# Wide–Coverage Spanish Named Entity Extraction[⋆]

Xavier Carreras, Lluís Màrquez, and Lluís Padró

TALP Research Center, LSI Department
Universitat Politècnica de Catalunya
Jordi Girona, 1–3, E-08034, Barcelona
{carreras,lluism,padro}@lsi.upc.es

**Abstract.** This paper presents a proposal for wide-coverage Named Entity Extraction for Spanish. Correct identification of Named Entities appearing in a text provides very useful knowledge for many Natural Language Processing tasks and applications. The extraction of named entities is treated using robust Machine Learning techniques (AdaBoost) and simple attributes requiring non tagged corpora complemented with external information sources (a list of trigger words and a gazetteer). A thorough evaluation of the task on real corpora is presented in order to validate the appropriateness of the approach.

## 1 Introduction

There is a wide consensus about that Named Entity Extraction is a Natural Language Processing (NLP) task which provides important knowledge not only for anaphora and correference resolution, but also to improve the performance of many applications, such as Information Extraction, Information Retrieval, Machine Translation, Query Answering, Topic detection and tracking, etc.

From 1987 to 1999, the *Message Understanding Conferences* (MUC), devoted to Information Extraction, included a *Named Entity Recognition* task, which *de facto* determined what we usually refer to with the term *Named Entity*, and established standard measures for the accuracy of a system performing this task.

In MUC, the Named Entity Recognition task is divided into three subtasks: the Name Extraction (ENAMEX), the Time Extraction (TIMEX), and the Number Extraction (NUMEX) tasks. The first consists of recognizing and classifying the names for persons, locations and organizations. The second refers to the extraction of temporal expressions (dates, times), and the last one deals with monetary and percentage quantities.

The techniques used in systems addressing this task cover a wide spectrum of approaches and algorithms traditionally used in NLP and AI. Some systems

[⋆] This research has been partially funded by the Spanish Research Department (HERMES TIC2000-0335-C03-02, PETRA TIC2000-1735-C02-02), by the European Comission (NAMIC IST-1999-12392), and by the Catalan Research Department (CIRIT's consolidated research group 1999SGR-150 and CIRIT's grant 1999FI 00773).

rely on heavily data-driven approaches [3, 1], while others use only hand–coded knowledge, [4, 2, 13, 8]. Finally, there are also hybrid systems combining corpus evidence and hand-coded knowledge, or external information sources [14, 5, 9]. It is remarkable that most research is performed on English, and that —to our knowledge— there is no work referring to Spanish.

We approach the task excluding the equivalent to the NUMEX and TIMEX tasks in MUC (i.e., we do not consider time or numerical expressions, which being frequent and easy to detect and classify, have the effect of raising the final accuracy figures). In addition, the task we approach is somewhat more difficult than MUC ENAMEX since we consider not only PERSON, LOCATION, and ORGANIZATION classes, but also a fourth category OTHERS which includes named entities such as documents, measures and taxes, titles of art works —cinema, music, literature, painting, etc.— and others.

The system uses Machine Learning (ML) components for the recognition and classification of simple entities. The ML modules use a large set of extremely simple contextual and orthographic information, which do not require any previous linguistic processing. Some experimental evaluation is presented confirming the validity of the approach proposed. We also test whether the NE classification performance significantly improves when using external knowledge sources (such as gazetteers or lists of trigger words).

The overall organization of the paper is the following: Section 2 presents the addressed task. Sections 3 and 4 are devoted to describe the algorithms and sources of information used to recognize and classify NEs. Section 5 describes the experimental evaluation of the model in a general Spanish corpus from a news agency. Finally, section 6 presents the main conclusions of the work and outlines some directions for future research.

## 2 Named Entity Extraction

In what respects to NE extraction, two sub-tasks must be approached: Named Entity Recognition (NER) —consisting of detecting the boundaries for each entity— and Named Entity Classification (NEC) —consisting of deciding whether the NE refers to a person, a location, an organization, etc.

We follow the approach of performing each task as soon as the necessary information is available. In this way, NER is performed during morphological analysis, since it requires only context information on word forms, capitalization patterns, etc. NEC is performed after the morphological analysis and before the tagging since no significant improvement were obtained when adding lemma and PoS information. NER and NEC tasks will be performed sequentially but independently.

Formally, NER can be seen as the task of segmenting a sequence of words into non–overlapping and non–recursive chunks (i.e., the NEs). From this point of view, a NE is defined by its starting and ending points. There are several possible classification models for this task:

- OpenClose: Two classifiers are used: one classifier to decide if a certain word in the sentence opens a new NE and another one to decide if it closes an already opened NE.
- IOB: A single classifier decides whether each word is the beginning of a NE (B tag), if it is a component of a NE, but not the first word (I tag), or if it is outside a NE (O tag).

There are several ways of using these classifiers to perform the NER task. The simplest and most efficient way consists in exploring the sequence of words in a certain direction and applying the *open* and *close* classifiers (or, alternatively, the IOB classifier) coherently. This greedy approach is linear in the number of words of the sentence. Given that the classifiers are able to provide predictions, which may be translated into probabilities, another possibility is to use dynamic programming for assigning the sequence of tags that maximize a global score over the sequence of words, taking into account the coherence of the solution [10,7].

In this work, for simplicity and efficiency reasons, the greedy approach has been followed. Initial experiments on applying global inference have provided no significant improvements.

Finally, it is worth noting that NEC is simply a classification task, consisting of assigning the NE type to each potential, and already recognized, NE. In this case, all the decisions are taken independently, and the classification of a certain NE cannot influence the classification of the following ones.

## 3 Learning the Decisions

The AdaBoost algorithm has been used to learn all the binary decisions involved in the extraction of NEs. The general purpose of the AdaBoost algorithm is to find a highly accurate classification rule by combining many *base* classifiers. In this work we use the generalized AdaBoost algorithm presented in [12], which has been applied, with significant success, to a number of problems in different research areas, including NLP tasks [11]. In the following, the AdaBoost algorithm will be briefly sketched —see [12] for details.

Let $(x_1, y_1), \ldots, (x_m, y_m)$ be the set of $m$ training examples, where each $x_i$ belongs to an input space $\mathcal{X}$ and $y_i \in \mathcal{Y} = \{+1, -1\}$ is the class of $x_i$ to be learned. AdaBoost learns a number $T$ of base classifiers, each time presenting the base learning algorithm a different weighting over the examples. A base classifier is seen as a function $h : \mathcal{X} \to \mathbb{R}$. The output of each $h_t$ is a real number whose sign is interpreted as the predicted class, and whose magnitude is a confidence rate for the prediction. The AdaBoost classifier is a weighted vote of the base classifiers, given by the expression $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$, where $\alpha_t$ represents the weight of $h_t$ inside the combined classifier. Again, the sign of $f(x)$ is the class of the prediction and the magnitude is the confidence rate.

The base classifiers we use are decision trees of fixed depth. The internal nodes of a decision tree test the value of a boolean predicate (e.g. "the word *street* appears to the right of the named entity to be classified"). The leaves of a tree define a partition over the input space $\mathcal{X}$, and each leave contains the prediction of the tree for the corresponding part of $\mathcal{X}$.

In [12] a criterion for greedily growing decision trees and computing the predictions in the leaves is given. Our base learning algorithm learns trees following this criterion, having a maximum_depth parameter as the stopping criterion. These base classifiers allow the algorithm to work in a dimensional feature space that contains conjunctions of simple features.

## 4 Information Sources and Features

The features used to take the decisions in NER and NEC tasks can be obtained from untagged text and may be divided into context features and external knowledge features. For the latter, we used a 7,427 trigger-word list typically accompanying persons, organizations, locations, etc., and a 10,560 entry gazetteer containing geographical and person names.

Due to the nature of AdaBoost, all features are binarized, that is, there is a feature for each possible word form appearing at each position in the context window. Although this creates large feature spaces, the AdaBoost algorithm is able to deal with such dimensionality appropriately (i.e., efficiently and preventing overfitting to the training examples).

### 4.1 NER Features

All features used for training the classifiers in the NER task, refer to context. Contrary to the NEC task, it has been empirically observed that the addition of knowledge from gazetteers and trigger words provides only very weak evidence for deciding the correct segmentation of a NE.

Since the basic AdaBoost algorithm is designed for binary classification problems, we have binarized the 3–class IOB problem by creating one binary problem for each tag. Therefore, each word in the training set —labelled as I, O, or B— defines an example, which is taken as positive for its class and negative for the rest. The following features are used to represent these examples:

- The form and the position of all the words in a window of $\pm 3$ words, and including the focus word (e.g., *word*(-1)="estadio").
- An orthographic feature and the position of all the words in the same $\pm 3$ window. These orthographic features are binary and not mutually exclusive and consider whether the $\pm i$_th word is: *initial-caps*, *all-caps contains-digits*, *all-digits alphanumeric*, *roman-number*, *contains-dots*, *contains-hyphen acronym*, *lonely-initial*, *punctuation-mark*, *single-char*, *function-word*, and *URL*.
- I, O, B tags of the three preceding words.

In the OpenClose scheme, the *open* classifier is trained with the words at the beginning of the NEs as positive examples, and the words outside the NEs as negative examples. The feature codification of these examples is the same as in the IOB case.

The *close* classifier is trained only with examples coming from words internal to the NEs, taking the last word of each NE as a positive example, and the rest as negative examples. In this case, the decision of whether a certain word should close a NE strongly depends on the sequence of words between the word in which the NE starts and the current word (i.e., the structure of the partial NE). For the words in a [-2,+3] window outside this sequence, exactly the same features as in the IOB case have been considered. The specific features for the inner sequence are the following:

- Word form and orthographic features of the focus word and the word starting the NE.
- Word form and orthographic features of the words inside the sequence taking its position with respect to the current word.
- Length in words of the sequence.
- Pattern of the partial entity, with regard to capitalized or non-capitalized words, functional words, punctuation marks, numbers, and quotations.

### 4.2 NEC Features

A binary classifier is trained for each NE class. Each training occurrence is used as a positive example for its class and as a negative example for the others. All classifiers use the following set of features:

- Context features: Form and position of each word in a $\pm 3$–word window (e.g. *word*(-2)="presidente").
- Bag-of-words features: form of each word in a $\pm 5$–word window of five words left and right of the entity being classified. (e.g. banco $\in$ *context*).
- NE inner features: Length (in words) of the entity being classified, pattern of the entity with regard to acronyms, numbers, capitalized words, prepositions, determiners, and punctuation marks.
- Trigger word features: Class and position of trigger words in a $\pm 3$–word window. Pattern of the entity immediate left context, with regard to punctuation marks, prepositions, determiners, trigger words denoting person, location, organization, or other entities, and trigger words denoting geographical origin,
- Gazetteer features: Class (geographical, first name, or surname) and position of gazetteer words in a $\pm 3$ window. Class in gazetteer of the NE being classified and class in the gazetteer of its components.

## 5 Evaluation

### 5.1 The EFE Spanish Corpus

The EFE corpus used for the evaluation of the whole named entity processing system is a collection of over 3,000 news agency articles totalling 802,729 words, which contain over 86,000 hand tagged named entities. A corpus subset containing 65,000 words (4,820 named entities) is reserved for evaluation tests and the remaining is used as training material.

For the NER task a only a subset of 100,000 words of the training set has been used[1]. The exact number of examples and features derived from the training corpus for each binary decision of the NER task is described in table 1. For the NEC task, the whole training set, consisting of some 81,000 NE occurrences, has been used. According to the features defined in section 4, these examples produce near 89,000 features.

Both in NER and NEC, features occurring less than 3 times in the training corpus have been filtered out. For the NEC task, this reduces the feature space to some 22,000 dimensions.

| | #Exs. | #Feat. | #Pos.examples |
|---|---|---|---|
| open | 91,625 | 19,215 | 8,126 (8.87%) |
| close | 8,802 | 10,795 | 4,820 (54.76%) |
| I | 97,333 | 20,526 | 5,708 (5.86%) |
| O | 97,333 | 20,526 | 83,499 (85.79%) |
| B | 97,333 | 20,526 | 8,126 (8.35%) |

**Table 1.** Sizes and proportions of positive examples in the NER binary decisions

## 5.2 Experimental Methodology

We trained the system using different feature sets and number of learning rounds, using base classifiers of different complexities, ranging from stumps (simple decision trees of depth 1) to decision trees of depth 4.

The evaluation measures for NER are: number of NE beginnings correctly identified ($B$), number of NE endings correctly identified ($E$), and number of complete NEs correctly identified. In the last case, recall ($R$, number of entities correctly identified over the number of expected entities), precision ($P$, number of entities correctly identified over the number of identified entities), and $F$-measure ($F_1 = 2 \cdot P \cdot R/(P + R)$) are computed.

The evaluation measures for NEC task include the accuracy of the binary classifiers for each category, as well as the evaluation of the combined classifier, which proposes the final decision based on the outcomes of all binary classifiers. The combination performance is measured in terms of recall, precision, and $F_1$. The accuracy ($Acc$) of the system when forced to choose one class per entity is also evaluated.

## 5.3 NER Results

Figure 1 contains the performance plots ($F_1$ measure) with respect to the number of rounds of the AdaBoost algorithm in the I, O, B binary decisions of the NER

[1] It has been empirically observed that using a bigger corpus does not result in a better performance for the task, while the number of examples and features greatly increase.

task. Note that decision trees perform significantly better than stumps and that a further increasing of the depth of the trees provides a small gain. Also, it can be noticed that all NER binary classifiers are quite accurate, with $F_1$ measure over 96%. The learning curves present a satisfactory behaviour, with no significant overfitting with larger number of rounds, and achieving maximum performance after a quite reduced number of rounds. Exactly the same properties hold for the curves corresponding to the open and close classifiers.

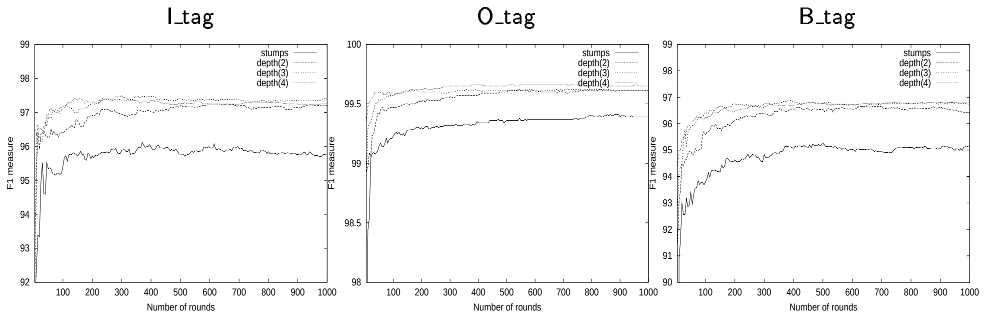

**Fig. 1.** $F_1$ measure w.r.t. the number of rounds of the I, O, B classifiers of NER task

Table 2 contains the results of the OpenClose and IOB approaches on the whole NER task, using depth–3 base classifiers. It can be observed that both variants perform significantly better than the baseline Maco+ [6]. The Maco+ NE module is a heuristic rule based NE recognizer, which takes into account capitalization patterns, functional words and dictionary lookup.

The performance of the OpenClose scheme is slightly worse than the IOB. This can be explained by the fact that when the *close* classifier wrongly decides not to end a NE then its output for the following words becomes unpredictable, since it enters into a situation not seen in the training phase.

This is confirmed by an additional experiment with a modified scheme consisting of applying the OpenClose scheme but after each negative prediction of the *close*, the I classifier is asked to confirm whether the following word is still inside the NE. If the I classifier gives a positive answer then the process continues normally, otherwise it is assumed that the *close* classifier was wrong. The positive answers of the *close* classifier are never questioned, since it is very accurate in its predictions. As it can be seen in table 2, this scheme (OpenClose&I) achieves the best results on the task.

Finally, table 3 presents the NER results depending on the length of the NE to recognize, as well as depending on whether the entity begins with uppercase or lowercase letter. As it could be expected, the performance degrades with the length of the sequence to be detected (specially on recall). However, a reasonable high accuracy can be expected for NEs of length up to six words. The set of NEs that begin with a lowercase word poses a very challenging problem for the NER module, specially due to the very shallow semantic treatment of the

| **Method** | $B$ | $E$ | $P$ | $R$ | $F_1$ |
|---|---|---|---|---|---|
| MACO+ | 90.83% | 87.51% | 89.94% | 87.51% | 88.71% |
| OpenClose | 94.61% | 91.54% | 92.42% | 91.54% | 91.97% |
| IOB | 95.20% | 91.99% | **92.66%** | 91.99% | 92.33% |
| OpenClose&I | **95.31%** | **92.14%** | 92.60% | **92.14%** | **92.37%** |

**Table 2.** Results of all methods in the NER task

training examples (captured only through the word forms, without any kind of generalization). We find very remarkable the precision achieved by the system on this subset of words (85.40%). The recall is significantly lower (63.93%), basically because in many occasions the *open* classifier does not have enough evidence to start a NE in a lowercase word.

| **Subset** | #NE | $B$ | $E$ | $P$ | $R$ | $F_1$ |
|---|---|---|---|---|---|---|
| length=1 | 2,807 | 97.04% | 95.80% | 94.64% | 95.65% | 95.15% |
| length=2 | 1,005 | 99.30% | 93.73% | 94.01% | 93.73% | 93.87% |
| length=3 | 495 | 93.74% | 88.69% | 91.65% | 88.69% | 90.14% |
| length=4 | 237 | 89.45% | 84.81% | 84.81% | 84.81% | 84.81% |
| length=5 | 89 | 87.64% | 76.40% | 77.27% | 76.40% | 76.84% |
| length=6 | 74 | 93.24% | 79.73% | 81.94% | 79.73% | 80.82% |
| length=7 | 22 | 59.09% | 54.55% | 60.00% | 54.55% | 57.14% |
| length=8 | 22 | 77.27% | 68.18% | 88.24% | 68.18% | 76.92% |
| length=9 | 11 | 90.91% | 72.73% | 80.00% | 72.73% | 76.19% |
| length=10 | 3 | 66.67% | 33.33% | 50.00% | 33.33% | 40.00% |
| uppercase | 4,637 | 96.42% | 93.25% | 92.81% | 93.25% | 93.03% |
| lowercase | 183 | 67.21% | 63.93% | 85.40% | 63.93% | 73.13% |
| TOTAL | 4,820 | 95.31% | 92.14% | 92.60% | 92.14% | 92.37% |

**Table 3.** Results of OpenClose&I on different subsets of the NER task

### 5.4 NEC Results

The binarization of the NEC problem used in this work consists of a binary classifier for each class (*one-vs-all* scheme). All NEC binary classifiers achieve an accuracy between 91% and 97% and show very similar learning curves in terms of number of rounds. As in the NER task, decision trees significantly outperform stumps.

With respect to the complete NEC system, the combination of binary decisions is performed selecting the classes to which binary predictors assigned a positive confidence degree. The system can be forced to give exactly one prediction per NE by selecting the class with higher confidence degree.

The results of all NEC systems are presented in table 4 (left part). The *basic* row refers to the model using context word, bag-of-words, and NE features as described in section 4.1. Results when the models include features obtained from lists of trigger words (*tw*) and gazetteers (*gaz*) are also presented. As a baseline we include the results that a dumb *most-frequent-class* classifier would achieve.

In all cases, the use of extra information improves the performance of the system, both in the binary decisions and in the final combination. The best result is achieved when both external resources are used, pointing out that each of them provides information not included in the other. Note that, although the individual performance of the binary classifiers was over 91%, the combined classifier achieves an accuracy of about 88%.

| **Method** | $P$ | $R$ | $F1$ | $Acc$ | $P$ | $R$ | $F1$ | $Acc$ |
|---|---|---|---|---|---|---|---|---|
| Most frequent | 39.78% | 39.78% | 39.78% | 39.78% | 37.47% | 37.47% | 37.47% | 37.47% |
| basic | 90.19% | 84.44% | 87.22% | 87.51% | 83.84% | 79.39% | 81.55% | 81.85% |
| basic+tw | 90.11% | 84.77% | 87.36% | 88.17% | 85.08% | 79.30% | 82.09% | 82.15% |
| basic+gaz | 90.25% | **85.31%** | 87.71% | 88.60% | 85.05% | 79.57% | 82.22% | 82.15% |
| basic+tw+gaz | **90.61%** | 85.23% | **87.84%** | **88.73%** | **85.32%** | **79.80%** | **82.47%** | **82.31%** |

**Table 4.** Results of all feature sets in the NEC task assuming perfect NE segmentation (left part), and using the real output of the NER module (right part)

Finally, the complete system is evaluated by testing the performance of the NEC classifier on the output of the NER module. Again, table 4 (right part) presents the results obtained. Performance is rather lower due to the error propagation of the NER module and to the worst-case evaluation, which counts as misclassifications the entities incorrectly recognized.

## 6 Conclusions and Further Work

We have presented a Named Entity Extraction system for Spanish based on robust Machine Learning techniques. The system relies on the usage of a large set of simple features requiring no complex linguistic processing. The performance of the learning algorithms is fairly good, providing accurate and robust NE recognizers and classifiers, that have been tested on a large corpus of running text. By adding extra–features from a gazetteer and a list of trigger words the performance has been further improved.

Comparing to state–of–the–art English systems, we have achieved a similar performance, though a cross–language comparison with other systems results is not reliable since, besides the differences from the linguistic phenomena addressed in each language, the evaluation corpora used as well as the criteria used to annotate them are not homogeneous.

The following are some lines of current work to improve the performance and quality of the system:

– Classification algorithms other than AdaBoost must be tested at NE extraction and results compared. Although this task fits well AdaBoost capabilities,

other algorithm such as Support Vector Machines may offer similar or better performances.
- Although the initial experiments have not reported good results, we think that the use of global inference schemes (instead of the one-pass greedy approach used in this paper) for assigning the sequence of tags deserves further investigation.
- NEC: The combination of the four binary classifiers obtains lower performance than any of them. Further combination schemes must be explored, as well as the use of multi-label AdaBoost algorithms instead of the binary ones.

## References

1. J. Aberdeen, J. Burger, D. Day, L. Hirschman, P. Robinson, and M. Vilain. MITRE: Description of the *ALEMBIC* System Used for MUC-6. In *Proceedings of the 6th Messsage Understanding Conference*, 1995.
2. D. E. Appelt, J. R. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Myers, and M. Tyson. SRI International FASTUS System MUC-6 Test Results and Analysis. In *Proceedings of the 6th Messsage Understanding Conference*, 1995.
3. D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: A High Performance Learning Name-Finder. In *Proceedings of the 5th Conference on ANLP*, Washington DC, 1997.
4. W. J. Black, F. Rinaldi, and D. Mowatt. FACILE: Description of the NE System Used for MUC-7. In *Proceedings of the 7th MUC Conference*, 1998.
5. A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. NYU: Description of the MENE Named Entity System as Used in MUC-7. In *Proceedings of the 7th Message Understanding Conference*, 1998.
6. J. Carmona, S. Cervell, L. Màrquez, M. Martí, L. Padró, R. Placer, H. Rodríguez, M. Taulé, and J. Turmo. An Environment for Morphosyntactic Processing of Unrestricted Spanish Text. In *Proceedings of the 1st LREC*, Granada, Spain, 1998.
7. X. Carreras and L. Màrquez. Boosting Trees for Clause Splitting. In *Proceedings of the 5th CoNLL Conference*, Tolouse, France, 2001.
8. G. R. Krupka and K. Hausman. IsoQuest, Inc.: Description of the NetOwl$^{TM}$ Extractor System as Used for MUC-7. In *Proceedings of the 7th Message Understanding Conference*, 1998.
9. A. Mikheev, C. Grover, and M. Moens. Description of the LTG System Used for MUC-7. In *Proceedings of the 7th Message Understanding Conference*, 1998.
10. V. Punyakanok and D. Roth. The Use of Classifiers in Sequential Inference. In *Proceedings of the 13th NIPS Conference*, 2000.
11. R. E. Schapire. The boosting approach to machine learning. an overview. In *Proceedings of the MSRI Workshop on Nonlinear Estimation and Classification*, Berkeley, CA, 2001.
12. R. E. Schapire and Y. Singer. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37(3), 1999.
13. R. Weischedel. BBN: Description of the PLUM System as Used for MUC-6. In *Proceedings of the 6th Messsage Understanding Conference*, 1995.
14. S. Yu, S. Bai, and P. Wu. Description of the Kent Ridge Digital Labs System Used for MUC-7. In *Proceedings of the 7th Message Understanding Conference*, 1998.