

A Named Entity Recognition System based on a Finite Automata Acquisition Algorithm

Muntsa Padró and Lluís Padró
TALP Research Center
Universitat Politècnica de Catalunya
{mpadro, padro}@lsi.upc.edu

Resumen: En este artículo presentamos un nuevo sistema para el reconocimiento de nombres propios en español. Este sistema está basado en el algoritmo CSSR (Causal-States Splitting Reconstruction) (Shalizi and Shalizi, 2004) que aprende un autómata de estados finitos partiendo de datos secuenciales. Los resultados obtenidos son ligeramente peores que los mejores sistemas presentados en la “shared task” del CoNLL 2002, pero dada la simplicidad de los atributos utilizados, estos resultados son realmente prometedores y creemos que pueden ser fácilmente mejorados introduciendo más información al sistema.

Palabras clave: Reconocimiento de Nombres Propios, Automatas de Estados Finitos, Aprendizaje Automático

Abstract: In this work, a new Named Entity Recognition system for Spanish is presented. This system is based on Causal-State Splitting Reconstruction algorithm (Shalizi and Shalizi, 2004), which learns a finite automaton from data sequences. The obtained results are slightly below the best systems presented in CoNLL 2002 shared task, though given the simplicity of the used features, they are really promising. Furthermore, we think that these results can be easily improved by introducing more information in the system.

Keywords: Named Entity Recognition, Finite State Automaton, Machine Learning

1 Introduction

Named Entity Extraction (NEE) task consists of detecting lexical units in a word sequence, referring to concrete entities and of determining which kind of entity the unit is referring to (persons, locations, organizations, etc.). This information is used in many NLP applications such as Question Answering, Information Retrieval, Summarization, Machine Translation, Topic Detection and Tracking, etc., and the more accurate the extraction of Named Entities (NE) is, the better the performance of the system will be.

NEE consists of two steps that could be approached either sequentially or in parallel. The first step is Named Entity Recognition (NER) which consists of detecting which parts of a text belong to a NE. The second step is Named Entity Classification (NEC) that consists of classifying the NEs into a set of predefined classes (person, location, organization, etc). This work presents a system to perform the first step (NER) using an algorithm that learns finite state automata from data.

There are several approaches to Named Entity Extraction task. *Message Understanding Conferences* (MUC), devoted to Information Extraction, included a NEE task that led to different approaches to the task. Some of them were

hand-made, knowledge-based, such as (Black, Rinaldi, and Mowatt, 1998; Appelt et al., 1995; Weischedel, 1995; Krupka and Hausman, 1998). There were also data-driven approaches, (Bikel et al., 1997; Aberdeen et al., 1995) and hybrid approaches combining machine learning techniques with gazetteer information or hand-made rules (Yu, Bai, and Wu, 1998; Borthwick et al., 1998; Mikheev, Grover, and Moens, 1998). More recent machine learning approaches to NEE task are framed in CoNLL-2002 (Tjong Kim Sang, 2002) and CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003) shared tasks. Both tasks consisted of detecting and classifying NEs, using different languages

In this paper, a new method to perform NER is presented. This method is applied to detect NEs in Spanish texts. It is based on CSSR (Causal State Splitting Reconstruction) algorithm, which learns a finite state automaton given sequential data. Since NEs present some regular-expression patterns, it is expected that this kind of algorithm can learn this structure and use it for the detection of NEs

The results presented in this paper are preliminary, since the performed experiments take into account few features. Nevertheless, the obtained results are quite promising since they are

not far from those of the state-of-the-art systems and there is still a large margin for improvement to the presented preliminary results. At the sight of which it can be said that NER task can be successfully approached using this automaton learning algorithm.

Next section presents the CSSR algorithm and its theoretical basis. Section 3 studies how the algorithm learns automata from text sequences. Section 4 defines our approach to apply the algorithm to NER task. In section 5 the performed experiments with the obtained results are discussed and section 6 states some conclusions and future work.

2 CSSR algorithm

The CSSR algorithm (Shalizi and Shalizi, 2004) performs the blind construction of asymptotically optimal nonlinear predictors of discrete sequences. It infers the causal states from data, searching for optimal predictors for discrete random processes, in the form of Markov Models.

Given a discrete alphabet Σ , consider a sequence x^- (history) and a future Z^+ . Z^+ can be observed after x^- with a probability $P(Z^+|x^-)$. Two histories, x^- and y^- , are equivalent when $P(Z^+|x^-) = P(Z^+|y^-)$, i.e. when they have the same probability distribution for the future.

The different future distributions build the equivalence classes which are named *causal states* of the process. Each causal state is a set of suffixes (sequences of symbols drawn from alphabet) that represent histories (up to a preestablished maximum length) with the same probability distribution for the future. The causal states of a process form a deterministic machine and are recursively calculable.

2.1 The algorithm

Causal-State Splitting Reconstruction (CSSR) estimates an HMM inferring the causal states from sequence data. The main parameter of this algorithm is the maximum length (l_{max}) the suffixes can reach. That is, the maximum length of the considered histories. In terms of HMMs, l_{max} would be the potential maximum order of the model (the HMM would have l_{max} order if all the suffixes belonged to different states).

The algorithm starts by assuming the process is an identically-distributed and independent sequence with a single causal state, and then iteratively adds new states when it is shown by statistical tests that the current states set is not sufficient. The causal state machine is built in three phases (see (Shalizi and Shalizi, 2004) for details):

1. Initialize:

Create a state set with only one state containing only the null suffix. Set $l = 0$ (length of the longest suffix so far).

2. Sufficiency:

Iteratively build new states depending on the future probability distribution of each possible suffix extension (suffix sons). Before doing so it is necessary to estimate the probability distribution $\hat{P}(X_t|S = s)$ (where X_t is the random variable for the next alphabet symbol in the sequence) for each state s . This is necessary because this probability can change at each iteration when the new suffixes are added to a given state. This probability distribution is estimated (via maximum likelihood, for instance) using the data.

At this phase, the suffix sons (ax) for each longest suffix (x) are created adding each alphabet symbol (a) at the beginning of each suffix. The future distribution $P(X_t|X_{t-l}^{t-1})$ (probability of each alphabet symbol given the last l symbols) for each son is computed and a hypothesis test with the following null hypothesis is performed,

$$P(X_t|X_{t-l}^{t-1} = ax) = P(X_t|S = s); \quad \forall a \in \Sigma$$

This hypothesis is true if the new distribution is equal (with a certain confidence degree) to the distribution of an existing state (s). In this case, the suffix son is added to this state. If the hypothesis is rejected for all states, a new state for the suffix son is created. To check the null hypothesis we can use an statistical test such as χ^2 or Kolmogorov-Smirnov.

As the suffix length grows, l is increased by one at each iteration. This phase goes on until l reaches some fixed maximum value l_{max} , the maximum length to be considered for a suffix, which represents the longest histories taken into account. The results of the system will be significantly different depending on the chosen l_{max} value, since the larger this value is, the more training data will be necessary to learn a correct automaton with statistical reliability. Also, the time needed to learn the automaton grows linearly with l_{max} . So it is necessary to tune the best maximum length for the amount of available data (or viceversa, the amount of necessary data for the required suffix length).

3. Recursion:

Since CSSR models stationary processes, first of all the transient states are removed. Then the states are splitted until a deterministic machine is reached. To do so, the transitions for each suffix in each state are computed and if two suffixes in one state have different transitions for the same symbol, they are splitted into two different states.

At the end of this recursion phase, a deterministic automaton is obtained.

For extended details about CSSR algorithm see (Shalizi and Shalizi, 2004).

3 Studying the Algorithm Behaviour

Before applying CSSR algorithm to NER task, some experiments to study the algorithm behaviour were performed. The goal of these experiments was to prove that this method is able to capture sentence patterns, as well as studying the different automata learned using different CSSR parametrization. The CSSR input was CoNLL-2002 training corpus for Spanish encoded as a token sequence in a given alphabet. In our experiments, the following feature-sets were mapped to the alphabet symbols:

- **G**: Beginning of the sentence, capitalized, not containing numbers, not in the dictionary¹..
- **S**: Beginning of the sentence, capitalized, not containing numbers, one of its possible analysis being a common noun.
- **M**: Not at the beginning of the sentence, capitalized.
- **a**: Not at the beginning of the sentence, non-capitalized, functional word².
- **w**: Other.

Note that these features encoded in this alphabet are quite oriented to NER task, since it is the same alphabet that will be used later in NER experiments (Section 5). This alphabet is based on the FreeLing analyzer (Carreras et al., 2004) which has a NER system that uses a simple hand-built automaton of four states. The alphabet presented above is the same –and encode the same features– than the one used by FreeLing analyzer NE detection module. The automata learned via

¹The used dictionary is the one provided by FreeLing (Carreras et al., 2004)

²Functional words are articles or preposition that are often found inside a NE

CSSR are expected to capture the pattern of a sentence in terms of the features presented above.

Once the alphabet was set, the training corpus used in CoNLL-2002 was translated into this alphabet and then, different automata were learned using CSSR with different maximum lengths.

The obtained results show that CSSR is able to reproduce sentence patterns with this kind of alphabets. As CSSR algorithm builds probabilistic automata, some states that would be melted by a minimizing algorithm are maintained separately because they have different probability distributions. If the probabilities are ignored and the automata are minimized, the automata shown in figures 1 and 2 are obtained. It can be seen that they are logical automata that reproduce the intuitive pattern of sentences containing NEs.

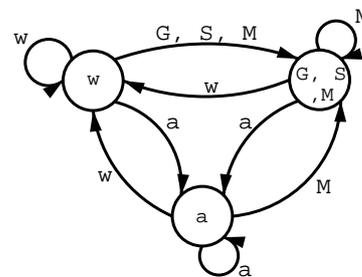


Figure 1: Learned automata with $l_{max} = 2$

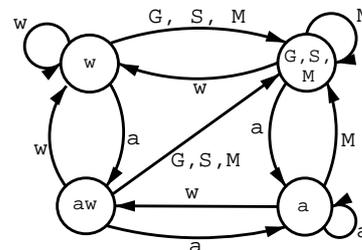


Figure 2: Learned automata with $l_{max} = 3$

These obtained automata can be compared with the hand-built automaton used by FreeLing NE detection module. In fact, the FreeLing automaton is the same that the obtained by CSSR using $l_{max} = 2$ (but with the information about a state being inside or outside a NE). It can be observed that the difference between the obtained automata with different maximum lengths is that as the length grows, the automata become more informed and tend to commit less over generalization. For example, for $l_{max} = 3$, there is an extra state that prevents the existence of the combination "awG" or "awS". In fact, this combination (which is accepted by the $l_{max} = 2$ automaton) is never seen in the data, because G and

S are symbols that only appear at the beginning of the sentence, so the w symbol preceding a G or a S will be the punctuation mark that ends the previous sentence and it couldn't be preceded by a preposition or an article (a) because a sentence never finishes with such a word.

Note that in these minimized automata there is no difference between the symbols G and S . In fact, the only difference is observed when the not-minimized automata, where G and S present different future probabilities so produce in different transitions. Furthermore, this distinction between G and S is maintained because it may be useful when performing NER.

4 Applying CSSR to Named Entity Recognition task

Following CoNLL 2002 and 2003 shared task, we worked with the "B-I-O" approach (Ramshaw and Marcus, 1995) to tag Named Entities. Each word has a B, I or O tag, being B the tag for a word where a NE begins, I the tag if the word is part of a NE but not the beginning, and O the tag for the words not belonging to any NE. There are other possible approaches to tagging NEs (Tjong Kim Sang and Veenstra, 1999) but this is one of the most widely used.

The general idea of our approach is to use CSSR to learn an automaton for NE structure. Once the automaton is learned, it can be applied to detect NEs in untagged text.

4.1 Learning the automaton

To learn the automaton that must reproduce NE structure, different information about the words is used. This information can be orthographic, morpho-syntactic, about the position in the sentence, etc. Using these features, the words in a sentence are translated to a closed set of symbols, that will be the alphabet of the automaton. The sentence translated in such a way will be the sequence that we use to learn the automaton via CSSR.

A problem of using that algorithm for this task is that it is conceived to model stationary processes, but NE patterns are not in this category. So, what we did was to regard a text sequence as a stationary process in which NEs occur once a while. Doing so implies the automaton is modeling the pattern of the sequence (the text), not the pattern of a NE.

To allow CSSR to learn the pattern of the NEs, we introduce in the alphabet the information of the NE-tag (B, I or O) available in the supervised training corpus. To do so, each symbol in the ba-

sic alphabet ($G, S, M...$) is splitted into three symbols, one for each NE-tag (the hidden information). So the correct NE-tag is taken into account for each kind of word when building the automaton.

In this way, although we obtain an automaton modeling the entire text sequence as an stationary process, we have information encoded in the transitions about B-I-O tags for NEs in the text. Thus, we can later use this information to compute the best path for a sequence and use it to tag NEs in a new text.

4.1.1 An Example

For instance, let's suppose an approach where the features taken into account are the same presented in section 3. In this case, the alphabet will consist of fifteen symbols, which are the possible combinations of each feature and a B-I-O tag ($G_B, G_I, G_O, S_B, S_I, S_O, M_B, M_I, M_O, a_B, a_I, a_O, w_B, w_I, w_O$). Each word will be translated into sequences of these symbols depending on whether it is capitalized and on its NE-tag.

Figure 3 shows an example of a possible training sentence and its translation into this alphabet. The first two columns would be the sentences as they are in the training corpus: a word and its right B-I-O tag. Last column is its translation into the alphabet, which will be used as input for the CSSR algorithm.

Word	Correct Tag	Alphabet Symbol
Fuentes	O	S_O
del	O	a_O
Vaticano	B	M_B
comentaron	O	w_O
la	O	a_O
salud	O	w_O
del	O	a_O
Papa	B	M_B
de	I	a_I
Roma	I	M_I
.	O	w_O
Navarro-Valls	B	G_B
afirma	O	w_O
que	O	w_O
está	O	w_O
estable	O	w_O
.	O	w_O
El	O	w_O
portavoz	O	w_O
...		

Figure 3: Example of a training sentence and its translation to the chosen alphabet

Once the data are properly translated into the alphabet, the automaton is built using CSSR.

4.2 Tagging NEs with the learned automaton

When a sentence has to be tagged, the information about the correct NE tag is not available, so there are several possible alphabet symbols for that word. It is only possible to know the part of the translation that depends on the word or sentence features. In our example, it would be possible to translate each word to a “*G*”, “*S*”, “*M*”, “*a*” or “*w*”, but not to know the part of the symbol that depends on the NE-tag, which is, in fact, what we want to know.

To find this most likely tag for each word in a sentence –that is, to find the most likely symbol of the alphabet (e.g. G_B , G_I , G_O for a G word), a Viterbi algorithm is applied. That is, for each word in a sentence, the possible states the automaton could reach if the current word had the tag B, I, or O, and the probabilities of these paths are computed. Then, only the highest probability for each tag is recorded. That means that for each word, the best path for this word having each tag is stored. At the end of the sentence, the best probability is chosen and the optimal path is backwards recovered. In this way, the most likely sequence of B-I-O tags for each word in the sentence is obtained. There are some forbidden paths, which are those that lead to the *OI* tag-combination. The paths including this combination are pruned out.

4.3 Managing Unseen Transitions

When performing the tagging of NEs given a text, it is possible to find symbol sequences that haven’t been seen in the training corpus. This will cause the automaton to fall in a *sink* state, which receives all the unseen transitions. This state can be seen as the state that contains all the unseen suffixes. All unseen transitions probabilities are smoothed to have a small probability of arriving to the sink state. Actually, the only sequences that have zero probability are those that have a forbidden combination of tags or of states being the beginning or the end of a NE.

When the automaton falls in the *sink* state, it can not follow the input sequence using transition information because, as the transitions weren’t seen, they are not defined. To allow the system to continue tagging the text, when the automaton falls into *sink* state, the suffix of length l_{max} is built using the last $l_{max} - 1$ symbols and the next symbol from the input. A state containing this new suffix is searched over the automaton and, if found, the automaton goes to this state and continues its normal functioning. If not, the process

is repeated, getting more symbols from the input sequence, until a state containing the new suffix is found.

This may cause skipping some part of the input, and is caused by the fact that the text sequence is considered as an stationary process, and so, when the CSSR-acquired automaton fails, we have to resynchronize it with the input data.

5 Experiments and Results

For the performed experiments, the data for the CoNLL-2002 shared task (Tjong Kim Sang, 2002) for Spanish were used. These data contain three corpora: one for the train and two for the test: one for the development of the system and the other one for the final test. The amount of data in each corpus is shown in table 1.

Corpus	# of words	# of NEs
Train	264,715	18,797
Test a	52,923	4,351
Test b	51,533	3,558

Table 1: Number of words/NEs in each corpus

With these data, some experiments were performed using CSSR as a NER system, in order to evaluate the accuracy of this system and to study the influence of the different parameters in its behaviour.

CSSR algorithm has three important parameters. One is the chosen maximum length (l_{max}), which is the most significant parameter. The other two are the test used to check the null hypothesis and the parameter α , controlling the test significance degree. We made several experiments for different l_{max} values and with two different statistical test: χ^2 and Kolmogorov-Smirnov. For each test, the experiments were performed with several α values. The used alphabet was the same one presented before.

For the discussion of the results we will focus on those obtained using Kolmogorov-Smirnov test. The results obtained with χ^2 test have a similar behaviour but are slightly worse and lead to bigger automata.

Figure 4 shows how the parametrization of the model affects to the size of the generated automaton.

As it was expected, the α value not only affects the performance of the system, but also changes the number of states of the generated automata. The larger α is, the greater the number of states will be. l_{max} also is very influent on the

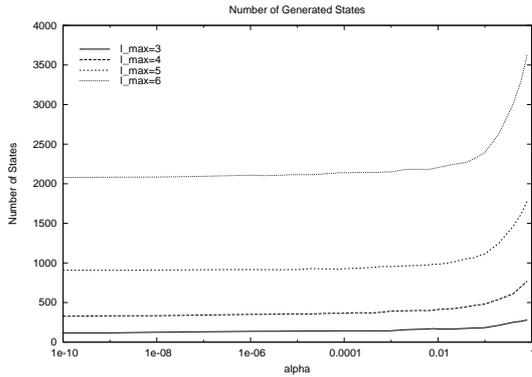


Figure 4: Number of states of the different learned automata using different l_{max} and α values

size of the automaton, growing rapidly the number of generated states with the chosen maximum length.

Figure 5 shows the obtained F_1 results with the different learned automata depending on the CSSR parametrization.

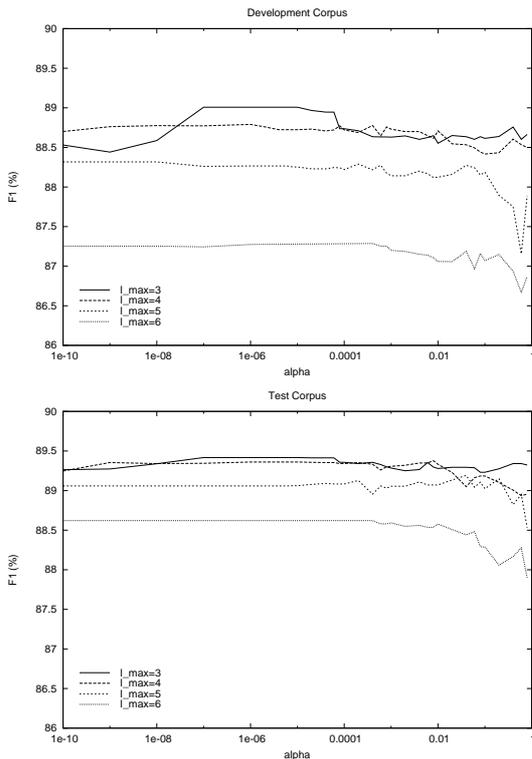


Figure 5: Obtained results with different l_{max} and α values for both test corpora

In this figure it can be seen that the significance degree value is not as influent as the l_{max} value. In fact, for α under 0.01 the reached results and the size of the built automata don't vary significantly with α .

About the influence of l_{max} , the results show

that best performance is obtained with small l_{max} , likely caused by the limited size of the training corpus, which seems not to allow statistically reliable acquisition of automata with too long histories.

The best performance is obtained with $l_{max} = 3$ and $\alpha = 1e - 5$. With these values, the system reaches a precision of 89.81%, a recall of 88.22% and $F_1 = 89.01\%$ for the development corpus (test a) and a 90.03% precision, 88.81% recall and $F_1 = 89.42\%$ for the test corpus (test b).

These results can be compared with the winner system of CoNLL-2002 shared task (Carreras, Màrquez, and Padró, 2002). This system was developed with the same training and testing data and performs the NE recognition and classification separately, so it is possible to compare our system with the part that performs the NE recognition.

That system obtained a F_1 of 91.66% for the Spanish development corpus and a 92.91% for the test corpus. This results are higher than the results presented in this work, which was expected since the feature set used by that system is much richer (bag of words, disambiguated PoS tag, many orthographic features, etc.) than the used in our experiments.

Furthermore, it is possible to apply the NEC system used by (Carreras, Màrquez, and Padró, 2002) to the output of our NE detector. Doing so over our best results yields to a $F_1 = 76.30\%$, which would situate our system in the fifth position in CoNLL-2002 ranking table for complete NER systems in Spanish as shown in table 2 (Tjong Kim Sang, 2002).

System	Precision	Recall	F_1
Carreras et.al.	81.38%	81.40%	81.39%
Florian	78.70%	79.40%	79.05%
Cucerzan et.al.	78.19%	76.14%	77.15%
Wu et.al.	75.85%	77.38%	76.61%
CSSR	76.82%	75.78%	76.30%
Tjong Kim Sang	76.00%	75.55%	75.78 %
Patrick et.al.	74.32%	73.52%	73.92 %
Jansche	74.03%	73.76%	73.89%
Malouf	73.93%	73.39%	73.66%
Tsukamoto	69.04%	74.12%	71.49%
Black et.al.	68.78%	66.24%	67.49%
McNamee et.al.	56.28%	66.51%	60.97%
baseline	26.27%	56.48%	35.86%

Table 2: Results presented by the participants of the CoNLL-2002 shared task compared to the results obtained by CSSR NER-system

6 Conclusions and Further Work

In this work, a new Named Entity Recognition system for Spanish has been presented. This system is based on a finite automata acquisition algorithm.

Firstly, the behaviour of the algorithm has been studied and it has been seen that the algorithm is able to learn the expected automata given word sequences translated into an adequate alphabet.

Secondly, our approach to use this algorithm to detect the NEs in a text and the performed experiments over the CoNLL-2002 corpora have been presented. It has been shown that this algorithm can build automata that give pretty good results when applied to recognize the NEs of a text. In fact, the system results are not too far from those obtained by the winner system on CoNLL 2002 shared task and they may be expected to improve by introducing more information in the system, since we use a much simpler knowledge than all CoNLL 2002 participants.

The main conclusion of this work, is that CSSR algorithm can be satisfactorily applied to NER tasks, which opens a door to applying it to other basic NLP tasks which need to learn sequential pattern information from data (PoS tagging, chunking, etc.)

The further work to be developed is focused on improving this NER system by introducing more orthographic and morpho-syntactic information in the alphabet in order to build more accurate automata. Similarly, external information such as trigger word lists or gazetteers could be also used.

Other future directions consist of applying CSSR algorithm to other NLP tasks such as chunking, PoS tagging or subcategorization pattern acquisition.

Acknowledgments

This research is being funded by the Catalan Government Research Department (DURSI), by the Spanish Ministry of Science and Technology (ALIADO TIC2002-04447-C02) and by the European Commission projects: Meaning (IST-2001-34460) and CHIL (IST-2004-506909). Our research group, TALP Research Center, is recognized as a Quality Research Group (2001 SGR 00254) by DURSI.

References

Aberdeen, John, John D. Burger, David Day, Lynette Hirschman, Patricia Robinson, and

Marc Vilain, 1995. MITRE: *Description of the ALEMBIC System Used for MUC-6*, pages 141–155. Proceedings of the 6th Message Understanding Conference. Morgan Kaufmann Publishers, Inc., Columbia, Maryland.

Appelt, Douglas E., Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Kehler, David Martin, Karen Myers, and Mabry Tyson, 1995. *SRI International FAS-TUS System MUC-6 Test Results and Analysis*, pages 237–248. Proceedings of the 6th Message Understanding Conference. Morgan Kaufmann Publishers, Inc., Columbia, Maryland.

Bikel, Daniel M., Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: A high performance learning name-finder. In *Proceedings of the 5th Conference on Applied Natural Language Processing, ANLP*, Washington DC. ACL.

Black, William J., Fabio Rinaldi, and David Mowatt. 1998. FACILE: Description of the new system used for muc-7. In *Proceedings of the 7th Message Understanding Conference*.

Borthwick, A., J. Sterling, E. Agichtein, and R. Grishman. 1998. NYU: Description of the MENE named entity system as used in muc-7. In *Proceedings of the 7th Message Understanding Conference*.

Carreras, Xavier, Isaac Chao, Lluís Padró, and Muntsa Padró. 2004. Freeling: An open-source suite of language analyzers. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal.

Carreras, Xavier, Lluís Màrquez, and Lluís Padró. 2002. Named entity extraction using adaboost. In *Proceedings of CoNLL Shared Task*, pages 167–170, Taipei, Taiwan.

Krupka, George R. and Kevin Hausman. 1998. Isoquest, inc.: Description of the netowlTM extractor system as used for muc-7. In *Proceedings of the 7th Message Understanding Conference*.

Mikheev, Andrei, Claire Grover, and Marc Moens. 1998. Description of the LTG system used for muc-7. In *Proceedings of the 7th Message Understanding Conference*.

Ramshaw, L. and M. P. Marcus. 1995. Text chunking using transformation-based learn-

- ing. In *Proceedings of the Third ACL Workshop on Very Large Corpora*.
- Shalizi, Cosma and Kristina Shalizi. 2004. Blind construction of optimal nonlinear recursive predictors for discrete sequences. *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference*.
- Tjong Kim Sang, Erik F. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan.
- Tjong Kim Sang, Erik F. and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.
- Tjong Kim Sang, Erik F. and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99*, pages 173–179. Bergen, Norway.
- Weischedel, Ralph, 1995. BBN: *Description of the PLUM System as Used for MUC-6*, pages 55–69. Proceedings of the 6th Message Understanding Conference. Morgan Kaufmann Publishers, Inc., Columbia, Maryland.
- Yu, Shihong, Shuanhu Bai, and Paul Wu. 1998. Description of the kent ridge digital labs system used for muc-7. In *Proceedings of the 7th Message Understanding Conference*.