NP       O
POS: +3   )             )         O          O
```

In the YamCha input file format, each line corresponds to an individual token from a document or a collection of documents (corpus). Individual tokens with their associated features make up the training and test examples. Thus, the first column corresponds to the token form (i.e. the token itself), the last column indicates the target classification for that token (also known as the *golden tag*), and the remaining columns correspond to the rest of the token's features. The generation of contextual features in a certain frame, left and right, of the current token is handled automatically by YamCha —basically, the features of tokens that fall into the context window are themselves appended as additional features of the current token. The length of the context window, to the left and to the right, are adjustable parameters of YamCha. Moreover, YamCha provides the ability to use the golden tags (the correct classification) of the tokens in a context from the current token as dynamic features.

As briefly mentioned already in the former section, a SVM classifier requires that examples be represented as a vector of numeric features (that is, if the usual kernels involving dot products are to be used). Many features commonly used in NLP learning tasks are categorial (e.g. the token form and variations thereof like lowercase and patterns, the POS tag, the syntactic tag, ...); and, moreover, some can have a domain of considerably many values —even several thousands—. For instance, in the case of the token form, every distinct word that appears in the training corpus becomes a value of the "token form" feature. Therefore, a transformation that maps categorial features into sets of *binary features* becomes a necessary step prior to training. The resulting feature vectors are very high-dimensional and sparse vectors (not unfrequently tens or hundreds of thousands of dimensions), with a majority of binary features indicating the presence or absence of a value of one of the original categorial features. If, in addition, we use contextual features from a number of tokens to the left and right, the number of dimensions grows accordingly. So, an expanded feature vector $\vec{x} = (x[1], \ldots, x[D])$ may look like this:

```
x[1] = 0            // Current word is not 'international'
x[2] = 1            // Current word is 'increase'
x[3] = 0            // Current word is not 'March'
x[4] = 0            // Current word is not '2000-06-12'
             .
             .
x[22379] = 0        // Current word is not a proper noun
x[22380] = 1        // Current word is a common noun
x[22381] = 0        // Current word is not an adjective
```

```
                .
                .
x[22432] = 1          // Current word is the start a noun phrase
x[22433] = 0          // Current word is not inside a noun phrase
x[22434] = 0          // Current word is not the start of a prep. phrase
                .
                .
x[68134] = 0          // Previous word is not 'sales'
x[68135] = 1          // Previous word is 'spectacular'
                .
                .
x[145372] = 0         // Next word is not a verb
x[145373] = 1         // Next word is a preposition
                .
                .
```

This "expansion" of the feature vectors is also internally handled by YamCha. With regard to the type of kernel, YamCha supports only polynomial kernels.

Support vector machines are in principle binary classifiers, whereas ours is a multiclass classification problem —each individual token is classified as I (inside), O (outside) or B (beginning)—. There exist two different strategies to tackle multiclass problems with SVMs, by reducing the multiclass classification to a series of binary decisions: one vs. rest (training one binary classifier per class in the data, treating examples of one class as positive and the rest as negative), and one vs. one (training one binary classifier per each pair of classes, to discriminate between examples of the two). While both strategies are implemented in YamCha, we have chosen to use one vs. rest classification with a view to minimizing the total time required for training of the models and for classification, although in our particular problem (3 classes) the number of binary classifiers to be trained turns out to be the same in both cases: three. Moreover, in the general case, the additional difficulty involved in combining the outputs of $\binom{n}{2}$ classifiers (one vs. one) to produce a coherent tag assignment would be greater than the difficulty of combining the outputs of n classifiers (one vs. rest), although this combination step is also handled internally by YamCha.

We have carried out experiments in which we altered different parameters of learning: the feature set, the degree of the polynomial kernel, and the length of the context window. The motivation behind these additional experiments was to quantify how variations in these three factors affected the classification performance. The results of the experiments are developed in the following section.

4 Results

In all of the following tables, the columns *precision*, *recall* and *F-1 measure* refer to these three performance metrics as they were defined in section 1. The precision, recall and F-measure scores are calculated in terms of mentions of time expressions: only cases where the classifier recovers a time expression whose boundaries match exactly those of the correct mention (as conveyed by the

	PREC	RECALL	F-1	ACC
Round 1	81.33	75.23	78.16	98.68
Round 2	77.74	70.46	73.92	98.60
Round 3	75.92	71.22	73.50	98.47
Round 4	80.05	73.71	76.75	98.65
Round 5	80.34	72.54	76.24	98.66
AVERAGE	**79.08**	**72.63**	**75.71**	**98.61**
STD DEV.	2.20	1.91	1.97	0.17

Table 1: 5-fold cross-validation with all features, quadratic kernel and a 2-token context window

returned vs. correct I/O/B tag assignments) are counted as a hits. The fourth column, *accuracy*, refers to the percentage of correct I/O/B tag assigments predicted by the classifier at the token level (i.e. whether the predicted tag coincides with the target tag).

First, we performed experiments using what we believed to be the optimal combination of features, learning algorithm parameters and context window length. For this process, the ACE 2005 corpus was split in five partitions. Special care was taken in order to obtain balanced partitions with regard to the number of documents from each of the five categories (i.e. newswire, broadcast news, broadcast conversations, conversational telephone speech and weblogs). In each round of 5-fold cross-validation, 80% of the corpus was used for training (4 of the 5 partitions) and 20% (the remaining partition) for test.

Table 1 shows the results of this 5-fold cross-validation experiment. We employed the full set of token features described in section 3, a polynomial kernel of degree 2, and a context window of 2 tokens to the left and 2 to the right of the current token (only 2 tokens to the left for the dynamic feature). The only reason supporting our choice of the quadratic kernel as part of the "presumedly optimal" configuration is that the quadratic kernel is known to give, in general, the best results in NLP chunking and classification tasks (e.g. NE recognition and classification). In other words, the basis for this choice is merely empirical.

The reason for the overwhelwingly high score in the accuracy column — which measures, as said, matches between the assigned and correct IOB tags at the level of single tokens—, as opposed to the much more modest 75.71% in F-measure is that the vast majority of tokens carry the O tag (for being outside of any time expression), which is easy to classify. Also, the lower score in recall with respect to precision, indicates that there exist an important subset of time expressions that the classifier cannot identify correctly.

Other sets of experiments look at the effect of varying the set of features used, the degree of the polynomial kernel and the length of the context window. Owing to the rather long time required to train one SVM model for classification (approx. 8 hours with YamCha running in a computer cluster), these sets of experiments have not been carried out using the full 5-fold cross-validation

KERNEL	PREC	RECALL	F-1	ACC
pol. lineal	72.39 (-7.66)	70.08 (-3.63)	71.21 (-5.54)	98.25 (-0.40)
pol. quadratic	**80.05**	**73.71**	**76.75**	**98.65**
pol. cubic	81.30 (+1.25)	71.73 (-1.98)	76.21 (-0.54)	98.65 (+0.00)

Table 2: Effects on performance of varying the degree of the kernel

procedure as above. Instead, we have trained a single model for each different configuration, using the same data partitions for training and test as in Round 4 of the cross-validation (because the results for that round were closer to the average). Note that we are using a corpus of considerable size for training (440 documents and approximately 205000 tokens [examples] in the training sets), and that training a model involves 3 different classifiers (one for each of the I, O and B tags).

Table 2 shows the gain in performance achieved by varying the degree of the polynomial kernel. The increments between brackets indicate the difference with respect to the reference configuration: the quadratic kernel, in this case. As seen, the quadratic kernel gives the best performance, and the best balance between results and complexity. The linear kernel lacks discriminating power, and the small improvement achieved (if at all) with a cubic kernel does not compensate for the added time complexity.

In order to study which types of features are relevant, we have designed three incremental models that use increasingly complex sets of features. The features used in each of these models are as follows:

- Model 1: the token form and the token form in lowercase.
- Model 2: Model 1 + POS tags + format features (isAllCaps, isAllCapsOrDots, isAllDigits, isAllDigitsOrDots, initialCap) + the token form removing alphabetic characters + the token form removing alphanumeric characters.
- Model 3: Model 2 + syntactic chunk tags + classes of words (isNumber, isMultiplier, isDay, isMonth).

Table 3 shows the effect of incrementally using a richer feature set. The increments between brackets indicate the difference with respect to the reference configuration: the maximal feature set (model 3). The results indicate that all of the features initially considered (i.e. those described previously in section 3) contribute, at least positively, to classification performance. A context window of 2 tokens left and right and a quadratic kernel were used in all three models.

And last, table 4 shows the results obtained from increasing the size of the context window used for the contextual features. It can be observed that passing from a context of 2 tokens to the left and right to a context of 3 tokens in each direction goes in detriment of classification performance. Enlarging the context window causes a dramatical increase in the number of dimensions of the feature

FEATURES	PREC	RECALL	F-1	ACC
Model 1	80.00 (-0.05)	66.89 (-6.82)	72.86 (-3.89)	98.56 (-0.09)
Model 2	80.10 (+0.05)	71.73 (-1.98)	75.68 (-1.07)	98.60 (-0.05)
Model 3	**80.05**	**73.71**	**76.75**	**98.65**

Table 3: Effects on performance of incrementally extending the feature set

WINDOW	PREC	RECALL	F-1	ACC
-2 .. +2	**80.05**	**73.71**	**76.75**	**98.65**
-3 .. +3	80.30 (+0.25)	71.29 (-2.42)	75.52 (-1.23)	98.59 (-0.06)

Table 4: Effects of increasing the context window

vectors, which may be a reason for decrease in performance (too many new unuseful features being introduced).

5 Conclusions and future work

Our optimal model for temporal expression chunking yields a F-1 score of 75.71%, which although far from being a success (provided that we are only tackling the task of detecting mentions of time expressions, not normalization), is not a despicable result. As a matter of comparison, the best system in the ACE 2005 evaluation[3] for the task of Named Entity Recognition (at the level of identifying the entity mentions) —different from, but somehow comparable to time expression recognition— achieved an average correctness of 85.1%. The best system for the task of Time Expression Recognition *and Normalization* in ACE 2005 achieved a score of 63.7%. The task of identification of the mentions of time expressions alone (for the English language) was not evaluated in ACE 2005.

The best performance in a task like time expression recognition, which has an important syntactic component, will still be achieved by a system that integrates grammatical analysis of some kind. The main drawback of using grammars is that grammar rules have to be tailored specifically to the task at hand. This is one of the reasons why we wanted to try out a purely statistical method such as SVMs and observe the classification performance that such a method affords.

Two potentially positive aspects of SVMs, regarding their application to our problem, are: first, SVM's resistance to overfitting, even in the presence of very high dimensional feature spaces (as those that arise typically in NLP problems); second, the fact that SVMs allow one to freely experiment with differents sets of features, and that by the use of non-linear kernels one can entrust the learning method with automatically finding interesting combinations of these features.

A deeper analysis of the test results, looking for the actual tokens where the returned and the actual time expression mentions mismatch, reveals that

[3] http://www.nist.gov/speech/tests/ace/ace05/doc/ace05eval_official_results_20060110.htm

most of the errors revolve around a small set of short-length time expressions that repeat a lot (e.g. *now*, *new*, *the day*, *today*, *future*, *year*, *right now*, *once*, *past*, *latest*, ...). On the other side of the spectrum, there are large numbers of errors due to long and complex time expressions that appear only once. We believe that a good percentage of the errors produced with these short-length time expressions are due to these words having different interpretations depending on the context, some of which are part of a time expression while some are not. Studying the context where these errors occur and trying to achieve better discrimination of these short and frequent time expressions (by contextual features) would most likely improve the results. Conversely, it would be difficult to attain improvements by focusing on the infrequent and complex time expressions.

In the meanwhile, we are exploring relational learning approaches to the time expression recognition task (e.g. ILP, inductive logic programming), and expect to be able to reach better recognition rates by using some combination of the statistical (SVMs, for the time being) and relational learners. A possible approach to try out could be to integrate new features based on some inductively learned logic predicates (ILP), as additional token features for SVM classification. Another line of improvement would be to perform a more in-detail analysis of in which contexts time expressions appear in the corpus —and in particular, which contexts are responsible for most classification errors—, and to introduce lists of *trigger words* that provide a clue as to the presence of a time expression mention.

Bibliography

[BOG05] Boguraev, B. and Ando. R. K. (2005). TimeML-Compliant Text Analysis for Temporal Reasoning. In *Proc. of the 19th International Joint Conference on Artificial Intelligence,* pp. 997–1003. Edinburgh, Scotland.

[FER03] Ferro, L. et al. (2003). *TIDES Standard for the Annotation of Temporal Expressions v1.3.* Technical Report, MITRE Corporation.

[FRE98] Freitag, D. (1998). *Machine Learning for Information Extractionin Informal Domains.* PhD thesis, Carnegie Mellon University, 1998.

[ISO02] Isozaki, H. and Kazawa H. (2002). Efficient Support Vector Classifiers for Named Entity Recognition. In *Proc. of the 19th International Conference on Computational Linguistics*, Vol. 1, pp. 1–7. Association for Computational Linguistics, Morristown, NJ, USA.

[MAY05] Mayfield, J., McNamee, P. and Piatko, C. (2003). Named Entity Recognition using Hundreds of Thousands of Features. In. *Proc. of the 7th Conference on Natural Language Learning.* HLT-NAACL 2003. Association for Computational Linguistics.

[PER05] Perrig, C. (2005). *Identification of Time Expressions, Signals, Events and Temporal Relations in Texts.* ETH Zurich, Switzerland.

[PLA99] Platt, J. (1999). Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Advances in Large Margin Classifiers*, in Smola, A., Bartlett, P., Scholkopf, B., Schuurmans, D. eds. MIT Press.

[SAQ08] Saquete, E., Muñoz, R. and Martínez-Barco, P. (2004). Event Ordering Using TERSEO System. In *Proc. of the 9th International Conference on Application of Natural Language to Information Systems,* NLDB 2004, pp. 39–50. LNCS Vol. 3136. Springer.

[SUR05] Surdeanu, M., Turmo, J. and Comelles, E. (2005). Named Entity Recognition from Spontaneus Open-Domain Speech. In *Proc. of the 9th European Conference on Speech Communication and Technology (Interspeech).*

[TUR02] Turmo, J. and Rodriguez, H. (2002). Learning Rules for Information Extraction. *Natural Language Engineering* 8, pp. 167–191. Cambridge University Press.

[WIL01] Wilson, G., Mani, I., Sundheim, B., Ferro, L. (2001). A Multilingual Approach to Annotating and Extracting Temporal Information. In *Proc. Workshop on Temporal and Spatial Information Processing*, Vol. 13, pp. 1–7. ACM.